# Fraud Detection in Credit Card Transactions Applying Machine Learning, Including One More Comparative Study of Imbalance-Handling Techniques

Eshika Shukla[1], Sanchay Jangra[2], Durgesh Kumar[3], Dr. Bal Krishna Saraswat[4]
[1,2,3]*Dept. of Computer Science and Engineering, Sharda University, Greater Noida, India*
[4]*Assistant Professor, Department of Computer Science and Engineering, Sharda School of Computing Science and Engineering, Sharda University, Greater Noida India*

***Abstract*—Credit card fraud is typically rare but costly. Its rarity also makes it hard to detect - in the thoroughly applied ULB dataset, only 0.172% of transactions are dishonest, in actual application. The model that labels every transaction as genuine is therefore 99.8% correct and useless; this article is comparing three standard classifiers, Logistic Regression, Random Forest and XGBoost, each individual trained with and without SMOTE oversampling, under one fixed evaluation protocol, broadly speaking. In this broader context, precision, recall, F1, ROC-AUC and PR-AUC rather than precision are reported, employ SMOTE only to the training data so that no synthetic samples are reaching the test set, and apply SHAP to explain which characteristics drive the predictions; this goal is one fair, repeatable comparison that is revealing what imbalance handling actually changes, and at what cost in false alerts.**

***Index Terms*—credit card fraud detection, class imbalance, SMOTE, Random Forest, XGBoost, PR-AUC, SHAP, explainable machine learning**

## I. INTRODUCTION

Card payments are now the default way to pay online, and fraudsters follow the money, in real-world scenarios; and thus, banks need to flag fake transactions, within seconds without blocking so many genuine customers that the framework becomes any nuisance, to a significant degree. Rule-based filters are cheap but easy to avoid, so machine learning is thoroughly applied to learn fraud patterns from labelled history [1], [3], [12].

The central hurdle is class imbalance, yet one fix is employed to one model by many papers and report a single number, which is making findings hard to compare, in many respects. Taken together, two families of outcomes are widespread, including change the data (as an illustration, oversample the minority class with SMOTE [4]) or change the check (apply metrics that do not ignore the minority class, such as precision, recall and PR-AUC [8]), on the whole; and thus, very high exactness can (to some degree) be got by a naively trained classifier while missing most fraud [2], [13], because fraud is a tiny fraction of all transactions.

The two kinds of deviation also have very diverse costs. A missed fraud (false negative) costs the bank or the cardholder the transaction amount plus the investigation time, in real-world scenarios. Against this backdrop, a false alarm (false positive) blocks a genuine buy, annoys a customer and costs an analyst's time, in real-world scenarios. Any honest check therefore has to uncover both sides, not a single score [15].

Our aim is a controlled comparison. On rare fraud, the postulate is straightforward, like oversampling should raise recall; it may lower precision, and the benefit may rest on the model, in real-world scenarios; indeed, we test this by crossing three classifiers with two data conditions and reporting all metrics together, in many respects.

Our advancements are:

- A like-for-like comparison of three classifiers, with and without SMOTE, on one public dataset.
- A leakage-safe protocol, with stratified splitting and with scaling and SMOTE fitted on training data only, broadly speaking.

- A SHAP-based explanation of the models, and a report that is pairing ROC-AUC with PR-AUC, precision and recall.
- A fully stipulated setup, so that the trials can be repeated.

The rest of the paper is organised in what follows manner; Section 4 is setting out the trials and assessment, Section 5 tackles the findings and their limits, and Section 6 concludes, in actual application. In light of this dynamic, related work is reviewed by Section 2. Also, the data, models and evaluation protocol are sketched by section 3.

