
Implementing Adaptive Machine Learning Frameworks for Financial Fraud Prevention

Ayzed Hugo ^{1*}, Amira Nowak ²

¹ Warsaw University of Technology, POLAND

² University of Nairobi, KENYA

Keywords

Adaptive Machine Learning, Fraud Prevention, Concept Drift, Class Imbalance, SMOTE, Model Retraining, Ablation Study, Mobile Money

ABSTRACT

Adaptive machine-learning frameworks for fraud prevention combine several design choices the learning algorithm, the treatment of class imbalance, the training window, the alert threshold and the policy for updating the model yet these are usually evaluated together, so implementers cannot tell which choices matter. This study implements the temporal-drift framework of Zuberi et al. [1] as the monitoring and retraining core of an adaptive fraud system and evaluates each design choice in a one-factor-at-a-time ablation. A simulated 40-week mobile-money stream of 320,000 transactions contained a mule-network fraud pattern emerging in week 15, a gradual shift in legitimate transfer behaviour and a two-week label delay. Fourteen configurations were compared over five replications using prequential evaluation. Adaptation was the decisive factor: a static model reached an average precision of 0.629, against 0.846 for the drift-triggered reference (gradient boosting, undersampling, four-week window, F2-tuned threshold). Among the remaining choices, the treatment of class imbalance and the length of the training window mattered most: SMOTE raised average precision to 0.876 and an eight-week window to 0.867, both improving on the reference in all five replications, mainly by stabilising models trained on few fraud cases. Combining them gave 0.878 and the highest F2 (0.817) with 3.6 retraining events, compared with 0.859 for weekly retraining with 35 events and ten times the training time. Learner choice had smaller effects, incremental stochastic-gradient learning was cheap but weaker (0.777), and re-tuning the threshold left average precision unchanged while modestly stabilising alert volumes. The results yield an implementation checklist: monitor and retrain on evidence of drift, give the learner enough fraud examples through resampling and window length, and gate every retrained model before deployment.

Introduction

Financial fraud prevention increasingly relies on machine-learning models that score transactions in real time [2], [3]. Because fraud tactics and customer behaviour change continuously, these models must be adaptive: they need to detect when the data have drifted and update themselves accordingly [4]–[6]. An adaptive framework is not a single algorithm but a set of design choices. Implementers must choose a learner, a way of handling the extreme class imbalance of fraud data [7], [8], a training window, a decision threshold and an update policy [9], [10], and each choice interacts with the delay before fraud labels are confirmed [11].

Most published frameworks report the performance of one complete configuration. That shows the framework can work, but not which of its components carry the benefit — information an implementer needs when constraints force compromises. Zuberi et al. [1] took an important step by evaluating 85 combinations of models and oversampling methods under strict chronological splitting and by measuring temporal drift with a battery of statistical tests. They found that no oversampling method dominated across model families and that drift within their 48-hour benchmark was modest, and they called for evaluation over longer horizons [1].

This study implements that framework as the core of a continuously operating adaptive system and asks which implementation choices matter over a 40-week horizon. Its contributions are:

1. an implementation of the drift-measurement and retraining logic of Zuberi et al. [1] — chronological windows, SHAP-selected monitoring, PSI/KS/JSD drift tests, prevalence-shift tests and F2-tuned thresholds — as an adaptive fraud-prevention pipeline;
2. a one-factor-at-a-time ablation of fourteen configurations covering four learners, four treatments of class imbalance, four training windows, two threshold policies and four update policies, each replicated five times;
3. evidence on accuracy, alert volume, alert stability, retraining frequency, training cost and scoring latency, and on a failure mode — unstable retrained models — that the ablation exposed;
4. an implementation checklist derived from the results.

Related Work

Adaptive learning under concept drift

Concept drift is a change over time in the relationship between inputs and outcomes or in the input distribution itself [4], [12]. Adaptive systems respond either by monitoring and retraining when drift is detected [13], [14], by updating continuously on a sliding window, by incremental learning that updates parameters with each new batch [9], [15], or by ensembles that reweight their members as performance changes [10]. In fraud detection, Dal Pozzolo et al. [6], [11] showed on real card data that regular updating is necessary and that label delay shapes which strategies are feasible, and Carcillo et al. [16] demonstrated a streaming architecture for continuous scoring.

