Benchmarking SHAP Explanation Strategies for Real-Time Transaction Fraud Detection Under Latency Constraints

Ahmad Ameen[1,*], Halleluyah Aworinde[1]
[1] Department of Information Technology, School of Computing, Miva Open University, Abuja, Nigeria
* Corresponding author: a.ameen0047@miva.edu.ng

Abstract

Machine learning models, particularly gradient-boosted trees, dominate real-time transaction fraud detection but provide no rationale for individual decisions, a limitation that regulators increasingly refuse to accept. SHAP (SHapley Additive exPlanations) offers theoretically grounded feature attributions, yet its computational cost is widely assumed to be incompatible with the sub-100 millisecond latency budgets of payment authorisation systems, and this assumption has not been tested empirically in a production-equivalent pipeline. We present TxWatch, a real-time fraud detection system built on LightGBM, ONNX Runtime, and FastAPI, and use it to benchmark seven SHAP explanation configurations spanning five strategy families: exact TreeSHAP, reduced-background TreeSHAP (50, 100, and 200 background samples), feature-sampled SHAP, KernelSHAP, and asynchronous post-hoc SHAP. Each configuration is evaluated on the IEEE-CIS Fraud Detection dataset for latency at the 50th, 95th, and 99th percentiles and for fidelity measured as mean absolute error against exact TreeSHAP values. We find that exact TreeSHAP is viable within a 100 ms budget on a 39-feature production model (p99 of 85.9 ms), that reduced-background variants offer a monotonic and tunable latency-fidelity tradeoff (an 18x speedup at a mean absolute error of 0.017), and that feature sampling is counterproductive, adding latency without reducing computation. We distil these results into a practitioner decision framework mapping latency budgets and explanation use cases to strategy selection, with specific guidance for the Nigerian financial services environment.

Keywords: explainable AI, SHAP, fraud detection, real-time inference, latency, LightGBM, FinTech

1. Introduction

Digital payment fraud has grown alongside the expansion of digital financial services. Global payment card fraud losses reached $33.83 billion in 2023 (Nilson Report, 2025), and in Nigeria, financial fraud incidents increased by 186% between 2019 and 2020, driven predominantly by electronic payment channels (NIBSS, 2021). Gradient-boosted tree models such as XGBoost (Chen & Guestrin, 2016) and LightGBM (Ke et al., 2017) have become the dominant approach to automated fraud detection, offering strong performance on imbalanced, high-dimensional tabular data at inference latencies compatible with real-time transaction processing.

These models, however, are opaque. They produce risk scores without interpretable rationales, which creates operational problems for analysts, communication problems for customers, and compliance problems for institutions. Regulatory pressure on this front is mounting. GDPR Recital 71 indicates that safeguards for automated decision-making, including fraud screening, should include the ability of

affected individuals to obtain an explanation of the decision reached (European Parliament, 2016). The EU AI Act classifies AI systems for creditworthiness assessment as high-risk while explicitly exempting systems used purely for detecting financial fraud (Annex III, point 5(b)); the exemption does not remove the pressure for explainability, since fraud decisions that restrict access to financial services remain subject to GDPR transparency obligations and to growing supervisory expectations (European Parliament & Council of the EU, 2024). In Nigeria, the Central Bank's open banking guidelines emphasise accountability and transparency in automated financial technology deployments (CBN, 2023).

SHAP (Lundberg & Lee, 2017) is the most widely adopted principled method for explaining individual predictions, and its TreeSHAP variant (Lundberg et al., 2020) computes exact Shapley values for tree ensembles in polynomial time. Yet SHAP computation adds measurable latency to the inference path, and payment authorisation systems typically operate within end-to-end budgets of 100 milliseconds (Visa, 2022). Several approximation strategies trade explanation fidelity for speed, but there is no published empirical characterisation of these tradeoffs in a production-equivalent real-time pipeline. Practitioners choosing between strategies currently do so on intuition.

This paper addresses that gap. Our contributions are:

1. **TxWatch**, an open-source, production-equivalent real-time fraud detection pipeline (LightGBM, ONNX Runtime, FastAPI, Redis, Kafka, PostgreSQL) with five interchangeable SHAP explanation strategies.
2. To our knowledge, the first controlled empirical benchmark of SHAP approximation strategies under real-time latency constraints, evaluating seven configurations on both tail latency (p50, p95, p99) and fidelity (mean absolute error against exact TreeSHAP).
3. Three actionable empirical findings: exact TreeSHAP is viable within a 100 ms budget when the feature set is production-constrained; reduced-background TreeSHAP provides a predictable, tunable latency-fidelity lever; and feature sampling is counterproductive as a latency optimisation.
4. A practitioner decision framework mapping operational requirements to strategy selection, framed with specific reference to the Nigerian FinTech environment of USSD transactions and thin user histories.