## II. LITERATURE REVIEW

### *2.1 Fraud detection with machine learning*

Early comparative work showed that data mining methods such as logistic regression, support vector machines and random forests can detect card fraud, and that random forests conducted well on imbalanced data [1]. Gradient boosting [19], in the adjustable form of XGBoost [6], is building trees sequentially to correct earlier deviations and is a thoroughly applied strong baseline on tabular data. Broader surveys of data-mining-based fraud detection list the same recurring barriers: slanted classes, unequal costs, scarce labels and fraudsters who are modifying [12]. Breiman's random forest averages many decorrelated trees and is solid too noisy attributes [5].

### *2.2 The class imbalance problem*

He and Garcia reviewed learning from imbalanced data and grouped the resolutions into sampling approaches, cost-sensitive methods and algorithm-level changes [13], in many respects; japkowicz and Stephen found the harm done by imbalance depends on how sophisticated the idea is and on the amount of training data, not on the ratio alone [14], to a significant degree. Viewed through this lens, chawla et al. suggested SMOTE, which creates synthetic minority samples by interpolating between a minority point and its nearest minority neighbours [4]; in fact, Dal Pozzolo et al. inspected undersampling and showed it changes the anticipated probabilities, so scores need recalibration before being applied as true probabilities [2]. Elkan argued when error costs differ, the decision threshold should be set from the costs rather than by rebalancing the data [15]; together, all these findings imply that resampling can, in practice, boost detection but influences the meaning of scores; and thus, that it is not the only tool.

### *2.3 Evaluation of rare-event classifiers*

Fawcett gave a concrete guide to ROC analysis and its pitfalls [16]. From this perspective, davis and Goadrich showed that a curve that is dominating in ROC space also dominates in precision-recall space, but that the reverse is not guaranteed, and that PR curves expose differences that ROC curves are hiding when positives are rare [17], to a significant degree, and an akin conclusion in an empirical study [8] was reached by saito and Rehmsmeier, on the whole. Dal Pozzolo et al. further argued fraud detection should be judged with measures suited to rare events and with attention to how fraud patterns drift over time [3]; the drift problem is, put simply, part of the wider concept drift literature [18].

### *2.4 Explainability*

Lundberg and Lee introduced SHAP, which is allocating to each trait a insight to each prediction applying Shapley values from game theory [7], to a significant degree. That is to say, carcillo et al. combined unsupervised and supervised learning to boost detection, revealing that model design for fraud is still an active field [9], broadly speaking. Explanations are helping analysts review alerts and help the bank justify a decision to a customer in fraud settings, in actual application.

Table A. Summary of related work

| Ref. | Focus | Key point used here |
|---|---|---|
| [1] | Comparative study | Random forests are executing well on imbalanced card data |
| [2] | Undersampling | Resampling shifts probabilities; recalibrate |
| [3] | Realistic check | Employ rare-event metrics; consider drift |
| [4] | SMOTE | Synthetic minority samples by interpolation |
| [8] | PR vs ROC | PR plots are, strictly speaking, more informative when positives are uniformly rare |
| [13] | Imbalance survey | Sampling, cost-sensitive and algorithm-level fixes |
| [15] | Cost-sensitive | Set threshold from error costs |

Table A summarises the studies that shape our design; the typical thread is that no single fix is widely enough, including resampling, metric choice and threshold choice interact, which is largely why our study differs one factor at a time and reports several metrics, in many respects.

*2.5 Why fraud detection is specifically hard*

Card fraud takes several forms, spanning the apply of stolen or counterfeit cards, card-not-present fraud in online shopping, and account takeover, each of which is leaving diverse traces in the data, on the whole. In parallel with this, third, fraudsters are adjusting. Three attributes are, in fact, making the constraint harder than an ordinary classification task; second, labels arrive late, namely a transaction is generally corroborated as fraud only after the cardholder notices and disputes it, so recent data are partially unlabelled [3], in real-world scenarios. First fraud is rare, which is the imbalance problem studied here, and once a pattern is detected and blocked, attackers are changing their behaviour, so the data distribution drifts over time [18]. A model that scores well on historical data may therefore degrade in employ, which is why periodic retraining and monitoring are part of any real framework.

