National College of Ireland

A Hybrid Active Learning GraphBoost Approach for Anti-Money Laundering in Cryptocurrency Transactions

MSc Research Project
FinTech

Chaolu
Student ID: 24110442

School of Computing
National College of Ireland

Supervisor: Dr. Noel Cosgrave

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Chaolu
Student ID:	24110442
Programme:	FinTech
Year:	2025
Module:	MSc Research Project
Supervisor:	Dr. Noel Cosgrave
Submission Due Date:	15/9/2025
Project Title:	A Hybrid Active Learning GraphBoost Approach for Anti-Money Laundering in Cryptocurrency Transactions
Word Count:	6753
Page Count:	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	*Chaolu*
Date:	10th September 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Hybrid Active Learning GraphBoost Approach for Anti-Money Laundering in Cryptocurrency Transactions

Chaolu
24110442

Abstract

Cryptocurrencies present significant anti-money laundering challenges due to their pseudonymity, evolving laundering tactics, and severe class imbalance. Traditional rule-based and static machine learning methods struggle to adapt to these dynamic transaction networks. This research introduces a Hybrid Active Learning GraphBoost Framework to address these limitations. The approach integrates graph neural networks with active learning to model complex transaction topologies while minimizing labeling costs. Key innovations include: Active learning strategies to prioritize high-value transactions for expert annotation, reducing labeling effort while improving detection. GraphBoost Model, a stacking ensemble combining Graph Neural Networks with Random Forest via a Logistic Regression meta-learner, leveraging both graph structural and node feature information. Comprehensive benchmarking showing GraphBoost outperforms standalone models, achieving a 0.9885 F1-score, 0.9957 AUC-ROC, and 0.9794 AUC-PR on the Elliptic Bitcoin dataset. Experiments confirm the superiority of graph-aware models over feature-based baselines. The Active Learning framework enhances a base GCN model's F1-score from 0.9478 to 0.9492 with only a 2.68% expansion in labeled data. The refined GraphBoost Model identifies 11,169 potentially illicit unlabeled transactions, demonstrating scalability and robustness for real-world AML deployment.

1 Introduction

The rapid proliferation of cryptocurrency ecosystems represents one of FinTech's most transformative innovations, enabling decentralized finance and borderless transactions. However, this disruptive potential is paralleled by significant regulatory challenges, particularly in anti-money laundering (AML) compliance.

Globally, UN (2025) estimated that between 2% and 5% of the world's GDP is laundered annually. While the majority of these funds flow through traditional financial systems, cryptocurrencies present a novel and complex channel for illicit finance. Criminals exploit pseudonymity and cross-chain interoperability to launder substantial sums through techniques like Chain Hopping, where funds are moved across different blockchains to obscure their origin; Peeling Chains, which involve siphoning off small amounts from a large wallet through a complex series of transactions; and the use of

Algorithmic Mixers, which pool and scramble funds from multiple users to break the transaction trail (Gong et al.; 2023; Almeida et al.; 2023; Mariani and Homoliak; 2025).

Traditional AML systems, which often depend on rule-based heuristics and static machine learning (ML) models, are ill-equipped to adapt to these evolving and intricate transaction topologies (Liu et al.; 2025). A primary challenge is the severe scarcity of labeled data for illicit activities. In prominent academic datasets, such as the Elliptic dataset, transactions labeled as illicit can constitute as little as 2% of the total data, which significantly hampers the training of effective supervised learning models (Weber et al.; 2019). This gap imposes unsustainable operational costs on financial institutions and undermines regulatory effectiveness, highlighting an urgent FinTech imperative: developing scalable, adaptive AI-driven detection frameworks that can learn from these complex, evolving patterns in real-time (Liang et al.; 2025).

Recent advances leverage graph neural networks (GNNs) to model transactional relationships as dynamic networks. While GNNs effectively capture structural anomalies indicative of illicit behavior (Wu et al.; 2021), they face critical FinTech-specific constraints. These include temporal dynamics, as money laundering patterns and adversarial tactics evolve across extended time horizons, requiring models that can adapt to concept drift without complete, continuous retraining (Di Gennaro et al.; 2024).