2. Related Work

Explainability in fraud detection research. Academic fraud detection work has established the strong baseline performance of gradient-boosted trees on benchmark datasets (Dal Pozzolo et al., 2015; Awoyemi et al., 2017; Ileberi et al., 2022), but treats fraud detection primarily as a predictive problem. More recent studies have applied XAI methods to fraud classifiers: Kurshan et al. (2020) examined graph-based approaches to financial crime detection and the practical constraints of deploying them, and Obeng et al. (2025) compared LIME (Ribeiro et al., 2016) and SHAP on credit fraud classifiers, finding SHAP more stable across similar transactions. Critically, these evaluations were conducted offline in batch mode. None measured the latency overhead of explanation computation in a real-time inference pipeline, leaving explanation deployability uncharacterised.

Commercial platforms. The dominant commercial systems treat explainability as absent, heuristic, or decoupled. FICO Falcon remains a black-box scoring engine with explainability handled by separate tooling (FICO, 2022). Stripe Radar surfaces curated rule descriptions rather than principled attributions (Stripe, 2023). AWS Fraud Detector provides ranked prediction reasons that lack Shapley-value grounding (Amazon Web Services, 2023). Feedzai orients its explanations toward asynchronous analyst review and does not document their latency characteristics (Feedzai, 2022). No platform publishes empirical fidelity-latency tradeoffs.

SHAP and its approximations. SHAP unifies additive attribution methods under cooperative game theory (Shapley, 1953; Lundberg & Lee, 2017), guaranteeing local accuracy, consistency, and missingness. TreeSHAP (Lundberg et al., 2020) reduces exact computation to polynomial time for tree ensembles. Approximations include reducing the background reference dataset, restricting returned attributions to the top-k features, and the model-agnostic KernelSHAP, which fits a weighted linear model to perturbed samples. The theoretical properties of these variants are documented, but their empirical behaviour under production latency constraints is not.

The Nigerian context. Existing fraud detection research reflects Western market data richness. The Nigerian environment differs materially: a large share of transactions flow through USSD channels carrying minimal metadata (GSMA, 2023), and many users have thin transaction histories (Demirgüç-Kunt et al., 2022), weakening behavioural features and altering the information value of explanations. Prior Nigerian studies (Adewumi & Akinyelu, 2017; Amaefule et al., 2022) address prediction without explainability or deployment constraints.

3. System Design

3.1 Architecture

TxWatch comprises three Python/FastAPI services and three infrastructure components, deliberately separated into a synchronous hot path and an asynchronous audit path (Figure 1). The **Feature Service** is the single entry point: it assembles the model's 39-feature vector from the raw transaction payload, records per-user velocity data in Redis sorted sets, forwards the vector to the Inference Sidecar over asynchronous HTTP, returns the decision to the client, and publishes the decision event to Kafka via a non-blocking buffered producer. The **Inference Sidecar** loads the ONNX model, the LightGBM pickle (required for TreeSHAP's access to tree structure), and the background dataset once at startup; per request it runs ONNX inference, applies three-tier decision thresholds (BLOCK at 0.90 and above, FLAG from 0.50 to 0.90, APPROVE below 0.50), and computes SHAP values via the configured strategy. The **Audit Service** consumes decision events from Kafka and persists complete records, including SHAP attributions and per-component latencies, to PostgreSQL with idempotent inserts.

Two design decisions matter for the benchmark's validity. Inference and explanation are co-located in one service, avoiding a network hop that would confound attribution of overhead to the explanation strategy. And audit persistence is decoupled via Kafka, so measured hot-path latency reflects only decision and explanation computation, not database I/O.