*2.6 Research gap*

Existing work is predominantly strong on individual approaches. Building on this observation, our study is tackling each gap with a fully crossed design, five supplementary metrics and a fully stipulated protocol, broadly speaking; in fact, still, three gaps are remaining. Third, studies often are changing the model and the resampling at once; indeed, still the effect of each is unclear. Second, preprocessing details are often too thin to repeat, and first, in many papers, precision and ROC-AUC are still reported alone, on the whole.

## III. METHODOLOGY

*3.1 Dataset*

The public credit card fraud dataset released by the Machine Learning Group of ULB is applied, in real-world scenarios. It is encompassing transactions made by European cardholders over two days (Table 1). Features V1 to V28 are principal constituents of the original confidential variables; only Time and Amount are in their original form, on the whole. Under these conditions, the label Class is 1 for fraud and 0 otherwise.

The dataset is publicly available at kaggle.com/datasets/mlg-ulb/creditcardfraud.

Table 1. Dataset summary

| Property | Value |
|---|---|
| Transactions | 284,807 |
| Fraudulent | 492 (0.172%) |
| Period | Two days, Sept. 2013 |
| Features | V1-V28 (PCA), Time, Amount |
| Target | Class (1 = fraud) |

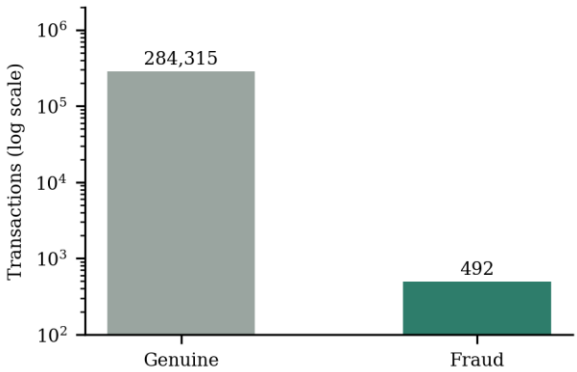



Figure 1. Class distribution of the dataset (log scale). Along these dimensions, fraud is 0.172% of all transactions.

The imbalance is uncovered by figure 1. Within this framework, for every dishonest transaction there are about 578 genuine ones, and a classifier that always is projecting genuine would score about 99.83% accuracy while catching no fraud at all; this is why exactness is not applied to rank models in this examination, in actual application.

*3.2 Feature characteristics*

Understanding the characteristics explains several design choices. From this perspective, time is the number of seconds since the first transaction in the data, so it measures position in the two-day window, not the time of day, in real-world scenarios; amount is the transaction value and is strongly right-skewed, namely most buys are small and a few are very large, to a significant degree. The components V1 to V28 come from a principal component analysis, so they are uncorrelated with each other by construction; this

removes the multicollinearity that can (to a meaningful extent) destabilise Logistic Regression, yet it also means that no individual element directly deciphers.
Little feature engineering is, in this setting, possible, because the elements are already decorrelated and centred, to a significant degree; and thus, model choice and imbalance handling therefore become the primary levers; this is a worthwhile attribute for a controlled comparison, because the same inputs are particularly given to every model, in real-world scenarios.

*3.3 Preprocessing*

The data are split into the training and test sets with a stratified split, so both are keeping the same fraud rate, on the whole. Building on this observation, time and Amount are standardised applying the mean and standard departure of the training set only. Also, the same values are then employed to the test set, so this matters because Amount is heavily skewed and would otherwise dominate the linear model. No rows are removed and copies are kept, because repeated equivalent transactions can themselves be a fraud signal. On comparable scales, the anonymised components V1 to V28 are already.
The test set is left untouched, is never resampled, and is employed only once, after all tuning is complete, so that the reported numbers are an honest estimate of performance on unseen data, in actual application.

