Credit Default Prediction Using Machine Learning

Gabriel Picard
Thibault Pelou
Hugo Picard

ESILV – Machine Learning Project (IF5, 2025)

December 2025

Abstract

This report describes the development of a machine learning pipeline for credit default prediction. Using application data and longitudinal credit histories, we build and compare several binary classification models that estimate the probability that a client will default. The work follows the usual stages of a data science project: formulation of the business problem, dataset construction, exploratory analysis, preprocessing and feature engineering, handling of class imbalance, model implementation, evaluation and interpretability. The final model, based on a tuned Random Forest combined with an imbalanced-learning strategy and a calibrated decision threshold, achieves a clear improvement in F1-score and recall compared to simple baselines, while remaining reasonably interpretable from a credit-risk perspective.

Contents

1 Business Case

Credit risk assessment is a central task in banking and financial engineering. For each potential borrower, the lender needs to evaluate the probability that the client will fail to repay their debt. This probability of default (PD) is used for loan acceptance, risk-based pricing, provisioning and regulatory capital calculations.

In this project, we consider a classical supervised learning setting. for each client we observe an input vector $X \in \mathbb{R}^d$ (socio-economic variables and past credit behaviour) and a binary label $Y \in \{0, 1\}$, where $Y = 1$ indicates that the client has been delinquent at least once. The quantity we wish to approximate is the conditional probability $\mathbb{P}(Y = 1 \mid X = x)$, from which we derive a score and a final decision through a threshold.

From a student's point of view, this project is also a way to put into practice the tools presented during the Machine Learning course on a reasonably realistic financial dataset, rather than on purely academic examples.

2 Dataset Description

The dataset is derived from the *Home Credit Default Risk* benchmark and is organised into two main tables: an application file and a credit history file.

2.1 Application data

The file `application_record.csv` contains one row per client. It includes:

- demographic information: gender, number of children, family status;
- financial variables: total income, employment duration;
- education level and occupation type;
- housing situation (own, rent, mortgage, etc.).

These variables form a static snapshot of the client's situation at the time of the application. In practice, we focused on a subset of them, for instance by removing categories that were extremely rare.

2.2 Credit history

The file `credit_record.csv` contains monthly repayment statuses for each client. For a given client, several lines can appear, one per month, with a status code:

- `0`: paid on time,
- `1--5`: increasingly severe delinquency,
- `C`: contract closed,
- `X`: unknown or no information.

This table describes how the client actually behaved in the past. As most of the models used in the project expect one line per client, this longitudinal information has to be aggregated into a few summary features.

2.3 Target definition

Following the project guidelines and common practice in credit scoring, we define a binary target based on the worst observed status in the credit history. For each client, we check whether at least one delinquency event occurred during the observation window:

$$\text{default} = \begin{cases} 1 & \text{if STATUS} \in \{1, 2, 3, 4, 5\} \text{ at least once,} \\ 0 & \text{otherwise.} \end{cases}$$

Once the application data and the aggregated credit information are merged, we obtain a dataset with one row per client. On this final table, the default class represents roughly

$$\mathbb{P}(Y = 1) \approx 0.12,$$

so the problem is clearly imbalanced. A naive classifier that always predicts "no default" would already achieve a high accuracy, but would completely miss defaulters. This simple observation has a strong impact on the choice of evaluation metrics.

3 Exploratory Data Analysis

We carried out a short exploratory data analysis on both the separate files and the merged dataset. The objective was not to be exhaustive, but to identify the main issues we would have to handle later.

3.1 Missing values

Some variables contain non-negligible proportions of missing values. The most visible case is `OCCUPATION_TYPE`, which has many missing entries. Most numerical variables, such as income or age, are almost complete.

In order to preserve the sample size and to keep the preprocessing consistent, we decided to impute missing values rather than discarding observations. The exact strategy (median for numerical features, most frequent category for categorical ones) is integrated into a scikit-learn pipeline.