Class imbalance and learner choice

Fraud typically accounts for well under 1% of transactions. Common remedies are class weighting, random undersampling of legitimate transactions, and synthetic oversampling with SMOTE or its hybrid with edited-nearest-neighbour cleaning (SMOTEENN) [7], [8], [17]; resampling can, however, distort probability estimates [6], [18]. Among learners, tree ensembles such as random forests and gradient boosting remain strong on tabular data [19]–[21], while logistic regression and neural networks are common alternatives [3]. Feature engineering that aggregates transaction histories is often as important as the learner [22], [23]. Cost-sensitive thresholding links model scores to operational objectives [24].

The framework implemented in this study

Zuberi et al. [1] proposed a robust machine-learning framework for detecting temporal drift in financial fraud prevention. On the European card benchmark it applied a chronological 70/15/15 split with all scaling, feature selection and oversampling fitted on training data only; selected features by SHAP importance [25]; compared classical, neural and ensemble learners with six oversampling methods; tuned decision thresholds to maximise F2; and measured drift between partitions with PSI, KS and JSD tests plus prevalence-shift z-tests. Its best configurations reached an F2 of 0.772 (a multilayer perceptron without oversampling) and an MCC of 0.806 (a soft-voting ensemble) [1]. Table 1 shows how each element is implemented here.

Table 1: Elements of the framework of Zuberi et al. and their implementation in the adaptive pipeline

Framework element [1]	Implementation in this study
Chronological splitting; preprocessing fitted on training data only	Every model is trained only on weeks whose labels have arrived (two-week delay); resampling and scaling are fitted inside each training window

Framework element [1]	Implementation in this study
SHAP-based feature selection	After each retraining, the six features with the largest mean SHAP in a gradient-boosting model are selected for drift monitoring
PSI, KS and JSD drift tests	Weekly comparison of the newest unlabelled week with a two-week reference; alarm when ≥ 2 features show $PSI \geq 0.10$ with $KS\ p < 0.001$, or $PSI \geq 0.25$
Prevalence-shift z-test	Two-proportion z-test of each newly labelled week against the training window; $ z > 3$ triggers retraining
Comparison of oversampling methods	Class weighting, random undersampling, SMOTE and SMOTEENN compared as ablation factors
F2-oriented threshold tuning	Threshold re-tuned on the newest labelled week after each update (or kept fixed, as an ablation)

Materials and Methods

Simulated mobile-money stream

Because long-horizon production data with documented drift are not publicly available, we simulated a mobile-money transaction stream in the tradition of PaySim [26]. Each of 40 weeks contains 8,000 transactions described by ten features: log amount; transfer and cash-out flags; the share of the sender's balance transferred; whether the sender's balance was emptied; whether the destination account is new; the destination's 24-hour in-degree; hour of day; log sender account age; and the sender's 24-hour transaction count. Table 2 summarises the drift schedule. The fraud rate is 0.5% before week 15 and 0.8% afterwards, giving roughly 2,100 fraud cases per replication.

Table 2: Drift schedule of the simulated mobile-money stream

Weeks	Event	Drift type	Description
0–14	Account-takeover fraud only	Baseline	High-value transfers and cash-outs that empty the sender's balance, often at night
15	Mule-network pattern appears	Sudden, real	60% of fraud becomes smaller transfers to new accounts with high in-degree; fraud rate rises from 0.5% to 0.8%
22–30	Instant transfers to new payees spread	Gradual, virtual	Legitimate transfer share rises from 30% to 50%, new-payee share from 5% to 18%, destination in-degree increases