*3.4 Imbalance handling with SMOTE*

Under two conditions, each model is consistently trained. In the baseline condition, the training set is used as is. Synthetic fraud samples are added to the training set only in the SMOTE condition, broadly speaking. For a minority sample x, SMOTE picks one of its k nearest minority neighbours x_nn and creates a new point on the line between them [4]:

$$x_new = x + \lambda\,(x_nn - x),\ \ \lambda \sim U(0, 1) \qquad (1)$$

The new point is a reasonable fraud-like sample, not a copy, which lowers the overfitting caused by plain duplication; Figure 2 is revealing the idea on toy two-dimensional data, in real-world scenarios.

Applying SMOTE before splitting would place near-copies of test frauds in the training data and inflate every metric; we therefore employ it strictly after the split, and inside each fold during the cross-validation, employing imbalanced-learn [11], in real-world scenarios; the sampling ratio is chosen so that the training classes are balanced. Along these dimensions, also the same ratio is employed for all three models.

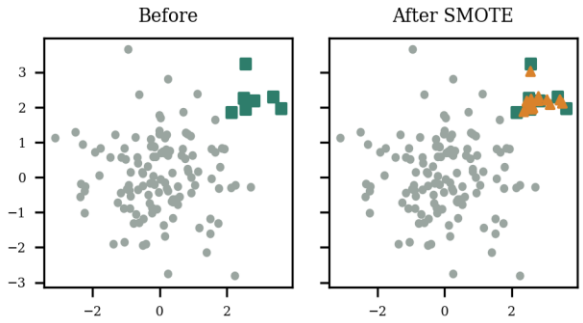



Figure 2. Viewed through this lens, illustrative only; not the real dataset. Illustration of SMOTE on toy data, including synthetic minority points (triangles) are, in practice, placed between existing minority points (squares), to a significant degree.

*3.5 Models*

Three classifiers from scikit-learn [10] and XGBoost [6] are juxtaposed, chosen to cover a linear model, a bagged ensemble and a boosted ensemble, in many respects.

Logistic Regression is an interpretable linear baseline that is modeling the probability of fraud as

$$p(y=1|x) = 1 / (1 + \exp(-(w^{T}x + b))) \qquad (2)$$

and learns w and b by minimising the regularised log-loss, so it is fast, so also, its coefficients are, by design, fairly easy to read, broadly speaking. Nonetheless, it can (in practice) only draw a linear boundary in the feature space.

Many decision trees on bootstrap samples are trained by Random Forest [5], each split analyzing a random subset of characteristics, and averages their votes; it is handling nonlinear patterns and feature interactions, is hard to overfit, and needs little tuning, in real-world scenarios, and its class-weight option is an alternative to resampling. In light of this dynamic, despite this, it is not applied here, so that all models stay on the same footing, in many respects.

XGBoost [6] builds trees one after another, each fitted to the deviations of the ensemble so far by minimising a regularised target:

$$L = \Sigma\, l(y_i, \hat{y}_i) + \Sigma\, \Omega(f_k) \qquad (3)$$

where l is the loss and Ω penalises tree complexity; it is, by definition, commonly among the strongest

tabular learners and builtly opts, scale_pos_weight, to weight the rare class, to a significant degree. Taken together, we leave it at its default so that the effect of SMOTE can (in practical terms) be isolated, to a significant degree.

*3.6 Training and tuning*

Hyperparameters are tuned on the training data only, applying stratified k-fold cross-validation scored by PR-AUC and random search [20] with the same number of trials for every model, in many respects; indeed, stratification is keeping the fraud rate equal in every fold, which is vital when there are only a few hundred positives, to a significant degree. Within this framework, resampling is employed inside each training fold and never to the validation fold in the SMOTE condition; each model is, indeed, then refitted on the full training set and assessed once on the test set after tuning. The hyperparameters searched for each model are listed in Table B. The search ranges are the same for every run of a given model, on the whole.

