KONTOGRAPH: Verified Point-in-Time Feature Consistency and Amortised Explanation for Real-Time Anti-Money Laundering under a 200 ms Decision Budget

Ahmed Abolfadl
Faculty of Media Engineering and Technology
German University in Cairo
Cairo, Egypt
ahmed.abuelfadel@student.guc.edu.eg

***Abstract*—Regulation (EU) 2024/886 obliges European payment service providers to settle euro credit transfers in under ten seconds, around the clock. This removes both the overnight batch window in which anti-money-laundering (AML) analytics traditionally ran and the settlement delay that made recovery possible, forcing detection, explanation and decision inside a single-digit-second envelope. We present KONTOGRAPH, an end-to-end AML pipeline for the SEPA Instant rail built under a self-imposed 200 ms 99th-percentile budget, and report an empirical study on 1,562,860 simulated payments with injected typologies and deliberately incomplete labels. Three findings are of interest beyond the system itself. First, a temporal graph network with per-node memory improves PR-AUC over a gradient-boosted tabular baseline from 0.0053 to 0.1717, a paired day-blocked bootstrap difference of $+0.166$ with 95% CI $[0.105, 0.241]$; per-node memory alone more than doubles the score. Second, expressing each feature once and compiling it to three execution backends, with equivalence enforced by property-based tests that perturb the future, surfaced three point-in-time violations that code review had passed—each of which would have inflated reported performance. Third, and most consequential for practice, exporting the deployed tree ensemble to ONNX changed only 7.4×10^{-8} in mean score yet altered 0.26% of decisions and inflated the alert volume by 12%, because 32-bit accumulation perturbs scores across a cost-optimal threshold of 3.98×10^{-4}. We argue that a serving-format conversion must be treated as a model change until measured, and that fidelity metrics for subgraph explainers can be vacuous when candidate neighbourhoods are small—a null result we report in full.**

***Index Terms*—anti-money laundering, temporal graph networks, real-time inference, data leakage, explainable AI, feature stores, model deployment, SEPA Instant**

I. Introduction

Instant payment schemes have changed the operating conditions of financial-crime detection more sharply than any modelling advance of the past decade. Regulation (EU) 2024/886 requires euro-area payment service providers to receive SEPA Instant Credit Transfers continuously and to complete them within ten seconds. Two properties that AML systems had implicitly relied upon disappeared simultaneously: the overnight window in which batch analytics ran, and the settlement lag that permitted funds to be recalled once a case was confirmed.

The consequence is a hard systems constraint. Scoring, explanation and decision must complete while the payer waits, and the decision must be defensible: Article 22 of the General Data Protection Regulation grants a data subject subject to a solely automated decision with legal effect the right to meaningful information about the logic involved, and a suspicious-activity report filed with a national financial intelligence unit must state grounds a human investigator can verify.

This paper describes a system built to that constraint and, more importantly, reports what measurement revealed about it. Our contributions are:

1) An architecture in which point-in-time correctness is a *tested invariant* rather than a convention: one feature specification is lowered to a typed intermediate representation and compiled to three backends (batch SQL, streaming SQL, and an incremental online executor), whose equivalence is checked by property-based tests that rewrite future events and assert that nothing computed earlier moves (Section IV).
2) An ablation on 1.5M simulated SEPA Instant payments isolating the contribution of continuous-time graph structure and of per-node memory, with paired day-blocked bootstrap confidence intervals throughout (Section VI).
3) Three failure modes that only measurement exposed—leakage bugs invisible to review, a serving-format conversion that shifted 12% of alerts, and explanation-fidelity metrics that were vacuous at the observed neighbourhood size (Section VII).

We emphasise at the outset that the evaluation uses *synthetic* data. No claim is made about performance on production banking traffic. What synthetic data does buy is a known ground truth, which permits measurement of label selection bias—

a quantity real institutions cannot observe about themselves, and whose absence we argue should temper the reading of published AML metrics.

II. Background and Related Work

A. Graph learning for financial crime

Modelling payments as a graph rather than as independent rows is well motivated: laundering typologies such as fan-in/fan-out mule networks, structuring and layering chains are defined by topology, not by the attributes of any single transfer. Weber et al. [5] applied graph convolutional networks to the Elliptic Bitcoin dataset, and Altman et al. [6] released large-scale synthetic AML transaction graphs in recognition that suitable labelled data is not publicly available.