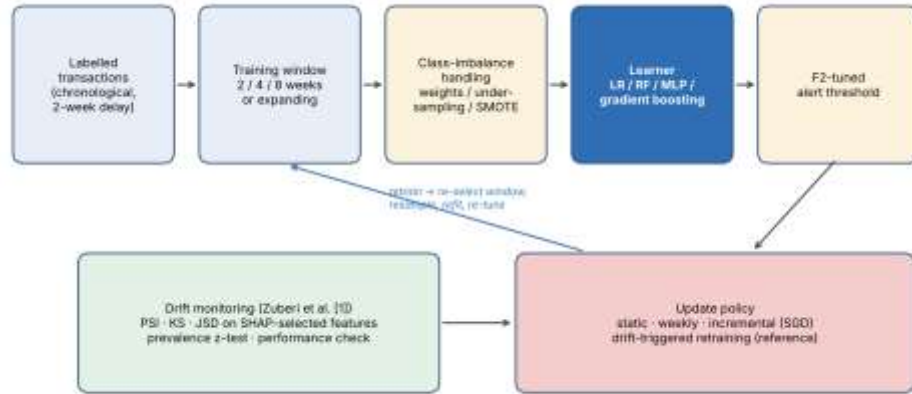
Reference configuration and ablation design

The reference configuration uses histogram-based gradient boosting [20], [27]; random undersampling of legitimate transactions to 10% with compensating weights; a four-week training window; an F2-tuned threshold re-tuned at every update; and drift-triggered retraining. Retraining is triggered when a newly labelled week's average precision falls below the recent mean by $\max(0.06, 3\ SD)$, when the prevalence z-test of the adopted framework gives $|z| > 3$, or when an unsupervised alarm from the PSI/KS/JSD battery is confirmed by a smaller performance drop [1]. Thirteen variants each change one factor (Table 3): learner (logistic regression, random forest, multilayer perceptron), class-imbalance treatment (class weights without resampling, SMOTE, SMOTEENN) [17], training window (two weeks, eight weeks, expanding), threshold (fixed after initial training), and update policy (static, weekly retraining, incremental stochastic-gradient

logistic regression updated weekly) [15]. A fourteenth configuration combines the two best-performing single changes. Figure 1 shows the pipeline and the factors varied.

Figure 1: Adaptive Pipeline and Ablation Factors

Figure 1 Adaptive fraud-prevention pipeline implementing the framework of Zuberi et al. [1], showing the design choices varied in the ablation



Evaluation

Each configuration was evaluated prequentially [28]: every week from week 4 to week 39 is scored by the model in force at that time, before its labels are used. We report average precision [29], recall when the top 1% of transactions are alerted, F2 and alert volume per 1,000 transactions at the deployed threshold, the standard deviation of weekly alert volume, the number of retraining events, total training time and mean scoring latency. Each configuration was run on five independent streams; differences from the reference are paired by stream.

Results

Overall comparison

Table 3: Results of the ablation over weeks 4–39 (mean over five replications; $\pm$ SD for average precision)

Configuration	Average precision	Recall @ 1%	F2	Alerts / 1,000	Retrains	Train time (s)
Reference (HGB, undersampling, 4 wk, re-tuned, drift-triggered)	0.846 $\pm$ 0.047	0.857	0.788	7.4	3.6	2.0
Learner: logistic regression	0.831 $\pm$ 0.011	0.828	0.767	11.0	3.2	0.0
Learner: random forest	0.845 $\pm$ 0.008	0.838	0.772	11.1	3.4	2.1
Learner: multilayer perceptron	0.822 $\pm$ 0.023	0.820	0.754	10.6	3.8	1.1
Imbalance: class weights only	0.822 $\pm$ 0.026	0.837	0.767	8.5	4.0	3.4
Imbalance: SMOTE	0.876 $\pm$ 0.006	0.874	0.807	7.0	3.8	1.4
Imbalance: SMOTEENN	0.862 $\pm$ 0.014	0.865	0.797	8.8	3.8	1.8
Window: 2 weeks	0.843 $\pm$ 0.052	0.848	0.753	5.9	3.6	2.9