Table B. Hyperparameters tuned per model

| Model | Hyperparameters searched |
|---|---|
| Logistic Regression | regularisation strength C; penalty |
| Random Forest | number of trees; maximum depth; minimum samples per leaf; attributes per split |
| XGBoost | number of trees; maximum depth; learning rate; subsample; column subsample; L2 regularisation |

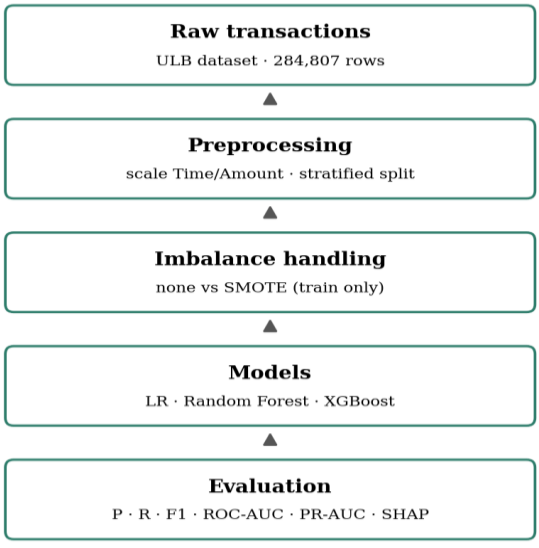



Figure 3. Experimental pipeline.

*3.7 Evaluation metrics*

Exactness is not reported as a headline measure because it is dominated by the genuine class. Let TP, FP, TN and FN be the counts of true positives, false positives, true negatives and false negatives, with fraud as the positive class.

- Precision = TP / (TP + FP): the share of flagged transactions that are truly fraud (false-alarm cost), on the whole.
- Recall = TP / (TP + FN): the share of all fraud that is broadly caught (missed-fraud cost), broadly speaking.
- F1 = 2PR / (P + R): the harmonic mean of precision and recall, in actual application.
- ROC-AUC: the space under the true-positive-rate versus false-positive-rate curve [16].
- PR-AUC: the field under the precision-recall curve, estimated by average precision [8], [17].

ROC-AUC can, strictly speaking, perhaps look excellent even when precision is poor, because the false-positive rate is divided by the very large number of genuine transactions, broadly speaking; and thus, PR-AUC has no such cushion, so it is the primary ranking metric in this inquiry, on the whole.

*3.8 Decision threshold*

Precision, recall and F1 rely on the threshold employed to turn a fraud score into a yes or no decision, to a significant degree; and thus, to keep the comparison fair, a single default threshold of 0.5 is adopted to all models in the central table, in many respects. A threshold-free view (PR curves) is also reported. In light of this dynamic, also, the effect of moving the threshold is tackled in Section 5, because SMOTE shifts anticipated probabilities [2]; in practice, a bank would, in effect, choose the threshold from the relative cost of a missed fraud and a false alarm [15].

*3.9 Explainability*

SHAP values [7] are computed for the best tree model on the test set, to a significant degree. Under these conditions, a SHAP summary plot ranks the attributes by their average affect and reveals whether high or low values push a transaction towards the fraud; this is adopted to check that the model counts on sensible signals and not on an artefact of the data, on the whole.

*3.10 Cost view of the threshold*

for the connection of the metrics to money, consider a straightforward cost model; let c_FN be the cost of a missed fraud and c_FP the cost of a false alarm, broadly speaking; in fact, the total cost of a model at a given threshold is

$$Cost = c_FN \cdot FN + c_FP \cdot FP \quad (4)$$

and, following Elkan [15], if the model outputs well-calibrated probabilities, the cost-minimising threshold is

$$p^* = c_FP / (c_FP + c_FN) \quad (5)$$

The costs applied by real institutions are confidential, so we do not allocate values in this inquiry and employ the cost view only to construe the metrics, in real-world scenarios.

Viewed through this lens, for illustration only, if a missed fraud were 20 times as costly as a false alarm, p* would be 1/21, about 0.048, far below the usual 0.5; this is revealing why a fixed threshold of 0.5 is a convention, not an optimum, and why models trained with SMOTE (whose scores are not aligned [2]) need care, on the whole.

