EAI Endorsed Transactions
on AI and Robotics

Research Article EAI.EU

# Temporal-Structural Stress Testing and Cross-Granularity Robustness Evaluation of Graph AI for Cryptocurrency AML Detection

Sadia Akter[1], Sabab Islam[2], Md Faysal Ahmed[1], Md Hossain Jamil[3], Md Fokrul Islam Khan[4], Partha Singha[5], Abu Kowshir Bitto[6]*, and Sanim Yousuf Fahim[6]

[1]School of Business, International American University, Los Angeles, California, USA
[2]College of Business, Texas A&M University–Commerce, USA
[3]MBA in IT, Humphreys University, USA
[4]College of Business, Westcliff University, USA
[5]College of Technology & Engineering, Westcliff University, USA
[6]Department of Software Engineering, Daffodil International University, Dhaka, Bangladesh

## Abstract

Cryptocurrency anti-money-laundering (AML) is typically framed as a graph-learning problem, since suspicious value flows rarely appear as isolated records. This study examines whether graph AI architectures retain an advantage over strong non-GNN baselines when evaluation is temporal, structurally explicit, and aligned with investigator triage. We answer that question using two public Bitcoin AML datasets: Elliptic, a transaction-node dataset, and Elliptic2, a subgraph-level dataset. Our comparison spans classical models, tree ensembles, graph-derived features, graph embeddings, neural baselines, and GNN variants, evaluated across AUPRC, AUROC, accuracy, precision and recall at an investigation budget, calibration, operating points, and bootstrap uncertainty. On the temporal Elliptic test period, Extra Trees achieves AUPRC 0.570 and AUROC 0.869, while GraphSAGE, the strongest standard GNN baseline, reaches AUPRC 0.311. A strict-temporal GraphSAGE variant improves to AUPRC 0.381 but still falls behind the leading tree ensemble model. The paired bootstrap difference between Extra Trees and GraphSAGE is 0.259 AUPRC, with a 95% interval of [0.222, 0.297]. On the full Elliptic2 subgraph benchmark, Random Forest achieves AUPRC 0.494 and AUROC 0.923 in the main split and remains the strongest model by mean AUPRC across repeated splits. These findings indicate that graph AI for cryptocurrency AML should be stress-tested against capable tree ensemble models and operational metrics before architectural complexity is treated as deployment evidence.

Received on 21 June 2026; accepted on 24 August 2026; published on 02 September 2026

**Keywords:** cryptocurrency AML, graph AI, graph neural networks, financial crime, temporal validation



doi:10.4108/airo.13658

## 1. Introduction

Cryptocurrency AML sits at the intersection of fraud detection, financial regulation, and network analysis. This combination is well suited to machine learning but difficult to evaluate rigorously. Public blockchains expose transaction flows, but they reveal little directly about the actors, intentions, or case histories behind those flows. Observations are relational and temporal, illicit cases are rare, and labels are necessarily incomplete. Aggregate accuracy is therefore a weak evaluation metric for AML usefulness. To be operationally useful, a model must help allocate scarce investigator capacity, keep false positives to a manageable level, and remain useful when applied to later transactions or to a related dataset with a different prediction unit [1].

Graph-based learning is a natural fit for this type of data. The Elliptic dataset represents Bitcoin transactions as a graph with more than 200,000 nodes, directed payment flows, temporal information, and partial

*Corresponding author. E-mail: abu.kowshir777@gmail.com

licit/illicit labels [2]. Elliptic2 shifts the unit of analysis from individual transactions to labeled subgraphs, which reflects the investigative intuition that laundering tends to appear as a local structure or path pattern rather than as a single suspicious transaction [3]. Together, these datasets make GNNs, graph embeddings, and graph-derived features reasonable predictors for AML research.

The harder empirical question is whether common graph-learning models outperform competitive non-GNN baselines once the validation design resembles deployment environments. This matters because GNNs add engineering complexity, hardware and memory requirements, monitoring burden, and explanation challenges. In regulated financial settings, that additional burden is defensible only when it produces a clear operational gain under a validation design that reflects the decision context [4].

We approach cryptocurrency AML as a model-governance testbed rather than as a venue for proposing another GNN architecture. The starting premise is deliberately simple: graph-structured data do not by themselves establish that graph neural models are better than well-tuned tree ensemble models. We examine this premise across two studies. Study 1 uses Elliptic as a transaction-level temporal test. Study 2 uses Elliptic2 as a full subgraph-level robustness exercise.

The contribution lies in the deployment-facing comparison. Instead of introducing a new architecture, we ask whether reproducible graph-AI baselines remain competitive against carefully tuned non-GNN models under temporal and cross-granularity evaluation. Specifically, the work makes four contributions:

1. It evaluates Elliptic with a temporal train-validation-test split, avoiding the inflation that can result from random shuffling.

2. It compares classical models, tree ensembles, graph features, graph embeddings, MLPs, and multiple GNN variants under common operational AML metrics.

3. It extends the analysis to full Elliptic2 subgraph classification with background-node feature aggregation and repeated stratified splits.

4. It reports bootstrap intervals, paired bootstrap comparisons, operating points, calibration diagnostics, and threats to validity to ensure reproducibility.

The empirical result is narrow but consequential for model governance in AML studies. Tree ensemble models outperform the evaluated GNNs on the temporal Elliptic experiment and remain highly competitive on full Elliptic2. This finding does not render graph structure irrelevant, nor does it rule out more specialized temporal, directed, heterogeneous, or subgraph-native GNNs. It shows that graph-AI claims in AML need to be tested against capable non-GNN baselines, temporal splits, and investigation-budget metrics before they are treated as deployment evidence in real-world scenarios.

The remainder of this manuscript is organized as follows. Section 2 presents the related work and reviews the existing studies relevant to the research topic. Section 3 describes the materials and methods employed in the study, including the proposed approach, simulation setup, and evaluation procedures. Section 4 presents the obtained results, while Section 5 provides a detailed discussion and interpretation of the findings in comparison with existing studies. Finally, Section 6 concludes the manuscript by summarizing the key findings, highlighting the main contributions of the study, and outlining potential directions for future research. The relevant references are provided at the end of the manuscript.

## 2. Related Work

### 2.1. Cryptocurrency AML and Public Blockchain Benchmarks

Early Bitcoin measurement work established a fundamental but important point that transaction flows are visible on public ledgers even when the parties behind them remain pseudonymous [5]. Later forensic work demonstrated how address clustering and entity-level reasoning can convert ledger traces into investigative signals [6]. These studies are not AML classifiers, but they explain why public ledgers are useful for financial experiments and why identification remains difficult, since a transaction identifier is not a person or institution, and laundering behavior may be distributed across addresses, entities, chains, and time in different situations.

The Elliptic dataset has become one of the main public reference points for Bitcoin AML research. It represents transactions as nodes and directed payment flows as edges, with labels for illicit, licit, and unknown transactions [2]. Part of its value comes from its imperfections: labels are partial, features are anonymized, and observations are organized into temporal slices. Subsequent work has built on this foundation to explore temporal and graph-based evaluations, including recurrent and temporal graph approaches [7].

Elliptic2 changes the framing by moving from individual transaction nodes to subgraphs. Rather than asking whether a single transaction is suspicious, it labels connected components or local laundering shapes as suspicious or licit [3]. That design is useful because laundering schemes often appear as multi-node structures, layering patterns, or conversion paths. In this paper, preprocessed Elliptic2 serves as a second AML

prediction unit, not as a direct feature-space transfer test.

## 2.2. Classical Machine Learning and Tree Ensembles

Financial fraud and AML detection have a long statistical tradition in which rare-event sensitivity must be balanced against the cost of false alerts [8]. Comparative work in financial fraud detection has repeatedly shown that classical baselines can be difficult to outperform [9]. Tree ensembles remain especially relevant for tabular financial crime data. Random Forests [10], Extra Trees [11], gradient boosting [12], and XGBoost [13] can capture nonlinear interactions and heterogeneous feature effects without requiring explicit graph message passing. Since the Elliptic feature matrix already contains anonymized transaction and neighborhood-derived attributes, tree ensembles should be treated as serious competitors rather than weak baselines.

For RegTech research, this baseline question is not merely technical. If a simpler model family such as a tree ensemble performs as well as, or better than, a graph neural network under realistic validation, the graph model has to justify its additional engineering, monitoring, and model-risk burden. This view is consistent with high-stakes machine-learning guidance that cautions against unnecessary black-box complexity when the performance gain is unclear [14].

## 2.3. Graph Representation Learning

Graph neural networks learn representations by passing information across neighboring nodes. GCN [15], GraphSAGE [16], and graph attention networks [17] are standard examples for graph-structured prediction models. Surveys organize these methods into spectral, spatial, recurrent, attention-based, and autoencoder-style model families [18]. EvolveGCN extends message passing to dynamic graph sequences [19], while Temporal Graph Networks offer another framework for evolving relational data [20]. Node2vec [21] and spectral embeddings provide non-neural graph representation baselines. In AML and fraud detection, these approaches are attracting attention because suspicious behavior can depend on the surrounding transaction neighborhood rather than only on the focal transaction itself.