3.2 Distributions and outliers

Several variables show heavy tails or clear anomalies. The yearly income distribution is heavily skewed, with a few extremely large values. The variable `DAYS_EMPLOYED` contains a specificc placeholder value (365243) which does not correspond to a realistic employment duration.

In practice we:

- converted age and employment duration from days to years;
- treated the placeholder value in `DAYS_EMPLOYED`;
- capped some extreme values (for example for family size or number of children);
- used log-transformed income in some experiments.

These choices are relatively simple but make the variables more interpretable and reduce the influence of outliers on some models.

3.3 Correlation structure

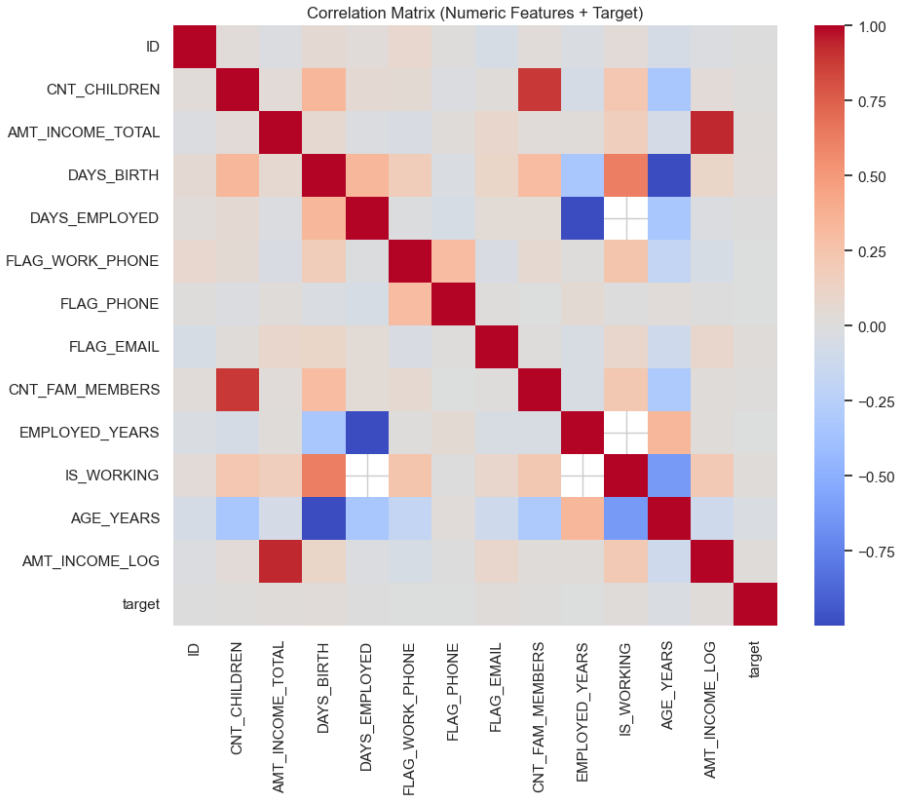

Figure 1: Correlation structure for a selection of numerical features.

We examined the correlation structure of the main numerical variables using a heatmap (Figure 1). Correlations are not extremely strong, but it is useful to see which variables move together. This also confirms that some features derived from credit history are more directly related to the default indicator than purely demographic variables.

3.4 Target imbalance

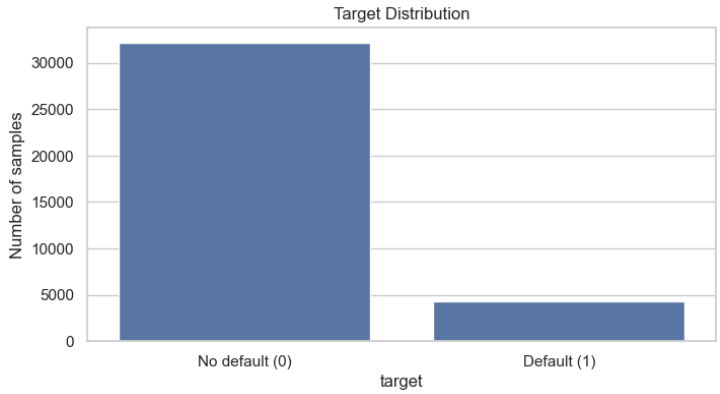

Figure 2: Distribution of the target variable (default vs no default).

