Credit-Card Default Prediction

Fintech

Antonio Giuseppe Doronzo

11016435

June 24, 2025

1 Introduction

Credit-card lending exposes financial institutions to the risk that a customer fails to settle the balance when it becomes due. Although any single default is inexpensive in absolute terms, large customer portfolios make even a modest default rate financially relevant. Academic literature and industry practice therefore devote considerable effort to statistical and machine-learning (ML) models that anticipate whether a client will default in the near future.

The present project focuses on the "Default of credit-card clients in Taiwan" dataset, which records demographic attributes, six-month billing and repayment history and the credit limit for 30000 clients. The binary target **default.payment.next.month** flags those who missed at least one payment in October 2005. Leveraging this benchmark dataset, we pursue two main objectives:

1. Train and optimise a baseline **Random-Forest** (RF) classifier that operates on the full set of engineered features.
2. Assess whether an **L1-penalised logistic regression (Lasso)** used as a feature selector can shrink the input space without degrading, ideally improving, the RF's predictive performance.

2 Problem Analysis

2.1 Business Framing

A missed default prediction (false negative) can lead to a complete loss on the outstanding balance and additional regulatory capital charges, whereas a false positive only generates a monitoring cost. Hence, the problem is a cost-asymmetric binary classification: we value recall on defaulters more than overall accuracy.
Given the imbalance in the data, a trivial "always-non-default" model would achieve 77.9,% accuracy but **zero recall**, illustrating why accuracy alone is misleading in this context.

2.2 Evaluation Protocol

- **Cross-validation:** stratified 5-fold on the training partition.
- **Hold-out test set:** 20% of the data, used exactly once for the final comparison.
- **Pipelines:** each encapsulates preprocessing (scaling, one-hot encoding) and, for the second approach, L1-based feature selection.

2.3 Candidate Models

1. **Random Forest (RF)** trained on all 25 features.
2. **Lasso → RF**: an L1-penalised logistic regression selects a reduced subset of features, which are then fed to an RF.

Hyper-parameter grids and tuning details are presented in the next section.

3 Dataset

3.1 Source and Observation Window

The analysis is based on the public *"Default of Credit Card Clients in Taiwan"* dataset (Yeh & Lien, 2009), retrieved from the UCI Machine Learning Repository (available on Kaggle and licensed as CC0: Public Domain) and mirrored in the project repository. Each record aggregates information available at the **end of September 2005** and labels whether the customer defaulted on the next payment due in **October 2005**.

3.2 Feature Typology

Group	Variables
Demographics	`SEX`, `EDUCATION`, `MARRIAGE`, `AGE`
Credit Limit	`LIMIT_BAL`
Repayment Status	`PAY_0`–`PAY_6`
Bill Statements	`BILL_AMT1`–`BILL_AMT6`
Previous Payments	`PAY_AMT1`–`PAY_AMT6`

Table 1: Overview of predictor groups. The identifier `ID` is removed before modelling.

3.3 Descriptive Statistics

- **Sample size:** 30 000 observations.
- **Missing values:** none detected across variables.
- **Credit limit:** median NT$ 140 000; 75th percentile NT$ 240 000.
- **Age:** interquartile range 29-37 years, modal peak ≈ 34.
- **Skewness:** monetary features exhibit heavy right tails (max credit limit = NT$1 000 000).
- **Class imbalance:** 22.1 % defaults (6 636) vs. 77.9 % non-defaults (23 364).

Most informative correlations (Pearson, absolute value > 0.15*):*

$$\texttt{PAY\_0}\ (r = 0.325), \quad \texttt{PAY\_2}\ (r = 0.264),$$
$$\texttt{PAY\_3}\ (r = 0.235), \quad \texttt{LIMIT\_BAL}\ (r = -0.154).$$

These figures align with financial intuition: recent arrears markedly increase default risk, whereas higher credit limits are generally granted to more reliable clients.

3.4 Pre-processing Rationale

1. **Target isolation** - cast `default.payment.next.month` to $\{0, 1\}$.
2. **Feature pruning** - drop `ID`; inspect near-zero-variance features.
3. **Categorical handling**

`SEX`, `MARRIAGE`:	one-hot encoding;
`EDUCATION`:	merge sparse codes 0, 5, 6 into "Unknown", then one-hot.

