Credit Card Default Prediction Using Supervised Learning in R

Kasra Ghasemipoo

Academic Year: 2024/2025

Instructor: Prof. Silvia Salini

Università degli Studi di Milano

Introduction

Credit risk prediction is a core challenge in financial services. Accurately identifying customers who are likely to default on their credit card payments enables banks to minimize losses and manage risk more effectively.

In this project, I developed a supervised machine learning solution to predict credit card default using a high-quality **synthetic version of the UCI Credit Card Default dataset**. The goal was to build and evaluate classification models that can distinguish between customers who are likely to default (Yes) and those who are not (No) in the following month.

Using the R programming language, I implemented a complete pipeline including:

- Data preprocessing and factor conversion
- Exploratory data analysis (EDA)
- Class imbalance handling via upsampling
- Model training using **logistic regression** and **random forest**
- Performance evaluation using metrics like **accuracy**, **balanced accuracy**, and **AUC-ROC**

This project demonstrates how supervised learning techniques can be used to support **data-driven credit risk management**, with a focus on interpretability, fairness, and predictive performance.

Data Setup and Initial Exploration

1.1) Loading the Dataset

At the beginning of the project, I worked with a synthetic version of the UCI Credit Card Default dataset. This dataset simulates customer behavior for predicting default risk where the goal is to predict whether a customer will default (Yes) or not (No) in the next month.

1.2. Understanding the Dataset

The dataset had **49,999 rows** and **25 columns**, covering:

- **Demographics**: such as SEX, EDUCATION, MARRIAGE, and AGE
- **Financial Behavior**: including credit limit (LIMIT_BAL), past bill amounts (BILL_AMT1–6), and payments (PAY_AMT1–6)
- **Payment Status**: delays across past 6 months (PAY_0–PAY_6)
- **Target Variable**: default.payment.next.month (which I renamed to default for simplicity)

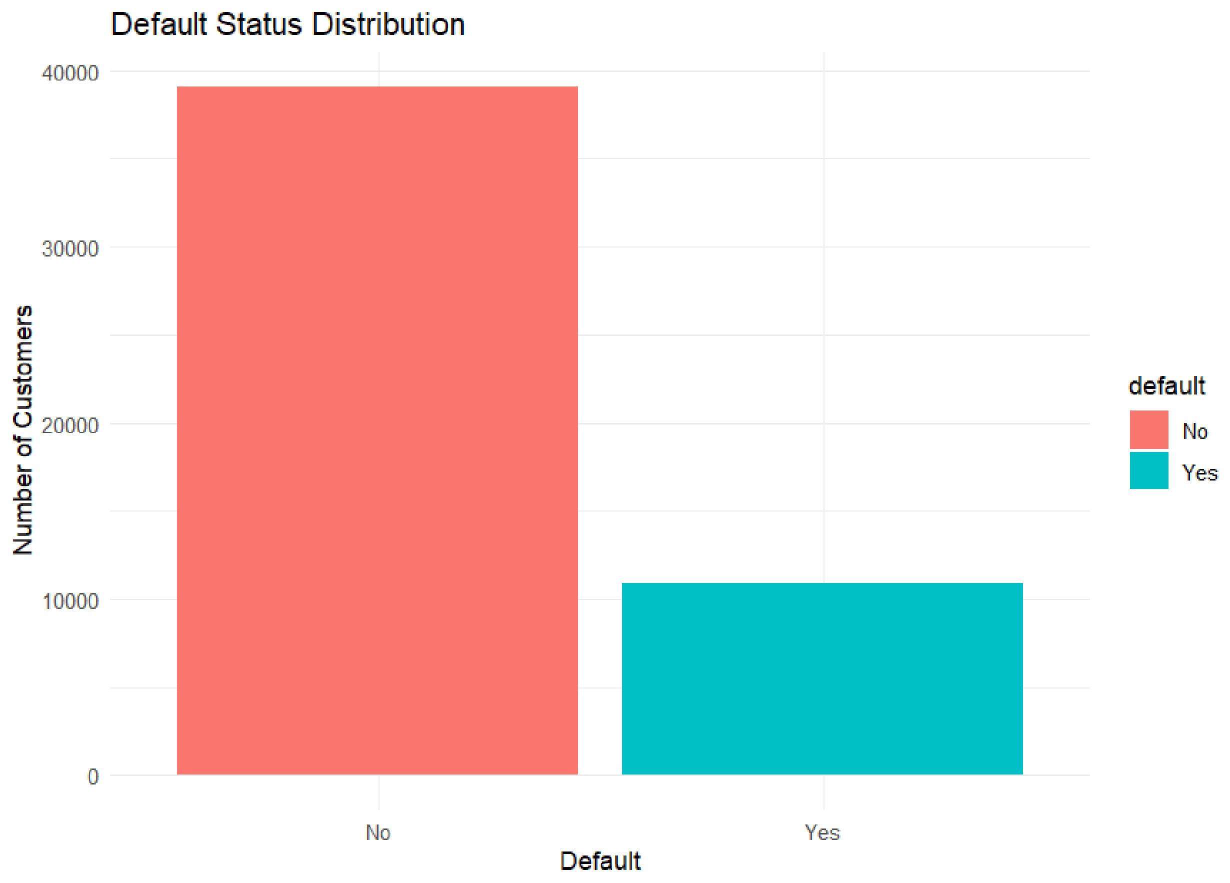
I also converted categorical variables (SEX, EDUCATION, MARRIAGE, and default) into factors, which is important for proper model interpretation in R