Static graph methods discard timing, which is precisely the signal in a velocity-driven typology. Continuous-time dynamic graph models address this: JODIE [2], DyRep [3] and TGAT [4] model interaction streams directly, and Temporal Graph Networks (TGN) [1] unify these under a memory module updated per event combined with a temporal attention embedding. Our detector follows the TGN formulation; the ablation in Section VI isolates the memory module specifically, since it is the component whose contribution is most often asserted rather than measured.

B. Explanation under a latency budget

Post-hoc explanation for graph models is dominated by perturbation search. GNNExplainer [9] optimises a soft edge mask per instance; PGExplainer [10] amortises this by training a mask generator; SubgraphX [11] searches subgraphs via Monte Carlo tree search. Attribution methods such as SHAP [12] require many model evaluations per instance. All are orders of magnitude outside a 200 ms end-to-end budget that must also accommodate feature retrieval and scoring.

We follow the amortisation strategy of PGExplainer in spirit—pay the search cost offline, learn a function that predicts its output in one forward pass—and evaluate faithfulness with comprehensiveness and sufficiency in the sense of DeYoung et al. [13]. For the narrative delivered to investigators we prefer counterfactual statements in the sense of Wachter et al. [14]: "removing these two payments moves the score below threshold" is checkable against an account, whereas a feature attribution is not. Rudin [15] argues that post-hoc explanation of opaque models is itself hazardous in high-stakes settings; our position is narrower, namely that if post-hoc explanation is used, its fidelity gap must be published rather than assumed, and Section VI-D reports ours including the case where the metric turned out to measure nothing.

C. Leakage and evaluation discipline

Kaufman et al. [16] characterise leakage as the introduction of information about the target that would not legitimately be available at prediction time, and note that it typically produces excellent offline results and silent production failure. In temporal settings, random cross-validation is invalid; Bergmeir and Benítez [17] analyse evaluation for time-dependent data. Saito and Rehmsmeier [18] show that under severe class imbalance the precision-recall curve is more informative than ROC, which motivates our choice of headline metric.

Our contribution here is not the observation that leakage matters but the mechanism: we treat point-in-time correctness as a property amenable to automated falsification using property-based testing [19], and report that this located defects that had survived review.

TABLE I
Generated dataset (profile `full`)

Payments	1,562,860
Accounts	6,576
Simulated days	270
Criminal payments (ground truth)	2,995
Criminal rate	0.19%
Confirmed labels	3,495
Criminal label coverage	**14.5%**

III. System Architecture

A. Data generation

Absent public SEPA-labelled data, we generate an agent-based simulation of a German retail payment ecosystem. Rather than sampling transfers from a fitted distribution, the simulator instantiates households, merchants, small enterprises and banks whose behaviour follows an economic calendar: salaries on the last business day of the month, rent to a stable counterparty, Poisson grocery arrivals, utilities, e-commerce and peer-to-peer transfers. German public holidays, including movable feasts derived by computus, modulate activity. Account identifiers are IBANs with valid ISO 7064 MOD-97-10 check digits and bank codes drawn from the three-pillar structure of German banking, whose segments exhibit different customer behaviour.

Five typologies are injected under known ground truth—mule networks, structuring, authorised push payment fraud, circular flows and layering chains—each in a standard and an *evasive* variant that introduces dwell time, fresh-account fan-out and sub-threshold amounts.

Crucially, the simulator models *label generation* as a separate process from crime. A criminal transaction is confirmed only if a simulated incumbent rules engine flags it and an investigator concurs, with a verification latency recorded in a `decided_ts` field distinct from the event timestamp. This reproduces the selection bias of real AML label streams: what the legacy system never flagged is silently recorded as legitimate.

Table I summarises the generated corpus. The dataset is a deterministic function of (configuration, seed, code revision) and is identified by a content hash rather than stored, with a determinism test asserting byte-identical regeneration.

B. Streaming and storage

Events are serialised in Apache Avro against a schema registry with a backward compatibility gate enforced in continuous integration, published to a Kafka-compatible broker

keyed by debtor account so that per-account ordering is preserved within a partition, and landed into Apache Iceberg tables in a bronze/silver/gold arrangement. Promotion between layers passes a quality gate whose expectations are tagged BLOCKING, QUARANTINE or WARN; the pipeline halts, diverts or records accordingly.