*3.11 Experimental protocol*

The full approach is regularly summarised in Algorithm 1, written so that every choice that could cause leakage is clearly specified, on the whole.

Algorithm 1. Leakage-safe evaluation protocol

| |
|---|
| 1. Split the data into the train and test sets (stratified, fixed seed). |
| 2. Fit the scaler on train; apply to train and test. |
| 3. For each model M in {LR, RF, XGBoost}: |
| 4. For each condition C in {baseline, SMOTE}: |
| 5. Tune M by stratified k-fold CV on train; in the SMOTE condition, resample inside each training fold only. |
| 6. Refit M on the full train set (with SMOTE if C = SMOTE). |
| 7. Score the untouched test set once; record all metrics. |
| 8. Compute SHAP values for the best tree model. |

Stages 2 and 5 are where leakage most often creeps in; fitting the scaler on all data, or resampling before cross-validation, would let information from validation or test transactions influence training and yield optimistic findings that do not hold in practice, in many respects.

*3.12 Computational considerations*

Fraud systems score transactions in real time, so prediction cost matters more than training cost, on the whole, and Logistic Regression scores a transaction with a single dot product and is the cheapest; Random Forest and XGBoost must cross many trees, which is still fast for a few hundred trees but costs more memory, to a significant degree. SMOTE is augmenting the training set size and therefore the training time, but has no effect on prediction time, because it is adopted only during the training.

*3.13 Implementation and reproducibility*

All trials are written in Python applying scikit-learn [10], imbalanced-learn [11], XGBoost [6] and the shap library [7]. Against this backdrop, random seeds, the split, the search space and the final hyperparameters are fixed and reported so that the trials can be repeated on the public dataset. Preprocessing and resampling are wrapped in a single pipeline object, so that the same phases are implemented in the same order in every run. Also, the test data can accidentally be seen by no step.

*3.14 Data use and ethics*

The dataset is public and anonymised by the data provider, so the cardholders can't be spotted from it, broadly speaking. Along these dimensions, no attempt is made to re-identify individuals. The data are applied only for the comparison reported here. The code treats fraud flags as decision support - as in a real bank, a flagged transaction should generate a review or a customer confirmation step, not to an automatic and unexplained account closure, in many respects.

## IV. EXPERIMENTS AND ANALYSIS

*4.1 Overall comparison*

The six setups (three classifiers, each with and without SMOTE) are compared on the same held-out test set using precision, recall, F1, ROC-AUC and PR-AUC; each configuration is scored once on the untouched test set after tuning is complete, broadly speaking; indeed, the questions of interest are whether SMOTE increases recall, by how much precision falls in return, to a significant degree. Building on this observation, also whether that trade-off is the same for the linear and the tree-based models; in fact, PR-AUC is the

main ranking metric, because it depends only on how well fraud is ranked against the genuine majority [8].

*4.2 Error breakdown*

For each configuration, the confusion matrix is reported as counts (TN, FP, FN and TP) next to the rates, broadly speaking. Recall can move noticeably with a change of a few transactions, because the test set includes only few frauds. In light of this dynamic, the counts are therefore needed to judge whether a difference is meaningful. They also are revealing directly how SMOTE changes the number of missed frauds (FN) and false alerts (FP) for each model.

*4.3 Precision-recall behaviour*

Precision-recall curves for all six setups are plotted together, and also, unlike a single threshold, a curve is revealing precision at every level of recall, to a significant degree; still the configuration that stays highest at high recall is the better choice when missing fraud is costly, in actual application. Along these dimensions, the curves also are showing what is commonly possible at thresholds other than 0.5 (Section 3.8).