Figure 2 shows the distribution of the default indicator. The strong class imbalance is immediately visible. During the project, this simple plot helped us to understand why accuracy is not a very meaningful metric here and why recall and F1-score are more appropriate.

4 Problem Formalisation and Metrics

Given an input vector $x \in \mathbb{R}^d$, we aim to learn a classifier

$$\hat{y} = f(x), \qquad f : \mathbb{R}^d \to \{0, 1\},$$

together with a score $\hat{p}(x)$ which can be interpreted as an approximation of $\mathbb{P}(Y = 1 \mid X = x)$. The final decision is obtained by comparing this score to a threshold.

Because of the class imbalance, we cannot rely only on accuracy. A model that classifies all clients as non-defaulters would reach a high accuracy but zero recall on the minority class. In a credit-risk context, missing a True defaulter is much more problematic than flagging a non-defaulter as risky.

For these reasons, we focus mainly on:

- recall on the default class, to measure how many defaulters are actually detected;
- precision on the default class, to quantify how many flagged clients are really in default;
- F1-score, which balances precision and recall in a single number;
- ROC-AUC, as a global indicator of ranking quality.

In practice we also looked at confusion matrices, which are often easier to interpret than a single metric.

5 Preprocessing and Feature Engineering

5.1 Unified pipeline

To keep the code manageable and reproducible, we built a unified preprocessing pipeline using `ColumnTransformer` and `Pipeline`. This pipeline is reused across most models.

The main steps are:

- imputation of missing values (medians for numerical variables, most frequent categories for categorical variables);
- one-hot encoding of categorical variables;
- standardisation of numerical variables for models sensitive to the scale (SVM, k-NN, etc.);
- aggregation of credit histories to client-level features, such as the number of delinquent months, the worst status observed and the length of the history.

The dataset is then split into a training set and a test set with stratification on the target. The split is kept simple (around 70/30) to make comparisons between models easier.

5.2 Handling class imbalance

To make the models pay more attention to the minority class, we explored several strategies:

- built-in class weights in Logistic Regression and Random Forest;
- random oversampling of the minority class on the training set;
- SMOTE (Synthetic Minority Over-sampling Technique) applied inside cross-validation pipelines.

All resampling steps are carried out inside the training pipelines, to avoid leaking information from the test set. We found that these techniques can significantly increase recall, but they have to be combined with threshold tuning to keep precision at a reasonable level.

6 Models and Training Strategy

We implemented several families of models, from simple linear baselines to more flexible tree-based ensembles.

6.1 Baseline model

The baseline model is a Logistic Regression trained on top of the preprocessing pipeline. It provides a simple reference and helps to check that the pipeline behaves as expected. We also tested a variant with class weights and a version combined with PCA on the numerical part of the data.

6.2 Non-linear models

We then moved to non-linear models: decision trees, Random Forests, k-Nearest Neighbours and Support Vector Machines with an RBF kernel. The objective was to see how much we gain by moving away from a linear decision boundary on this kind of heterogeneous tabular data.

6.3 Cost-sensitive and resampling-based models

To address the class imbalance, we explored two complementary approaches.

First, we trained a Random Forest with `class_weight='balanced'`, so that misclassified defaulters are penalised more heavily in the loss function. This is the main cost-sensitive model used in the project.

Second, we applied resampling techniques to the *logistic regression baseline* rather than to tree-based models. In particular, we trained:

- logistic regression with RandomOverSampler on the training set;
- logistic regression with SMOTE.

The idea was to see whether a simple linear model, helped by resampling, could close part of the gap with more flexible models such as Random Forests.