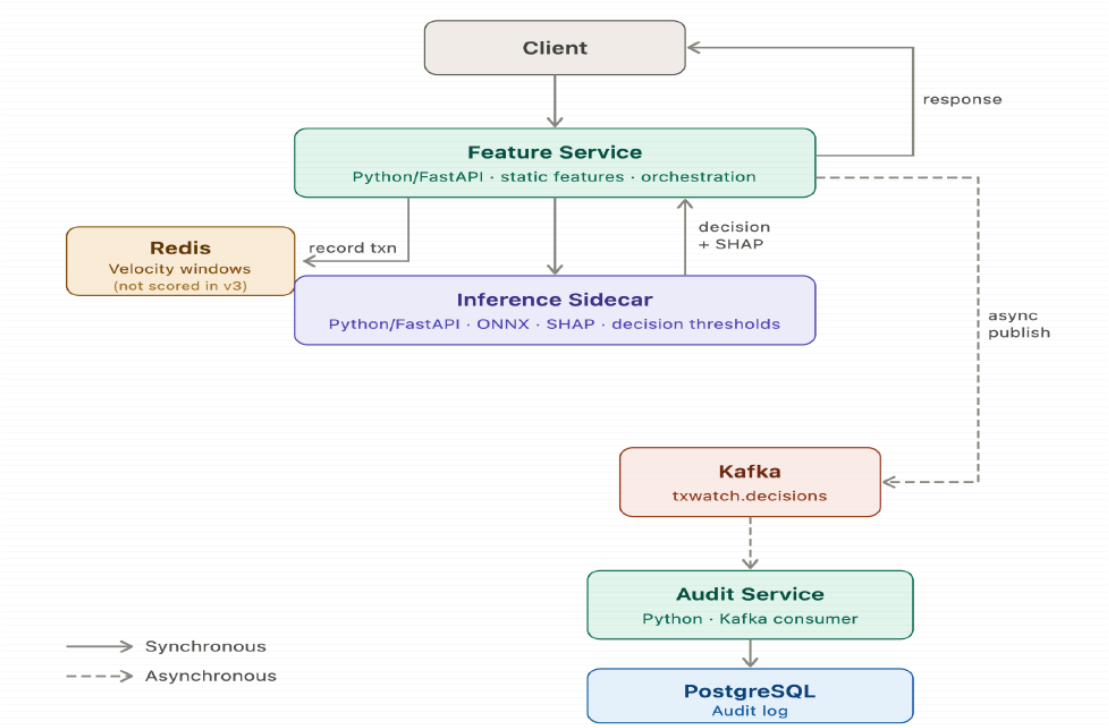

Figure 1. TxWatch system architecture. Solid arrows denote the synchronous hot path; dashed arrows denote the asynchronous audit path.

3.2 Model Configurations

Three LightGBM configurations were trained on the IEEE-CIS dataset with identical hyperparameters (500 estimators, learning rate 0.05, 31 leaves, positive class weight of approximately 27.6 to counter the 3.5% fraud rate). Table 1 summarises them.

Table 1. Model configurations.

Configuration	Features	Purpose	Deployable	Threshold
v1	18	Curated baseline (depends on Vesta features)	No	0.83
v2	432	Offline research ceiling	No	0.83
v3	39	Production-realistic; benchmark model	Yes	0.90

The v3 model uses only features computable from a live request payload: 15 raw transaction, card, address, and email fields; nine match flags (M1 to M9); and 15 recency timedeltas (D1 to D15). Its 0.90 operating threshold was chosen above the F1-optimal 0.83 to prioritise precision (0.71 versus 0.54), routing borderline scores to a FLAG tier for review, consistent with production practice where false blocks carry high cost.

3.3 SHAP Strategy Implementations

Five strategies implement a common interface and are selected at service startup. **Exact TreeSHAP** computes interventional Shapley values over a 1,000-sample background dataset and serves as the fidelity

ground truth. **Reduced-background TreeSHAP** runs the identical algorithm over 50, 100, or 200 background samples. **Feature-sampled SHAP** computes full TreeSHAP internally but zeroes attributions outside the top-k globally important features. **KernelSHAP** is the model-agnostic weighted linear approximation. **Asynchronous post-hoc SHAP** returns zero attributions on the hot path and dispatches exact TreeSHAP as a background task after the response is sent.

4. Experimental Setup

The benchmark follows a controlled experimental design: the explanation strategy is the sole independent variable, with the model (v3), background dataset, transaction sample, and hardware held constant. Dependent variables are per-transaction latency and fidelity, the latter measured as mean absolute error (MAE) against exact TreeSHAP attributions on the same transactions. Latency is reported at the 50th, 95th, and 99th percentiles rather than as a mean, since tail latency governs user-facing experience and service-level compliance in low-latency systems (Dean & Barroso, 2013).