Graph learning is not automatically useful in all cases. It can struggle when labels are sparse, graph directionality is simplified, topology is noisy, or tabular features already carry most of the useful predictive signal. CARE-GNN illustrates how fraud-specific graph architectures can address relation selection and camouflage [22]. More recent graph-anomaly surveys emphasize that node, edge, subgraph, and graph-level anomaly tasks require different assumptions and evaluation protocols [23]. More recent studies have begun to confront the temporal and directed structure of blockchain transaction graphs directly rather than treating them as generic homogeneous graphs. Egressy et al. show that standard message-passing GNNs cannot provably recover certain directed transaction patterns unless they are augmented with direction-aware mechanisms [24]. Lin et al. apply a wavelet-based temporal graph transformer to illicit-transaction detection in Bitcoin [25]. Li et al. survey the broader graph-learning landscape for financial fraud and identify directionality and temporal handling as open challenges rather than solved problems [26]. This study complements that literature by asking a prior model-governance question: are graph-native architectural advances necessary at all once strong non-GNN baselines are properly tuned and temporally validated? It therefore does not propose a new direction-aware architecture. A simple GCN or GraphSAGE result should not be interpreted as evidence against graph learning as a whole. Large graph-evaluation efforts make the same methodological point: realistic splits and application-specific metrics are central to meaningful graph machine learning evaluation [27]. This study evaluates graph learning as one model family within that broader comparison for AML detection.

## 2.4. Evaluation, Leakage, and Rare Event Detection

Illicit transactions and suspicious subgraphs are rare, which makes accuracy a poor guide for evaluation. A model can appear accurate simply by predicting the majority class while missing the cases that matter most. Imbalanced-learning research recommends metrics that focus on minority-class recovery and error trade-offs rather than overall hit rate [1]. Precision-recall analysis is particularly informative for skewed binary classification [28]. This study therefore uses AUPRC, precision at an investigation budget, recall at an investigation budget, calibration, and operating-point analysis.

Evaluation design is also a leakage problem: if future information leaks into training features, graph neighborhoods, or validation folds, an AML model can appear far stronger than it would be at deployment time [4]. The main Elliptic experiment uses temporal train-validation-test separation, leakage-aware feature construction, and paired comparisons on the same test observations. Calibration is included because risk scores are triage tools and probability reliability is distinct from discrimination [29]. Explainability adds another governance challenge for graph models, since explanations may involve features, nodes, edges, paths, and subgraphs and still lack a widely accepted evaluation standard [30]. Bootstrap intervals [31] and paired bootstrap comparisons are used to characterize

uncertainty, with the caveat that graph dependence limits purely observation-level inference.

Related applications of AI in financial decision-making include large language model applications to financial fraud and customer-facing services [32], predictive-analytics frameworks for financial risk classification under class imbalance [33], and explainable graph-learning approaches to detection tasks that share this study's concern with interpretability under skewed labels [34]. The present study extends this line of work by focusing specifically on the graph-versus-non-graph model-selection question for cryptocurrency AML rather than on a single downstream application or architecture.

## 3. Materials and Methods

### 3.1. Elliptic Transaction-Level Dataset

The Elliptic dataset contains 203,769 Bitcoin transaction nodes and 234,355 directed edges [2]. Of these nodes, 4,545 are labeled illicit, 42,019 are labeled licit, and 157,205 remain unlabeled. The supervised classification sample is limited to the 46,564 labeled nodes. After removing transaction identifiers, time variables, and labels from the predictors, the modeling table contains 165 anonymized transaction features, and time steps run from 1 to 49.

Unknown labeled nodes are not treated as licit negatives. They are simply excluded from supervised metrics, though the graph retains them for topology and message passing. This avoids the mistake of treating missing labels as evidence of benign activity.

### 3.2. Elliptic2 Subgraph-Level Dataset

Elliptic2 shifts the prediction unit from a transaction node to a connected component or subgraph [3]. The experiment uses the full Elliptic2 dataset, including all 121,810 labeled subgraphs. The label table contains 2,763 positive subgraphs, giving a positive rate of 0.0227. Because the background-node feature file is large, it was read in chunks, and 444,521 background-node rows were retained and aggregated. The final subgraph modeling table contains 140 features, combining structural fields with aggregated background-node features. Identifier-like and label-like fields are removed to reduce leakage risk.

The constructed Elliptic2 modeling table did not contain a usable temporal variable. The subgraph experiment uses stratified 70/15/15 train-validation-test splits and repeats the experiment over five random seeds. This design is less deployment-realistic than the Elliptic temporal split, but it still checks whether the full subgraph result is stable across partitions.

Table 1. Dataset and evaluation design summary

| Dataset | Unit | Obs. | Pos. | Pos. rate | Evaluation design |
|---|---|---|---|---|---|
| Elliptic | Txn node | 46,564 | 4,545 | 0.0976 | Temporal train-validation-test split |
| Elliptic2 | Subgraph | 121,810 | 2,763 | 0.0227 | Full subgraph benchmark with repeated stratified 70/15/15 splits |

### 3.3. Research Design

The empirical design has two levels and is intended to resemble deployment more closely than a single-model tuning exercise. Let $G = (V, E)$ denote a directed transaction graph, $X \in \mathbb{R}^{|V| \times d}$ denote node attributes, and $y_i \in \{0, 1\}$ denote the supervised AML label for labeled observation $i$, where 1 represents illicit or suspicious activity. Unknown labels are not used as supervised negative features. A model $f_\theta$ maps each observation to a risk score

$$\hat{p}_i = f_\theta(x_i, G) \in [0, 1], \tag{1}$$

interpreted as the estimated probability that observation $i$ belongs to the positive AML class. The central question is whether model families that explicitly use $G$ outperform non-GNN alternatives when validation respects time, class imbalance, and shifts in prediction granularity.

**Study 1: temporal transaction-node validation.** In the Elliptic study, each prediction concerns one transaction node. Each labeled node $i$ has feature vector $x_i$, time step $\tau_i$, and binary label $y_i$. The temporal split follows the chronology of the dataset, placing later time steps outside model fitting until validation or testing. The final temporal test set contains 8,841 labeled transactions, including 408 illicit ones. This design measures the operational situation in which an AML model is trained on past behavior and evaluated on future behavior.

**Study 2: full subgraph-level cross-granularity validation.** In the Elliptic2 study, the prediction unit is a connected component or subgraph $S_j = (V_j, E_j)$ rather than a single transaction. For each labeled subgraph, the analysis constructs a feature vector

$$z_j = \big[|V_j|,\ |E_j|,\ \rho_j,\ \log(1 + |V_j|),\ \log(1 + |E_j|), \\ \mathrm{Agg}_{v \in V_j}(b_v)\big], \tag{2}$$

where $\rho_j$ is a directed density-style structural feature, and Agg denotes mean, standard deviation, maximum, and minimum aggregations of selected background-node features $b_v$. Identifier and label fields are excluded to reduce leakage risk. The full Elliptic2 experiment

uses 121,810 labeled subgraphs present in the dataset. Because the constructed Elliptic2 feature table has no reliable temporal ordering variable, the main split is stratified 70/15/15 train-validation-test, and robustness is assessed through repeated stratified splits over different seeds $\{42, 123, 456, 789, 2024\}$.

**Cross-granularity interpretation.** The two studies are not treated as direct transfer learning. Elliptic is a transaction-node dataset, whereas Elliptic2 is a subgraph dataset. The aim is to assess model-family robustness and determine whether tree ensembles remain competitive across different AML prediction units and whether the evaluated GNNs dominate in the transaction-node setting.

## 3.4. System Architecture

Figure 1 summarizes the evaluation pipeline from raw public datasets to reported evidence. The workflow has seven layers: (i) dataset input validation for Elliptic and Elliptic2; (ii) schema validation and leakage checks; (iii) temporal splitting for Elliptic and repeated stratified splitting for Elliptic2; (iv) tabular, graph-handcrafted, spectral, node2vec, and subgraph-background feature construction; (v) training of classical models, tree ensembles, MLPs, and GNN model variants; (vi) bootstrap uncertainty, paired model comparisons, calibration, operating-point analysis, and repeated-split robustness evaluation; and (vii) generation of reported tables, figures, predictions, and model diagnostics. Fitting and prediction remain separated throughout the process, and transformations are fitted on training data and then applied to validation and test folds [4].

## 3.5. Feature Availability and Leakage Controls

Because these AML datasets are relational and only partly labeled, leakage control is part of the research design from the outset, not a post-processing step. Table 2 summarizes how each information source enters the learning pipeline. The main rule is simple: validation and test labels are never used to create supervised training signals. Graph topology and unsupervised embeddings are treated as observed dataset structure, while strict-temporal GNN variants are reported as a sensitivity check for future-edge availability.