1.3. Class Distribution

After preparing the data, I explored the **distribution of the target variable** (default). I discovered that:

- 78% of customers did **not default** (label = "No")
- 22% of customers **did default** (label = "Yes")

This imbalance alerted me that **accuracy alone wouldn't be a good metric**, and that I needed to carefully evaluate how well the model predicts **both classes**, especially the minority class (Yes).

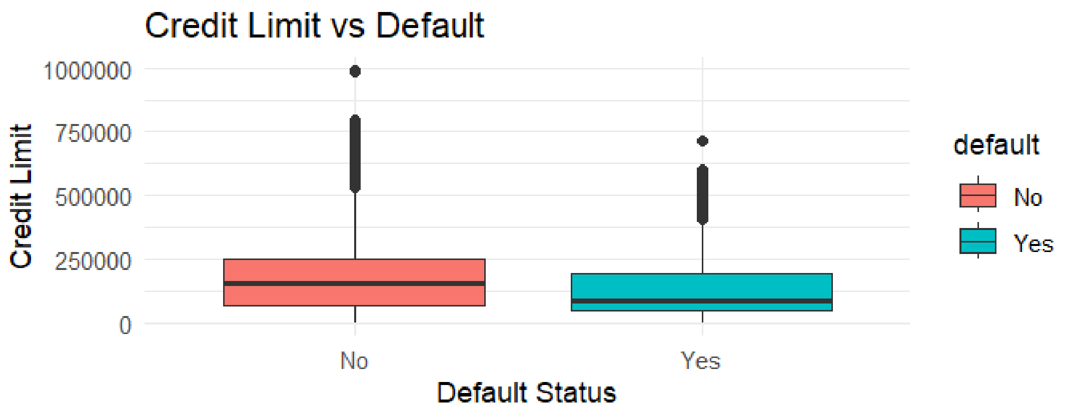


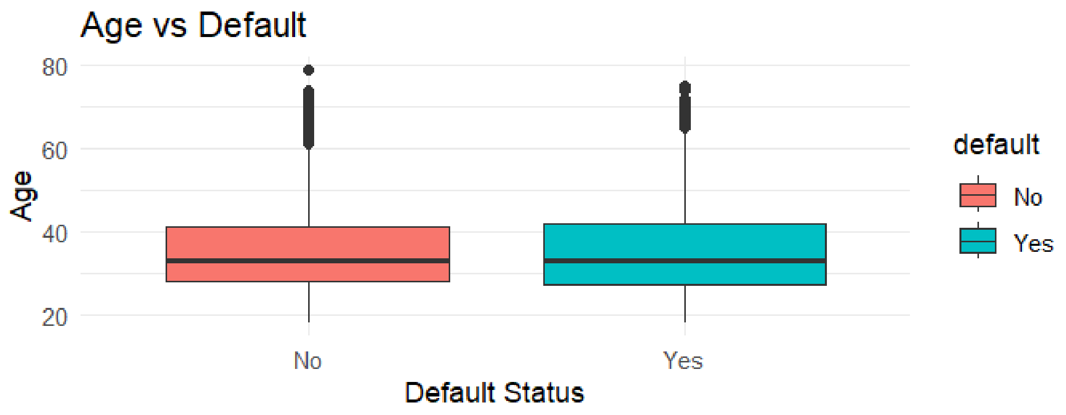
Exploratory Data Analysis (EDA) and Feature Insights:

2.1. Exploring the Target Variable

In this step, I started my **exploratory data analysis (EDA)** to better understand the relationships between input features and the target variable (default). I began by visualizing the class distribution using a bar plot. This confirmed what I had seen in Step 1: the dataset is imbalanced, with **significantly more non-defaulters (No)** than defaulters (Yes).