Seven configurations were evaluated on 100 transactions drawn from the held-out test split (118,108 rows, 3.5% fraud), processed sequentially; KernelSHAP was evaluated on 20 transactions owing to its per-request cost. The benchmark harness runs in-process, loading each strategy through a common factory and timing all seven configurations in a single execution, which guarantees identical model, background data, and machine state across configurations. All experiments ran on a MacBook Pro (Apple M-series, 16 GB RAM), with the Redis, Kafka, and PostgreSQL infrastructure containerised via Docker Compose and the Python services running as independent processes, eliminating inter-node network variability. Reported latency is the total in-process scoring time (ONNX inference plus SHAP computation); isolated SHAP timings differ from these totals by less than one millisecond.

5. Results

5.1 Model Performance

Table 2. Model performance on the 118,108-transaction held-out test set.

Configuration	Features	Threshold	ROC-AUC	Precision	Recall	F1
v1	18	0.83	0.8963	0.6323	0.4486	0.5248
v2	432	0.83	0.9479	0.7239	0.5981	0.6550
v3	39	0.90	0.9284	0.7072	0.3465	0.4651

The v2 figures represent an offline ceiling unreachable in production, since its Vesta-engineered features cannot be computed from a live payload. v3 outperforms v1 on ROC-AUC and precision using a strictly deployable feature set; its lower recall reflects the deliberate precision-first threshold, with borderline cases routed to analyst review rather than hard-blocked.

5.2 SHAP Benchmark

Table 3. SHAP strategy benchmark on the v3 model (39 features). Latencies are total in-process scoring time.

Strategy	n	p50 (ms)	p95 (ms)	p99 (ms)	MAE vs exact
exact_tree	100	67.6	82.5	85.9	0.0 (baseline)
reduced_tree_50	100	3.79	5.0	6.6	0.0167
reduced_tree_100	100	8.03	10.1	10.3	0.0100
reduced_tree_200	100	15.31	18.7	20.0	0.0089
feature_sampled	100	68.96	112.0	138.7	0.0150
async_posthoc	100	0.008	0.017	0.020	N/A
kernel	20	17.33	41.7	52.4	0.0940

Figure 2 plots each configuration in fidelity-latency space. On a warmed request with exact TreeSHAP active, full pipeline latency from transaction receipt at the Feature Service to decision response was 95.6 ms (feature extraction 3.1 ms, ONNX inference 0.9 ms, SHAP computation 82.0 ms, inter-service HTTP approximately 9 ms), within the 100 ms budget.

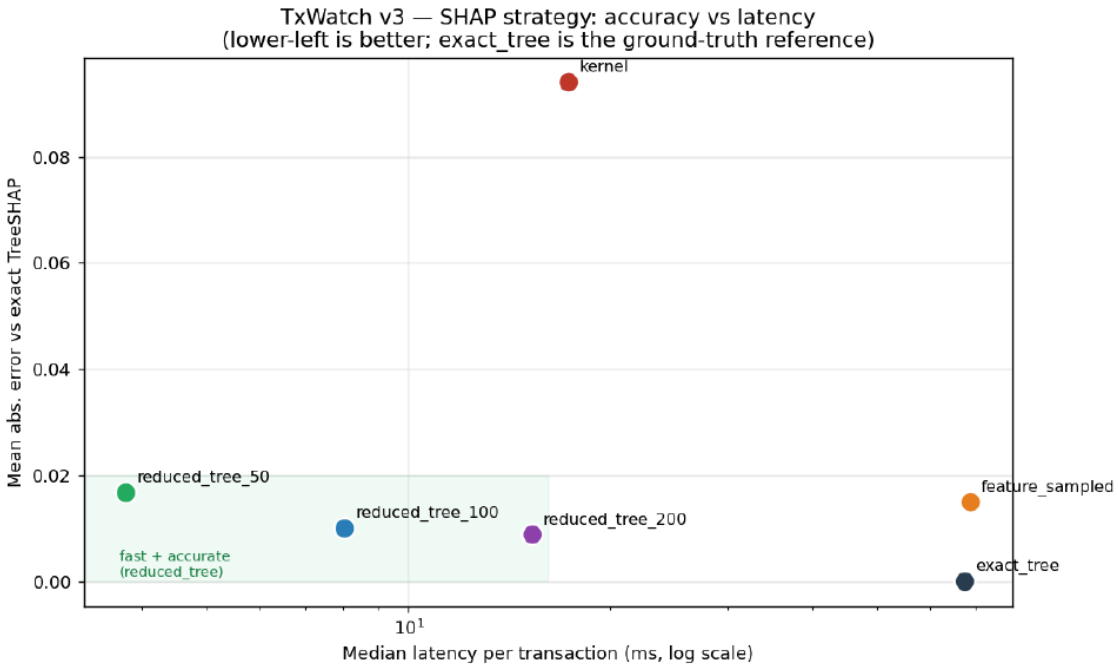

Figure 2. Fidelity versus median latency for the seven configurations (log scale). Lower-left is better; exact_tree is the ground-truth reference.