## 3.6. Model Families

Each supervised model produces a score $\hat{p}_i \in [0, 1]$, used as the AML risk score for observation $i$. Class weighting is applied where it is available, and continuous variables are standardized only for scale-sensitive estimators. Classical estimators and preprocessing steps follow the scikit-learn interface where feasibly applicable [35].

**Tabular and neural baselines.** The tabular baseline set includes logistic regression [36], Random Forest [10], Extra Trees [11], HistGradientBoosting [12], XGBoost [13], and a feed-forward multilayer perceptron (MLP) [37]. Logistic regression provides a linear probabilistic reference. The tree ensembles and boosting models serve as competitive nonlinear references for anonymized transaction features. The MLP separates generic neural function approximation from graph-specific message passing.

**Graph-derived feature baselines.** The graph-feature ablations use degree, temporal edge counts, PageRank [38], core-style summaries [39], spectral embeddings, and node2vec-style random-walk embeddings [21]. These features are evaluated alone and in combination with the original tabular features to test whether explicit graph summaries add predictive value.

**Graph neural models.** The evaluation includes GCN [15], GraphSAGE [16], residual GCN, and a lightweight GAT-style graph-based attention model [17]. These models use message passing over the observed transaction graph and output node-level risk scores. In the Elliptic experiment, GraphSAGE is the strongest standard GNN baseline.

**Strict-temporal GNN variants.** The strict-temporal GNN check uses three horizon-specific adjacency matrices. For a horizon $h$, an edge $(u, v)$ is retained only when both endpoint time steps satisfy $\tau_u \leq h$ and $\tau_v \leq h$. The training adjacency uses $h = \max(\mathcal{T}_{\text{train}})$, the validation adjacency uses $h = \max(\mathcal{T}_{\text{val}})$, and the test adjacency uses $h = \max(\mathcal{T}_{\text{test}})$. The model is optimized only with the training mask and the training-horizon adjacency. Validation AUPRC for early stopping is computed with the validation-horizon adjacency, and final test probabilities are computed with the test-horizon adjacency. Unknown-label nodes remain in the message-passing graph when their time steps are available at the relevant horizon, but their labels never enter the supervised loss or metric computation. The sparse GCN and GraphSAGE adjacencies are still symmetrized, so this protocol is best interpreted as a leakage-sensitivity check rather than as a fully causal directed transaction-flow model.

## 3.7. Implementation Configuration

The experiments used all available labeled observations, with no subsampling. The global random seed was 42, the tree ensemble size was 350, the primary top-K budget was $K = \lceil 0.05n \rceil$, and uncertainty intervals used 1,000 bootstrap resamples. The Elliptic2 repeated-split

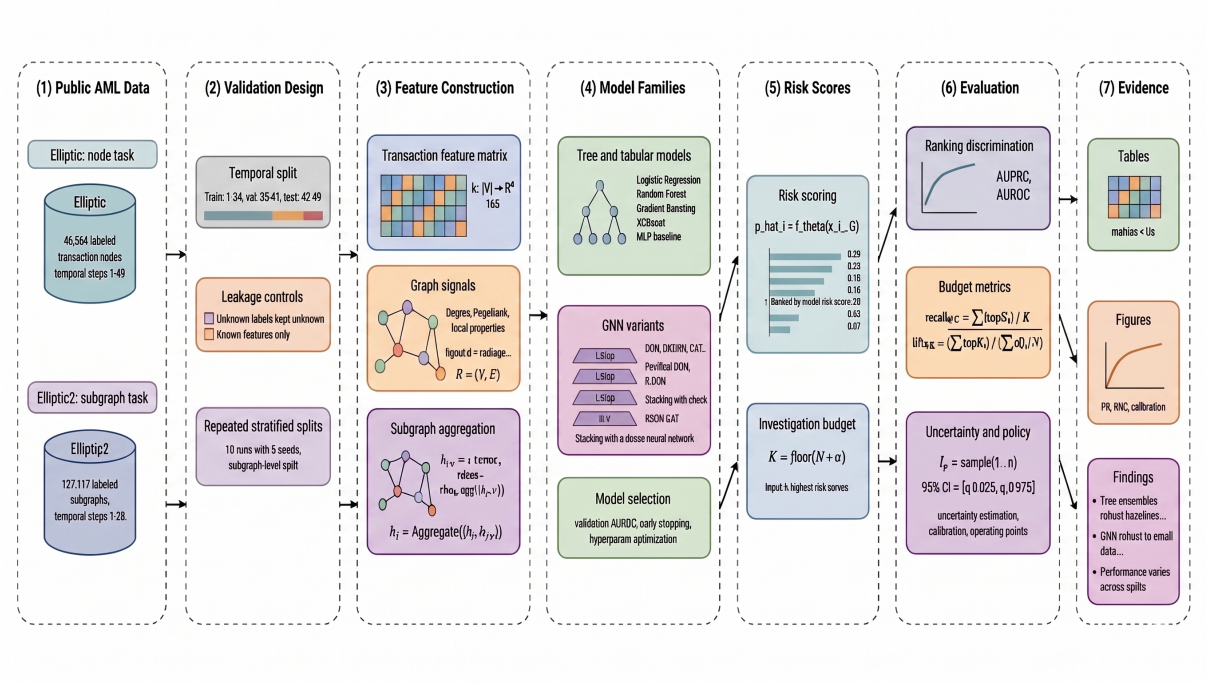



**Figure 1.** System architecture of the cryptocurrency AML evaluation pipeline. The diagram traces data ingestion, validation, feature construction, model-family training, statistical evaluation, and final result generation

Table 2. Feature availability and leakage controls used in the evaluation pipeline

| Component | Availability rule | Leakage-control implication |
|---|---|---|
| The Elliptic tabular features | Transaction identifier, time step and class label are separated from the 165 anonymized modeling features. Scaling is fitted on the training period and then applied to validation and test periods. | Test labels and test-period distributional statistics do not enter the fitted scaler or supervised estimators. |
| Graph handcrafted features | Degree, temporal edge counts, PageRank, and core-style structural summaries use graph structure. Neighbor-label risk features are computed with training labels only. | Validation and test labels are not used as neighbor labels and unknown labels are retained as unknown rather than redefined as licit. |
| Spectral and node2vec embeddings | Spectral embeddings and bounded node2vec-style embeddings are learned without class labels and are evaluated as ablation feature sets. | Embeddings test unsupervised topology value but are not used as evidence of causal laundering mechanisms. |
| Standard GNN adjacency | GCN, GraphSAGE, residual GCN, and GAT-style models use symmetrized observed edges with unknown nodes retained for message passing. | This is the common graph-learning evaluation condition and the paper reports diagnostics because symmetrization and future-edge availability can affect AML claims. |
| Strict-temporal GNN check | Split-specific adjacency matrices are constructed for strict-temporal GCN and GraphSAGE scoring. | This sensitivity analysis tests whether GNN performance depends on graph information unavailable at the evaluated temporal horizon. |
| The Elliptic2 subgraph features | Identifier-like and label-like fields are excluded and selected background-node features are chunk-read and aggregated to subgraph level. Scaling is fitted on the Elliptic2 training fold only. | The subgraph experiment is framed as cross-granularity robustness, not direct model transfer or a second temporal deployment test. |

analysis used fixed seeds $\{42, 123, 456, 789, 2024\}$. The hyperparameter configuration is recorded below.

**Classical Elliptic models.** Logistic regression used 1,000 iterations, balanced class weights, and the `lbfgs` solver. The tree-based Random Forest and Extra Trees models used 350 trees, unrestricted full-run depth, `min_samples_leaf=2`, and balanced class weighting. HistGradientBoosting used 250 iterations, learning rate 0.06, 31 leaf nodes, and

$L_2$ regularization 0.05. The multilayer perceptron (MLP) used hidden layers $(64, 32)$, $\alpha = 10^{-4}$, learning rate $10^{-3}$, 160 iterations, early stopping, and validation fraction 0.15. XGBoost used 450 trees, depth 4, learning rate 0.05, 0.85 row and column subsampling, logistic objective, log-loss evaluation, histogram tree construction, and positive-class scaling from the training fold.

**Graph-derived features and embeddings.** The spectral graph embeddings used 32 dimensions with truncated SVD and 7 iterations. Node2vec-style embeddings used at most 60,000 start nodes, one walk per node, walk length 12, embedding dimension 32, and 5 Word2Vec epochs.

**GNN models.** Graph-based GCN, GraphSAGE, and residual GCN used hidden dimension 96 and dropout 0.35. GATLite used hidden dimension 48, and GNNs used AdamW with learning rate $10^{-3}$, weight decay $10^{-4}$, weighted binary cross-entropy, validation-AUPRC monitoring, and early stopping with patience 15 over a maximum of 100 epochs. The strict-temporal GCN and strict-temporal GraphSAGE sensitivity analyses used the same optimizer and loss with a 60-epoch cap. The multi-seed neural robustness analysis used seeds $\{42, 123, 456, 789, 2024\}$ and a 40-epoch cap.