This reinforced the need to use **metrics beyond accuracy** and to consider techniques like **resampling** later in the modeling phase.





2.2. Feature Relationships with Default

I explored how key features relate to default status using boxplots and proportional bar charts. This gave me early insights into which variables might be strong predictors.

Visuals and Key Findings:

1. Credit Limit vs Default

Interpretation:

- The median credit limit is higher for customers who didn’t default.
- Customers who did default tend to have lower credit limits.
- There are outliers on both sides (some people with very high credit limits still defaulted or didn’t), but the distribution is lower for defaulters.

Conclusion:

Lower credit limits are associated with higher default risk, possibly because they reflect lower creditworthiness or tighter financial constraints.

2. Age vs Default

Interpretation:

- The median age for both groups is very similar.
- However, the boxplot for defaulters (Yes) seems slightly younger overall.
- There are some outliers among older customers, but they’re not common.

Conclusion:

Younger customers may be slightly more likely to default, but age alone is not a strong predictor.

3. PAY_0 (Repayment Status) vs Default

What PAY_0 means:

- It's the repayment status for the most recent month:
    - 0 = paid on time
    - 1 = 1-month delay
    - 2, 3, ..., 8 = longer delays
    - -1 or -2 = paid in advance or no due

Interpretation:

- People with on-time or advance payment (values 0, -1, -2) mostly do not default.
- As PAY_0 increases (payment delay gets worse), the proportion of defaulters grows significantly.
- At PAY_0 = 2, 3, 4, 5, etc., more than half of the customers default.

Conclusion:

PAY_0 is a strong predictor: recent payment delays are strongly associated with future default.

2.3. Insights Gained from EDA

This phase gave me two important lessons:

1. **Behavioral features** like PAY_0 and LIMIT_BAL carry more predictive power than demographic features like SEX or MARRIAGE.
2. I confirmed that I was dealing with a **real-world type of class imbalance** and that I'd need to use smarter evaluation methods and resampling strategies to avoid misleading model performance.

Modeling and Evaluation:

Algorithms We Will Use

1. Logistic Regression

- **Why?** Simple, fast, interpretable baseline
- **How it works:** Models the probability of default using a logistic function
- **Pros:** Easy to understand, works well with linear relationships

2. Random Forest

- **Why?** Powerful and handles nonlinear relationships & interactions
- **How it works:** Builds an ensemble (forest) of decision trees using random subsets of data/features
- **Pros:** Handles missing values and categorical variables well, provides feature importance

Class Distribution Summary:

- **No Default (No)**: 39,089 customers (≈ 78%)
- **Defaulted (Yes)**: 10,910 customers (≈ 22%)

Insight:

This is a **class imbalance** .most customers do **not** default.

This matters because:

- If we train a model without addressing it, it might just predict “No” all the time and still get high accuracy.
- We’ll need to look at **precision, recall, and F1-score**, not just accuracy.

3.1. Train-Test Split

I split the dataset into **80% training** and **20% testing**.This resulted in:

- **Training set**: 40,000 observations
- **Test set**: 9,999 observations

3.2. Logistic Regression (Baseline Model)

I first trained a **logistic regression model** as a baseline. It achieved:

- Accuracy: 80.9%

- Sensitivity (No): 97.5%
- Specificity (Yes): 21.3%
- Balanced Accuracy: 59.4%
- AUC: 0.7287

Confusion Matrix Overview:

	Actual No	Actual Yes
Predicted No	**7625**	**1718**
Predicted Yes	**192**	**464**

Key Metrics Explained:

Metric	Value	What It Means
Accuracy	0.809	~81% of total predictions are correct
Sensitivity (Recall)	0.9754	97.5% of the *non-defaulters* were correctly identified (true negatives rate)
Specificity	0.2126	Only 21.3% of actual *defaulters* were correctly identified (true positives)
Balanced Accuracy	0.594	Average of Sensitivity and Specificity (better for imbalanced data)
Kappa	0.25	Indicates low agreement beyond chance .common in imbalanced data

Interpretation:

What the model does well:

- It's excellent at catching non-defaulters (Sensitivity = 97.5%)
- High overall accuracy (81%)

What the model does poorly:

- It struggles to detect defaulters (only 21% of them are correctly flagged)
- This is because the model is biased toward the majority class ("No")

Summary for Logistic Regression

Logistic Regression model gives a strong baseline, but:

It's biased toward non-defaulters, which is common when the dataset is imbalanced (like yours: 78% No, 22% Yes).