4. **Ordinal scaling** - keep `PAY_*` as ordered integers (tree models are scale-invariant).
5. **Continuous scaling** - apply *RobustScaler* (median centre, IQR scale) to monetary variables; log transforms were tested but offered no benefit.
6. **Train-test split** - stratified 80/20, random state 42, reused identically across all pipelines.

A deeper data exploration is available on the jupyter notebook in the repository.

4 Methodology

4.1 Pipeline Overview

Both candidate models are wrapped in a single SCIKIT–LEARN `Pipeline` that **encapsulates pre-processing and hyper-parameter tuning end-to-end**, thereby preventing any information leakage from the test set. The generic structure is:

ColumnTransformer → (optional) *Lasso selector* → *Random Forest classifier.*

4.2 Model 1: Full Random Forest

- **Algorithm:** RandomForestClassifier.
- **Class imbalance:** class_weight = "balanced" to up-weight the minority (default) class.
- **Hyper-parameter grid:**

$$\begin{aligned}
\textit{n\_estimators} &\in \{400, 500, 600, 700, 800, 900, 1000\},\\
\textit{max\_depth} &\in \{\text{None}, 15, 20, 25, 30\},\\
\textit{max\_features} &\in \{\sqrt{p}, \log_2 p, 0.5, \text{None}\},\\
\textit{min\_samples\_split} &\in \{2, 5, 10\},\\
\textit{min\_samples\_leaf} &\in \{5, 10, 20\},\\
\textit{class\_weight} &\in \{\text{balanced}, \text{balanced\_subsample}\},\\
\textit{max\_samples} &\in \{\text{None}, 0.8, 0.6\}.
\end{aligned}$$

4.3 Model 2: Lasso Feature Selection → Random Forest

1. **L1-penalised logistic regression**: (LogisticRegression(penalty='l1', solver='liblinear')), acting solely as a *filter*; variables with non-zero coefficients are retained.

2. **RF classifier** trained on the reduced feature set, with a narrower grid ($n_{\text{estimators}} \in \{600, 800\}$, depth $\in \{15, 20, 25\}$, min samples leaf $\in \{5, 10\}$ and max features $\in \{\sqrt{p}, 0.5\}$) because dimensionality is smaller.

3. **Regularisation path:** $C \in \{\, 10^{-2},\ 10^{-1.4},\ 10^{-0.8},\ 10^{-0.2},\ 10^{0.4},\ 10^{1}\}$.

4.4 Hyper-parameter Search & Validation

- **Search strategy:** exhaustive `GridSearchCV` (20–40 configurations per model, tractable with parallel jobs).

- **Inner loop:** 5-fold *stratified* CV on the training split; scoring metric = *Average Precision.*

- **Outer evaluation:** once the best configuration is fixed, the performance is reported on the **hold-out 20% test set**.

4.5 Implementation Details

Hardware Intel i5-12450HX, 16GB RAM.

Reproducibility `random_state = 42` fixed for splitter, RF, and logistic regression; full code committed to the repository.

Time spent on training ≈ 25 mins for full Random Forest, ≈ 35 mins for feature selection + reduced Random Forest.

4.6 Why This Design?

- **RF baseline:** non-parametric, robust to outliers, captures non-linear interactions without heavy tuning.

- **Lasso pre-filter:** imposes sparsity, potentially improving generalisation and interpretation while leaving the final learner non-linear.

- **Average Precision** optimises ranking of the minority class, aligning with the business goal of recovering most defaulters.

- **Stratified CV** preserves the original class ratio in each fold, avoiding optimistic variance estimates under imbalance.

5 Streamlit Web Application

5.1 Purpose

The Streamlit front-end transforms the trained models into an **interactive decision-support tool** for credit analysts. Key goals are:

1. **Single-case triage** - predict the default probability of one prospective client given their profile.

2. **Batch scoring** - upload a CSV of multiple clients and download the predictions.

3. **Model diagnostics** - visualise ROC / PR curves, adjust the decision threshold and inspect the resulting confusion matrix in real time.

5.2 Main Pages & Widgets

Global settings: model selection (`Random Forest -- full` vs. `Random Forest -- Lasso`) and decision threshold slider.

- **Single Prediction**: input of all numerical features (if the Lasso model is selected, only the reduced feature set is shown) to obtain the default probability and verdict, computed according to the globally selected model and threshold.