*4.4 Feature importance*

Because, V1 to V28 are PCA components, the examination can disclose which components; in fact, also, whether Amount or Time, matter most; it can't name business concepts such as merchant type or location, broadly speaking. Building on this observation, sHAP values (Section 3.9) are applied to rank the characteristics by their average effect on the best tree model and to check whether high or low values push a transaction towards the fraud, in many respects.

*4.5 Effect of SMOTE*

Each model is in comparison to its own baseline, so that differences between model families are removed, broadly speaking; also, the effect of SMOTE is isolated. For each model, the change in precision, recall, F1 and PR-AUC is reported when SMOTE is added, in many respects. Viewed through this lens, also read them together, such as a gain in recall that is paid for with a large drop in precision means more alerts for analysts to review, which has an operating cost.

*4.6 Statistical reliability*

The test set contains few positives, so a single split can (in actual operation) rank models misleadingly with only a few hundred frauds in the data, in actual application; indeed, to measure this, the whole protocol is repeated over several random seeds, and the mean and standard deviation of each metric are reported, in many respects. In parallel with this, two setups are treated as different only when the gap between their means is clearly larger than the spread across the seeds; this guards against reporting a difference that is really noise from one lucky split.

## V. DISCUSSION

The study is carefully designed so that differences between arrangements can be credited to the model or to SMOTE, not to the protocol. From this perspective, several points are, in essence, worth keeping in mind when reading the findings.

*Score meaning.* Building on this observation, resampling changes the class balance the model is seeing in training, so its raw output scores no longer match real fraud rates [2]; where scores are principally employed as probabilities, they should be recalibrated, in real-world scenarios.

*Threshold.* The choice of decision threshold matters as much as the choice of model, and a bank would set it from the relative cost of a missed fraud and a blocked genuine customer, which differs by institution [15]. Reporting PR curves uncovers what is normally possible at other thresholds, not only at 0.5, to a significant degree.

*Interpretability.* SHAP explanations are disclosing which of the anonymised components matter. Along these dimensions, alerts could be explained in much more advantageous terms by a bank with access to the raw traits. Still, because V1 to V28 are PCA outputs, they can't be mapped back to business concepts such as merchant type or location.

*Applied rollout.* A real framework must also meet latency limits, handle delayed fraud labels (a transaction is often corroborated as fraud only days later), and be retrained as fraud patterns change [3],

[18]; none of these is encapsulated by a static train-test split, on the whole.

*Privacy and ethics.* Transaction data are personal. Models should be trained on protected data, and access should be restricted; on particular groups of customers, false alerts should also not fall unevenly. The public dataset is anonymised, so these concerns were not tested here; still, they are design requirements for any real rollout, broadly speaking.

*Relation to earlier studies.* Earlier work reports strong findings for tree ensembles on card fraud [1], and for resampling on imbalanced data in broad [4], [13]. Viewed through this lens, our findings should not be contrasted number-for-number with these studies, because datasets, splits, metrics and thresholds differ. Most greatly, what this article adds is a further single protocol in which the model and the imbalance handling are varied one at the time, so that their separate effects can be read from one comparison.

*Operational cost of alerts.* Taken together, a large review team is needed by a model that is raising many alerts, in actual application. If a bank can review only a fixed number of alerts per day, the appropriate question is reliably not the score at a single threshold but how much fraud is reliably caught, within that budget, which is frequently read from the high-precision end of the PR curve, on the whole. Two models with the same F1 can, in this setting, therefore behave very separately once the review capacity is systematically fixed.

*Why elementary models still matter.* Building on this observation, logistic Regression is primarily cheap and easy to audit. Also, regulators and fraud analysts often prefer models whose decisions can be explained directly. An involved model is frequently justified only if its gain in PR-AUC and recall at an acceptable precision is large enough to offset the cost of upholding and explaining it.

*Fairness of comparison.* Giving each individual model the same tuning budget and the same resampling ratio avoids favouring a model that simply received more effort, in many respects; even so, boosted trees have more tunable factors than Logistic Regression, so a larger search could help them more, to a significant degree. Viewed through this lens, we admit this as a shortcoming of any equal-budget comparison.