3.3. Logistic Regression with Upsampling

To address the imbalance, I applied **upsampling** to the training data and retrained the model. Results:

- Accuracy: 67.4%
- Sensitivity: 67.6%
- Specificity: 66.4%
- Balanced Accuracy: 67.0%

- Kappa: 0.26

Interpretation

What improved:

- model is now much better at identifying defaulters ("Yes")!
- Specificity jumped from 21% → 66%, which means:

Now your model is correctly catching 1 in 3 defaulters, not just 1 in 5.

What dropped:

- Accuracy decreased to 67%, but that’s expected:

Accuracy is not reliable when data is imbalanced, so the drop is acceptable.

- Sensitivity also dropped, but that just means the model doesn’t say “No” as confidently . it’s now more balanced in its predictions.

Summary:

we traded a little accuracy for **a big gain in fairness**: now model predicts **both defaulters and non-defaulters much more evenly** ,and that’s exactly what we want in a real-world risk model.

Comparison: Before vs After Upsampling

Metric	Original Model	Upsampled Model	Change
Accuracy	80.9%	67.4%	Dropped (expected)
Sensitivity (No)	97.5%	67.6%	Lower (less biased)
Specificity (Yes)	21.3%	66.4%	Big improvement
Balanced Accuracy	59.4%	67.0%	Higher overall
Kappa	0.25	0.26	Slightly better

Why We Chose Upsampling:

Because:

1. we have a big dataset (≈50,000 rows)

   → We can afford to add some duplicates without harming model quality.

2. Undersampling would drop 78% of the "No" class

   → You'd go from 39,000 → 10,000 "No" cases = huge data loss!

3. Logistic regression (and later Random Forest) can handle duplicated data fairly well.

4. We're trying to improve model sensitivity to rare class (defaulters), and upsampling directly targets that.

NOTE:

We didn’t use undersampling because it would throw away too many good "No" examples .instead, upsampling let us balance the classes while keeping all the data.

3.4. Random Forest with Upsampling

Next, I trained a **Random Forest** model with the same upsampled data. It significantly outperformed logistic regression:

- Accuracy: 86.9%
- Sensitivity: 95.7%
- Specificity: 55.2%
- Balanced Accuracy: 75.5%
- Kappa: 0.57
- AUC: 0.8813 ✅

Random Forest Performance Summary

Metric	Value	What It Means
Accuracy	86.9%	Very strong .much better than the upsampled logistic regression
Sensitivity (No)	95.7%	Most non-defaulters correctly classified
Specificity (Yes)	55.2%	55% of defaulters correctly caught .much better than before!

Balanced Accuracy	75.5%	Strong . average of sensitivity & specificity
Kappa	0.57	Shows good agreement beyond chance

Interpretation

- Random Forest model is the best so far:
    - Maintains high sensitivity (finds most "No" correctly)
    - Significantly improves on catching defaulters (Yes) over the baseline logistic model
    - Has good balance and much stronger overall performance

This model achieved the **best balance** between catching defaulters and maintaining high accuracy.

Comparison Recap

Model	Accuracy	Sensitivity (No)	Specificity (Yes)	Balanced Accuracy	Kappa
Logistic (original)	80.9%	97.5%	21.3%	59.4%	0.25
Logistic (upsampled)	67.4%	67.6%	66.4%	67.0%	0.26
Random Forest (upsampled)	86.9%	95.7%	55.2%	75.5%	0.57

Feature Importance and Interpretation

4.1. Feature Importance (Random Forest)

I extracted variable importance from the final Random Forest model.

Top 5 Features:

1. **PAY_0** – recent repayment status (most predictive)
2. **LIMIT_BAL** – credit limit
3. **PAY_AMT4** – payment made 4 months ago
4. **PAY_AMT1** – recent payment
5. **PAY_AMT6** – older payment

➡️ These features reflect payment behavior, confirming that recent and consistent repayments reduce default risk.