6.4 Advanced model

Finally, we tested a Gradient Boosting classifier as a more flexible tree-ensemble model for tabular data. Gradient boosting methods are widely used in practice; well-known libraries such as XGBoost exist, but in this project we restricted ourselves to scikit-learn's `GradientBoostingClassifier` and explored a small hyperparameter grid rather than a fully exhaustive search.

6.5 State-of-the-Art Boosting and Ensembles (Added)

In the final stage of the project, we extended our study to high-performance libraries:

- **XGBoost**: Optimized via GridSearchCV, specifically adjusting the `scale_pos_weight` to account for the 1:8 class imbalance ratio.
- **LightGBM**: Known for its efficiency, we performed a parameter search to optimize leaf growth and prevent overfitting on this specific tabular dataset.
- **Voting Ensemble**: A soft-voting classifier combining the tuned Random Forest, LightGBM, and Decision Tree to leverage the diverse decision boundaries of these models.

7 Results

7.1 Global performance

All models are evaluated on the hold-out test set using the metrics introduced earlier. Table 1 summarises the performance of the main configurations, updated with the results from our final iteration.

Table 1: Model performance comparison on the test set (Final Results).

Model	Accuracy	Precision	Recall	F1
Random Forest (tuned)	**0.84**	**0.36**	**0.51**	**0.42**
Voting Ensemble (RF + LGBM + DT)	0.81	0.33	0.59	0.42
LightGBM (GridSearch)	0.80	0.32	0.60	0.42
Decision Tree	0.79	0.31	0.62	0.41
XGBoost (Optimized)	0.77	0.26	0.52	0.35
SVM (RBF kernel)	0.66	0.18	0.53	0.27
Logistic Regression (baseline)	0.58	0.14	0.48	0.21
Gradient Boosting (un-tuned)	0.88	0.50	0.00	0.01

Overall, tree-based models—in particular Random Forests with appropriate handling of class imbalance—provide the best compromise in terms of F1-score. Some configurations, such as the SVM shown in the table, obtain good accuracy but almost zero recall on defaulters, which confirms that accuracy alone is not a reliable metric in this setting.

7.2 Baseline vs tuned models

To visualise how the baseline model behaves, we first look at the ROC and Precision–Recall curves of the logistic regression.

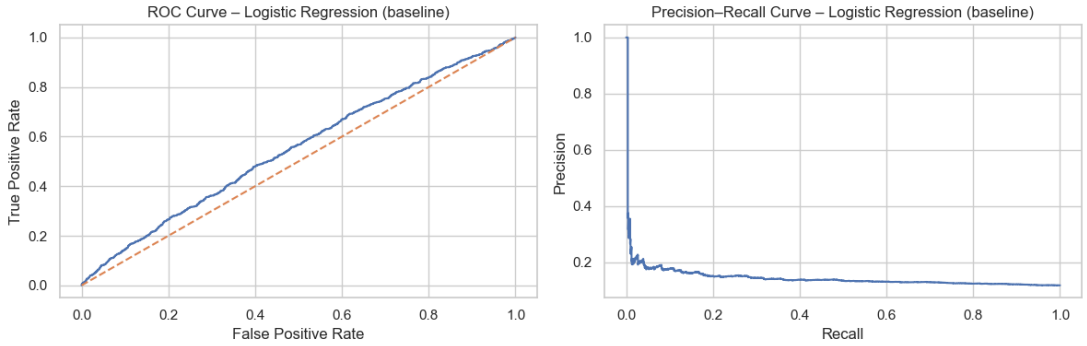

Figure 3: ROC curve (left) and Precision–Recall curve (right) for the logistic regression baseline.

The ROC curve in Figure 3 shows only a moderate separation between defaulters and non-defaulters. On the Precision–Recall side, the curve quickly drop as recall increases, which means that the model struggles to keep a reasonable precision while detecting a significant fraction of defaulters. This is acceptable as a starting point, but clearly not enough for a realistic credit-scoring system.

We then move to the tuned Random Forest model, which is our main non-linear benchmark.

