Proceedings of the 5th International Conference on Financial Technology and Business Analysis
DOI: 10.54254/2754-1169/2026.39082

Application of Intelligent Agents in Anti-money Laundering Transaction Detection

Yunzhe Wan

Surrey International Institute, Dongbei University of Finance and Economics, Dalian, China
1967165393@qq.com

Abstract. Anti-money laundering (AML) transaction detection faced scarce labels, extreme class imbalance, relational dependencies, and future-information leakage. This study developed a validation-gated ensemble detection framework combining temporally constrained transaction features, historical account statistics, explicit transaction-network motifs, and seven heterogeneous XGBoost experts. A large language model (LLM) proposed candidate rules from aggregated validation evidence, which a programmatic rule-validation agent assessed through threshold fitting and out-of-time validation. A validation-gated router integrated expert predictions and validated rule signals. On the temporal test set of the synthetic HI-Small dataset, the framework achieved PR-AUC, F1, and recall of 0.5965, 0.5532, and 0.4299, respectively. These values exceeded those of the evaluated XGBoost and GIN-family baselines; the explicit-motif XGBoost baseline achieved a PR-AUC of 0.4788. Stage-wise analysis attributed most gains to explicit transaction-network motifs and validation-weighted expert voting. The router provided only a small improvement, and the rule branch did not improve PR-AUC. Under this experimental setting, the findings supported using the LLM for candidate-rule generation and auxiliary analysis, but did not establish it as the source of the overall performance gain.

Keywords: anti-money laundering, transaction detection, multi-expert ensemble, transaction-network motifs, intelligent agents

1. Introduction

1.1. Historical development of AML transaction detection

AML transaction detection was a central component of financial institutions' obligations for customer risk management and suspicious-transaction monitoring. Its primary purpose was to identify high-risk transactions from customer information, account behavior, and fund-flow records, and to refer relevant cases for compliance review. Early automated detection systems relied mainly on regulatory requirements, business experience, and known money-laundering patterns, with experts specifying thresholds for transaction amounts, frequencies, and combinations of conditions. Such rules were auditable and readily embedded in compliance workflows, and they remained an important component of AML systems. As electronic payments expanded and transactional relationships became more complex, fixed thresholds could no longer exploit high-dimensional data

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2754-1169/2026.39082

adequately; statistical analysis and machine learning were consequently introduced for customer risk profiling, anomalous-behavior detection, and alert prioritization [1, 2].

The evolution of AML technology was not a simple replacement of rules and statistical analysis by artificial intelligence; rather, it represented a continuing expansion in the information available and the levels at which it could be analyzed. Conventional rules converted regulatory knowledge into executable conditions; statistical and machine-learning models estimated risk from historical data; network analysis and graph learning characterized account relationships; and LLMs and agents were beginning to support natural-language knowledge processing, analytical task decomposition, and result interpretation. Although newer technologies might improve detection efficiency and risk identification, they still had to satisfy requirements for risk-based safeguards, data protection, interpretability, and human oversight [1]. The suitability of each method therefore remained jointly constrained by data quality, the regulatory environment, privacy requirements, computational scalability, and the need for explanation [3, 4].

1.2. Principal challenges in AML transaction detection

First, reliable labels were exceptionally scarce. A data label was the supervised-learning annotation that represented the ground-truth class of an observation; in this task, transactions were generally categorized as normal or money laundering. In banking data, money-laundering labels typically originated from investigations, manual reviews, or law-enforcement feedback. They were slow and costly to obtain and covered only activities that had already been detected. A transaction that was not labeled as money laundering was therefore not necessarily confirmed to be legitimate. AML data consequently combined a low positive-class prevalence with incomplete labels and limited public availability. Extreme class imbalance reduced the usefulness of overall accuracy, while incomplete labels might cause models to learn from yet-undetected laundering transactions as if they were negative examples [2]. These limitations had also motivated interest in self-supervised learning, semi-supervised learning, and synthetic data [5-7].

Second, money-laundering risk often depended on transaction-network structure. A single transaction could be described by time, amount, currency, payment method, and the counterparties involved, but complex fund movements generally required analysis of sequential relationships among multiple accounts. If accounts were represented as nodes and timestamped fund transfers as directed edges, the transaction data formed a time-evolving directed multigraph. Practices such as splitting, aggregation, dispersion followed by re-aggregation, circular transfers, and multilayer pass-through transactions generated characteristic connection structures and fund-flow patterns in local transaction networks. Prior work showed that network indicators could add customer-risk information beyond individual attributes and transaction statistics; studies of time-evolving transaction graphs further suggested that transaction order, transfer velocity, and local subgraph structure were informative for detecting particular laundering patterns [8, 9]. Treating transactions as mutually independent observations might therefore omit the account relationships and multistep fund paths on which risk depended.

Third, real-world deployment was subject to strict point-in-time constraints. Deployment meant connecting a model to a continuously arriving transaction stream and producing a risk score at, or shortly after, each transaction. For a transaction arriving at time t, the model was expected in principle to use only that transaction and the history available at or before t. If account statistics, graph structures, or motif features were constructed using transactions after t, future-information leakage occurred. Those future transactions were unavailable at prediction time but might be incorporated prematurely during offline processing, thereby overstating validation and test

DOI: 10.54254/2754-1169/2026.39082

performance. Research on streaming graph features showed that dynamic transaction graphs needed to be updated chronologically and that network features for the current transaction needed to be calculated from the historical graph state [10]. Strict temporal ordering and information isolation were therefore prerequisites for credible deployment-oriented evaluation.

1.3. Evolution of AML transaction-detection methods

Rule-based methods applied conditions derived from expert knowledge and regulatory requirements, such as large transaction amounts, high-frequency short-term transfers, and transaction splitting. Their decision logic was explicit, their outputs were readily interpretable, and they could directly encode known risk typologies. Their limitations were that thresholds and conditions required continual maintenance, overlapping rules might produce duplicate alerts, and fixed rules were poorly suited to novel laundering behavior that had not yet been formalized. Rule-based approaches were thus appropriate for compliance constraints and screening for known patterns, but their effectiveness depended on rule quality, the business environment, and update mechanisms [1, 2].

Statistical and conventional machine-learning methods formulated transaction detection as classification, anomaly detection, or risk ranking. Common models included logistic regression, decision trees, random forests, support vector machines, and gradient-boosted trees. Compared with fixed rules, these models could learn nonlinear relationships among multiple variables and could support alert prioritization through probabilities or risk scores. Their performance, however, remained highly dependent on label quality and manually engineered features. When the input consisted mainly of single-transaction attributes or aggregated account indicators, the representation of multi-account relationships and multistep fund flows remained limited [2-4].

As the graph structure of transaction data had received greater attention, network analysis, graph neural networks, and self-supervised graph representation learning had increasingly been applied to AML. Social-network analysis supplemented risk profiles with centrality, adjacency, and community structure [8]; temporal graph-pattern search identified structures such as fund splitting and aggregation [9]; LaundroGraph learned account and transaction representations from weakly labeled transaction networks through self-supervised tasks [5]; and graph-contrastive pre-training created learning signals from unlabeled networks to reduce dependence on manual labels [6]. Another line of work did not use graph neural networks directly for classification, but instead mined fan-in, fan-out, dispersed aggregation, and circular subgraph patterns in streaming data and encoded them as inputs to gradient-boosting models [10]. These studies showed that graph learning and tree-based models were complementary rather than mutually exclusive, and that network structure could be supplied to conventional classifiers through explicit transaction-network motifs.

LLMs and agents provided new functional modules for AML detection, although the relevant evidence remained preliminary. In financial-agent research, LLMs had been used for information organization and language-based analysis, but they faced limitations in numerical-reasoning stability, prompt sensitivity, real-time adaptation, privacy compliance, and architectural transparency [11]. Existing studies had converted local financial subgraphs into structured text and then used in-context learning for suspiciousness assessment and explanation generation, but the available evidence was derived mainly from synthetic settings [12]. Accordingly, the appropriate role of the LLM in this study was constrained rule-hypothesis generation rather than direct replacement of numerical detection models.

Motivated by these limitations, this study developed a validation-gated ensemble detection framework that satisfied point-in-time constraints. The framework built historical snapshots in chronological order, constructed transaction-local features, historical account statistics, and

DOI: 10.54254/2754-1169/2026.39082

transaction-network motifs, and trained seven XGBoost detection experts with different feature sets and error preferences. From aggregated evidence such as feature importance, class separability, and false-positive and false-negative statistics, an LLM proposed structured candidate rules. A programmatic rule-validation agent then performed threshold fitting, out-of-time validation, and structural deduplication in independent temporal windows. Finally, a validation-gated router combined expert scores, expert disagreement, and validated rule signals, reverting to an anchor expert when the gain did not meet a prespecified criterion. This design enabled testing of whether transaction-network motifs, multi-expert ensembling, the rule branch, and validation-gated routing improved extremely imbalanced transaction detection without using future information.

2. Dataset

2.1. IBM HI-Small dataset

This study used the HI-Small subset of the publicly available AMLworld synthetic AML datasets. Altman et al. developed AMLworld as a multi-agent virtual environment that simulated transactions among banks, individuals, and companies and included actors that held illicit funds and conducted laundering activities. The data-generation agents were simulation units representing economic entities and their behaviors; they were distinct from the rule-generation agent used for detection in this study. Rather than being seeded with real individuals or transactions, the generator simulated account relationships, economic activities, and money-laundering processes from statistical distributions [7].