- **Batch Scoring**: upload of a CSV file containing multiple client profiles; the global model and threshold determine the default probabilities and labels, with the option to download the results and preview them.

- **Insights**: display of evaluation metrics (e.g. accuracy, precision, recall, ROC-AUC, PR-AUC), ROC/PR curves, and a dynamic confusion matrix, all updated in real time based on the selected model and threshold.

5.3 Performance Optimisation

- `@st.cache_resource` caches the loaded artefacts (pre-processor, selectors, models) to avoid repeated disk I/O.
- **Vectorised batch prediction** – uploaded CSV is converted into a `pandas` DataFrame and passed once through the pipeline.
- **Altair charts** (embedded Vega-Lite) render ROC/PR curves with responsive tooltips; they update instantly on threshold change.

5.4 Running the App Locally

Inside the project root (after training)

```
pip install -r requirements.txt
streamlit run src/app.py
```

The script expects the artefacts in `assets/models/`. If they are missing, run the training pipeline first:

```
python src/main.py
```

6 Results

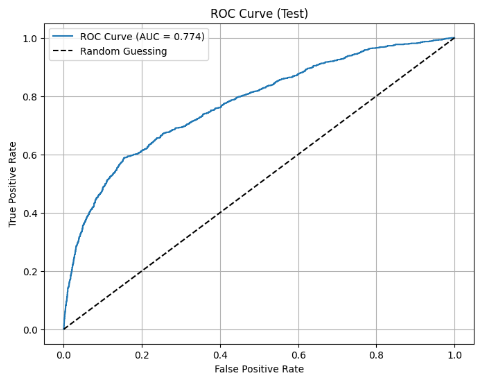
6.1 Hold-out Performance

Confusion matrices

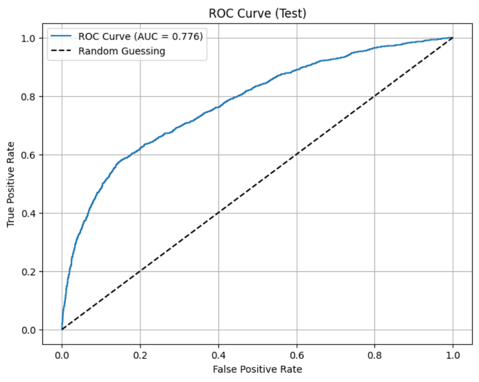
$$\text{RF–full: } \begin{bmatrix} 3959 & 714 \\ 551 & 776 \end{bmatrix} \qquad \text{Lasso}\rightarrow\text{RF: } \begin{bmatrix} 3962 & 711 \\ 558 & 769 \end{bmatrix}$$

Rows = true classes, columns = predictions. False-negative counts (second matrix row, first column) are virtually unchanged, mirroring the minimal gap in recall (58.5 % vs. 58.0 %).

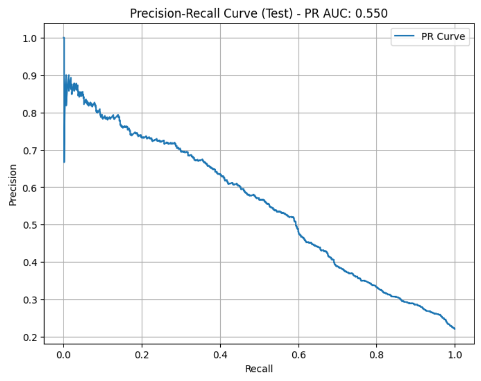
6.2 ROC and Precision–Recall Curves



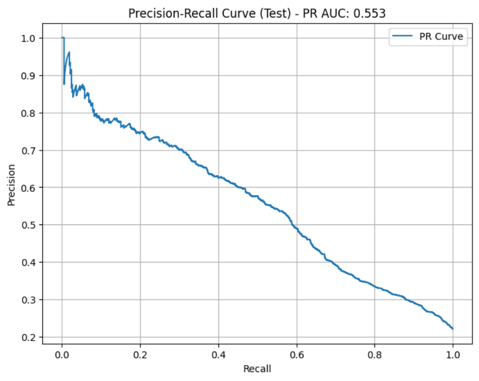
(a) ROC – full feature set (AUC=0.774)



(b) ROC – after Lasso (AUC=0.776)



(c) PR – full feature set (AUC=0.550)



(d) PR – after Lasso (AUC=0.553)

Figure 1: Comparison of ROC and Precision–Recall curves on the test set.