C. Feature platform

Twenty-nine features are declared in a small domain-specific language over (entity, aggregation, window, filter). Each declaration is lowered to a typed intermediate representation in which the window semantics are stated exactly once:

$$t - W \le h.\texttt{acceptance\_ts} < t,$$

left-closed and right-open on event time, strictly excluding the scored event itself and anything sharing its millisecond. The IR is then compiled to three targets: DuckDB SQL for historical backfill, Flink SQL for streaming, and an incremental executor holding per-entity accumulators for online serving.

This is the anti-skew mechanism. Training-serving skew conventionally arises because offline features are written in one language and reimplemented in another; here there is one specification and three compilers, and their agreement is a test rather than an aspiration.

D. Serving

A single asynchronous FastAPI process holds the model, the online feature state and the explainer. Scoring executes synchronously inside the asynchronous handler by design: the work is CPU-bound and takes single-digit milliseconds, so dispatching to a thread pool would add scheduling latency without parallelism benefit. Decision-log persistence and alert publication are asynchronous, so disk latency cannot enter the decision path.

Every decision is appended to a hash-chained log recording the model version, feature-set fingerprint, code revision, per-stage latency and the explanation presented—captured at decision time rather than reconstructed later, which is what an Article 22 obligation actually requires. Each entry commits to its predecessor, so tampering is both detectable and localised: verification returns the index of the first broken entry, bounding the trustworthy prefix.

IV. Point-in-Time Correctness as a Tested Invariant

We state the invariant as a property and attempt to falsify it automatically. Let f be any compiled feature and E an event stream. For an event e at time t, the property is

$$f(e, E) = f(e, \{e' \in E : e'.\texttt{ts} < t\}),$$

that is, the computed value must be invariant to every event at or after t.

Two test families operate on Hypothesis-generated streams. The first computes each feature through the batch and online backends and asserts elementwise equality across every value, so any divergence fails the build. The second is stronger and makes no assumption about the mechanism of a leak: it perturbs the future, replacing all events at or after t with different ones, and asserts that no value computed before t changes. A leak through a window boundary, a join condition, an ordering error or a future-looking enrichment is caught identically, because the test does not encode a hypothesis about which of these occurred.

This suite located three defects that had passed code review:

1) A `LEFT JOIN` combined with `COUNT(*)` returned 1 rather than 0 for an account's first observed payment, because SQL counts the null-extended row. Every entity's first event carried a fabricated history of size one.
2) In the online executor, entity keys were not namespaced by role, so account B as debtor and B as creditor collided. A feature requesting outbound history was served the account's inbound history.
3) An accumulator could not exclude events sharing the scored event's millisecond, admitting simultaneous events into a strictly-prior window.

All three would have inflated reported performance, and none was apparent from reading the code. We regard this as the central methodological claim of the paper: point-in-time correctness is falsifiable by machine, and treating it as a matter of discipline forgoes that.

V. Experimental Setup

A. Protocol pre-registration

The evaluation protocol—metrics, splits, cost model, ablation ladder, and an explicit commitment to publish a negative result—was written and committed to version control *before any model existed*. The repository history is the evidence that no metric was selected after observing outcomes. The protocol is identified by hash `2eacb6e2` and referenced by every generated report.

B. Splits

Splits are chronological with purge and embargo. The purge band is one day, chosen after measuring that a 30-day band—initially specified—removed 79% of usable data and was therefore not a defensible configuration. A guard rejects any band exceeding 25% of a fold. Labels are admitted to a fold only if their `decided_ts` precedes that fold's availability cutoff, so a model never trains on a verdict that had not yet been reached.

The held-out test fold contains 303,129 payments of which 44 carry a confirmed positive label, a labelled positive rate of 1.45×10^{-4}.

C. Metrics and operating point

We report PR-AUC as the ranking metric, following [18], with precision and recall at an analyst capacity of 200 alerts per day, and expected cost per 10,000 payments under a stated cost model (investigation EUR 35, customer friction EUR 12, prevention fraction 0.85). Confidence intervals are day-blocked bootstrap: resampling units are whole days, not payments,

TABLE II
ABLATION LADDER ON THE HELD-OUT TEST FOLD (303,129 PAYMENTS, 44 CONFIRMED POSITIVES). RECALL IS AT 200 ALERTS/DAY.