AMLworld provided six datasets: high-illicit-ratio (HI) and low-illicit-ratio (LI) groups, each at Small, Medium, and Large scales. HI-Small spanned 10 days and contained approximately 515,000 bank accounts, 5 million transactions, and 3,600 transactions labeled as money laundering. The subset combined a large transaction volume with severe class imbalance and was therefore suitable for studying rare laundering-transaction detection.

Each record represented a transaction from a source account to a destination account. The principal fields included timestamp, source bank and account, destination bank and account, transaction amount, currency, payment method, and money-laundering label. When accounts were represented as nodes and transactions as attributed, timestamped directed edges, the dataset became a time-evolving directed multigraph. Because the same pair of accounts might transact repeatedly at different times, retaining parallel edges and their temporal order was necessary for constructing historical account statistics, local-neighborhood features, and transaction-network motifs.

HI-Small was well matched to the proposed method for three reasons. First, the synthetic environment recorded the generation and subsequent movement of illicit funds, providing relatively complete transaction-level labels for comparing detection experts and routing strategies. Second, the data contained transaction attributes, account relationships, timestamps, and multiple laundering patterns, allowing transaction-local features, historical account statistics, and transaction-network motifs to be constructed within a unified setting. Third, the subset was large enough to exhibit severe class imbalance and complex transaction networks, yet was more tractable than the Medium and Large datasets for repeated feature ablations, rule validation, and router calibration. HI-Small therefore served as a controlled benchmark for comparing components of the proposed ensemble framework.

DOI: 10.54254/2754-1169/2026.39082

2.2. Dataset source and supporting reference

HI-Small originated from Realistic Synthetic Financial Transactions for Anti-Money Laundering Models by Altman et al., published at NeurIPS 2023 [7]. That study introduced the AMLworld data generator, released six synthetic datasets with different scales and illicit-transaction ratios, and reported gradient-boosted tree and graph-neural-network baselines. Unlike proprietary banking data, AMLworld contained no real personal information and its simulator provided transaction-level laundering labels. Its limitation was that transaction behavior, account relationships, and laundering patterns were constrained by the data-generation mechanism and its parameterization; performance on the synthetic data could not therefore be extrapolated proportionally to operational settings.

This study accordingly used HI-Small only for method development, internal comparison, and mechanism analysis, focusing on the relative contributions of feature classes, detection experts, validated rules, and validation-gated routing within this dataset.

3. Methods

This study addressed temporally ordered AML transaction detection under extreme class imbalance with an ensemble framework comprising point-in-time-constrained features, heterogeneous detection experts, a rule-generation agent, a rule-validation agent, and a validation-gated router. The framework separated high-frequency numerical prediction from low-frequency rule discovery. XGBoost detection experts produced a risk score for each transaction; during model development only, an LLM read aggregated validation evidence and proposed candidate rules; a programmatic rule-validation agent fitted thresholds and evaluated the rules; and a validation-gated router used an independent validation interval to decide whether, and to what extent, the routing output was to be combined with the anchor expert. The design targeted transaction-level ranking while maintaining auditable rules and temporal isolation.

3.1. Problem definition and framework overview

3.1.1. Task definition

The i th transaction was represented as

$$\mathrm{e_i} = (\mathrm{u_i}, \mathrm{v_i}, \mathrm{t_i}, \mathrm{x_i}, \mathrm{y_i}) \tag{1}$$

where u_i and v_i were the payer and payee accounts, respectively; t_i was the transaction time; x_i contained features of the current transaction and its historical network; and $y_i \in \{0,1\}$ was the label. The model learned a risk function f that produced the risk score that the transaction belonged to the money-laundering class

$$\widehat{\mathrm{p}}_\mathrm{i} = f(\mathrm{e_i}) \in [0,1] \tag{2}$$

Because the positive class was far less prevalent than the negative class, PR-AUC was used as the primary measure of ranking performance. In the experimental code, PR-AUC was calculated using average_precision_score and therefore corresponded numerically to average precision (AP). To

DOI: 10.54254/2754-1169/2026.39082

remain consistent with the figures and experimental outputs, the term PR-AUC was used throughout:

$$AP = \sum_k (R_k - R_{k-1}) P_k \tag{3}$$

where P_k and R_k were the precision and recall, respectively, at the k th threshold. F1, precision, recall, ROC-AUC, the Brier score, and expected calibration error were additionally reported to evaluate ranking, classification, and probabilistic calibration.

3.1.2. Temporal partitioning and point-in-time constraints

All transactions were sorted in ascending temporal order. Split points were moved to the end of tied timestamps so that the same timestamp did not span adjacent intervals. The full experiment sequentially defined a context interval, a training target interval, a validation target interval, and a test target interval at cumulative proportions of 55%, 75%, 85%, and 100%, respectively. In the complete HI-Small data, the training, validation, and test target intervals contained 1,015,727, 507,772, and 761,639 transactions, including 1,061, 542, and 1,561 positive instances, respectively.

For any target interval Is, account statistics and transaction-network motifs were constructed only from transactions preceding the start of that interval. The training target interval used the context interval as its historical snapshot; the validation target interval used the context and training target intervals; and the test target interval used all context, training, and validation history available before the test interval began. Standardization parameters fitted during training were retained for validation and testing.

The complete experiment used stage-wise historical snapshots: all transactions within a target interval shared the historical state available before the interval began, and the account statistics and transaction-network motifs were not updated transaction by transaction within that interval. The code additionally provided a rolling diagnostic that reconstructed history by test-time buckets, but this option was not enabled in the complete experiment. Point-in-time constraints in this chapter therefore meant that no information after the target interval was used; they did not imply online updating after every transaction.

3.1.3. Processing pipeline

The method comprised five steps. First, the feature-construction module generated transaction-local features, historical account statistics, full-history transaction-network motifs, multi-scale transaction-network motifs, and engineered interactions from temporal snapshots, producing a unified feature matrix. Second, seven XGBoost detection experts were trained with different feature subsets, class weights, and negative-sample compositions, yielding risk scores with heterogeneous inductive biases. Third, the error-analysis module compressed feature importance, class separability, quantiles, and false-positive and false-negative statistics into an evidence package from which the LLM proposed a limited number of structured rule hypotheses. Fourth, the rule-validation agent fitted thresholds in temporally partitioned validation subintervals, evaluated support and lift, and converted accepted rules into binary features. Fifth, a shallow XGBoost router combined expert scores, expert disagreement, important raw features, and rule features, and produced the final risk probability after confirmation gating, anchor fallback, and Platt calibration.

DOI: 10.54254/2754-1169/2026.39082

3.2. Construction of point-in-time-constrained features

3.2.1. Transaction-local features

Transaction-local features comprised 12 continuous variables and one-hot encodings of categorical variables. The 12 continuous variables were the log payer amount, log payee amount, log foreign-exchange ratio, payer amount deviation, payee amount deviation, same-bank indicator, self-transfer indicator, cross-currency indicator, and sine and cosine encodings of hour and day of week. This study denoted the payer and payee amounts by a_i^{pay} and a_i^{rec}, respectively:

$$\mathrm{x_{i,1}} = log(1 + \mathrm{a_i^{pay}}),\ \mathrm{x_{i,2}} = log(1 + \mathrm{a_i^{rec}}) \tag{4}$$

$$\mathrm{x_{i,3}} = \log \frac{\mathrm{a_i^{pay}} + 10^{-4}}{\mathrm{a_i^{rec}} + 10^{-4}} \tag{5}$$

The payer- and payee-side amount deviations were defined as the difference between the current log amount and the corresponding historical mean log amount:

$$\mathrm{x_{i,4}} = log(1 + \mathrm{a_i^{pay}}) - log(1 + \bar{\mathrm{a}}_{\mathrm{u_i}}^{\mathrm{out}}) \tag{6}$$

$$\mathrm{x_{i,5}} = log(1 + \mathrm{a_i^{rec}}) - log(1 + \bar{\mathrm{a}}_{\mathrm{v_i}}^{\mathrm{in}}) \tag{7}$$

Hour and day of week were cyclically encoded. If the transaction occurred at hour h_i and had day-of-week index d_i, then:

$$\mathrm{x_{i,9}} = \sin \frac{2\pi \mathrm{h_i}}{24},\ \mathrm{x_{i,10}} = \cos \frac{2\pi \mathrm{h_i}}{24} \tag{8}$$

$$\mathrm{x_{i,11}} = \sin \frac{2\pi \mathrm{d_i}}{7},\ \mathrm{x_{i,12}} = \cos \frac{2\pi \mathrm{d_i}}{7} \tag{9}$$

The categorical variables were payment channel, payer currency, and payee currency. A fixed category dictionary was learned for each group from the training data and used for one-hot encoding. In this experiment, the three encoded groups contributed 37 dimensions, giving 49 transaction-local features in total. Continuous features were standardized using means and standard deviations estimated during training; categorical features were binary.

DOI: 10.54254/2754-1169/2026.39082

3.2.2. Historical account statistics

For each account u in the historical snapshot, 10 account statistics were calculated: incoming count, outgoing count, incoming amount, outgoing amount, mean incoming amount, mean outgoing amount, net-amount ratio, net-count ratio, interbank-transaction ratio, and self-transfer ratio. The first six were $log(1+x)$ transformed; the net-amount and net-count ratios were defined as:

$$\mathrm{b_u^{vol}} = \frac{\mathrm{A_u^{in}} - \mathrm{A_u^{out}}}{\max(\mathrm{A_u^{in}} + \mathrm{A_u^{out}}, 1)} \tag{10}$$

$$b_u^{cnt} = \frac{C_u^{in} - C_u^{out}}{max(C_u^{in} + C_u^{out}, 1)} \tag{11}$$

Here, A denoted cumulative amount and C denoted cumulative transaction count. Interbank-transaction and self-transfer ratios used the total number of incoming and outgoing edges of an account as their denominator. Standardization parameters for account statistics were fitted only on active accounts.