5.3 Findings

Exact TreeSHAP is viable within a 100 ms budget. At a p99 of 85.9 ms on the 39-feature model, exact, theoretically grounded explanations are compatible with real-time requirements when feature dimensionality is production-constrained. This contradicts the common assumption that exact SHAP must be excluded from the hot path.

Reduced background offers a monotonic, tunable tradeoff. reduced_tree_50 delivers an 18x p50 speedup (67.6 ms to 3.8 ms) at an MAE of 0.0167; reduced_tree_200 improves fidelity to 0.0089 at 15.3 ms. More background samples predictably buy fidelity with latency, giving operators a clean tuning lever. reduced_tree_100 (8 ms, MAE 0.010) is a reasonable default where exact values are not required.

Feature sampling is counterproductive. It is the worst synchronous strategy on both axes, breaching the budget at p99 (138.7 ms) with a higher MAE than reduced_tree_200, which is over six times faster. The mechanism is clear: it computes full TreeSHAP internally and merely discards information, reducing nothing.

KernelSHAP pays a heavy fidelity cost. Its MAE of 0.094 is an order of magnitude above the TreeSHAP variants, justifiable only when tree structure is inaccessible to the explainer.

Asynchronous post-hoc SHAP eliminates hot-path overhead entirely (p99 of 0.020 ms), at the categorical cost of unavailability at decision time. It is the only strategy whose latency guarantee is independent of model complexity.

6. Discussion

6.1 Practitioner Decision Framework

Table 4 distils the benchmark into deployment guidance, mapping the latency budget and the required availability of explanations to a recommended strategy.

Table 4. SHAP strategy selection framework.

Use case	Latency budget	Availability needed	Recommended strategy
Real-time customer communication	Under 100 ms	At decision time	exact_tree (up to ~39 features)
Analyst review dashboard	Flexible	Seconds after decision	reduced_tree_100 or async_posthoc
Regulatory audit logging	No hot-path constraint	Post-hoc	async_posthoc
High-throughput approximate	Under 20 ms	At decision time	reduced_tree_50
Model-agnostic deployment	Any	At decision time	kernel (accept higher MAE)
Budget-critical production	Under 10 ms	Not on hot path	async_posthoc

6.2 Implications for the Nigerian Financial Services Environment

A significant share of Nigerian transactions flow through USSD channels and NIP-routed transfers carrying minimal metadata, and thin transaction histories weaken behavioural features, so SHAP attributions computed over sparse vectors carry lower information value. The latency cost of synchronous explanation is correspondingly harder to justify on those channels, making async_posthoc particularly appropriate for them, while richer card-network transactions warrant reduced_tree_100 as a balanced default. As platforms such as OPay, Moniepoint, and PalmPay scale under the Central Bank of Nigeria's

accountability expectations, this channel-differentiated strategy selection offers a practical deployment path.

6.3 Limitations

Several limitations bound the generality of these results. All measurements were taken on a single machine; distributed deployments introduce inter-service network latency not captured here. The benchmark is sequential rather than concurrent, so the reported percentiles do not reflect CPU contention, under which the synchronous tree-based strategies would degrade first. The sample sizes (100 transactions per configuration, 20 for KernelSHAP) expose the order-of-magnitude separation between strategies but do not support confidence intervals on tail percentiles, and KernelSHAP's p95 and p99 estimates are correspondingly less reliable. Fidelity is assessed by mean absolute error alone; rank-based agreement over the top-k attributions would better reflect how explanations are consumed by analysts. The completion lag of the asynchronous strategy, that is, how soon after the decision its explanation becomes available, was not measured. Finally, the benchmark covers a single dataset and model family; the latency hierarchy should transfer to other tree ensembles, since TreeSHAP cost is governed by ensemble structure, but the specific fidelity values are dataset-dependent. Future work should extend the benchmark to concurrent load, larger KernelSHAP samples, rank-based fidelity metrics, asynchronous completion-lag measurement, and evaluation against live African transaction streams.