Another significant barrier is label inefficiency. The expert-led annotation of illicit transactions is prohibitively costly, time-consuming, and often delayed, resulting in a scarcity of labeled data for training robust supervised models (Wang et al.; 2019). Furthermore, a major challenge is operational scalability. The computational overhead of GNNs makes real-time inference infeasible for large-scale financial networks that can process over 1 million daily transactions, demanding highly optimized and efficient system architectures (Deprez et al.; 2025).

Existing solutions in academic literature typically address these challenges in isolation. For example, temporal GNNs designed to capture evolving patterns often overlook active learning (AL) strategies to reduce labeling costs, and hybrid frameworks that combine different techniques frequently lack validation for real-world, large-scale efficiency and performance (Liu et al.; 2025).

1.1 Research Question

How effectively can an AL framework, which leverages graph models, be used to obtain labels to address imbalance labeling and improve AML detection in cryptocurrency transactions?

1.2 Research Objectives

This research introduces an integrated AL and GraphBoost framework for cryptocurrency AML. It contributes to FinTech through:

- An AL framework that leverages graph models to strategically acquire labels, aiming to reduce labeling costs and improve detection performance over static training baselines. This includes an ablation study evaluating various AL query strategies (Random, Uncertainty, Centrality, and their combined variants) to provide insights into their effectiveness in selecting informative nodes for labeling.

- Benchmarking of various Graph Neural Networks (GNNs), including GCN, GAT, and GraphSAGE, alongside a traditional model (Random Forest), demonstrating the superior performance of graph-aware models in cryptocurrency AML.
- Development and evaluation of a GraphBoost hybrid model, which combines graph and feature-based learning via a stacking ensemble, achieving the highest performance among all benchmarked models.

The findings suggest that integrating AL with powerful graph-based and hybrid models is a promising direction for improving AML detection in dynamic and partially labeled transaction networks.

The paper is structured as follows. Section 2 reviews graph-based AML and AL. Section 3 outlines the methodology. Section 4 details the architectural design. Section 5 presents the implementation. Section 6 discusses the evaluation. Section 7 summarises the conclusion.

2 Related Work

This section reviews advancements and challenges in graph-based and AL approaches for detecting illicit cryptocurrency activities. Graph-based models, including Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and Temporal GNNs (TGNs), leverage transaction structures and temporal dynamics for improved anomaly and money laundering detection (Scarselli et al.; 2009; Kipf and Welling; 2016; Veličković et al.; 2018; Rossi et al.; 2020). AL addresses data scarcity by prioritizing high-value transactions for annotation, reducing labeling costs and false positives (Settles; 2009). Key challenges include the computational burden of large, dynamic graphs, severe class imbalance, the scarcity and delayed nature of ground-truth labels, and the continuous evolution of sophisticated money laundering tactics that evade current detection methods.

2.1 Graph Neural Network Detection Systems

Graph-based models have become central to detecting illicit cryptocurrency activities like money laundering, leveraging the inherent transaction graph structure. Early approaches focused on extracting structural features from Bitcoin transaction networks for classification. For instance, Alarab et al. (2020) explored GCNs on the Elliptic dataset, highlighting challenges like network complexity but demonstrating GCNs' potential. To address heterophily in Ethereum transaction networks where dissimilar nodes connect, Huang et al. (2022) proposed EH-GCN, which incorporates both homophilic and heterophilic neighborhood information via bi-path neighbor sampling and attention, improving account classification performance.

Capturing temporal dynamics is crucial. Zheng (2022) combined Bidirectional GRUs to model sequential transaction features with GATs to weight spatial features, enhancing anomaly detection in Bitcoin. Similarly, Di Gennaro et al. (2024) introduced Amatriciana, a TGNN model designed to process the entire evolving transaction graph without splitting it into discrete time windows, significantly reducing false positives in money laundering detection on Bitcoin. Ensemble approaches combining graph models have also emerged; Li et al. (2024) presented BELFAL, a blockchain-based framework enabling flexible scheduling and parallel execution of heterogeneous graph models like GCNs for improved AML prediction.