For transaction (u_i, v_i), this study extracted payer-account vector $\boldsymbol{n}_{u_i}$ and payee-account vector $\boldsymbol{n}_{v_i}$, and constructed their absolute difference and elementwise product:

$$\mathrm{x_i^{node}} = [\mathrm{n_{u_i}}, \mathrm{n_{v_i}}, \mathrm{n_{u_i}} \odot \mathrm{n_{v_i}}, |\mathrm{n_{u_i}} - \mathrm{n_{v_i}}|] \tag{12}$$

The historical-account component expanded 10 base statistics across four relational representations, yielding 40 dimensions. This representation preserved the historical states at both ends of a transaction, their differences, and their interaction strengths.

3.2.3. Full-history transaction-network motifs

This study represented historical transactions as a directed multigraph $G = (V, E)$. Accounts were nodes and transactions were directed edges; multiple parallel edges might connect the same account pair. For current transaction (u, v), the full-history motif representation contained 13 dimensions of features:

1. $u \to v$ parallel-edge count and its log transformation;
2. $v \to u$ reverse-edge count, its log transformation, and an indicator of whether a reverse edge existed;
3. the payer account's numbers of unique outgoing and incoming neighbors;
4. the payee account's numbers of unique incoming and outgoing neighbors;
5. $N^{out}(u) \cap N^{in}(v)$ corresponding diverging–converging two-hop path count;
6. $N^{out}(v) \cap N^{in}(u)$ corresponding three-node cycle-path count;
7. $N^{out}(u) \cap N^{in}(v)$ number of common payees;
8. $N^{out}(v) \cap N^{in}(u)$ number of common payers;
9. the fan-in–fan-out balance of the payer and payee accounts.

DOI: 10.54254/2754-1169/2026.39082

Except for existence indicators and balance measures, all count variables underwent a $log(1+x)$ transformation. The fan-in–fan-out balance of account u was defined as:

$$g_u = \frac{2min(d_u^{in}, d_u^{out})}{max(d_u^{in}+d_u^{out}, 1)} \tag{13}$$

where d_u^{in} and d_u^{out} were the numbers of unique incoming and outgoing neighbors. Neighbor-intersection calculations were capped at 256 to control the computational cost for highly connected nodes.

3.2.4. Multi-scale transaction-network motifs

To describe recent activity, this study constructed six features in each of three windows—6 h, 24 h, and 168 h: the numbers of same-direction parallel transactions, reverse transactions, payer outgoing transactions, payee incoming transactions, unique payees of the payer, and unique payers of the payee. Each window contributed six dimensions, producing 18 multi-scale motif features. The window length was denoted by Δ and the historical-snapshot cutoff time by T_s. The edge set within the window was:

$$E_\Delta(T_s) = \{e_j | T_s - \Delta \le t_j \le T_s\} \tag{14}$$

All windows were defined relative to the cutoff time of the historical snapshot preceding the target interval. The 13 full-history motif features and 18 multi-scale motif features together formed a 31-dimensional explicit-motif representation.

3.2.5. Engineered interactions and feature groups

In addition to transaction-local features, historical account statistics, and transaction-network motifs, the complete feature set included 31 engineered interactions in four groups: amount differences, account fund-flow relationships, cross-window differences, and motif interactions. Amount-difference features comprised the difference between payer and payee log amounts, its absolute value, the maximum amount deviation at either end, and the absolute difference between the two deviations. Account fund-flow features comprised the sum and minimum of the payer's historical outgoing amount and the payee's historical incoming amount, the difference in net-amount ratios at the two ends, and the product of the payer's outgoing count and the payee's incoming count.

For each of the six recent motif types, short-minus-long, short-minus-medium, and medium-minus-long differences were calculated, yielding 18 cross-window features. This study further included five motif interactions: the products of parallel-edge and reverse-edge counts and of two-hop and cycle-path counts, the difference between the two account-balance measures, the interaction between amount and cycle paths, and the interaction between amount and two-hop paths. The four groups contained 31 engineered features in total.

DOI: 10.54254/2754-1169/2026.39082

3.3. Multiple detection experts

3.3.1. Common training configuration

All seven detection experts used XGBoost binary classifiers [13] with a binary logistic objective, aucpr evaluation metric, hist tree-building method, and base learning rate of 0.05. Each model used a fixed number of trees without early stopping. All positive training instances were retained, and at most 30 negative instances were sampled per positive instance. In this experiment, each expert used 1,061 positive and 31,830 negative instances, for a total of 32,891 training observations. The base positive-class weight was the ratio of sampled negative to positive instances, multiplied by an expert-specific weight coefficient.

The detection experts were not repeated fits of the same model. Instead, they used different feature subsets, tree depths, numbers of trees, class weights, and proportions of hard negative samples to induce heterogeneous biases.

3.3.2. Hard-Negative expert

The Hard-Negative Precision expert ranked negative training instances using risk scores from the Engineered Motif Champion. Of the negative instances required by this expert, 70% were selected from those with the highest risk scores and 30% were sampled without replacement from the remaining negatives. If N_- negative instances were required, the number of hard negatives was:

$$\mathrm{N}_-^{\mathrm{hard}} = round(0.7\mathrm{N}_-) \tag{15}$$

The remaining instances were sampled without replacement from ordinary negatives. This expert also used shallower trees, a larger minimum child weight, and a lower positive-class weight to improve discrimination between positive instances and difficult negative instances.

3.3.3. Expert outputs

Each expert output a risk probability p_{im} for the validation and test sets, where $m = 1, \cdots, 7$. The model also stored gain-based feature importance. The seven probabilities were used both to compare experts and as primary router inputs. Expert names described their design objectives and did not prespecify their ranking on the test set.

3.4. Error evidence and the rule-generation agent

3.4.1. Evidence compression

Rule discovery analyzed the Engineered Motif Champion using only the earliest, Discovery interval of the validation set. This study first determined the F1-maximizing threshold used to define error groups within that interval. For each candidate feature, this study then calculated gain importance; the difference between positive- and negative-class means divided by the overall standard deviation; the 50th, 90th, and 95th percentiles of the negative class; and the mean feature values among false-positive and false-negative instances. Features were ranked by the sum of their importance rank and class-separation rank, and at most 28 were retained in a structured evidence package. The package

DOI: 10.54254/2754-1169/2026.39082

also recorded the Discovery sample size, number and proportion of positive instances, and the Champion model's PR-AUC in that interval.

This compression restricted the LLM to aggregated development-set evidence; it had no access to raw transaction records or final test labels. Feature names appearing in the evidence package also defined the whitelist for candidate rules.

3.4.2. Model and inference configuration

The rule-generation agent used Qwen3-4B-Instruct-2507 [14], loaded on a local GPU with 4-bit NF4 quantization, double quantization, and FP16 as the 4-bit computation type. The model was invoked once, with a maximum generation length of 1,200 tokens and do_sample set to False. If local model loading, inference, or parsing failed, the detection pipeline continued using a deterministic rule-generation program. The prompt, raw model response, parsing status, and candidate rules were written to output files for audit.

Within this framework, the LLM proposed only verifiable feature relationships. It did not score transactions directly, execute code, access the test set, generate numerical thresholds, or decide whether a rule was accepted. Online inference executed only validated and fixed rule programs and did not invoke the LLM.

3.4.3. Structured rule hypotheses

Each candidate-rule object contained at least a rule name, rule type, first feature, and direction. A bivariate rule additionally included a second feature, its direction, and a brief rationale. The permitted rule types were:

$$K = single, and, contrast, product \tag{16}$$

The allowed direction set was $\{high, low\}$. Model output had to be parseable as JSON; feature names had to be on the whitelist; and rule type and direction had to belong to the permitted sets. Otherwise, the candidate was discarded. The system also constructed single-feature, conjunction, and product candidates automatically from Discovery evidence, and removed exact duplicates using a signature comprising rule type, feature order, and directions.

3.5. Rule-validation agent

3.5.1. Rule semantics and threshold search

The rule-validation agent converted natural-language hypotheses into fixed numerical operations. A candidate rule used standardized features x_{ia} and x_{ib}. The four rule types had the following forms:

$$\mathrm{r}_{\mathrm{i}}^{\mathrm{single}} = l(\mathrm{x}_{\mathrm{ia}} \circ_{\mathrm{a}} \tau_{\mathrm{a}}) \tag{17}$$

$$\mathrm{r}_{\mathrm{i}}^{\mathrm{and}} = l(\mathrm{x}_{\mathrm{ia}} \circ_{\mathrm{a}} \tau_{\mathrm{a}}) l(\mathrm{x}_{\mathrm{ib}} \circ_{\mathrm{b}} \tau_{\mathrm{b}}) \tag{18}$$

DOI: 10.54254/2754-1169/2026.39082

$$r_i^{contrast} = l(x_{ia} - x_{ib} \circ_a \tau_a) \tag{19}$$

$$r_i^{product} = l(clip(x_{ia}x_{ib}, -10^4, 10^4) \circ_a \tau_a) \tag{20}$$

Here, when the direction was high, $\circ$ denoted $\geq$; when the direction was low, $\circ$ denoted $\leq$. Thresholds for single, contrast, and product rules were selected from the 75th, 85th, 90th, 95th, 97.5th, and 99th percentiles of the negative-class distribution in Discovery; the corresponding lower-tail percentiles were used for the low direction. For and rules, threshold combinations were searched from the 80th, 90th, and 95th percentiles of the negative-class distribution for each feature. All numerical thresholds were fitted by the validation program rather than generated by the LLM.