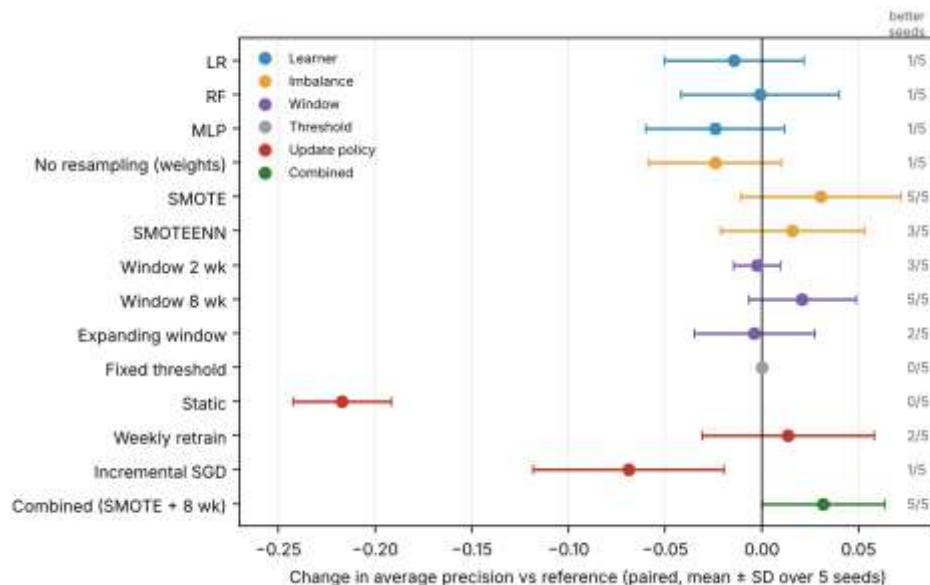
Configuration	Average precision	Recall @ 1%	F2	Alerts / 1,000	Retrains	Train time (s)
Window: 8 weeks	0.867 ± 0.023	0.871	0.806	7.4	3.6	3.6
Window: expanding	0.842 ± 0.030	0.859	0.793	9.7	3.6	3.7
Threshold: fixed	0.846 ± 0.047	0.857	0.789	7.9	3.6	3.1
Update: static	0.629 ± 0.036	0.630	0.567	4.1	0.0	0.5
Update: weekly retraining	0.859 ± 0.017	0.865	0.778	11.7	35.0	21.9
Update: incremental SGD	0.777 ± 0.008	0.828	0.777	10.0	35.0	0.1
Combined: SMOTE + 8 weeks	0.878 ± 0.016	0.876	0.817	7.6	3.6	1.6

Note. F2 and alert volume are measured at the deployed threshold; recall @ 1% is measured with the top 1% of transactions alerted each week. Scoring latency was below 12 μ s per transaction for every configuration.

Table 3 and Figure 2 summarise the ablation. The update policy dominated every other choice. Without updating, average precision fell to 0.629, 0.217 below the reference, and after the mule-network pattern appeared it averaged only 0.49, against 0.80 for the drift-triggered reference. Every other single change moved average precision by at most 0.030.

Figure 2: Effect of Each Design Choice

Figure 2 Paired change in average precision relative to the reference configuration for each ablation (mean $\pm$ SD across five replications); numbers at right give the replications in which the variant beat the reference



Class imbalance and training window

The two factors that consistently improved on the reference both increase the number of fraud examples available to the learner. SMOTE raised average precision to 0.876 ± 0.006 and beat the reference in all five replications, while also reducing its variability across replications (SD 0.006 against 0.047); SMOTEENN gave a smaller gain (0.862, better in three of five), and class weights without resampling were worse (0.822). This pattern agrees with Zuberi et al. [1], who found that the benefit of oversampling depends on the learner, with the largest gain for a gradient-boosting model combined with SMOTEENN. Extending the training

window from four to eight weeks also improved average precision in all five replications (0.867 ± 0.023); a two-week window was less stable and gave the lowest F2 among adaptive configurations (0.753), and an expanding window — which keeps every past week — raised alert volume to 9.7 per 1,000 transactions as outdated patterns accumulated.