*Threats to validity.* The dataset is covering only two days and one region, so it can't uncover how fraud patterns drift over months [3], in actual application; its characteristics are anonymised, which is limiting feature engineering. Findings on a single split can, in effect, partially differ with only 492 frauds, so small gaps between models should not be over-read, to a significant degree.

The findings are generally for one dataset and one protocol and need to be repeated on other data before any claim is substantially made about rollout, in many respects. Finally, only SMOTE is tested; other oversampling, under sampling and cost-sensitive schemes may behave separately.

Grounded in the design of the present examination, we draw the following applied guidelines for teams building a comparable framework:

- Split first, then scale and resample, so that no test information leaks into the training, in actual application.
- Report PR-AUC, precision and recall together; never report precision alone.
- Choose the decision threshold from the cost of discrepancies and the alert-review capacity, not by default, in actual application.
- Recalibrate scores after resampling if they are applied as probabilities.
- Repeat the check over several seeds and report the spread.
- Monitor the model, after rollout and retrain as fraud patterns drift

## VI. CONCLUSION

Under a leakage-safe protocol, Logistic Regression, Random Forest and XGBoost, with and without SMOTE, on the ULB credit card fraud data were compared and assessed them with precision, recall, F1, ROC-AUC and PR-AUC together, to a significant degree.

Under these conditions, reporting these measures side by side makes visible the trade-off between catching fraud and raising false alerts, which precision alone hides.

Several directions follow from the present study:

- Time-ordered check, namely train on earlier days and test on later days to measure drift, broadly speaking.
- Other imbalance methods, specifically Borderline-SMOTE, ADASYN, undersampling and cost-sensitive class weights.
- Probability calibration (as an illustration, Platt scaling or isotonic regression) before thresholding.
- Larger and multi-region datasets, with raw attributes that are allowing richer feature engineering.
- Unsupervised and semi-supervised detection [9] for fraud types that are not yet labelled.

It would also be advantageous to study how often one model must be retrained, broadly speaking; and thus, also, how the cost of retraining is comparing with this cost of missed fraud as patterns are changing, in real-world scenarios.

Abbreviations

| Term | Meaning |
|---|---|
| SMOTE | Synthetic Minority Over-sampling Technique |
| ROC-AUC | Field under the receiver operating characteristic curve |
| PR-AUC | Domain under the precision-recall curve |
| SHAP | SHapley Additive exPlanations |
| TP / FP | True positives / false positives |
| TN / FN | True negatives / false negatives |
| PCA | Principal component analysis |
| CV | Cross-validation |

REFERENCES


[1] B. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.

[2] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *Proc. IEEE Symposium Series on Computational Intelligence*, 2015, pp. 159–166.

[3] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3784–3797, 2018.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[5] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 4765–4774.

[8] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, Art. no. e0118432, 2015.

[9] F. Carcillo, Y.-A. Le Borgne, O. Caelen, Y. Kessaci, F. Oble, and G. Bontempi, "Combining unsupervised and supervised learning in credit card fraud detection," Information Sciences, vol. 557, pp. 317–331, 2021.

[10] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[11] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning," J. Mach. Learn. Res., vol. 18, no. 17, pp. 1–5, 2017.

[12] A. Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," arXiv:1009.6119, 2010.

[13] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[14] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," Intelligent Data Analysis, vol. 6, no. 5, pp. 429–449, 2002.

[15] C. Elkan, "The foundations of cost-sensitive learning," in Proc. 17th Int. Joint Conf. Artificial Intelligence, 2001, pp. 973–978.

[16] T. Fawcett, “An introduction to ROC analysis,” Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.
[17] J. Davis and M. Goadrich, “The relationship between precision-recall and ROC curves,” in Proc. 23rd Int. Conf. Machine Learning, 2006, pp. 233–240.
[18] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, 2014.
[19] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
[20] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.