3.5.2. Rule-acceptance criteria

For rule *q*, support, precision, recall, and lift were defined as:

$$support_q = \frac{1}{n}\sum_{i=1}^{n} r_{iq} \tag{21}$$

$$precision_q = \frac{\sum_{i=1}^{n} r_{iq}y_i}{\sum_{i=1}^{n} r_{iq}+\varepsilon} \tag{22}$$

$$Recall_q = \frac{\sum_{i=1}^{n} r_{iq}y_i}{\sum_{i=1}^{n} y_i+\varepsilon} \tag{23}$$

$$lift_q = \frac{precision_q}{\pi+\varepsilon} \tag{24}$$

Here, π was the positive-class prevalence in the corresponding validation interval. In Discovery, a candidate rule had to hit at least three positive instances, have lift of at least 1.5, and have support within $[10^{-4}, 0.25]$. In Verification, the rule had to hit at least two positive instances, have lift of at least 1.125, and have support greater than zero.

For every threshold combination satisfying these criteria, this study calculated:

$$L_q = \sqrt{L_q^D \max(L_q^V, L_q^D)} \tag{25}$$

$$R_q = \sqrt{R_q^D \max(R_q^V, R_q^D)} \tag{26}$$

DOI: 10.54254/2754-1169/2026.39082

$$\mathrm{S_q} = log(1 + \mathrm{L_q})\sqrt{\max(\mathrm{R_q}, 10^{-9})} \tag{27}$$

Superscripts D and V denoted Discovery and Verification, respectively. For each candidate, the threshold combination with the highest score was retained. All candidates were ranked in descending order of S_q, and at most 12 rules were accepted.

The current deduplication procedure removed only structurally identical rules on the basis of rule type, feature order, and direction signature. It did not yet perform semantic deduplication based on correlations between rule-hit vectors or logical equivalence. Overlap among accepted rules still had to be examined when interpreting them.

3.5.3. Internal partitioning of the validation set

The validation target interval was further divided chronologically into Discovery, Verification, Router Selection, and Calibration segments containing 35%, 20%, 25%, and 20% of the validation data, respectively.

Router Selection was then divided equally in time into a weight-tuning segment and a confirmation segment. This design ensured that rule discovery, confirmation of routing gains, and probability calibration did not access the final test interval. However, Verification was used both to confirm rules and to fit the router and therefore did not provide completely independent validation data. The subsequent confirmation gate on Router Selection was used to limit the resulting overfitting risk.

3.6. Validation-gated routing and calibration

3.6.1. Routing meta-features

The router input had four components. The first comprised seven expert probabilities pim and their logit transformations. The second contained the mean, standard deviation, minimum, maximum, and range of the expert probabilities. The third contained the 24 highest-ranked raw features with positive gain in Engineered Motif Champion. The fourth contained binary features for validated rules. If 12 rules were accepted, the router had 55 input dimensions.

The expert summary statistics represented the ensemble's overall risk level and disagreement; important raw features provided transaction context; and rule features contributed structured anomaly signals that had passed temporal validation.

3.6.2. Shallow router

The router was a shallow XGBoost model with 250 trees, maximum depth 3, learning rate 0.04, minimum child weight 2, and both row and feature subsampling rates of 0.90. The L1 and L2 regularization coefficients were 0.02 and 1.5, respectively. The positive-class weight was set to the negative-to-positive ratio in the router-fitting data, and the random seed was 2069. The router was fitted on Verification. Discovery and Verification were merged as a fallback training set only when Verification contained fewer than 12 positive instances or lacked one class. Verification contained 158 positive instances in the complete experiment, so this fallback was not triggered.

DOI: 10.54254/2754-1169/2026.39082

To assess the marginal contribution of rules, the system followed the same procedure to train a router without rule features and a complete router with validated rule features. Both configurations were subject to the subsequent confirmation gate; router outputs were not adopted by default.

3.6.3. Anchor fallback and confirmation gating

In the first half of Router Selection, the detection expert with the highest PR-AUC was selected as the anchor expert. This study used $\mathrm{p}_{\mathrm{i}}^{\mathrm{anchor}}$ to denote the anchor probability and $\mathrm{p}_{\mathrm{i}}^{\mathrm{router}}$ to denote the router probability. A candidate mixture was:

$$\mathrm{p}_{\mathrm{i}}^{\mathrm{mix}}(w)=(1-w)\mathrm{p}_{\mathrm{i}}^{\mathrm{anchor}}+w\mathrm{p}_{\mathrm{i}}^{\mathrm{router}},\qquad w\in\{0{,}0.1,\ldots,1.0\} \tag{28}$$

The value of w with the highest PR-AUC was selected in the first half of Router Selection and then evaluated in the second half. The weight was retained only if:

$$\mathrm{AP}_{\mathrm{mix}}\geq\mathrm{AP}_{\mathrm{anchor}}(1+0.005) \tag{29}$$

If the condition was not met, this study set $\mathrm{w}=0$ and fell back directly to the anchor expert. In this experiment, Hard-Negative Precision was selected as the anchor expert and $\mathrm{w}=0.30$. Thus, mixture weight 0.30 explicitly denoted the router weight and 0.70 the anchor-expert weight.

3.6.4. Platt calibration and operating threshold

Probabilities after confirmation gating were Platt-calibrated in the Calibration interval [15]. The uncalibrated probability p_i^{mix} was first clipped to $\left[10^{-6},1-10^{-6}\right]$; its logit was then calculated and a logistic-regression model was fitted:

$$\widehat{\mathrm{p}}_{\mathrm{i}}^{\mathrm{cal}}=\sigma(\mathrm{a}\,\mathrm{logit}(\mathrm{p}_{\mathrm{i}}^{\mathrm{mix}})+\mathrm{b}) \tag{30}$$

The calibration model was an L2-regularized logistic regression with $\mathrm{C}=1000$, using the lbfgs solver and a maximum of 2,000 iterations. On the precision–recall curve of the same Calibration interval, this study then selected the F1-maximizing threshold τ^* :

$$\tau^*=arg\max_{\tau}\frac{2\mathrm{P}(\tau)\mathrm{R}(\tau)}{\mathrm{P}(\tau)+\mathrm{R}(\tau)} \tag{31}$$

At test time, the trained experts, rules, router, mixture weight, calibrator, and threshold were all fixed; no component was adjusted using test labels. For a new transaction, the system constructed point-in-time-constrained features, obtained seven expert probabilities, calculated binary features for accepted rules and the routing meta-features, generated the router probability, combined it with the anchor probability using the fixed weights, obtained the final probability through Platt calibration, and raised an alert according to τ^*.

DOI: 10.54254/2754-1169/2026.39082

4. Results and analysis

This chapter addressed three questions: whether the validation-gated ensemble detection framework improved transaction-risk ranking under extreme class imbalance; how much performance was contributed by detection experts, validation-weighted expert voting, the router, and the rule branch; and which transaction-local attributes and transaction-network structures most strongly informed the model.

4.1. Experimental setting

4.1.1. Data partitioning

Experiments used the AMLworld HI-Small dataset. All transactions were sorted in ascending temporal order and divided sequentially into a context interval, a training target interval, a validation target interval, and a test target interval. The training, validation, and test target intervals contained 1,015,727, 507,772, and 761,639 transactions, including 1,061, 542, and 1,561 positive instances, respectively. The test set occupied the final approximately 15% of the transaction sequence and had a positive-class prevalence of approximately 0.205%. This partition evaluated risk detection in a later temporal interval and avoided future-information leakage from random splitting.

4.1.2. Comparative models

The comparative models fell into three groups. The first contained XGBoost baselines using transaction-local features only, transaction-local features plus point-in-time-constrained account statistics, and a further extension with explicit transaction-network motifs. In the figures, the label causal graph statistics denoted point-in-time-constrained account statistics. The second group comprised graph-neural-network baselines: IBM-style GIN and IBM-style Multi-GIN with multi-view edge updates, reverse message passing, and port encoding. The third group was the proposed validation-gated ensemble detection framework, which included seven XGBoost detection experts, validated rule features, and a validation-gated router; its figure label was LLM-Agent Validation-Gated Router.

4.1.3. Evaluation metrics

This paper reported PR-AUC, F1, and recall. Because laundering transactions were extremely rare, PR-AUC was more informative than accuracy for overall minority-class ranking. F1 measured the balance between precision and recall at a fixed operating threshold, whereas recall was the proportion of labeled laundering transactions detected by the model.

4.2. Overall detection results

4.2.1. Comparison of PR-AUC

Figure 1 showed that the validation-gated ensemble detection framework achieved a PR-AUC of 0.5965 on the common temporal test set, the highest among the compared models. Explicit-motif XGBoost was the strongest single baseline, with a PR-AUC of 0.4788. Transaction-local XGBoost and XGBoost with account statistics achieved PR-AUC values of 0.2022 and 0.2267, respectively.

DOI: 10.54254/2754-1169/2026.39082

IBM-style GIN and IBM-style Multi-GIN achieved 0.0370 and 0.0320, respectively, both below the tree-based models evaluated here.