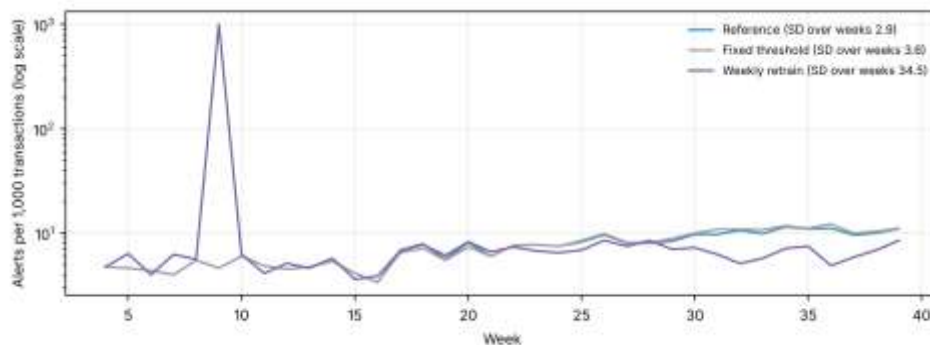
Combining SMOTE with an eight-week window gave an average precision of 0.878 ± 0.016 and the highest F2 of all configurations (0.817), improving on the reference in all five replications. The two gains were not additive — the combination was close to SMOTE alone — which suggests that both act through the same mechanism.

An instability that the ablation exposed

The variability of the reference configuration (SD 0.047) pointed to that mechanism. A four-week window contains only about 160–260 fraud cases, and gradient boosting fitted to so few positives was occasionally unstable. In a diagnostic experiment that refitted the reference model 40 times on the same four weeks of one replication with different undersampling draws, four refits produced models with an average precision below 0.30 on the following week, and one was no better than random. In the weekly-retraining configuration such a model was deployed once without any check: for one week it assigned near-identical scores to all transactions, its average precision fell to 0.004, and every transaction exceeded the alert threshold (Figure 3). Synthetic minority examples and longer windows both reduce this instability, but the more general lesson is procedural: every retrained fraud model should pass an automated validation gate before deployment [30], [31].

Figure 3: Alert Volume and an Unvalidated Retrain

Figure 3 Weekly alert volume at the deployed threshold for the reference, fixed-threshold and weekly-retraining configurations in the replication with the largest volatility; the spike in week 9 is an unvalidated weekly retrain that flagged every transaction



Learner, threshold and update policy

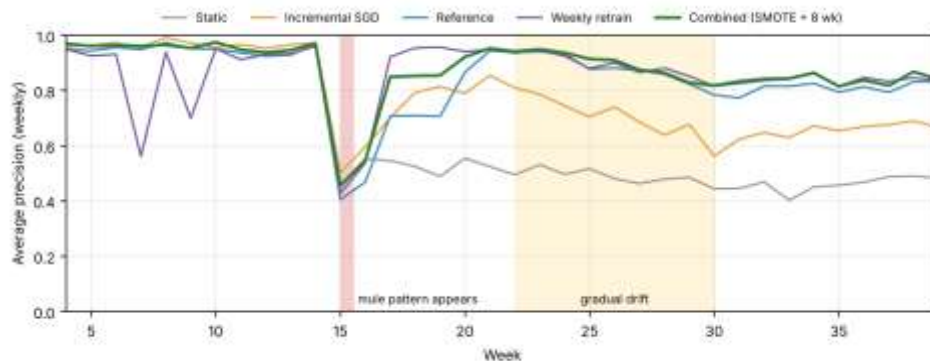
Learner choice mattered less than expected. Random forest matched gradient boosting in average precision (0.845 vs 0.846) with lower variability, while logistic regression (0.831) and the multilayer perceptron (0.822) were somewhat weaker; consistent with evidence on tabular data [21], tree ensembles were the safer choice. All three alternatives, however, produced higher alert volumes at their F2 thresholds (about 11 per 1,000 against 7.4) with lower precision, an operational cost that average precision alone does not reveal.

Re-tuning the threshold at each update cannot change average precision, which depends only on the ranking of scores, and in this stream it also left F2 almost unchanged (0.789 fixed vs 0.788 re-tuned). It did, however, reduce the week-to-week variability of alert volume (SD 3.6 vs 2.9 alerts per 1,000). Weekly retraining

reached an average precision of 0.859 ± 0.017 — better than the reference in only two of five replications — at the cost of 35 retraining events and 22 seconds of training time, about ten times that of the drift-triggered configurations. Incremental stochastic-gradient learning was by far the cheapest to update but adapted poorly to the new fraud pattern (average precision 0.777; Figure 4).