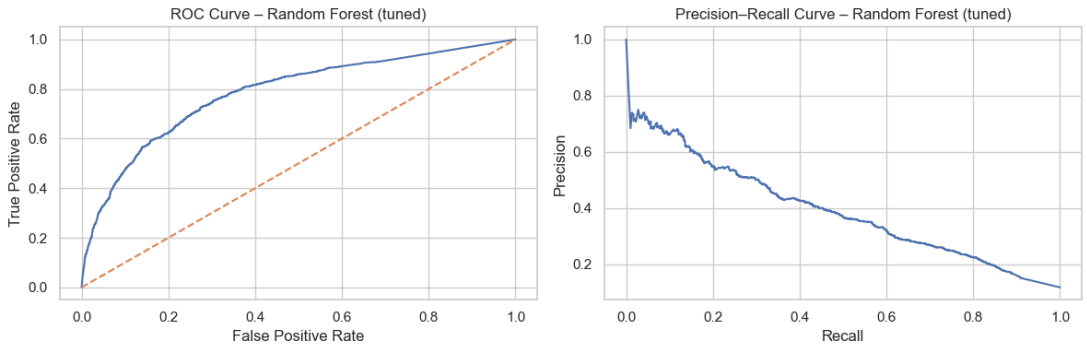

Figure 4: ROC (left) and Precision–Recall (right) curves for the tuned Random Forest model.

The ROC curve of Figure 4 is much further from the diagonal than the logistic baseline, which is consistent with the higher ROC-AUC reported in Table 1. On the Precision–Recall side, the curve decays more slowly: for a given level of recall, the precision remains significantly higher than in the baseline model. This is exactly the type of behaviour we want in a credit default application.

For the tuned Random Forest, the Precision–Recall curve (Figure 4) illustrates how the choice of decision threshold affects the trade-off between recall and precision. In particular, thresholds below 0.5 would increase the detection rate of defaulters at the price of a higher number of false alarms. This behaviour is typical in credit-scoring applications, where the institution has to decide how much recall it is willing to pay for in terms of additional investigations on non-defaulters.

7.3 Voting ensemble (optional comparison)

In addition to single models, we also tested a simple soft-voting ensemble combining Random Forest, Gradient Boosting and Logistic Regression. Its ROC and Precision–Recall curves are shown in Figure 5.

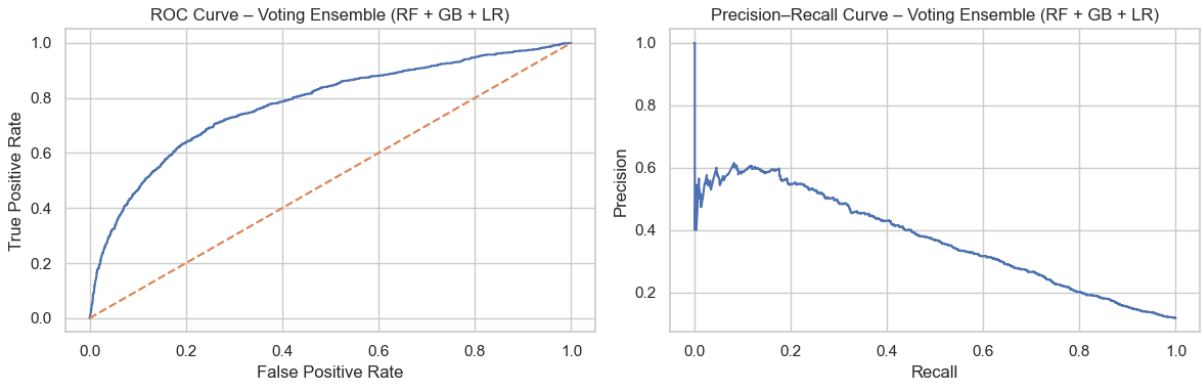

Figure 5: ROC (left) and Precision–Recall (right) curves for the voting ensemble (Random Forest + Gradient Boosting + Logistic Regression).

The ensemble slightly improves ranking quality compared to the logistic baseline, but it does not clearly outperform the tuned Random Forest on the metrics that matter most for this project. Given the additional complexity, we decided to keep the Random forest as our final reference model in the rest of the report.