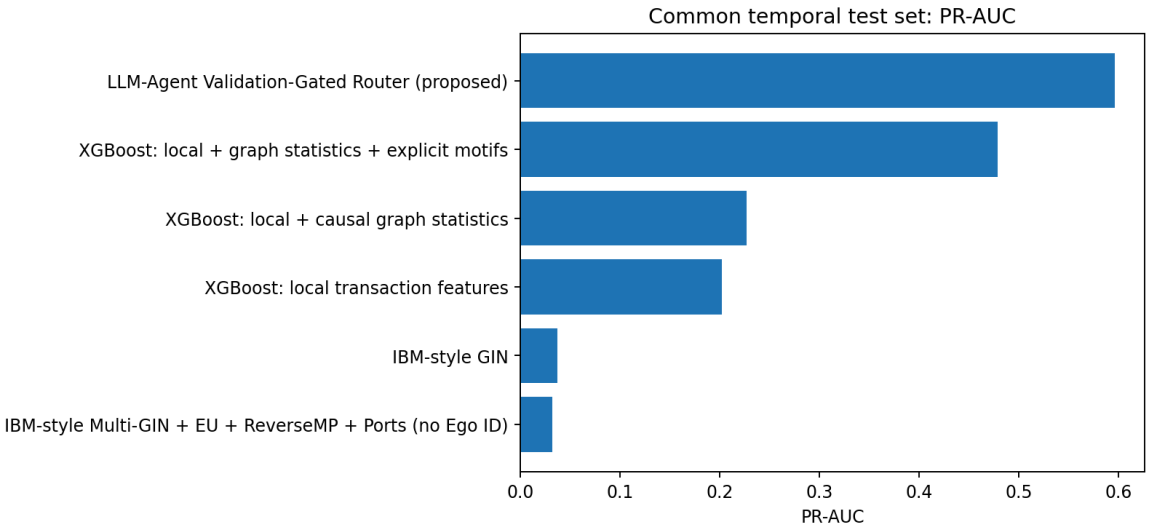


Figure 1. PR-AUC of different models on the common temporal test set

Table 1. Overall performance of each model on the common temporal test set

Model	PR-AUC	F1	Recall
Proposed validation-gated ensemble detection framework	0.5965	0.5532	0.4299
XGBoost: transaction-local + account statistics + explicit motifs	0.4788	0.4715	0.3863
XGBoost: transaction-local + point-in-time-constrained account statistics	0.2267	0.2694	0.2383
XGBoost: transaction-local features	0.2022	0.2756	0.2466
IBM-style GIN	0.0370	0.0614	0.0416
IBM-style Multi-GIN	0.0320	0.1033	0.0775

Relative to explicit-motif XGBoost, the ensemble framework improved PR-AUC by 0.1177 in absolute terms and 24.6% in relative terms. Explicit-motif XGBoost exceeded XGBoost with account statistics by 0.2521, suggesting that network structures such as repeated transfers, reverse transfers, fan-in/fan-out, and cycles contained discriminative information that was not fully captured by transaction-local attributes or account aggregates. Account statistics improved PR-AUC by only 0.0245 over transaction-local features, indicating that account-level activity summaries provided limited additional representation of relational structure.

4.2.2. Comparison of F1 and recall

The ensemble framework achieved an F1 of 0.5532, an improvement of 0.0817 over explicit-motif XGBoost, and a recall of 0.4299, an improvement of 0.0436. The advantage was therefore not restricted to aggregate ranking performance: at the selected operating threshold, the framework also identified more positive instances and achieved a better joint balance of precision and recall.

DOI: 10.54254/2754-1169/2026.39082

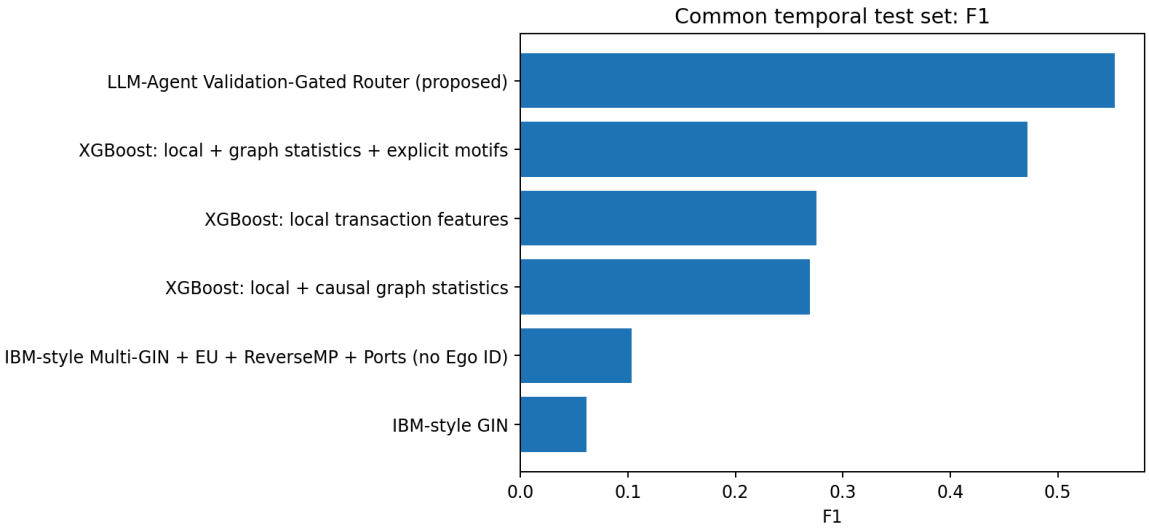


Figure 2. F1 of different models on the common temporal test set

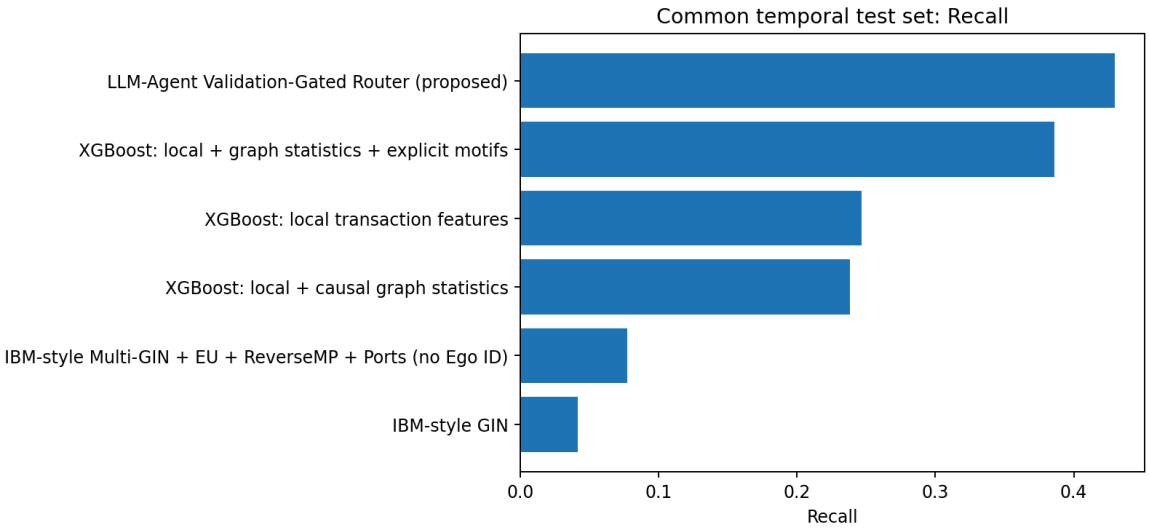


Figure 3. Recall of different models on the common temporal test set

Adding account statistics increased PR-AUC from 0.2022 to 0.2267, but reduced F1 from 0.2756 to 0.2694 and recall from 0.2466 to 0.2383. This divergence reflected the different scope of the metrics: PR-AUC evaluated ranking quality across thresholds, whereas F1 and recall depended on a single operating threshold. Account statistics improved overall ranking but did not guarantee a better classification point at the selected threshold. Models for extremely imbalanced tasks were therefore not considered suitable for evaluation using only a single-threshold metric.

Both graph-neural-network baselines had low overall values. IBM-style Multi-GIN had a slightly lower PR-AUC but higher F1 and recall than IBM-style GIN, indicating better minority-class identification around the fixed threshold without a corresponding improvement in overall ranking.