Figure 4: Weekly Detection Performance

Figure 4 Weekly average precision of selected configurations (mean of five replications); shading marks the appearance of the mule-network pattern (red) and the gradual behavioural drift (yellow)

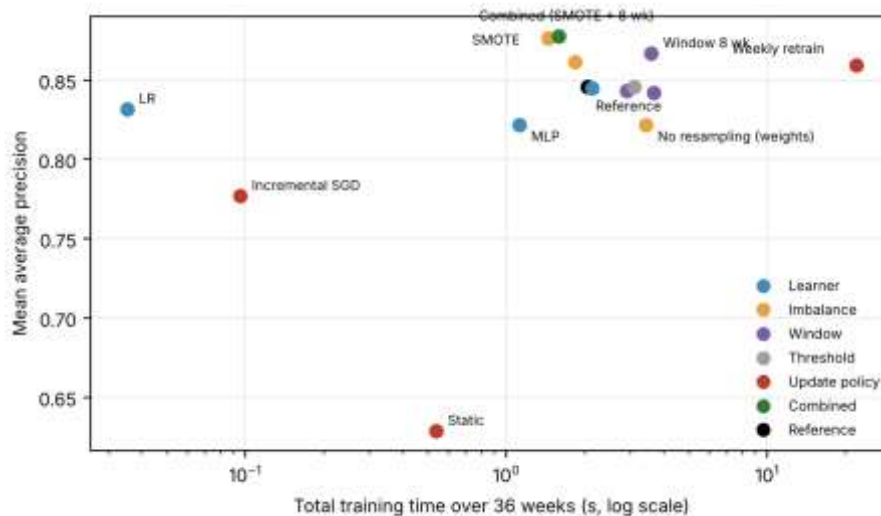


Accuracy and cost

Figure 5 relates average precision to total training time. The drift-triggered configurations with SMOTE or an eight-week window occupy the efficient frontier: they are as accurate as, or more accurate than, weekly retraining while training for one to four seconds in total over 36 weeks instead of 22. Scoring latency was below 12 μ s per transaction for every configuration, so none of the choices affected real-time feasibility.

Figure 5: Accuracy Against Training Cost

Figure 5 Mean average precision against total training time for all configurations (colours indicate the factor varied)



Implementation Checklist

The results translate into an ordered checklist for implementing an adaptive fraud framework of the kind proposed by Zuberi et al. [1] (Table 4). The order reflects the size of each effect in this study.

Table 4: Implementation checklist derived from the ablation

Priority	Design choice	Recommendation	Evidence (this study)
1	Update policy	Monitor drift and retrain on evidence (performance drop, prevalence shift, confirmed feature drift) rather than never or on every cycle	Static -0.217 AP; weekly retraining $10\times$ training time for no consistent gain
2	Fraud examples per model	Use SMOTE-type oversampling and/or an 8-week window so each model sees several hundred fraud cases	SMOTE and 8-week window each better in 5/5 replications
3	Validation gate	Block deployment of any retrained model whose validation performance falls below the current model	One unvalidated weekly retrain flagged every transaction
4	Learner	Prefer tree ensembles; check alert volume, not only ranking metrics	RF $\approx$ HGB; LR and MLP raised alert volume by $\sim 50\%$
5	Threshold	Re-tune the F2 threshold at each update to keep alert volume stable	Lower week-to-week alert variability
6	Window type	Avoid expanding windows once drift has occurred	Highest alert volume of the window variants

Discussion

The ablation shows that, once a fraud system adapts at all, the most important implementation question is how many fraud examples each retrained model sees. Resampling and window length both address this, and both reduce an instability of boosted models fitted to small minority samples that would otherwise surface as occasional broken models. This complements the finding of Zuberi et al. [1] that oversampling effects depend on the learner, by showing that over a long horizon the benefit is felt chiefly as stability. It also explains why weekly retraining, despite its apparent responsiveness, did not consistently beat drift-triggered retraining: each weekly model is fitted to a short window and inherits that instability, and frequent retraining multiplies the chance of deploying a poor model.

For implementers the cost dimension is equally important. Drift-triggered retraining with the monitoring logic of the adopted framework needed three to four retraining events in 36 weeks. In regulated institutions each retraining implies validation, documentation and approval under model-risk guidance [31], and engineering experience shows that every automated feedback loop adds operational risk [32]. A framework that retrains rarely but on evidence, and validates every candidate before deployment, therefore offers both better detection and lower operational burden.