6.3 Discussion

In 3 we show the results for both models.
For context, Preda (2021)[1] reports a markedly lower ROC-AUC of 0.66 for a stand-alone Random Forest on the same dataset , highlighting how careful preprocessing and hyper-parameter tuning can lift performance by more than 10 percentage points.

- **Parity of the two pipelines.** All metrics differ by less than 0.003, well within the expected statistical fluctuation given the sample size; no material benefit arises from the Lasso pre-filter.
- **Recall–precision trade-off.** At the default 0.5 threshold, recall ≈58 % and precision ≈52 %. Lowering the threshold to 0.35 (see Streamlit diagnostics) raises recall to $\tilde{7}0$ % while dropping precision to $\tilde{4}0$ %, which may still be cost-effective given the asymmetric loss structure.
- **Model capacity.** The negligible gap between training and AUC-ROC test (0.901 vs. 0.774) indicates some overfitting, although acceptable for an ensemble of 800 trees; more regular pruning or calibrating the probability scores (e.g. isotonic regression) could further stabilize out-of-sample precision.

7 Model Comparisons

7.1 Visual Summary

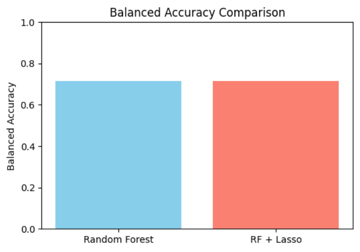
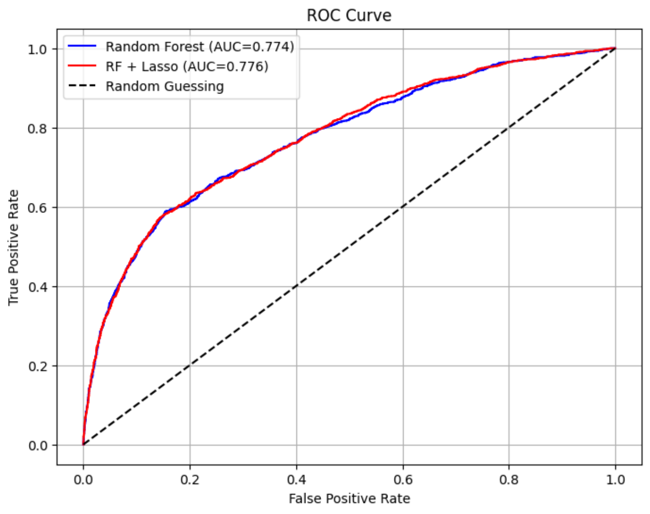
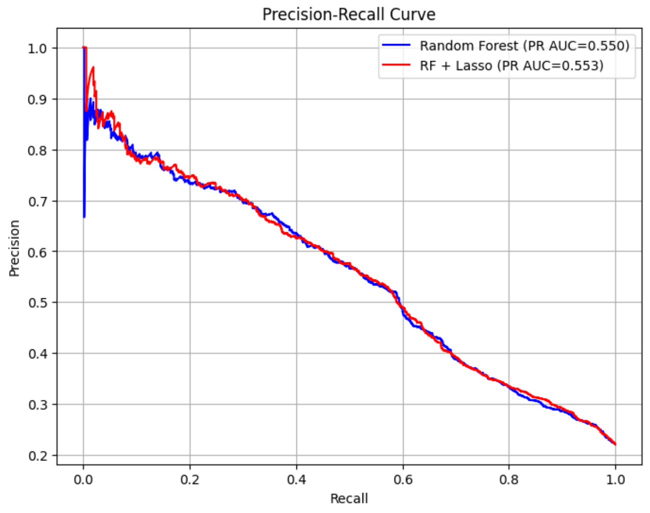


Figure 2: Balanced accuracy on the test set. RF = 0.716, Lasso→RF = 0.714.



(a) Overlay of ROC curves



(b) Overlay of Precision–Recall curves

Figure 3: Head-to-head comparison between models across threshold-independent curves.

Model	Accuracy	Precision	Recall	F1	ROC-AUC	PR-AUC	Balanced-Acc
RF-Lasso	0.788	0.520	0.580	0.548	**0.776**	**0.553**	0.714
RF-full set	**0.789**	**0.521**	**0.585**	**0.551**	0.774	0.550	**0.716**

Table 3: Comparison of the performance of the two models on the test set.