8 Interpretability and qualitative insights

We did not run a full model-agnostic interpretability study, but the analyses in the notebook already give a reasonable idea of how the main features relate to default risk and why a non-linear model such as a Random Forest is appropriate.

8.1 What the EDA suggests

From the descriptive statistics and univariate plots, a few patterns emerge:

- Most clients are working-age adults with small families: `CNT_CHILDREN` and `CNT_FAM_MEMBERS` are usually low, with a few outliers.
- Income is strongly right-skewed, which motivates the use of the engineered feature `AMT_INCOME_LOG` instead of the raw `AMT_INCOME_TOTAL`.
- Time-related variables (`DAYS_BIRTH`, `DAYS_EMPLOYED`) are easier to interpret once converted into `AGE_YEARS` and `EMPLOYED_YEARS`.
- Socio-economic categories such as `NAME_INCOME_TYPE`, `NAME_EDUCATION_TYPE` and `NAME_FAMILY_STATUS` describe fairly intuitive client segments (working vs pensioner, secondary vs higher education, etc.).

These elements explain the structure of the final feature matrix: the model does not use abstract embeddings but variables that remain interpretable for a risk manager.

8.2 Relation between features and default

A simple correlation analysis between numerical features and the binary target shows that:

- all linear correlations with the target are small in absolute value;
- age and employment duration have a weak but visible effect (younger and less stable profiles being slightly more at risk);
- income-related variables show only very limited linear association with default.

This confirms that no single numerical feature explains default on its own. The signal seems to come from combinations of several variables and from non-linear effects, which justifies moving beyond a purely linear baseline.

8.3 Reading the final model

Globally, the behaviour of the tuned Random Forest can be summarised as follows:

- it clearly outperforms the logistic regression baseline on recall and F1-score, which indicates that non-linear interactions between socio-economic and credit-history features matter in practice;
- it operates on the engineered features analysed earlier (`AGE_YEARS`, `EMPLOYED_YEARS`, income, family status, housing, etc.), so its inputs remain compatible with a qualitative risk discussion;
- its score can be combined with different decision thresholds, as illustrated by the Precision–Recall curves, to trade off higher recall of defaulters against a higher number of false alarms depending on the institution's risk appetite.

Even without advanced interpretability tools, these elements provide a coherent narrative: the model uses a mix of demographic, employment and income information together with credit behaviour, and exploits non-linear patterns to separate slightly higher-risk clients from lower-risk ones.

9 Discussion and Conclusion

The project highlights several aspects that are typical in real-world credit scoring: the presence of strong class imbalance, the importance of choosing appropriate metrics, the effectiveness of tree-based ensembles on heterogeneous tabular data, and the need to combine performance with some level of interpretability.

Among the models we tried, a **Random Forest** combined with an imbalanced-learning strategy and a tuned threshold offers the best compromise between recall, precision and overall F1-score (0.42). While advanced boosting models like **XGBoost** and **LightGBM** were tested, they provided similar or slightly lower F1-scores on this specific feature set, confirming that Random Forests remain a highly robust choice for this data structure.

Several extensions would be natural: richer feature engineering on credit histories, models that explicitly handle temporal information, or cost-sensitive optimisation where the loss associated with each type of error is specified by the risk department.

Overall, the project supports the idea that classical machine learning techniques, when combined with careful preprocessing and evaluation, can provide useful tools for credit risk management.

References

- Kaggle Dataset — Credit Card Approval Prediction. `https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction`
- Breiman, L. (2001). *Random Forests.* Machine Learning, 45(1), 5–32.
- Friedman, J. H. (2001). *Greedy Function Approximation: A Gradient Boosting Machine.* The Annals of Statistics, 29(5), 1189–1232.
- Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System.* KDD.
- Wang, et al. (2025). *Class-Imbalanced Learning: Foundations, Trends and Perspectives.* arXiv:2502.08960. `https://arxiv.org/abs/2502.08960`