4.2. ROC Curve Comparison

Why we do this:

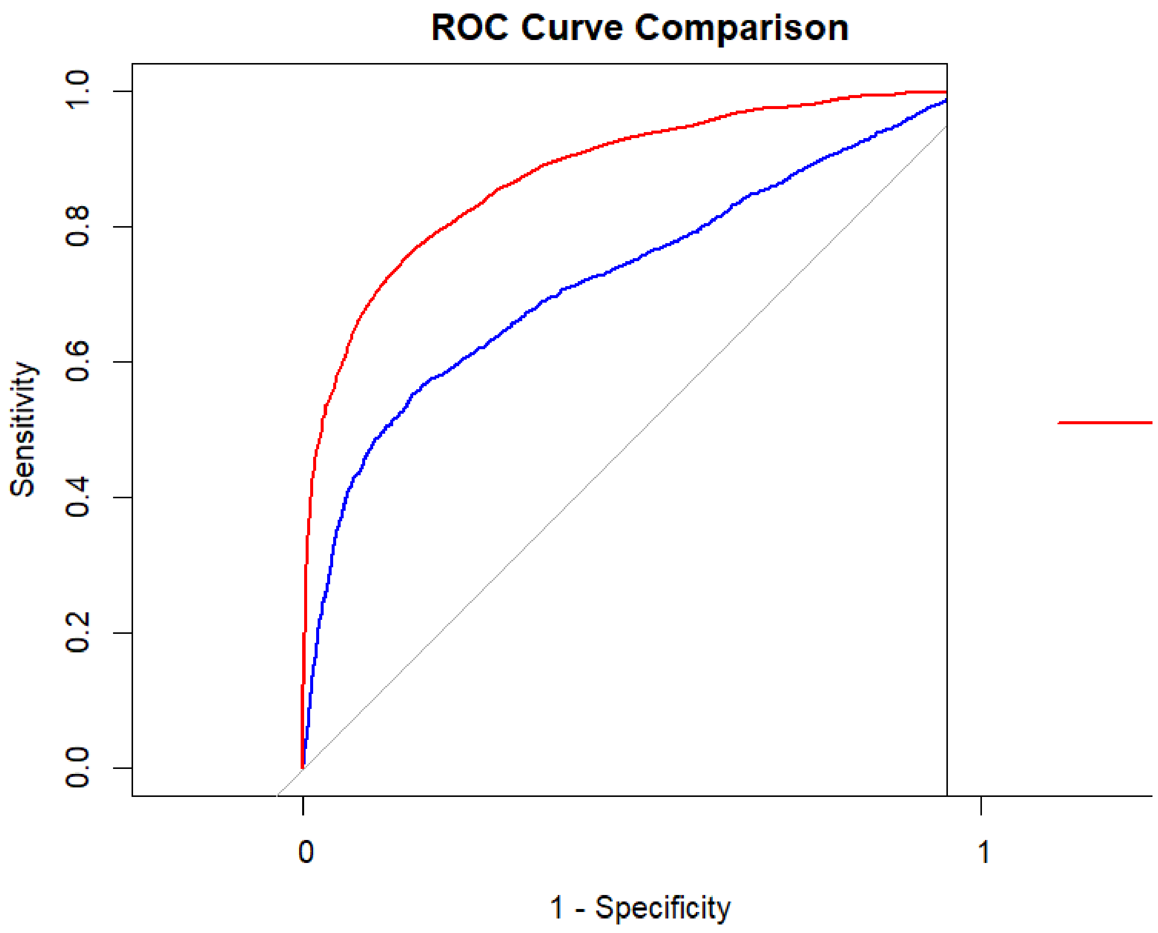
The **ROC curve** shows:

- Trade-off between True Positive Rate (Sensitivity) and False Positive Rate
- AUC (Area Under Curve) summarizes performance:
  - AUC = 1.0 → perfect classifier
  - AUC = 0.5 → random guess

I plotted ROC curves for both models:

Model	AUC
Logistic Regression	0.7287
Random Forest	0.8813 ✅

Red line : Random Forest   Blue line : regression



Summary and Conclusion

In this project, I developed a binary classification model to predict credit card default using a synthetic version of the UCI dataset. I followed a full machine learning pipeline in R, including:

- Data cleaning and factor conversion
- Class distribution analysis (78% No, 22% Yes)
- Exploratory Data Analysis (EDA)
- Model training with logistic regression and random forest
- Class imbalance handling through **upsampling**
- Performance evaluation using **accuracy**, **balanced accuracy**, and **AUC**

Best Model: Random Forest (with upsampling)

- Accuracy: 86.9%
- Balanced Accuracy: 75.5%
- AUC: 0.8813
- Kappa: 0.57

This model performed well on both classes, especially improving the detection of defaulters.

Key Insights:

- **PAY_0** (recent payment delay) is the most important predictor
- Credit limit (LIMIT_BAL) and payment amounts (PAY_AMT1–6) are also strong indicators
- Logistic regression was a useful baseline but struggled with class imbalance