DOI: 10.54254/2754-1169/2026.39082

4.2.3. Precision–recall curves

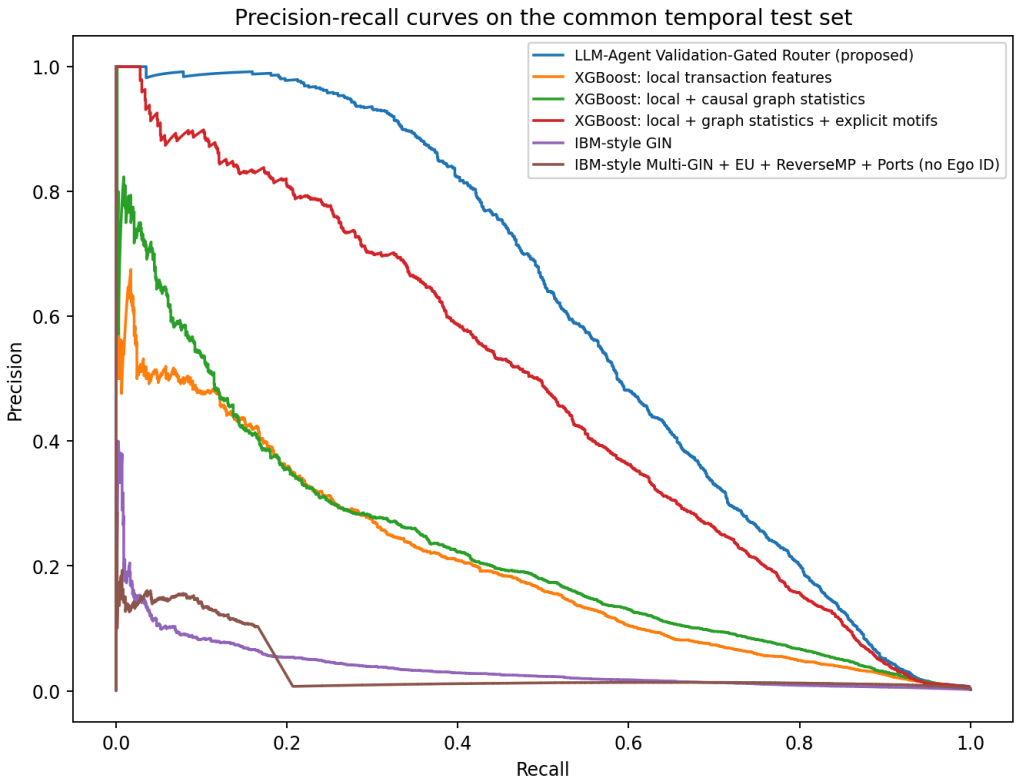


Figure 4. Precision–recall curves of different models on the common temporal test set

Figure 4 showed that the ensemble framework maintained the highest precision over most of the displayed recall range, with its curve generally above those of the other models. In the low-to-moderate recall range, it increased detection while retaining comparatively high precision. Explicit-motif XGBoost ranked second, whereas precision declined more rapidly for the transaction-local and account-statistics XGBoost models. This pattern agreed with the PR-AUC ranking and indicated that the framework's advantage spanned a broad range of thresholds.

As recall approached 1, the precision of every model approached the positive-class prevalence in the test set. This behavior was expected under extreme class imbalance: if a model labeled almost every transaction as positive, its precision converged to the base positive rate. Operational deployment had to select a region of the precision–recall curve that was compatible with the affordable alert volume.

4.3. Multi-expert and stage-wise contribution analysis

4.3.1. Comparison of detection experts

Hard-Negative Precision was the strongest single detection expert, with a PR-AUC of 0.5751. This result suggested that constructing training samples from high-risk negative instances helped the model learn the boundary between laundering transactions and difficult normal transactions. Engineered Motif Champion outperformed Exact Causal Motif by 0.0273, indicating that amount differences, account fund-flow relationships, cross-window differences, and motif interactions supplemented the original explicit motifs.

DOI: 10.54254/2754-1169/2026.39082

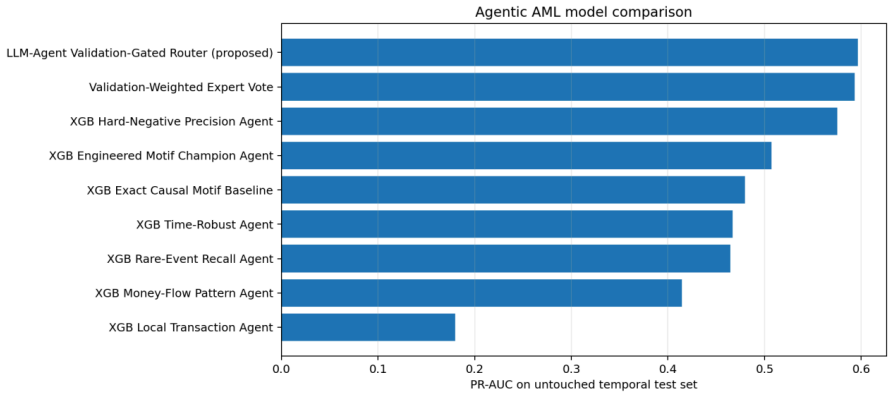


Figure 5. PR-AUC of the detection experts and ensemble framework

Local Transaction achieved a PR-AUC of 0.1798, below the detection experts that incorporated network information, further demonstrating that single-transaction attributes were insufficient to characterize laundering risk fully. Time-Robust, Rare-Event Recall, and Money-Flow Pattern did not individually outperform Hard-Negative Precision, but validation-weighted expert voting reached a PR-AUC of 0.5933, 0.0861 above Engineered Motif Champion and higher than every individual expert. Complementarity among the experts was therefore a principal source of the overall performance gain.

The final routing model improved PR-AUC by only approximately 0.0032 over validation-weighted expert voting. Thus, most of the gain arose from the base risk ranking formed by the expert ensemble, with the router providing only a small adjustment.

4.3.2. Stage-wise model evolution

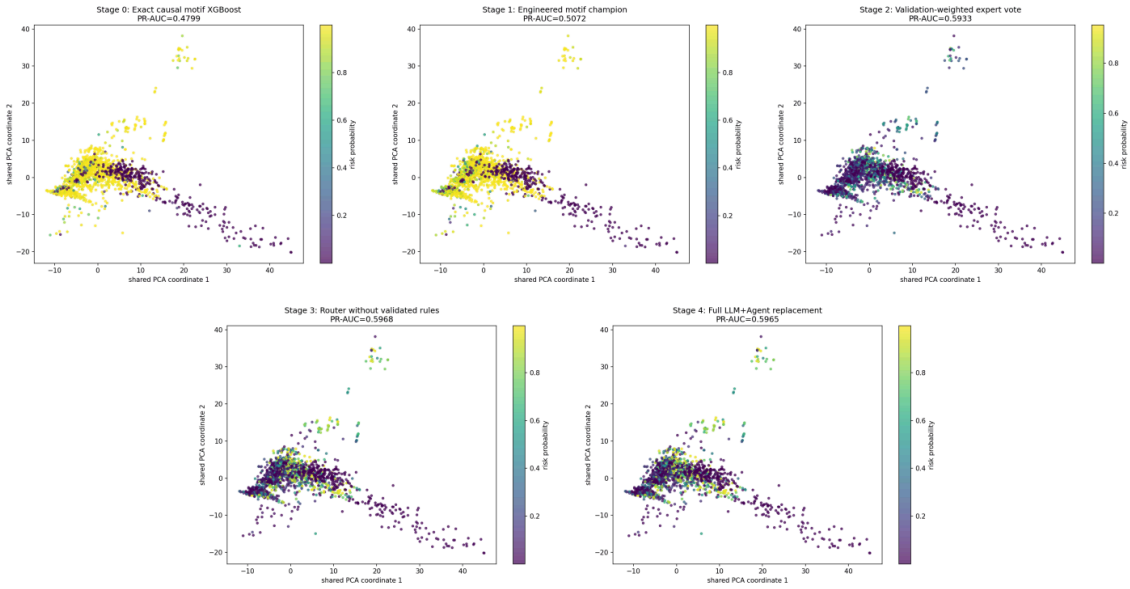


Figure 6. Composite view of model evolution from the explicit-motif expert to the complete framework

DOI: 10.54254/2754-1169/2026.39082

Figure 6 recorded PR-AUC across five stages. Stage 0 was Exact Causal Motif XGBoost; Stage 1 added engineered motif features; Stage 2 applied validation-weighted expert voting; Stage 3 used a router without rule features; and Stage 4 added the complete rule branch.

The stage-wise results further identified multi-expert ensembling as the dominant source of improvement. PR-AUC increased by 0.1166 from Stage 0 to Stage 4, with the gains at Stages 1 and 2 accounting for approximately 73.8% of the total. The router without rule features added only 0.0035, and adding rule features slightly reduced PR-AUC from 0.5968 to 0.5965.