Improving model interpretability remains a key challenge. Yu et al. (2023) developed AEtransGAT, a mutual learning-based GNN framework combining a transaction-aware GAT encoder to weight surrounding transactions based on flow amounts and a Graph Autoencoder decoder to capture structural information, aiming for better understanding alongside detection accuracy on the Elliptic dataset. These graph-based methods increasingly exploit the rich relational and temporal patterns within blockchain transaction data for AML tasks.

2.2 Active Learning Approaches

AL approaches have emerged as a pivotal strategy to address data scarcity and labeling costs in Bitcoin AML, leveraging uncertainty sampling and reinforcement learning to prioritize high-value transactions for expert annotation. Alarab and Prakoonwit (2023) pioneered this by integrating Monte Carlo adversarial attacks with graph-based LSTM models, using Bayesian uncertainty estimates to select ambiguous transactions in the Elliptic dataset, thereby improving illicit detection by 7.8% with 40% fewer labels. Wang et al. (2024) advanced this through GraphALM, a reinforcement learning framework that dynamically samples high-uncertainty nodes while incorporating feature-based risk scores; their method reduced false positives by 55% on simulated laundering patterns.

Complementing these, Karim et al. (2024) combined semi-supervised graph embeddings with AL, using topological centrality metrics to identify critical nodes for labeling, which boosted F1-scores by 20% on the AMLSim dataset with only 30% labeled data. These approaches collectively tackle concept drift and label sparsity while enhancing adaptability to evolving laundering tactics (Yang et al.; 2023), demonstrating that strategic querying of blockchain data can optimize compliance efficiency without compromising detection accuracy.

2.3 Open Challenges

Cryptocurrency transaction networks form complex, dynamic graphs with billions of nodes (addresses) and edges (transactions), posing significant computational and analytical hurdles (Wu et al.; 2021). As noted by Rossi et al. (2020), temporal graph networks struggle to efficiently capture evolving transaction patterns across extended time horizons due to memory constraints. Subgraph sampling techniques offer partial solutions but risk omitting critical long-range dependencies essential for identifying sophisticated laundering schemes, such as multi-hop transaction chains (Li et al.; 2023). Additionally, real-time analysis remains infeasible for networks like Ethereum, which processes $\sim$1.2 million daily transactions (Zheng et al.; 2020), necessitating distributed computing frameworks that current GNN architectures do not fully leverage.

Labelled illicit transactions are exceptionally rare, typically $<$2% of datasets, creating severe class imbalance that biases models toward majority (licit) classes. Semi-supervised approaches attempt to mitigate this by generating synthetic illicit samples, but often produce unrealistic patterns that degrade model generalizability (Alarab et al.; 2020). AL strategies improve annotation efficiency but require costly expert verification (Alarab and Prakoonwit; 2023). Furthermore, ground-truth labels derived from law enforcement cases suffer from temporal delays, causing concept drift where models trained on historical patterns fail to detect novel laundering tactics (Vassallo et al.; 2021). This scarcity also impedes cross-chain analysis, as labels from Bitcoin cannot directly transfer to Ethereum

due to structural differences in smart contract-enabled transactions (Chen et al.; 2019).

Money launderers continuously adapt strategies, such as *chain hopping* (cross-exchange transfers), *peeling chains*, and algorithmic mixing services, to evade detection (Sun et al.; 2022). These techniques generate Structurally Camouflaged transactions that mimic legitimate patterns. For instance, tumblers fragment transactions across hundreds of addresses, obscuring fund origins through high-volume micro-transactions (Rathore et al.; 2022). GNNs struggle to distinguish such behaviors without explicit temporal signals, as static graph embeddings ignore transaction sequencing critical for identifying layering phases (Zheng; 2022). Additionally, adversarial laundering techniques exploit model vulnerabilities; attackers inject Noise Transactions to distort neighborhood aggregations (Yang et al.; 2023). While methods like dynamic temporal convolution improve pattern capture (Guo et al.; 2023), they cannot generalize to unseen laundering topologies, such as decentralized mixer protocols.