**Elliptic2 models.** The Elliptic2 main and repeated-split models reused the same 350-tree Random Forest and Extra Trees settings. Elliptic2 HistGradientBoosting used 180 iterations and learning rate 0.06 and Elliptic2 MLP used hidden layers $(64, 32)$, 120 maximum iterations, and early stopping. Elliptic2 XGBoost used 220 trees, depth 4, learning rate 0.05, subsample 0.85, column subsample 0.85, logistic objective, log-loss evaluation, histogram tree construction, and split-specific positive-class scaling.

## 3.8. Metrics

Let $TP$, $FP$, $TN$, and $FN$ denote true positives, false positives, true negatives, and false negatives at threshold $c$, with $\hat{y}_i(c) = \mathbb{1}\{\hat{p}_i \geq c\}$. The analysis reports accuracy, precision, recall, F1, AUROC [40], AUPRC [28], Brier score [41], and investigation-budget metrics. Accuracy is included for completeness but is not treated as the primary metric because both datasets are highly imbalanced.

The threshold metrics are

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN}, \\ \text{Precision} &= \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}, \end{aligned} \tag{3}$$

with $F_1$ defined as the harmonic mean of precision and recall. AUPRC is the primary ranking metric because precision-recall analysis is more informative than accuracy under rare-event class imbalance.

For the operational budget analysis, the top-$K$ set $\pi_K$ contains the $K = \lceil 0.05n \rceil$ highest-scoring observations. Precision@K and Recall@K are computed as

$$\text{Precision@K} = \frac{1}{K} \sum_{i \in \pi_K} y_i, \qquad \text{Recall@K} = \frac{\sum_{i \in \pi_K} y_i}{\sum_{i=1}^{n} y_i}. \tag{4}$$

Calibration is summarized with Brier score and calibration deciles. Operating-point tables use thresholds selected on validation predictions and then applied only to the test set covering the default 0.50 threshold, validation maximum-F1, top-5% budget and precision-constrained thresholds.

## 3.9. Statistical Analysis

The uncertainty analysis uses the test-set predictions produced by each fitted model.

**Bootstrap intervals for metrics.** For a test set of size $n$, bootstrap index vectors are drawn as

$$I_b = (i_{b1}, \ldots, i_{bn}), \qquad i_{bk} \sim \text{Uniform}\{1, \ldots, n\}, \tag{5}$$

with replacement for $b = 1, \ldots, B$, where $B = 1{,}000$ in the main analysis. For metric $m$, the bootstrap statistic is

$$\hat{m}^{(b)} = m\left(\{(y_i, \hat{p}_i) : i \in I_b\}\right). \tag{6}$$

The 95% interval is the empirical percentile interval

$$\left[ Q_{0.025}\left(\hat{m}^{(1:B)}\right), Q_{0.975}\left(\hat{m}^{(1:B)}\right) \right], \tag{7}$$

following the bootstrap logic of [31].

**Paired bootstrap model comparisons.** For two models $A$ and $B$ evaluated on the same test observations, the paired bootstrap computes

$$\Delta^{(b)} = m\left(y_{I_b}, \hat{p}_{A,I_b}\right) - m\left(y_{I_b}, \hat{p}_{B,I_b}\right). \tag{8}$$

The reported effect size is

$$\widehat{\Delta} = m(y, \hat{p}_A) - m(y, \hat{p}_B), \tag{9}$$

with a percentile bootstrap interval from $\Delta^{(1:B)}$. We report effect estimates and percentile intervals rather than formal p-values, because observation-level bootstrap signs can be overinterpreted when graph-connected observations are dependent. The main prespecified contrast is Extra Trees versus GraphSAGE on Elliptic AUPRC, since it directly tests the paper's central graph-AI claim. These bootstrap quantities are descriptive conditional intervals over the labeled test

observations. They may understate uncertainty when graph-connected observations are statistically dependent but they do not account for label incompleteness, future distribution shift, or alternative graph construction choices.

**Repeated Elliptic2 split robustness.** For Elliptic2, the full subgraph experiment is repeated across five fixed seeds. For each seed $s$, a stratified 70/15/15 split is generated, and the scaler is fitted on the training fold only. Each model family is retrained from scratch. For metric $m$ and $R = 5$ repeated splits, the repeated-split mean and standard deviation are

$$\bar{m} = \frac{1}{R}\sum_{r=1}^{R} m_r, \qquad s_m = \sqrt{\frac{1}{R-1}\sum_{r=1}^{R}(m_r - \bar{m})^2}. \quad (10)$$

The analysis reports an approximate repeated-split interval

$$\bar{m} \pm 1.96\frac{s_m}{\sqrt{R}}. \quad (11)$$

This interval summarizes split-to-split stability, not observation-level uncertainty.

**Interpretation rule.** The statistical analysis supports model-family conclusions. It is not used to claim causal feature effects when AUPRC, Recall@K, calibration, and operating-point evidence point in different directions. The interpretation prioritizes AUPRC and investigation-budget metrics because they most closely match the needs of rare-event AML triage.

## 4. Results

### 4.1. Evaluation Set Audit

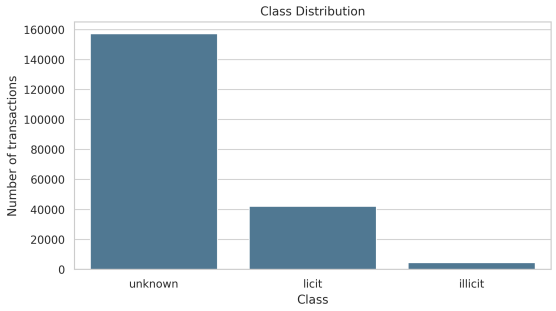
The Elliptic audit confirms 2 03,769 t ransaction rows in the node-feature and class files, 2 34,355 e dge rows, and 165 original transaction features. Because 157,205 nodes are unlabeled, the supervised Elliptic evaluation is restricted to 46,564 labeled transactions, of which 42,019 are licit and 4,545 are illicit.

Table 3. Elliptic temporal split used for the primary evaluation

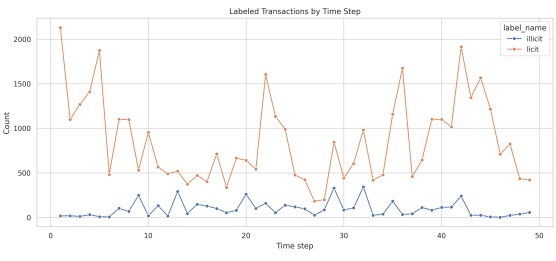
| Split | Trans. | Licit | Illicit | Rate | Time steps |
|---|---|---|---|---|---|
| Train | 29,894 | 26,432 | 3,462 | 0.116 | 1–34 |
| Val. | 7,829 | 7,154 | 675 | 0.086 | 35–41 |
| Test | 8,841 | 8,433 | 408 | 0.046 | 42–49 |

The temporal split creates a noticeably harder test period than the training period. The illicit rate falls from 11.6% in training to 4.6% in testing, so accuracy alone gives an incomplete and potentially misleading picture of model usefulness. For that reason, the main results emphasize AUPRC and investigation-budget recall alongside accuracy. For the Elliptic temporal test set, the top-5% investigation budget is $K = 443$.

Under random ranking evaluation, expected AUPRC is approximately the positive-class prevalence in the evaluated set. The leading Elliptic temporal-test model improves substantially over the 0.0461 test-set prevalence baseline. Extra Trees reaches AUPRC 0.5699, approximately 12.35 times the prevalence baseline. In the Elliptic2 subgraph test set, Random Forest reaches AUPRC 0.4940 against a 0.0227 prevalence baseline, approximately 21.75 times the baseline. These ratios place ranking performance in context of severe class imbalance, while leaving operating-point behavior as a separate deployment question.




(a) Class-label distribution.




(b) Labeled transactions over time.

**Figure 2.** Elliptic data context: severe label imbalance and temporal variation in labeled transactions

### 4.2. Elliptic Temporal Evaluation

Table 4 reports the leading Elliptic temporal-test models. Extra Trees achieves the highest AUPRC, reaching 0.570 with a 95% bootstrap interval of [0.525, 0.613], and obtains AUROC 0.869 with interval [0.851, 0.886]. HistGradientBoosting posts slightly higher AUROC and Recall@K, but lower AUPRC and lower default-threshold precision. XGBoost matches Extra Trees on Recall@K but falls below it on AUPRC and accuracy. The best feature-augmented model, Extra Trees with tabular, graph, and node2vec features, reaches AUPRC 0.553 with interval [0.507, 0.599]. Thus, the additional graph-derived inputs do not improve on the original-feature Extra Trees model.