7. Conclusion

We presented TxWatch and, to our knowledge, the first controlled empirical benchmark of SHAP explanation strategies in a production-equivalent real-time fraud detection pipeline. Exact explanations fit within real-time budgets when models are built on deployable feature sets; reduced-background approximation gives a predictable tuning lever; feature sampling should be avoided; and asynchronous computation remains the only strategy immune to model complexity. The accompanying decision framework converts these measurements into deployable guidance for institutions balancing regulatory accountability against payment-network latency requirements, with particular relevance to Nigerian and broader African FinTech operators.

Data and Code Availability

The complete implementation, trained model artefacts, benchmark harness, and reproduction instructions are openly available at https://github.com/mameen7/txwatch. The IEEE-CIS Fraud Detection dataset is publicly available on Kaggle (IEEE Computational Intelligence Society, 2019) and is not redistributed with the repository.

References

Adewumi, A. O., & Akinyelu, A. A. (2017). A survey of machine-learning and nature-inspired based credit card fraud detection techniques. International Journal of System Assurance Engineering and Management, 8(2), 937–953.

Amaefule, I. A., Eke, C. I., Nkpodoin, J. B., & Imoize, A. L. (2022). A hybrid model for detecting fraudulent electronic card transactions in Nigerian financial institutions. Scientific African, 18, e01392.

Amazon Web Services. (2023). AWS Fraud Detector documentation. https://docs.aws.amazon.com/frauddetector/

Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. Proceedings of the 2017 International Conference on Computing Networking and Informatics (ICCNI), 1–9.

Central Bank of Nigeria. (2023). Operational guidelines for open banking in Nigeria. https://www.cbn.gov.ng/Out/2023/CCD/Operational%20Guidelines%20for%20Open%20Banking%20in%20Nigeria.pdf

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence (SSCI), 159–166.

Dean, J., & Barroso, L. A. (2013). The tail at scale. Communications of the ACM, 56(2), 74–80.

Demirgüç-Kunt, A., Ansar, S., Klapper, L., & Singer, D. (2022). The Global Findex Database 2021: Financial inclusion, digital payments, and resilience in the age of COVID-19. World Bank Group.

European Parliament. (2016). General Data Protection Regulation (GDPR), Regulation (EU) 2016/679. Official Journal of the European Union.

European Parliament & Council of the EU. (2024). Artificial Intelligence Act, Regulation (EU) 2024/1689. Official Journal of the European Union.

Feedzai. (2022). Feedzai platform overview. https://feedzai.com

FICO. (2022). FICO Falcon platform. https://www.fico.com/en/products/fico-falcon-platform

GSMA. (2023). The state of mobile internet connectivity report 2023. GSMA Intelligence.

IEEE Computational Intelligence Society. (2019). IEEE-CIS fraud detection dataset. Kaggle. https://www.kaggle.com/competitions/ieee-fraud-detection

Ileberi, E., Sun, Y., & Wang, Z. (2022). A machine learning based credit card fraud detection using the GA algorithm for feature selection. Journal of Big Data, 9(1), 1–17.

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.

Kurshan, E., Shen, H., & Yu, H. (2020). Financial crime and fraud detection using graph neural networks: Application to real-world fraud data. Proceedings of the 2020 IEEE International Conference on Big Data, 1925–1932.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S. I. (2020). From local explanations to global understanding with explainable AI for trees. Nature Machine Intelligence, 2(1), 56–67.

Nigeria Inter-Bank Settlement System. (2021). Industry fraud report Q3 2020. NIBSS.

Nilson Report. (2025). Payment card fraud losses approach $34 billion [Press release]. Globe Newswire.

Obeng, S., Asante, M., & Gyamfi, E. (2025). Explainable AI for credit card fraud detection: Bridging the gap between accuracy and interpretability. World Journal of Advanced Research and Reviews, 25(3).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144.

Shapley, L. S. (1953). A value for n-person games. In H. W. Kuhn & A. W. Tucker (Eds.), Contributions to the theory of games (Vol. 2, pp. 307–317). Princeton University Press.

Stripe. (2023). Stripe Radar documentation. https://stripe.com/docs/radar

Visa Inc. (2022). Visa core rules and Visa product and service rules. https://usa.visa.com/dam/VCOM/download/about-visa/visa-rules-public.pdf