Limitations

The study uses a simulated stream, which makes drift controllable and known but cannot reproduce every feature of real payment data [26]; absolute performance values should not be read as production estimates. The ablation varies one factor at a time and tests only one combination, so interactions between factors are only partly explored. Hyperparameters were fixed rather than tuned at each retraining, the label delay was constant, and only the oversampling methods most relevant to the adopted framework were included.

Future Work

Validation on production data with known fraud campaigns is the most important next step. Factorial rather than one-factor designs would quantify interactions between resampling, window length and learner. Automated validation gates — for example, champion–challenger comparison on held-out recent data — deserve systematic evaluation, as does the extension of the adopted drift battery to monitor the alerted, high-score population where fraud is concentrated.

Conclusion

Implementing the temporal-drift framework of Zuberi et al. [1] as the core of an adaptive fraud-prevention system shows which design choices matter. Adapting to drift is essential; after that, giving each retrained model enough fraud examples — through SMOTE oversampling or a longer training window — produced the most reliable gains and removed a hidden instability, raising average precision from 0.846 to 0.878 with fewer than four retraining events. Learner choice and threshold re-tuning mattered less for ranking performance but shaped alert volumes. Retraining on evidence, with enough fraud examples per model and a validation gate before deployment, is the practical core of an adaptive fraud framework.

References

- [1] Zuberi, N., Rahman, M. M., Rashid, S. U., Ahsan, M. N., Rahman, M. S., & Mollah, M. H. O. R. (2026). A robust machine learning framework for detecting temporal drift in financial fraud prevention. *Scientific Reports*, 16, Article 27776.

- [2] Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50(3), 559–569.
- [3] West, J., & Bhattacharya, M. (2016). Intelligent financial fraud detection: A comprehensive review. *Computers & Security*, 57, 47–66.
- [4] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), Article 44.
- [5] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12), 2346–2363.
- [6] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8), 3784–3797.
- [7] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.
- [8] Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1), 20–29.
- [9] Losing, V., Hammer, B., & Wersing, H. (2018). Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275, 1261–1274.
- [10] Kolter, J. Z., & Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8, 2755–2790.
- [11] Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., & Bontempi, G. (2014). Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, 41(10), 4915–4928.
- [12] Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1), 69–101.
- [13] Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence – SBIA 2004* (Lecture Notes in Computer Science, Vol. 3171, pp. 286–295). Springer.
- [14] Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining* (pp. 443–448). Society for Industrial and Applied Mathematics.
- [15] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT 2010* (pp. 177–186). Physica-Verlag.
- [16] Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). SCARFF: A scalable framework for streaming credit card fraud detection with Spark. *Information Fusion*, 41, 182–194.

- [17] Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17), 1–5.
- [18] Van Calster, B., McLernon, D. J., van Smeden, M., Wynants, L., & Steyerberg, E. W. (2019). Calibration: The Achilles heel of predictive analytics. *BMC Medicine*, 17, Article 230.
- [19] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- [20] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 3146–3154.
- [21] Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35, 507–520.
- [22] Whitrow, C., Hand, D. J., Juszczak, P., Weston, D., & Adams, N. M. (2009). Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1), 30–55.
- [23] Correa Bahnsen, A., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51, 134–142.
- [24] Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (pp. 973–978). Morgan Kaufmann.
- [25] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 4765–4774.
- [26] Lopez-Rojas, E. A., Elmir, A., & Axelsson, S. (2016). PaySim: A financial mobile money simulator for fraud detection. In *Proceedings of the 28th European Modeling and Simulation Symposium* (pp. 249–255).
- [27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- [28] Gama, J., Sebastião, R., & Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine Learning*, 90(3), 317–346.
- [29] Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3), Article e0118432.
- [30] Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017). The ML test score: A rubric for ML production readiness and technical debt reduction. In *Proceedings of the 2017 IEEE International Conference on Big Data* (pp. 1123–1132). IEEE.
- [31] Board of Governors of the Federal Reserve System & Office of the Comptroller of the Currency. (2011). *Supervisory guidance on model risk management* (SR Letter 11-7).
- [32] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, 28, 2503–2511.