#	Model	PR-AUC [95% CI]	Recall	Cost/10k
0	Constant	0.0001 [0.0001, 0.0002]	0.045	473,280
1	LightGBM	0.0053 [0.0030, 0.0079]	0.909	20,459
2	+ static graph	0.0144 [0.0064, 0.0319]	0.795	53,529
3	+ TGN, no mem.	0.0734 [0.0348, 0.1353]	0.773	21,627
4	**+ memory**	**0.1717 [0.1011, 0.2445]**	0.909	19,997

because payments within a day are dependent and payment-level resampling would understate interval width.

For the GPU-trained rungs the operating point is the capacity constraint itself, i.e. the k-th highest score for k corresponding to 200 alerts per day. It is deliberately *not* fitted: the protocol selects thresholds on validation, the GPU experiments exported test scores only, and fitting a threshold on the test fold would constitute leakage presented as a cost figure.

D. Hardware

All components except graph-model training run on an Intel Core i7-8550U (4 cores, 16 GB RAM, no GPU). Graph models train on a single NVIDIA T4. TGN training with memory required 1,094 s for five epochs over 937,731 training events at batch size 200; without memory, 155 s.

VI. RESULTS

A. Ablation ladder

Table II reports the ladder. Each rung adds exactly one mechanism.

The paired day-blocked bootstrap difference between rung 4 and rung 1 is $+0.166$ with 95% CI $[0.105, 0.241]$, excluding zero. Continuous-time graph structure accounts for a large part of the improvement (rung 2 to rung 3), and per-node memory more than doubles PR-AUC again (0.0734 to 0.1717). Since memory is the component of the TGN formulation most often included without separate justification, we regard its isolated measurement as the more useful half of this result.

An independent training run (a separate notebook, different seed) reproduced the direction and approximate magnitude of the memory ablation, 0.0955 to 0.1555, which we note as weak evidence of stability rather than as a second measurement.

B. The cost metric contradicts the ranking metric

Rung 2 attains nearly three times rung 1's PR-AUC while incurring $2.6\times$ the expected cost (EUR 53,529 against EUR 20,459). This is not an inconsistency. PR-AUC integrates over all operating points, whereas cost is evaluated at one—the capacity constraint—and at that point rung 2's recall is lower (0.795 against 0.909). A model that ranks better on average can be worse at the only threshold that will be deployed.

We take this as an argument for making expected cost the decision metric and PR-AUC a diagnostic, and note that a ladder reported on PR-AUC alone would have recommended rung 2 over rung 1.

TABLE III
PER-STAGE DECISION LATENCY (2,000 PAYMENTS, SINGLE-THREADED CPU)

Stage	p50 (ms)	p99 (ms)
Feature fetch	1.84	4.67
Graph assembly	0.00	0.01
Inference	0.42	1.78
Explanation	0.00	1.87
Counterfactual search	0.00	3.59
Unaccounted	0.17	0.46
Total	**2.71**	**7.97**

TABLE IV
TEACHER–STUDENT EXPLANATION, 57 INSTANCES

Teacher (perturbation search), median	575.6 ms
Student (single forward pass), median	2.38 ms
Speed-up	$241\times$
Top-5 Jaccard vs. teacher	0.904
Spearman rank correlation vs. teacher	-0.038
Median candidate edges per instance	**1.0**

C. Latency

Table III gives the per-stage budget over 2,000 scored payments. Percentiles are nearest-rank rather than linearly interpolated, so each reported figure corresponds to a latency some request actually experienced; interpolation is optimistic in the tail, placing p99 at 14 ms for a sample of 99 fast requests and one at 900 ms.

The budget is met with substantial margin. Two caveats: median explanation and counterfactual times are zero because only alerting payments are explained (98 of 2,000), so the p99 is the meaningful figure for the explained path; and per-stage p99 values do not sum to the total p99, because each is measured independently and the slowest feature fetch does not occur in the same request as the slowest counterfactual search. We report them unreconciled rather than constructing a request that was worst at everything.

Repeated runs on the same machine varied between 3.6 ms and 8.0 ms p99 depending on background load. The claim we defend is the order of magnitude of headroom, not a specific figure.

D. Amortised explanation

Table IV reports the explainer over 57 explained instances.

The amortisation objective is met unambiguously. The perturbation teacher requires 575.6 ms per explanation—nearly three times the entire end-to-end budget—while the distilled student answers in 2.38 ms, comfortably inside it.