Table 4. Leading Elliptic temporal-test results and 95% bootstrap intervals

| Model | Family | Accuracy | Precision | Recall | AUROC | AUPRC | Recall @K |
|---|---|---|---|---|---|---|---|
| Extra Trees | Tabular | 0.974 | 0.923 | 0.473 | 0.869 [0.851, 0.886] | 0.570 [0.525, 0.613] | 0.498 |
| HistGradientBoosting | Tabular | 0.963 | 0.630 | 0.480 | 0.872 [0.857, 0.886] | 0.560 [0.512, 0.602] | 0.505 |
| Extra Trees + graph + node2vec | Ablation | 0.974 | 0.959 | 0.453 | 0.852 [0.832, 0.872] | 0.553 [0.507, 0.599] | 0.485 |
| XGBoost | Tabular | 0.958 | 0.546 | 0.490 | 0.863 [0.846, 0.879] | 0.553 [0.507, 0.598] | 0.498 |
| Extra Trees + graph + spectral | Ablation | 0.974 | 0.954 | 0.461 | 0.846 [0.826, 0.864] | 0.550 [0.504, 0.595] | 0.485 |
| Random Forest | Tabular | 0.974 | 0.923 | 0.468 | 0.850 [0.833, 0.867] | 0.543 [0.497, 0.587] | 0.483 |

## 4.3. Temporal Versus Random Split Robustness

The random-split robustness analysis is not the primary research design, but it illustrates how much easier the benchmark becomes once temporal ordering is removed. Under the random split, the top tree and boosting models achieve AUPRC above 0.97 and Precision@K equal to 1.000. Under the temporal split, the best AUPRC is 0.570. That gap is substantial and suggests that randomly shuffled evaluation would significantly overstate how well models perform on later, unseen transactions.

## 4.4. Graph Features and GNNs

The feature registry covers eight feature settings: 165 original tabular features, 24 handcrafted graph features, 32 spectral embedding features, 32 node2vec features, and their combinations. Adding graph-derived inputs helps some lower-performing settings but does not change the overall ranking. The best feature-augmented model is Extra Trees with tabular, handcrafted graph, and node2vec features (AUPRC 0.553). Graph-only feature sets fall well short of the transaction-feature models. Graph-handcrafted variants reach AUPRC near 0.066–0.068, and standalone spectral or node2vec embeddings also remain far below the original transaction-feature models. These results indicate that graph information adds useful context, but it is not sufficient on its own and does not displace the anonymized Elliptic transaction features.

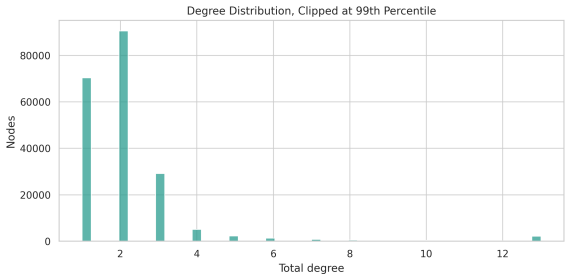



**Figure 3.** Elliptic transaction-graph degree distribution, clipped at the 99th percentile

Table 6 compares the leading model from each family. The best evaluated standard GNN baseline is GraphSAGE, with AUPRC 0.311 (95% bootstrap interval [0.267, 0.354]) and Recall@K 0.309. Strict-temporal GraphSAGE improves to AUPRC 0.381 (95% bootstrap interval [0.333, 0.426]) and Recall@K 0.395, but it remains well below Extra Trees. The standard GraphSAGE operating point is also difficult to use in practice at threshold 0.50, recall is 0.904 but precision is only 0.056, meaning that most default-threshold alerts are false positives.

The multi-seed neural summary points in the same direction. Across all the five seeds, GraphSAGE has mean AUPRC 0.318 with standard deviation 0.018, while GCN has mean AUPRC 0.108 with standard deviation 0.006. The MLP outperforms both GNNs on AUPRC, with mean AUPRC 0.379, but it still falls short of the leading tree ensemble models.

The paired bootstrap comparisons in Table 7 quantify the model-family contrast on identical temporal-test observations. Extra Trees exceeds GraphSAGE by 0.259 AUPRC, with a 95% interval of [0.222, 0.297] over 1,000 paired bootstrap resamples. The same Extra Trees advantage holds for AUROC, Precision@K, Recall@K, and accuracy. Against XGBoost, the Extra Trees AUPRC advantage is much narrower, 0.0166 with

Table 5. Random-split robustness compared with the temporal Elliptic evaluation

| Model | Random AUROC | Random AUPRC | Temporal AUPRC | Random Recall@K |
|---|---|---|---|---|
| XGBoost | 0.996 | 0.984 | 0.553 | 0.513 |
| HistGradientBoosting | 0.996 | 0.982 | 0.560 | 0.513 |
| Random Forest | 0.996 | 0.981 | 0.543 | 0.513 |
| Extra Trees | 0.995 | 0.973 | 0.570 | 0.513 |
| MLP | 0.983 | 0.943 | 0.446 | 0.510 |
| Logistic Regression | 0.964 | 0.757 | 0.199 | 0.440 |

Table 6. Best model by family on the Elliptic temporal test set

| Family | Model | Accuracy | Precision | Recall | AUPRC | Recall@K |
|---|---|---|---|---|---|---|
| Tabular ML | Extra Trees | 0.974 | 0.923 | 0.473 | 0.570 [0.525, 0.613] | 0.498 |
| Feature ablation | Extra Trees + graph + node2vec | 0.974 | 0.959 | 0.453 | 0.553 [0.507, 0.599] | 0.485 |
| Graph AI | GraphSAGE | 0.294 | 0.056 | 0.904 | 0.311 [0.267, 0.354] | 0.309 |
| Strict-temporal graph AI | Strict-temporal GraphSAGE | 0.836 | 0.153 | 0.561 | 0.381 [0.333, 0.426] | 0.395 |

interval [0.0032, 0.0316]. The XGBoost comparison does not show a meaningful separation in AUROC or investigation-budget metrics. The AUROC interval crosses zero, and the Precision@K and Recall@K differences are zero.

## 4.5. Operating-Point Analysis

AML deployment requires explicit threshold choices. At the default 0.50 threshold, Extra Trees identifies 193 of 408 illicit test transactions while producing 16 false positives. This corresponds to precision 0.923 (95% bootstrap interval [0.889, 0.959]), recall 0.473 (95% bootstrap interval [0.424, 0.519]), and 0.083 false positives per true positive (95% bootstrap interval [0.043, 0.125]). HistGradientBoosting and XGBoost identify slightly more positives at the same threshold, but at the cost of substantially lower precision and a higher false-positive burden.

The operating-point table exposes the limitation of accuracy as a headline metric. XGBoost has the highest default-threshold true-positive count among these three models, but it generates 166 false positives and 0.830 false positives per true positive. The top-5% budget policies sharply reduce false positives while also lowering recall. The appropriate operating point ultimately depends on the compliance team's investigation capacity and its tolerance for missed illicit transactions.

## 4.6. Calibration and Feature Diagnostics

The calibration and feature-diagnostic analyses add two important findings. First, the best AUPRC model is not the lowest-Brier model. Extra Trees has Brier score 0.0287, whereas the Extra Trees tabular, graph, and node2vec ablation has Brier score 0.0279 but lower AUPRC. Second, the feature-importance tables are useful for model diagnostics but not for economic interpretation, because Elliptic features are anonymized. The strongest Extra Trees built-in importance is assigned to feature `f_052` (importance 0.0571). The largest permutation AUPRC drops are observed for `f_054` (0.0448, standard deviation 0.0085), `f_051` (0.0440, standard deviation 0.0087), and `f_052` (0.0219, standard deviation 0.0056).

Table 9 reports Brier scores for representative Elliptic temporal-test models. Probability loss and ranking quality are related but meaningfully distinct. GraphSAGE has higher AUPRC than logistic regression, but its Brier score is worse on the temporal test set. Similarly, MLP has a better Brier score than XGBoost but a lower AUPRC.

## 4.7. Elliptic2 Full Subgraph Validation

The Elliptic2 analysis uses all 121,810 labeled subgraphs and constructs 140 subgraph-level features from connected-component labels, membership information, edge-derived structural information, and aggregated background-node features. The construction table reports 444,521 selected nodes, 444,521 retained background-node rows, 32 background feature columns, and 367,137 loaded edge rows. With a positive rate of 0.0227, Elliptic2 is more imbalanced than the Elliptic supervised transaction-node benchmark.