4.4. Feature-importance analysis

4.4.1. Router feature importance

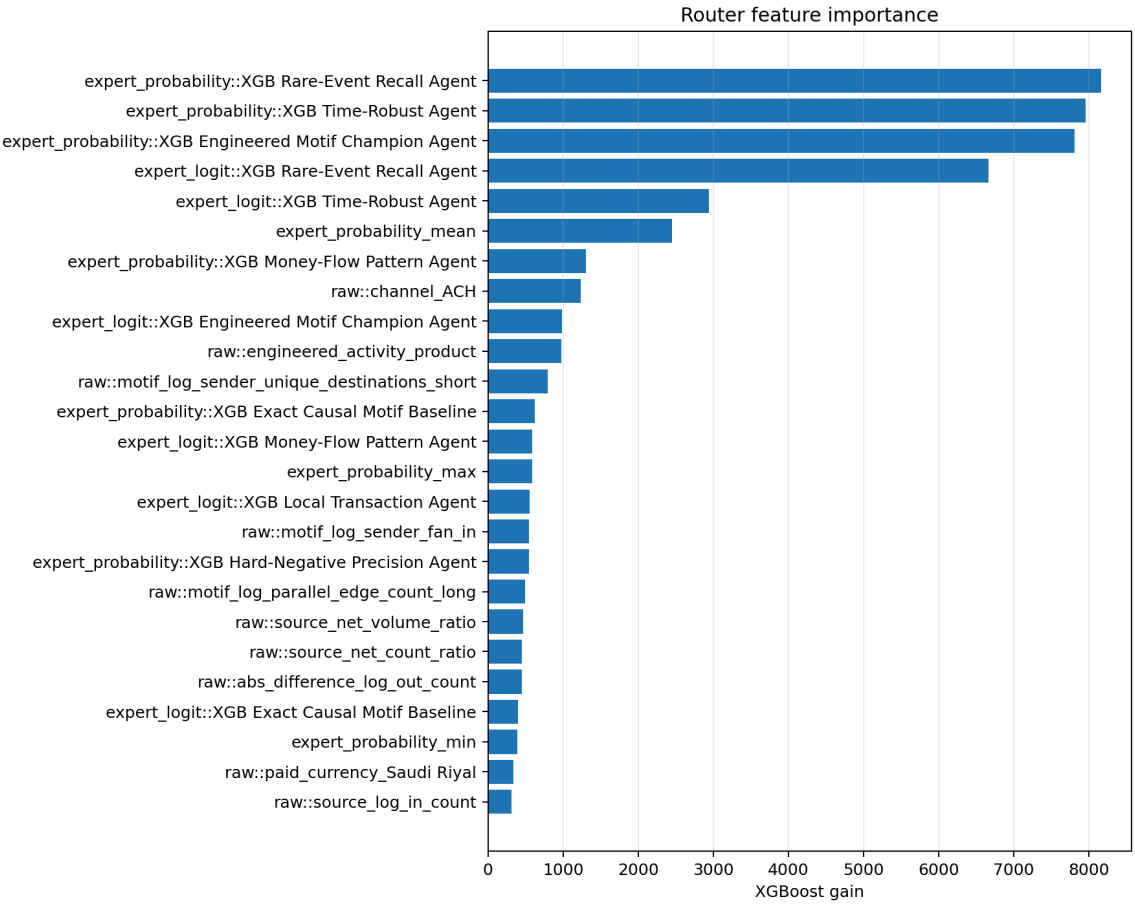


Figure 7. Router feature importance

Expert probabilities and their logit transformations dominated the current router's gain-importance ranking. The probabilities from Rare-Event Recall, Time-Robust, and Engineered Motif Champion ranked first, second, and third, respectively, while the logits of Rare-Event Recall and Time-Robust were also highly important. Although Hard-Negative Precision was the best individual detection expert and was selected as the anchor, its probability was not highly ranked by the router. Overall PR-AUC of an individual expert was therefore not equivalent to its marginal contribution to routing: the anchor expert provided stable fallback behavior, whereas other experts might contribute complementary information for particular instances.

DOI: 10.54254/2754-1169/2026.39082

Among raw features, the ACH channel, account-activity product, short-window number of unique payees, long-window parallel-edge count, and account net-flow ratio ranked highly. The fitted router therefore used transaction context in addition to expert scores. Validated rules did not appear among the leading features in the figure, consistent with the absence of a PR-AUC improvement from the rule branch.

Probabilities and their logits were monotonically related. Including both in a tree model created substantial redundancy and might distribute gain importance across correlated variables. Figure 4-7 therefore described the signals used by the fitted model but did not imply that these variables causally affected money laundering; direct comparisons of individual gain values were also considered inappropriate across highly correlated variables.

4.4.2. Transaction-local and amount features

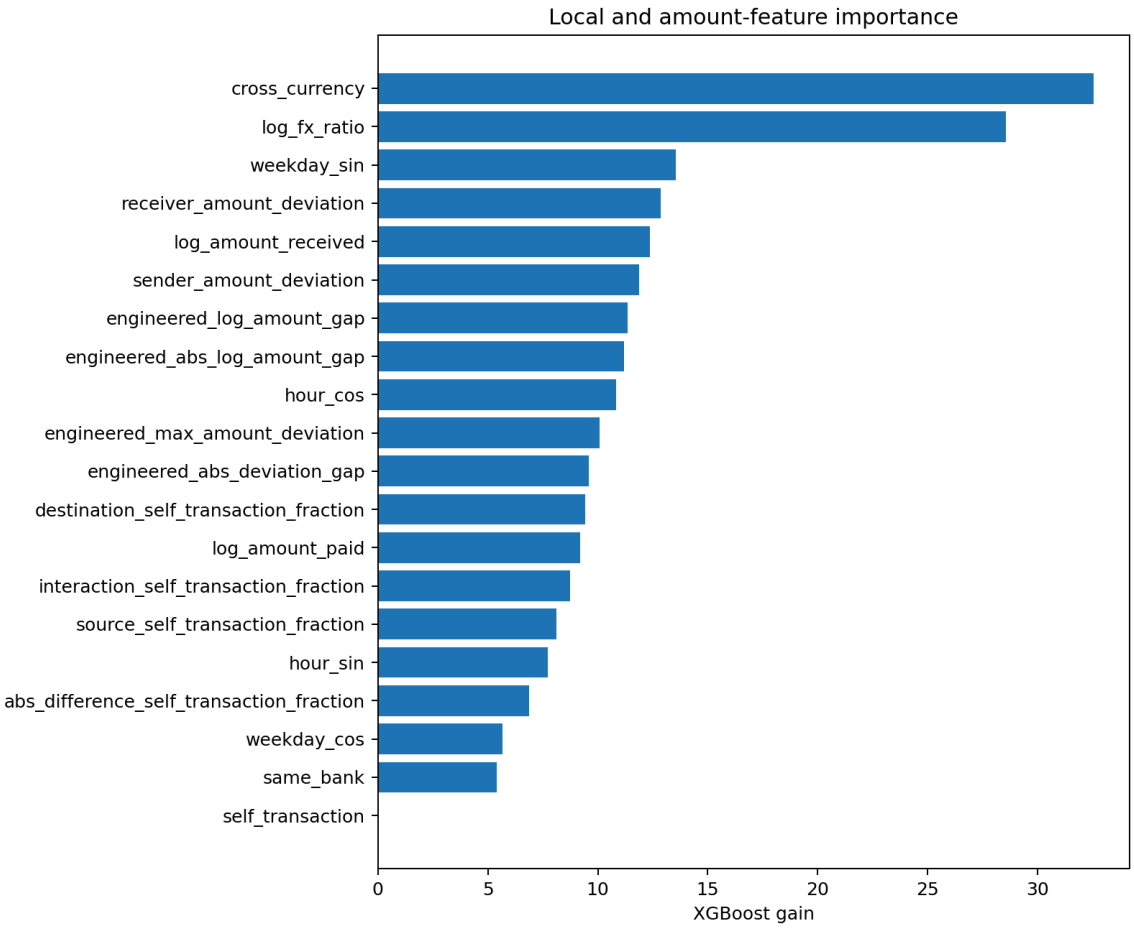


Figure 8. Importance of transaction-local and amount features

Figure 8 showed that the cross-currency indicator and log foreign-exchange ratio had the highest gain importance in the current model, exceeding the remaining transaction-local features. In HI-Small, currency conversion and the ratio between payer and payee amounts were therefore strongly associated with the laundering label. Payee amount deviation, payee amount, payer amount deviation, and amount-difference features were also important, indicating that deviations from an account's historical amount level provided additional discriminative information.

DOI: 10.54254/2754-1169/2026.39082

Day-of-week and hour cyclic features were also highly ranked, indicating temporal distribution differences in HI-Small. These signals might reflect the activity times of laundering transactions, the synthetic data-generation mechanism, or distribution shift between training and testing. The importance of temporal features therefore required interpretation together with the Time-Robust expert and was not considered a basis for extrapolation to a universal laundering pattern.

The self-transfer indicator had near-zero gain importance, suggesting limited marginal value in the current model and dataset. This did not imply that self-transfers lacked risk significance in real operations, because tree-model gain was also affected by feature sparsity, correlations, and substitute variables.