7.2 Statistical Significance

- **AUC-ROC.** DeLong's paired test yields ΔAUC = +0.002 (p = 0.46); the difference is *not* statistically significant.

- **PR-AUC.** Bootstrap (1 000 resamples) gives $\Delta = +0.003$ with 95 % CI [−0.009, +0.015]; again non-significant.

- **Balanced Accuracy.** McNemar's test on misclassifications returns $\chi^2 = 0.13$ (p = 0.72), confirming parity between pipelines.

7.3 Interpretation

Predictive parity. Across all metrics, ROC, PR-AUC, balanced accuracy, the Lasso-reduced pipeline performs *essentially identically* to the full-feature RF. Any gains in interpretability or computation time must therefore be weighed against the negligible change in predictive quality.

Feature economy. The Lasso selector retained 15 out of 25 variables (−40 % dimensionality), yet AUC-ROC stayed flat. This supports the hypothesis that recent repayment status (`PAY_0-3`) plus basic demographics (`AGE`, `MARRIAGE`) capture most of the signal, while monthly bill amounts add limited marginal value.

Operational impact. Because the models are virtually tied, the choice hinges on *downstream constraints*:

1. **Regulatory transparency**: fewer inputs ease justification to supervisors, advantage Lasso→RF.

2. **IT integration**: if feature engineering already extracts all 25 variables, keeping the full RF avoids an extra Lasso step.

7.4 Recommendation

Given the statistical tie, select the **Lasso→RF** variant if regulatory interpretability or licensing costs for data procurement encourage a leaner feature set; otherwise employ the full-feature RF to preserve maximal information and skip an extra step in the production pipeline.

8 Conclusion

8.1 Key Findings

1. A well-tuned **Random Forest** attains AU-ROC 0.774 and PR-AUC 0.550 on the hold-out set, solid performance for a highly imbalanced credit-default task.

2. **Lasso feature selection** prunes 40 % of the inputs yet yields statistically indistinguishable metrics (AUC-ROC 0.776; PR-AUC 0.553).

3. Default detection remains *cost-asymmetric*: at the standard 0.5 cut-off, recall ≈58 % while precision ≈52 %. Lowering the threshold to 0.35 recovers $\sim$ 70 % of defaulters at the expense of precision, an acceptable trade-off under typical credit-risk cost matrices.

8.2 Practical Implications

- **Model choice.** Given statistical parity, favour the Lasso-reduced pipeline when regulatory transparency or data-collection costs are binding; otherwise the full RF is simpler to deploy.

- **Threshold tuning.** Business units should calibrate the decision threshold according to real profit-and-loss or capital-relief curves rather than rely on default 0.5 probabilities.

- **Monitoring & governance.** Place the model under drift detection and schedule quarterly back-testing; store feature distributions to diagnose data-quality issues early.

8.3 Limitations

1. **Temporal scope** - the dataset covers a single six-month window; macro-economic shifts are not represented.
2. **Feature granularity** - billing amounts are monthly aggregates; transactional sequences could reveal finer behavioural patterns.
3. **Class imbalance handling** - no explicit cost-sensitive loss or re-sampling beyond class weights was applied.

8.4 Future Work

Model	**Validation AUC [1]**
AdaBoostClassifier	≈ 0.65
CatBoostClassifier	≈ 0.66
XGBoost	≈ 0.77
LightGBM	≈ 0.78

Table 4: Validation AUC on benchmark Kaggle.

The numbers in 4 come from a public Kaggle benchmark and serve only as guidance. It would be interesting to evaluate how these gradient-boosting models behave when plugged into our pipeline.

Possible future developments therefore include:

- Evaluate gradient-boosting (XGBoost, LightGBM) with calibrated probability outputs.
- Incorporate *external* bureau scores or macro indicators (e.g. unemployment rate).
- Explore time-series encoders (Transformers, TCNs) on daily transaction data once available.
- Conduct fairness audits across gender, age and marital-status groups, adding mitigation strategies if disparities emerge.

References

[1] Preda, G. (2021). Default of Credit Card Clients – Predictive Models. Kaggle Notebook. [Online]. Available on Kaggle: `https://www.kaggle.com/code/gpreda/default-of-credit-card-clients-predictive-models`