Table 7. Paired bootstrap model comparisons on the Elliptic temporal test set

| Model A | Model B | Metric | A | B | Difference | 95% interval |
|---|---|---|---|---|---|---|
| Extra Trees | GraphSAGE | AUPRC | 0.570 | 0.311 | 0.259 | [0.222, 0.297] |
| Extra Trees | GraphSAGE | AUROC | 0.869 | 0.784 | 0.086 | [0.064, 0.108] |
| Extra Trees | GraphSAGE | Recall@K | 0.498 | 0.309 | 0.189 | [0.149, 0.226] |
| Extra Trees | GraphSAGE | Precision@K | 0.458 | 0.284 | 0.174 | [0.135, 0.214] |
| Extra Trees | GraphSAGE | Accuracy | 0.974 | 0.294 | 0.680 | [0.668, 0.690] |
| Extra Trees | XGBoost | AUPRC | 0.570 | 0.553 | 0.0166 | [0.0032, 0.0316] |
| Extra Trees | XGBoost | AUROC | 0.869 | 0.863 | 0.0067 | [-0.0070, 0.0196] |
| Extra Trees | XGBoost | Recall@K | 0.498 | 0.498 | 0.000 | [-0.016, 0.017] |
| Extra Trees | XGBoost | Precision@K | 0.458 | 0.458 | 0.000 | [-0.016, 0.016] |
| Extra Trees | XGBoost | Accuracy | 0.974 | 0.958 | 0.016 | [0.014, 0.019] |

Table 8. Selected operating policies on the Elliptic temporal test set

| Model | Policy | Threshold | Precision | Recall | F1 | TP | FP/TP |
|---|---|---|---|---|---|---|---|
| Extra Trees | Default 0.50 | 0.500 | 0.923 | 0.473 | 0.626 | 193 | 0.083 |
| Extra Trees | Validation max F1 | 0.519 | 0.932 | 0.471 | 0.625 | 192 | 0.073 |
| Extra Trees | Top-5% budget | 0.848 | 0.994 | 0.397 | 0.567 | 162 | 0.006 |
| HistGradientBoosting | Default 0.50 | 0.500 | 0.630 | 0.480 | 0.545 | 196 | 0.587 |
| HistGradientBoosting | Validation max F1 | 0.861 | 0.914 | 0.471 | 0.621 | 192 | 0.094 |
| HistGradientBoosting | Top-5% budget | 0.999 | 1.000 | 0.328 | 0.494 | 134 | 0.000 |
| XGBoost | Default 0.50 | 0.500 | 0.546 | 0.490 | 0.517 | 200 | 0.830 |
| XGBoost | Validation max F1 | 0.784 | 0.836 | 0.475 | 0.606 | 194 | 0.196 |
| XGBoost | Top-5% budget | 0.992 | 0.981 | 0.373 | 0.540 | 152 | 0.020 |

Table 9. Calibration-oriented Brier scores on the Elliptic temporal test set. Intervals are 95% bootstrap intervals

| Model | Family | Brier score | PR AUC |
|---|---|---|---|
| Extra Trees | Tabular tree ensemble | 0.0287 [0.0262, 0.0315] | 0.570 |
| MLP | Tabular neural baseline | 0.0319 [0.0285, 0.0353] | 0.446 |
| HistGB | Boosted tree ensemble | 0.0332 [0.0300, 0.0365] | 0.560 |
| XGBoost | Boosted tree ensemble | 0.0357 [0.0325, 0.0391] | 0.553 |
| GCN | Graph neural baseline | 0.1086 [0.1049, 0.1125] | 0.294 |
| Strict-temp. GraphSAGE | Leakage-sensitivity GNN | 0.1103 [0.1058, 0.1149] | 0.381 |
| Logistic Reg. | Linear baseline | 0.1993 [0.1928, 0.2057] | 0.199 |
| GraphSAGE | Graph neural baseline | 0.3365 [0.3328, 0.3401] | 0.311 |

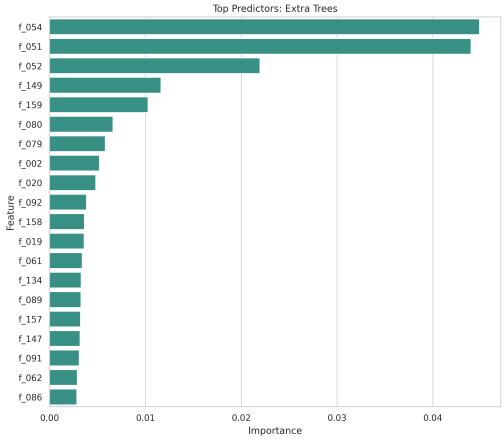



**Figure 4.** Top anonymized predictors for the Extra Trees model. The figure is used as a model diagnostic, not as a causal explanation of laundering behavior

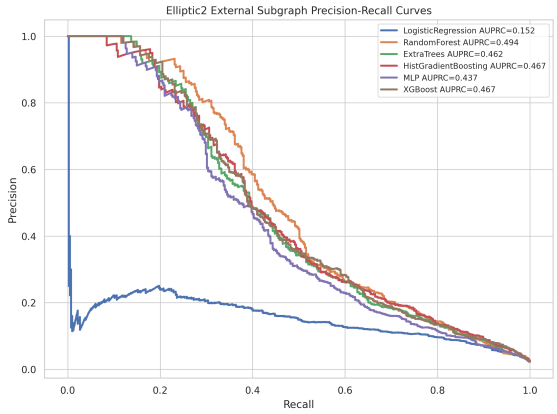
The main Elliptic2 split has 85,267 training, 18,271 validation, and 18,272 test subgraphs, with $K = 914$ for the top-5% investigation budget. Table 10 reports the single-split results. Random Forest achieves the highest AUPRC, 0.494 (95% bootstrap interval [0.444, 0.544]), and the lowest Brier score among the main Elliptic2 models, 0.0156. HistGradientBoosting has the highest AUROC, 0.929, and XGBoost has the highest Recall@K, 0.607. Both boosting models produce far lower default-threshold precision than Random Forest because they classify many more subgraphs as illicit at threshold 0.50.

## 4.8. Elliptic2 Repeated-Split Robustness and Cross-Granularity Synthesis

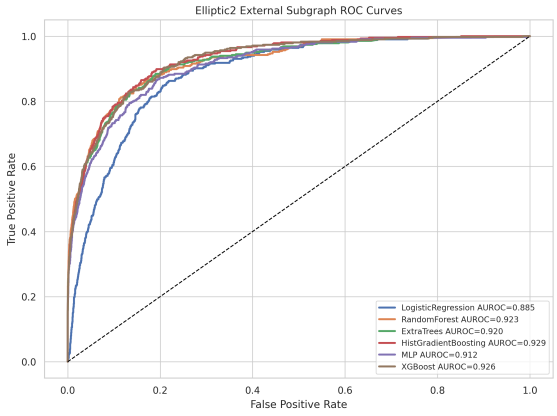
The Elliptic2 experiment is repeated over five stratified random splits as a descriptive stability check. Random Forest remains the strongest model by mean AUPRC, 0.488, with a split-to-split variability interval of [0.476, 0.500]. HistGradientBoosting has the highest mean AUROC, 0.929, and the highest mean Recall@K, 0.607,

Table 10. Full Elliptic2 subgraph classification results, main split. Intervals are 95% bootstrap intervals

| Model | Accuracy | Precision | Recall | AUROC | AUPRC | Precision @K | Recall @K |
|---|---|---|---|---|---|---|---|
| Random Forest | 0.982 | 0.929 | 0.219 | 0.923 [0.909, 0.936] | 0.494 [0.444, 0.544] | 0.270 | 0.595 |
| XGBoost | 0.858 | 0.121 | 0.831 | 0.926 [0.913, 0.940] | 0.467 [0.419, 0.518] | 0.276 | 0.607 |
| HistGradient Boosting | 0.864 | 0.125 | 0.834 | 0.929 [0.917, 0.941] | 0.467 [0.420, 0.515] | 0.268 | 0.590 |
| Extra Trees | 0.981 | 0.880 | 0.212 | 0.920 [0.907, 0.935] | 0.462 [0.412, 0.511] | 0.267 | 0.588 |
| MLP | 0.982 | 0.777 | 0.260 | 0.912 [0.899, 0.927] | 0.437 [0.389, 0.487] | 0.258 | 0.569 |
| Logistic Regression | 0.794 | 0.087 | 0.848 | 0.885 [0.871, 0.900] | 0.152 [0.130, 0.177] | 0.182 | 0.400 |




(a) Elliptic2 precision-recall curves




(b) Elliptic2 ROC curves

**Figure 5.** Elliptic2 discrimination curves from the full subgraph benchmark

but its mean threshold precision is 0.127 compared with 0.908 for Random Forest. XGBoost reaches mean AUPRC 0.467, Extra Trees 0.461, MLP 0.423, and logistic regression 0.145. The repeated splits support the main Elliptic2 result: tree ensembles lead on AUPRC,

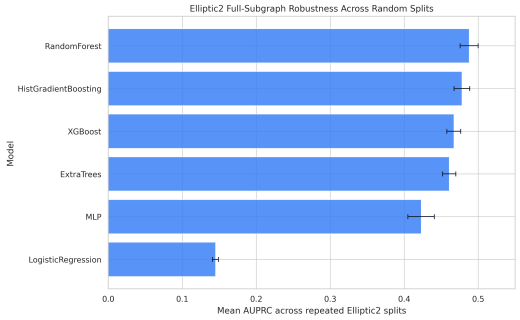



**Figure 6.** Repeated-split Elliptic2 AUPRC stability across fixed random seeds

while operating-point choices still involve a precision-recall trade-off. B ecause o nly fi ve se eds ar e us ed, the repeated-split interval is best read as descriptive split-to-split variability rather than as a formal inferential interval.