4.4.3. Motif and fund-flow features

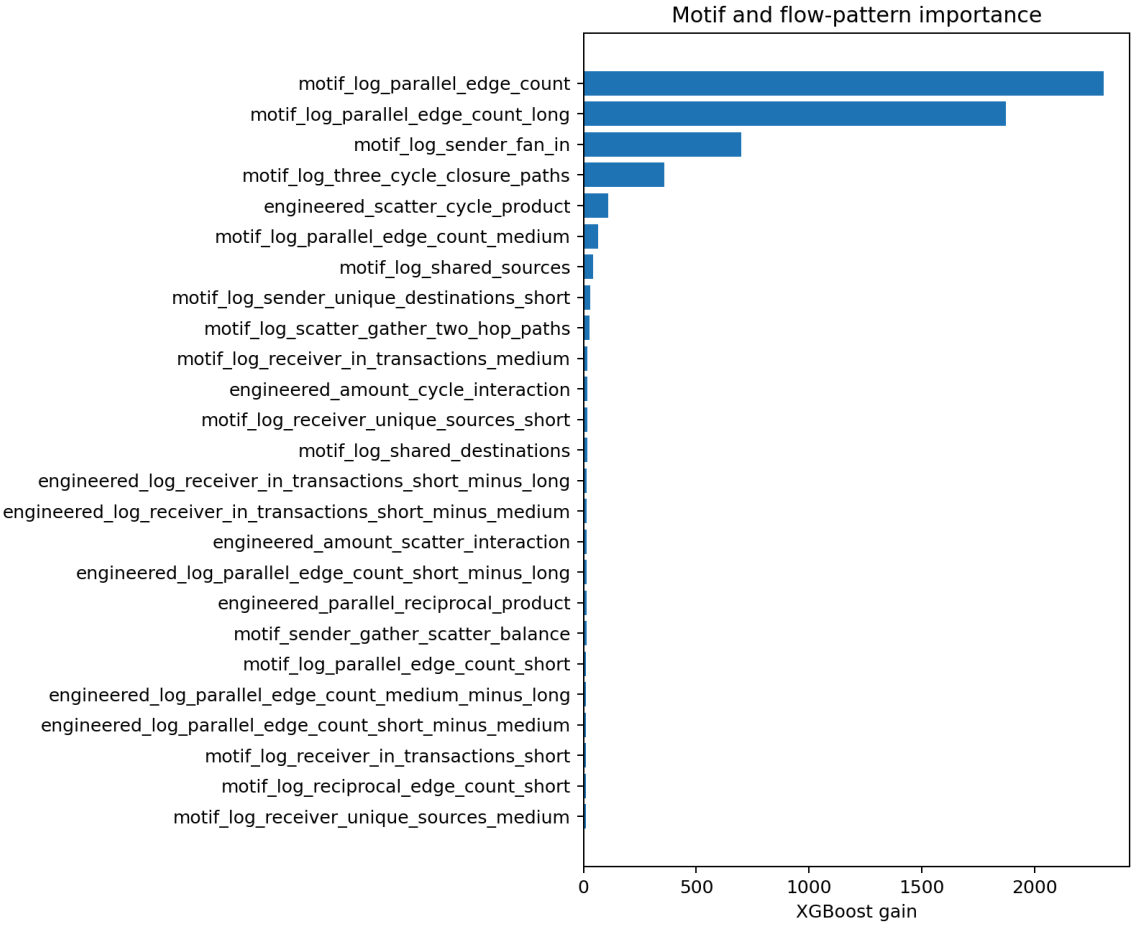


Figure 9. Importance of motif and fund-flow features

Among motif and fund-flow features, the full-history parallel-edge count and long-window parallel-edge count had the highest gain importance, followed by payer-account fan-in and three-node cycle paths. Repeated transfers between the same account pair, particularly persistent repetition over longer timescales, were therefore prominent network signals for the current model in HI-Small; payer fan-in and cycle paths additionally represented fund aggregation and circular transfer.

Individual medium- and short-window features and engineered interactions had lower gain, but this did not establish that they were uninformative. Correlated motifs might describe similar

DOI: 10.54254/2754-1169/2026.39082

structures, concentrating importance in a small subset of variables, and engineered features might be useful only to particular detection experts or local subsets of transactions. This interpretation was consistent with the stage-wise analysis: engineered features provided a limited gain, whereas validation-weighted expert voting contributed substantially more.

4.5. Overall discussion

The results showed that explicit transaction-network motifs enabled tree models to use relational structures such as repeated transfers, fund aggregation, and circular movement. Compared with single-transaction attributes and aggregated account statistics, these structural features more closely represented the organization of multi-account, multistep fund flows. Validation-weighted expert voting further reduced biases associated with any single feature set, class weight, or negative-sampling strategy and was therefore the primary performance source in the complete framework.

The router and rule branch warranted a more limited interpretation. The shallow router provided only a small incremental gain, suggesting that the expert ensemble already established a strong base ranking and left little additional routing capacity. The rule branch did not improve PR-AUC; in the present experiment, validated rules were therefore better viewed as auditable auxiliary signals than as a primary source of predictive performance. Because the experiment used one synthetic dataset and one temporal split, these conclusions explained mechanisms within the framework but could not yet be generalized directly to operational environments.

5. Conclusions and recommendations

5.1. Conclusions

This study developed an ensemble framework for temporally ordered AML transaction detection under extreme class imbalance. The framework combined point-in-time-constrained features, heterogeneous tree-based detection experts, a rule-generation agent, a rule-validation agent, and a validation-gated router. On the common temporal test set of HI-Small, the complete framework achieved PR-AUC, F1, and recall values of 0.5965, 0.5532, and 0.4299, respectively, yielding the highest risk-ranking performance among the models compared in this study. Stage-wise analysis identified explicit transaction-network motifs and validation-weighted expert voting as the principal sources of performance, with only a small gain from the shallow router; the rule branch did not improve PR-AUC. The evidence therefore supported using an LLM as a candidate-rule-generation tool rather than treating it as the direct source of overall detection performance.

The scope of these findings was correspondingly limited. The experiments used only the synthetic HI-Small dataset, one temporal split, and one run; they did not provide evidence across random seeds, confidence intervals, datasets, or operational banking environments. The GIN-family baselines used here also did not represent all graph-neural-network methods. The findings were therefore regarded as preliminary evidence about framework mechanisms and the relative contributions of its modules. This study recommended rolling temporal tests, stability analyses, and fair comparisons with strong baselines on other public or operational datasets. It also recommended assessing the practical value of the rule module under fixed alert budgets, manual review, and audit requirements.

DOI: 10.54254/2754-1169/2026.39082

5.2. Recommendations

First, this study recommended prioritizing detection pipelines built around point-in-time-constrained transaction-network motifs and multi-expert ensembling in subsequent prototype evaluations. In the experiments, explicit-motif XGBoost and validation-weighted expert voting outperformed the GIN-family baselines considered here, but this result was specific to the data, features, and training configuration and did not invalidate other graph-neural-network methods. This study recommended close examination of structural features such as parallel edges, fan-in/fan-out, and cycles, together with fair comparisons against strong baselines under identical temporal splits, feature budgets, and hyperparameter-tuning budgets.

Second, this study recommended restricting the LLM to candidate-rule generation and auxiliary analysis while retaining an independent programmatic validation gate. Because the rule branch did not improve PR-AUC and this study did not evaluate direct transaction-level scoring by the LLM, the results did not justify incorporating the LLM into transaction-level decisions. The proposed workflow involved providing the LLM only with aggregated error evidence and requesting a limited number of candidate rules in a fixed format. The proposed role of the rule-validation agent was to fit thresholds, evaluate support and lift, and perform structural deduplication in isolated temporal subintervals. Eligibility for subsequent model evaluation was restricted to rules that satisfied prespecified out-of-time validation criteria.

Third, this study recommended strict point-in-time isolation in deployment research and staged evaluation of transfer from synthetic data to operational settings. The proposed requirements for operational prototypes included chronological updates to account statistics and transaction-network motifs, the use of information available before each prediction, and retention of feature snapshots, rule-hit records, and model versions. Given that this study used only synthetic data, it recommended initial evaluation through shadow deployment and manual review without directly affecting business decisions. The proposed evaluation covered recall at a fixed alert budget, false-alert volume, temporal stability, and review cost, followed by model recalibration using feedback from subsequent investigations.

References

[1] Financial Action Task Force. (2021). *Opportunities and challenges of new technologies for AML/CFT*. FATF. https://www.fatf-gafi.org/en/publications/Digitaltransformation/Digital-transformation.html

[2] Jensen, R. I. T., & Iosifidis, A. (2023). Fighting money laundering with statistics and machine learning. *IEEE Access, 11*, 8889–8903. https://doi.org/10.1109/ACCESS.2023.3239549

[3] Castelao-López, J., Corzo Santamaría, T., & Lagoa-Varela, D. (2025). Analysis of the main techniques and tools to combat money laundering: A review of the literature. *Journal of Money Laundering Control, 28*(4–5), 645–664. https://doi.org/10.1108/JMLC-10-2024-0159

[4] Mousavian, S., & Miah, S. J. (2025). Review of artificial intelligence-based applications for money laundering detection. *Intelligent Systems with Applications, 27*, 200572. https://doi.org/10.1016/j.iswa.2025.200572

[5] Cardoso, M., Saleiro, P., & Bizarro, P. (2022). LaundroGraph: Self-supervised graph representation learning for anti-money laundering. *Proceedings of the Third ACM International Conference on AI in Finance*, 130–138. https://doi.org/10.1145/3533271.3561727

[6] Lu, H., & Wang, H. (2024). Graph contrastive pre-training for anti-money laundering. *International Journal of Computational Intelligence Systems, 17*, 307. https://doi.org/10.1007/s44196-024-00720-4

[7] Altman, E., et al. (2023). Realistic synthetic financial transactions for anti-money laundering models. *Advances in Neural Information Processing Systems, 36*, 29851–29874.

[8] Fronzetti Colladon, A., & Remondi, E. (2017). Using social network analysis to prevent money laundering. *Expert Systems with Applications, 67*, 49–58. https://doi.org/10.1016/j.eswa.2016.09.029

DOI: 10.54254/2754-1169/2026.39082

[9] Starnini, M., Tsourakakis, C. E., Zamanipour, M., et al. (2021). Smurf-based anti-money laundering in time-evolving transaction networks. In: *Machine Learning and Knowledge Discovery in Databases, ECML PKDD 2021*, 171–186. https://doi.org/10.1007/978-3-030-86514-6_11

[10] Blanuša, J., Cravero Baraja, M., Anghel, A., et al. (2024). Graph feature preprocessor: Real-time subgraph-based feature extraction for financial crime detection. *Proceedings of the Fifth ACM International Conference on AI in Finance*, 222–230. https://doi.org/10.1145/3677052.3698674

[11] Dong, Y., Wu, F., Zhang, K., et al. (2025). Large language model agents in finance: A survey bridging research, practice, and real-world deployment. *Findings of the Association for Computational Linguistics: EMNLP 2025*, 17889–17907. https://doi.org/10.18653/v1/2025.findings-emnlp.972

[12] Pirmorad, E. (2025). Exploring the in-context learning capabilities of LLMs for money laundering detection in financial graphs. arXiv:2507.14785. https://arxiv.org/abs/2507.14785

[13] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

[14] Qwen Team. (2025). Qwen3-4B-Instruct-2507 model card. *Hugging Face*. https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507

[15] Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers* (pp. 61–74). MIT Press.