The fidelity figures, however, are not interpretable at this neighbourhood size, and we report this as a null result. The median explained instance has a single candidate edge. With one candidate, a top-k Jaccard for $k = 5$ is 1.0 by construction and rank correlation is undefined (our implementation returns 0 for $n < 2$). The reported 0.904 and -0.038 therefore principally measure candidate-set cardinality, not student quality.

TABLE V
ONNX EXPORT PARITY AGAINST THE TRAINED BOOSTER

Mean absolute score difference	7.4×10^{-8}
Maximum absolute score difference	1.7×10^{-4}
Decisions changed	774 (0.26%)
Alerts, original	6,458
Alerts, ONNX	7,232
Alert volume inflation	**+12.0%**

We stress this because the tempting readings are both unsupported. Reporting "241× faster with 0.90 Jaccard agreement" would present an artifact as a success; reporting "241× faster but rank-uncorrelated with its teacher" would present the same artifact as a substantive negative finding. Distinguishing them requires the candidate-count distribution, which was not initially recorded. We now record it, and recommend that fidelity results for subgraph explainers be published alongside it as a matter of course.

The cause is explicable: the teacher explains the highest-scoring test events, and the detector's high scores concentrate on payments to fresh beneficiaries with almost no prior neighbourhood—so there is little to attribute among. A valid experiment must stratify explained instances by candidate count and report fidelity per stratum, or widen the candidate set. Whether pointwise distillation loss teaches ranking, which we consider the likeliest weakness, is untestable on the present evidence.

VII. Three Failure Modes Found by Measurement

A. An inverted cost model

An initial cost specification treated a true positive as revenue-generating, crediting recovered funds. The threshold optimiser consequently preferred 2.7% recall to 97%: alerting on almost nothing minimised the objective. Detection does not earn money, it avoids a loss accounted for elsewhere. We note the diagnostic value of the behaviour—an optimiser that proposes catching nothing is usually reporting that the objective is misspecified—and the protocol was amended with the correction recorded.

B. Serving-format conversion as a model change

The deployed gradient-boosted model was exported to ONNX for portability, and the export was then compared against the original on all 303,129 test payments (Table V).

The mean difference is the statistic that would ordinarily be quoted to establish that an export is faithful, and it is misleading. The ONNX tree ensemble operator accumulates in 32-bit floating point while the source library accumulates in 64-bit, and the converter rejects double-precision inputs for classifier graphs. At a cost-optimal operating threshold of 3.98×10^{-4}, that rounding is sufficient to move scores across the decision boundary.

The operational consequence is material: against a stated capacity of 200 alerts per day, adopting the export would have increased alert volume by 12% while all published metrics continued to describe a different model. We therefore serve the original artifact and ship the export as a portability artifact with its parity recorded. The general statement we would defend is that *a serving-format conversion is a model change until measured otherwise*, and that parity must be assessed in decisions at the deployed threshold, not in mean score error.

C. A generated report asserting absent evidence

The system drafts bilingual suspicious-activity narrative fragments from counterfactuals. Two forms exist: edge-space counterfactuals, which name specific transactions, and feature-space counterfactuals, which name an aggregate indicator and identify no transaction at all. Both were rendered through a single template, which given the latter produced the sentence "The alert rests on 0 related payment(s): ", asserting transactional evidence that did not exist in a document destined for a regulatory filing.

The generator now branches on counterfactual type and states explicitly that the feature-space form "is a statement about an aggregate indicator, not about any identified transaction, and does not on its own identify conduct to report". We include this case because automated narrative generation for regulatory reporting is an emerging practice, and the failure mode—fluent text asserting evidence the system does not possess—is characteristic rather than incidental.

VIII. Limitations and Threats to Validity

Synthetic data. All results are on simulated payments. The simulator's calibration targets are public aggregates, not licensed microdata, and no claim of transfer to production traffic is made. The generative process and the detector share an author, and although the detector receives no oracle information, we cannot exclude that the injected typologies are more separable than real ones.

Weak labels. Only 14.5% of genuinely criminal payments receive a confirmed label. Every precision figure is therefore an upper bound on error, and evasive campaign variants are confirmed at roughly half the rate of standard ones. This is deliberate and measured, but it means the reported ranking of models is a ranking under a biased label distribution.

Positives are few. The test fold contains 44 confirmed positives. Bootstrap intervals are correspondingly wide, and while the rung-4 versus rung-1 interval excludes zero, comparisons between adjacent middle rungs do not separate.