The cross-granularity synthesis identifies E xtra Trees as the best Elliptic transaction-node model and Random Forest as the best Elliptic2 subgraph model. This is not a direct model-transfer result, because Elliptic and Elliptic2 use different prediction units. Nor is it a second GNN comparison, because the Elliptic2 experiment evaluates subgraph-level classical, neural, and tree models rather than graph neural models. The synthesis is that, across two publi cryptocurrency AML prediction units, tree ensembles remain necessary baselines and the evaluated Elliptic GNNs do not automatically dominate under deployment-facing evaluation.

## 5. Discussion

### 5.1. Graph Structure Matters, but the Evaluated GNNs Do Not Dominate

The results challenge a common assumption in cryptocurrency AML research. Because transactions

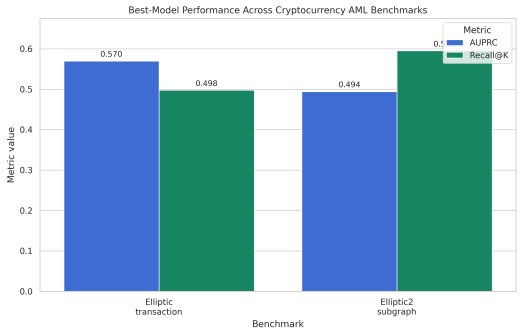



**Figure 7.** Best-model performance across the Elliptic transaction-node benchmark and the Elliptic2 subgraph benchmark

form a graph, GNNs are sometimes treated as the default modeling endpoint, an assumption not supported by these findings. On the Elliptic temporal evaluation, the leading standard GNN baseline falls substantially short of Extra Trees. Strict-temporal GraphSAGE improves over the standard GraphSAGE result, indicating that graph construction and temporal leakage controls matter, but it does not close the gap. The broader lesson is consistent with the graph ML literature: graph structure is valuable only when the model, labels, split design, and evaluation metric align with the deployment problem [27].

Several mechanisms may explain this pattern. The Elliptic transaction features are informative and include anonymized local and aggregated information, which tree ensembles appear to exploit more effectively than lightweight message-passing models. The graph also contains many unknown nodes, so neighborhoods can be noisy even when topology is retained. Common GNN implementations often symmetrize directed transaction flows, which may weaken AML-relevant money-flow directionality. Sparse illicit labels further limit the supervision available for neighborhood aggregation.

A further reason for the gap between Extra Trees and the standard GNN baselines may lie in how directionality is handled. The GCN, GraphSAGE, and GAT-style architectures in this study operate on a symmetrized adjacency structure and propagate information in both directions along an edge. Bitcoin transaction graphs, however, are directed multigraphs. An edge from node (i) to node (j) represents a specific transfer of value, and treating that edge as undirected discards the money-flow ordering that investigators rely on to trace layering and structuring patterns. Symmetrization can therefore erase exactly the directional information that distinguishes a source of funds from a destination, collapsing patterns that are meaningfully different from an AML perspective into the same local neighborhood representation. Egressy et al. show that standard message-passing GNNs are provably unable to recover certain directed transaction patterns unless they are augmented with direction-aware mechanisms such as port numbering, ego identifiers, and heterogeneous in/out message passing for directed multigraphs [24]. This suggests that part of the AUPRC gap reported in Table 6 may reflect an architectural mismatch rather than an inherent limitation of graph information: a direction-aware GNN could plausibly recover more of the money-flow signal than the symmetrized baselines evaluated here. Testing that hypothesis is a natural extension of this work rather than a claim made in the present study.

None of this constitutes evidence against graph learning as a whole. It is evidence against unqualified graph-AI claims that have not been tested against capable baselines. A more specialized temporal, directed, heterogeneous, or subgraph-native GNN might improve performance, but that claim has to be evaluated directly. The negative result here is an empirical finding from this evaluation design, not a theoretical impossibility.

## 5.2. Tree Ensembles Remain Important RegTech Baselines

The best Elliptic model is Extra Trees. On Elliptic2, Random Forest leads by both single-split and repeated-split AUPRC. These outcomes show that tree ensembles remain highly competitive in AML detection with anonymized but information-rich tabular features. From a governance perspective, this matters because tree ensembles are generally easier to train, deploy, compare, and monitor than complex GNN pipelines. The result also fits the broader statistical fraud-detection tradition, in which reliable baselines, threshold policy, and false-alert management are central to operational value [8].

This does not mean Random Forest or Extra Trees should always be the deployment choice for AML applications. It does suggest that they should be mandatory comparators in AML graph-learning studies. If a more complex graph model fails to outperform them under temporal or cross-granularity validation, the additional complexity needs clearer justification.

## 5.3. Model Governance and Interpretability

AML models are triage tools for investigators, not autonomous adjudication systems. A high-AUPRC model can still generate false alerts, miss illicit activity, or drift as laundering strategies evolve. For that reason, this paper reports operating points, calibration, and bootstrap intervals alongside rank-order metrics. The practical question is not just which model has the highest score, but whether that score can be translated into a defensible investigation policy and real-world implementation.

The findings also reinforce the case for model-governance discipline. When a complex model does

not clearly outperform a simpler one, the burden of explanation, monitoring, and validation becomes harder to justify. This is especially important in regulated or high-stakes domains where black-box complexity can create accountability problems [14]. Risk-management guidance similarly emphasizes documented validity, monitoring, and accountability for AI systems [42]. For graph models, the explanation challenge is further compounded by the fact that explanations may involve nodes, edges, paths, subgraphs, and features rather than only tabular covariates [30]. Complex graph AI should be compared with strong tree ensembles before deployment, not after the graph model has already been selected.

## 5.4. Evaluation Design Changes the Story

The central claim depends heavily on evaluation design. A temporal split is closer to deployment than a random split because future illicit behavior may differ from earlier behavior in ways a model has never seen. The analysis includes a random-split sensitivity analysis, but that is not the main design. The difference between random and temporal evaluation should be treated as part of model-risk assessment. Random splits can mix related or temporally adjacent observations across training and testing, which may inflate performance in relational data settings. The leakage literature is clear that predictive signals must be available at the time of decision and not merely present somewhere in the dataset [4].

Elliptic2 provides a complementary perspective. Its subgraph-level labels reflect the intuition that laundering may appear as a pattern rather than a single transaction. The full Elliptic2 experiment again favors tree ensembles by AUPRC. This cross-granularity evidence strengthens the main conclusion, while the difference in prediction units prevents direct transfer claims.

## 5.5. Practical Implications

For compliance teams and RegTech developers, the findings translate into a short set of practical guidelines:

1. Compare graph neural networks with tree ensembles before adopting them in the final solution.
2. Use AUPRC, Precision@K, Recall@K, calibration, and operating-point analysis as primary metrics.
3. Avoid selecting models based on only accuracy in rare-event AML settings.
4. Treat graph-AI outputs as triage signals rather than proof of illicit conduct.
5. Document temporal validation, false-positive trade-offs, and model-risk limitations before final deployment.

## 5.6. Limitations and Threats to Validity

The study has several limitations. First, some of the Elliptic features are anonymized, which limits economic interpretation and rules out causal claims about specific laundering mechanisms. Feature importance can identify which anonymized variables influence the model, but it cannot provide a substantive behavioral explanation. Second, unknown labels are not clean negatives and they are excluded from supervised evaluation but retained in graph structure where appropriate. This preserves topology, while acknowledging that neighborhoods may contain unlabeled behavior whose true compliance status is unknown.

A related question is whether the tree-ensemble advantage reflects the availability of Elliptic's anonymized handcrafted features rather than a general limitation of graph learning. Across the tabular-plus-graph ablations summarized in Table 6, neither node2vec nor spectral embedding augmentation improved on the tabular-only Extra Trees model, suggesting that the anonymized features already capture most of the exploitable signal in this dataset. This feature strength may blunt the potential advantage of a topology-only GNN. It remains possible that a GNN restricted to raw topology, with no anonymized node attributes, would close some of the gap because the tree ensemble would lose its main source of predictive signal. We did not run this feature-ablated comparison in the present study and therefore flag it explicitly as a limitation. The reported comparisons hold the feature budget roughly constant across model families and should not be read as evidence about which family would dominate in a topology-only regime. This is a natural direction for follow-up work.

Third, Elliptic and Elliptic2 use different prediction units. Elliptic is a transaction-node dataset, while Elliptic2 is a subgraph dataset. The cross-granularity result should be read as evidence about model-family robustness, not as direct model transfer. Fourth, the implemented GNNs are lightweight graph-based models. Specialized temporal, directed, heterogeneous, edge-feature, or subgraph-native architectures may perform better, but those possibilities require new experiments to test. Fifth, neither dataset contains the full KYC, exchange, sanctions, case-management, and investigator-feedback information used by real AML teams.

Sixth, bootstrap intervals quantify sampling uncertainty conditional on the observed labeled test sets, and they do not capture all sources of uncertainty from graph dependence, labeling, future distribution shift, or data-collection bias. Finally, Elliptic2 does not provide the same usable temporal split structure as Elliptic.

Because Elliptic2 lacks a usable temporal ordering variable, the repeated stratified splits used here can place structurally related subgraphs and background-node information on both sides of a given split. This

provides a substantially weaker leakage guard than the chronological separation enforced for Elliptic. As a result, the full Elliptic2 findings should be read as evidence of cross-granularity robustness under random partitioning, not as a second temporal deployment test. Any operational claim about how a subgraph-level model would perform on genuinely future laundering activity would require either recovering a usable temporal signal from the underlying blockchain timestamps or validating against a dataset release that preserves chronology at the subgraph level.

## 6. Conclusions

We set out to ask whether the graph-AI baselines outperform competitive non-GNN baselines for cryptocurrency AML under deployment-facing validation. In the Elliptic temporal experiment, Extra Trees achieves the highest AUPRC and materially exceeds GraphSAGE in paired bootstrap comparisons. On full Elliptic2, Random Forest achieves the highest single-split and repeated-split AUPRC among the evaluated subgraph-level models. The conclusion is not that graph information lacks value. Rather, graph structure alone does not justify graph neural network deployment without capable baselines, temporal validation and operational metrics.

For cryptocurrency AML research and real-world applications, future graph-AI studies should include tree baselines, temporal validation, cross-granularity validation, operational metrics, and uncertainty estimates. For financial institutions and RegTech teams, the practical message is straightforward: test model complexity before adopting it.

## References


[1] He H, Garcia EA. Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering. 2009;21(9):1263-84.

[2] Weber M, Domeniconi G, Chen J, Weidele DKI, Bellei C, Robinson T, et al. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. In: KDD '19 Workshop on Anomaly Detection in Finance. Anchorage, AK, USA; 2019. Available from: https://arxiv.org/abs/1908.02591.

[3] Bellei C, Xu M, Phillips R, Robinson T, Weber M, Kaler T, et al.. The Shape of Money Laundering: Subgraph Representation Learning on the Blockchain with the Elliptic2 Dataset; 2024. ArXiv preprint. Available from: https://arxiv.org/abs/2404.19109.

[4] Kaufman S, Rosset S, Perlich C, Stitelman O. Leakage in Data Mining: Formulation, Detection, and Avoidance. ACM Transactions on Knowledge Discovery from Data. 2012;6(4):15.

[5] Ron D, Shamir A. Quantitative Analysis of the Full Bitcoin Transaction Graph. In: Financial Cryptography and Data Security. vol. 7859 of Lecture Notes in Computer Science. Springer; 2013. p. 6-24.

[6] Meiklejohn S, Pomarole M, Jordan G, Levchenko K, McCoy D, Voelker GM, et al. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. In: Proceedings of the 2013 Internet Measurement Conference. ACM; 2013. p. 127-39.

[7] Alarab I, Prakoonwit S. Graph-Based LSTM for Anti-Money Laundering: Experimenting Temporal Graph Convolutional Network with Bitcoin Data. Neural Processing Letters. 2023;55:689-707.

[8] Bolton RJ, Hand DJ. Statistical Fraud Detection: A Review. Statistical Science. 2002;17(3):235-55.

[9] Lessmann S, Baesens B, Seow HV, Thomas LC. Benchmarking State-of-the-Art Classification Algorithms for Credit Card Fraud Detection. European Journal of Operational Research. 2016;249(2):666-81.

[10] Breiman L. Random Forests. Machine Learning. 2001;45(1):5-32.

[11] Geurts P, Ernst D, Wehenkel L. Extremely Randomized Trees. Machine Learning. 2006;63(1):3-42.

[12] Friedman JH. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics. 2001;29(5):1189-232.

[13] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016. p. 785-94.

[14] Rudin C. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. Nature Machine Intelligence. 2019;1:206-15.

[15] Kipf TN, Welling M. Semi-Supervised Classification with Graph Convolutional Networks. In: International Conference on Learning Representations; 2017. Available from: https://openreview.net/forum?id=SJU4ayYgl.

[16] Hamilton WL, Ying R, Leskovec J. Inductive Representation Learning on Large Graphs. In: Advances in Neural Information Processing Systems. vol. 30; 2017. Available from: https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba837a7ebea9-Abstract.html.

[17] Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph Attention Networks. In: International Conference on Learning Representations; 2018. Available from: https://openreview.net/forum?id=rJXMpikCZ.

[18] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems. 2021;32(1):4-24.

[19] Pareja A, Domeniconi G, Chen J, Ma T, Suzumura T, Kanezashi H, et al. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34; 2020. p. 5363-70.

[20] Rossi E, Chamberlain B, Frasca F, Eynard D, Monti F, Bronstein M. Temporal Graph Networks for Deep Learning on Dynamic Graphs; 2020.

[21] Grover A, Leskovec J. node2vec: Scalable Feature Learning for Networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge

Discovery and Data Mining; 2016. p. 855-64.
[22] Dou Y, Liu Z, Sun L, Deng Y, Peng H, Yu PS. Enhancing Graph Neural Network-Based Fraud Detectors against Camouflaged Fraudsters. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management; 2020. p. 315-24.
[23] Ma X, Wu J, Xue S, Yang J, Zhou C, Sheng QZ, et al. A Comprehensive Survey on Graph Anomaly Detection With Deep Learning. IEEE Transactions on Knowledge and Data Engineering. 2023;35(12):12012-38.
[24] Egressy B, von Niederhäusern L, Blanuša J, Altman E, Wattenhofer R, Atasu K. Provably Powerful Graph Neural Networks for Directed Multigraphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38; 2024. p. 11838-46.
[25] Lin Z, Luo Q, Wu D, Shen J, Li L, Nong X, et al. Detecting Illicit Transactions in Bitcoin: A Wavelet-Temporal Graph Transformer Approach for Anti-Money Laundering. Scientific Reports. 2026;16(1):1548.
[26] Li E, Chen M, Xiang S, Chen L. Graph Learning-Empowered Financial Fraud Detection: Progress and Future Directions. Intelligent Computing. 2025;4:0146.
[27] Hu W, Fey M, Zitnik M, Dong Y, Ren H, Liu B, et al. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In: Advances in Neural Information Processing Systems. vol. 33; 2020. p. 22118-33. Available from: https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.
[28] Saito T, Rehmsmeier M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. PLOS ONE. 2015;10(3):e0118432.
[29] Guo C, Pleiss G, Sun Y, Weinberger KQ. On Calibration of Modern Neural Networks. In: Proceedings of the 34th International Conference on Machine Learning. vol. 70 of Proceedings of Machine Learning Research. PMLR; 2017. p. 1321-30. Available from: https://proceedings.mlr.press/v70/guo17a.html.
[30] Yuan H, Yu H, Gui S, Ji S. Explainability in Graph Neural Networks: A Taxonomic Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2023;45(5):5782-99.
[31] Efron B, Tibshirani RJ. An Introduction to the Bootstrap. Chapman and Hall/CRC; 1994.
[32] Cao X, Li S, Katsikis V, Khan AT, He H, Liu Z, et al. Empowering Financial Futures: Large Language Models in the Modern Financial Landscape. EAI Endorsed Transactions on AI and Robotics. 2024;3(1).
[33] Hossan MZ, Riipa MB, Hossain MA, Dhar SR, Zaman AM, Hossain M, et al. AI-Powered Predictive Analytics for Financial Risk Management in U.S. Markets. EAI Endorsed Transactions on AI and Robotics. 2025 August;4(1). Available from: https://publications.eai.eu/index.php/airo/article/view/9532.
[34] Dang QV, Nguyen PL, Le D, Dinh MN. XHBot: eXplainable Heterophily-Aware Graph Neural Networks for Social Bot Detection. EAI Endorsed Transactions on AI and Robotics. 2026 July;5. Available from: https://publications.eai.eu/index.php/airo/article/view/12969.
[35] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825-30. Available from: https://www.jmlr.org/papers/v12/pedregosa11a.html.
[36] Cox DR. The Regression Analysis of Binary Sequences. Journal of the Royal Statistical Society: Series B (Methodological). 1958;20(2):215-32.
[37] Rumelhart DE, Hinton GE, Williams RJ. Learning Representations by Back-Propagating Errors. Nature. 1986;323(6088):533-6.
[38] Page L, Brin S, Motwani R, Winograd T. The PageRank Citation Ranking: Bringing Order to the Web. Stanford InfoLab; 1999. 1999-66. Available from: http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf.
[39] Seidman SB. Network Structure and Minimum Degree. Social Networks. 1983;5(3):269-87.
[40] Fawcett T. An Introduction to ROC Analysis. Pattern Recognition Letters. 2006;27(8):861-74.
[41] Brier GW. Verification of Forecasts Expressed in Terms of Probability. Monthly Weather Review. 1950;78(1):1-3.
[42] Tabassi E. Artificial Intelligence Risk Management Framework (AI RMF 1.0). National Institute of Standards and Technology; 2023. NIST AI 100-1.