The graph model is not deployed. The latency budget in Table III measures the gradient-boosted serving path. Promoting the TGN requires exporting validation scores, selecting a cost-optimal threshold on them, and re-measuring latency against a neural forward pass. None of this has been done, and no latency claim is made for the graph model.

No fairness assessment. The simulator generates names reflecting Germany's largest immigrant communities, which makes disparate-impact analysis across name origin feasible. It has not been performed. This is a gap, not evidence of its absence.

Single-institution framing. Cross-institutional typologies, which constitute a substantial share of real laundering, are out of scope.

IX. Conclusion

We presented an anti-money-laundering pipeline for instant euro payments that decides and explains within a 200 ms budget, and reported an empirical study whose most transferable results concern measurement rather than modelling. A temporal graph network with per-node memory substantially outperformed a gradient-boosted baseline on simulated data, with per-node memory alone more than doubling PR-AUC. But the findings we consider most useful to practitioners are the three cases in which measurement contradicted a reasonable expectation: point-in-time violations invisible to code review, a portable model export that altered 12% of alerts while changing mean score by 7×10^{-8}, and explanation-fidelity metrics that were vacuous at the observed neighbourhood size.

A common thread connects them. In each case a plausible summary statistic—the code reads correctly, the mean error is negligible, the Jaccard overlap is 0.90—was available and would have been reported. What falsified each was a test constructed to fail: perturbing the future, comparing decisions rather than scores, and recording the cardinality of the set being compared. We suggest that systems making claims of this kind should be designed around such tests, and that reviewers ask which statistic would have been reported had the test not been run.

The implementation, evaluation protocol, decision records and generated artifacts are available so that every reported figure can be regenerated from the corresponding command.

Reproducibility

Every number in this paper is regenerated from stored artifacts by a command-line tool rather than transcribed. The evaluation protocol was committed before any model existed. Metrics computed on the GPU were independently recomputed by the local evaluation code and agreed to four decimal places; the ingestion tool reports disagreement rather than silently preferring either value. Datasets are content-hashed functions of (configuration, seed, revision) and are regenerated rather than stored, with a determinism test asserting byte-identical output.

References

[1] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," in *ICML Workshop on Graph Representation Learning*, 2020.

[2] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proc. KDD*, 2019, pp. 1269–1278.

[3] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "DyRep: Learning representations over dynamic graphs," in *Proc. ICLR*, 2019.

[4] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *Proc. ICLR*, 2020.

[5] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson, "Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics," in *KDD Workshop on Anomaly Detection in Finance*, 2019.

[6] E. Altman, J. Blanuša, L. von Niederhäusern, B. Egressy, A. Anghel, and K. Atasu, "Realistic synthetic financial transactions for anti-money laundering models," in *Advances in Neural Information Processing Systems*, 2023.

[7] E. A. Lopez-Rojas, A. Elmir, and S. Axelsson, "PaySim: A financial mobile money simulator for fraud detection," in *Proc. European Modeling and Simulation Symposium*, 2016.

[8] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017.

[9] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems*, 2019.

[10] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," in *Advances in Neural Information Processing Systems*, 2020.

[11] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *Proc. ICML*, 2021.

[12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, 2017.

[13] J. DeYoung, S. Jain, N. F. Rajani, E. Lehman, C. Xiong, R. Socher, and B. C. Wallace, "ERASER: A benchmark to evaluate rationalized NLP models," in *Proc. ACL*, 2020.

[14] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *Harvard Journal of Law & Technology*, vol. 31, no. 2, pp. 841–887, 2018.

[15] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, pp. 206–215, 2019.

[16] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 4, pp. 1–21, 2012.

[17] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," *Information Sciences*, vol. 191, pp. 192–213, 2012.

[18] 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, e0118432, 2015.

[19] D. R. MacIver, Z. Hatfield-Dodds, and many other contributors, "Hypothesis: A new approach to property-based testing," *Journal of Open Source Software*, vol. 4, no. 43, p. 1891, 2019.

[20] European Parliament and Council, "Regulation (EU) 2024/886 amending Regulations (EU) No 260/2012 and (EU) 2021/1230 as regards instant credit transfers in euro," *Official Journal of the European Union*, 2024.

[21] European Parliament and Council, "Regulation (EU) 2016/679 (General Data Protection Regulation)," *Official Journal of the European Union*, 2016.