3 Methodology

This section outlines the comprehensive methodology employed for developing an AML detection framework that integrates GNNs with AL. It covers the stages from data preprocessing and graph construction to AL-driven label acquisition, concluding with a detailed benchmarking and refinement process for various graph and traditional models, as conceptually illustrated in Figure 1.

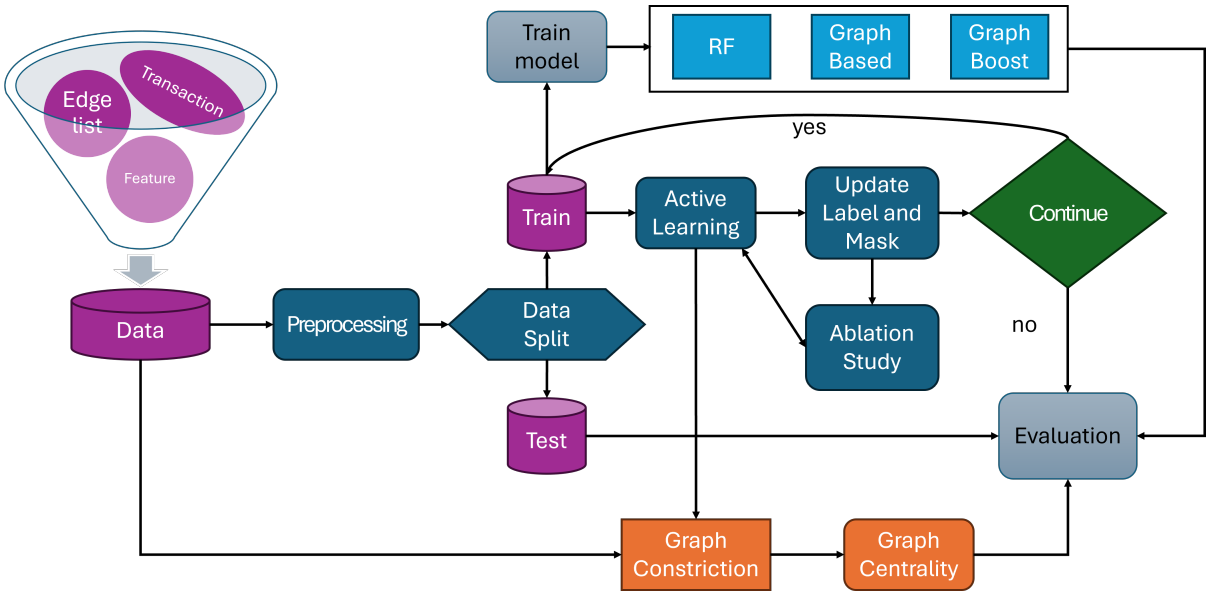

Figure 1: Integrated AML detection framework

3.1 Dataset

From Table 1, the Elliptic Bitcoin dataset [1] consists of three interconnected components 203,769 transactions, each with a transaction ID (txID), a temporal information (timestep) and 165 anonymized features; 234,355 directed edges that show payment flows;

[1] https://www.kaggle.com/datasets/ellipticco/elliptic-data-set

Table 1: The Elliptic Dataset Composition Breakdown

Transaction Category		Class	Count
Total Transactions	**Known Transactions**	licit	42,019
		illicit	4,545
	Unknown Transactions		157,205
		Total	203,769
Transaction Edges			234,355

and a three-class labeling system (illicit, licit, and unknown). This dataset has no missing values across any of its components, which helps ensure data integrity for modeling. The dataset is composed of three distinct CSV files:

- Elliptic_txs_classes.csv, providing labels for each transaction;
- Elliptic_txs_edgelist.csv, details the connections between transaction nodes;
- Elliptic_txs_features.csv, contains transaction features along with temporal information.

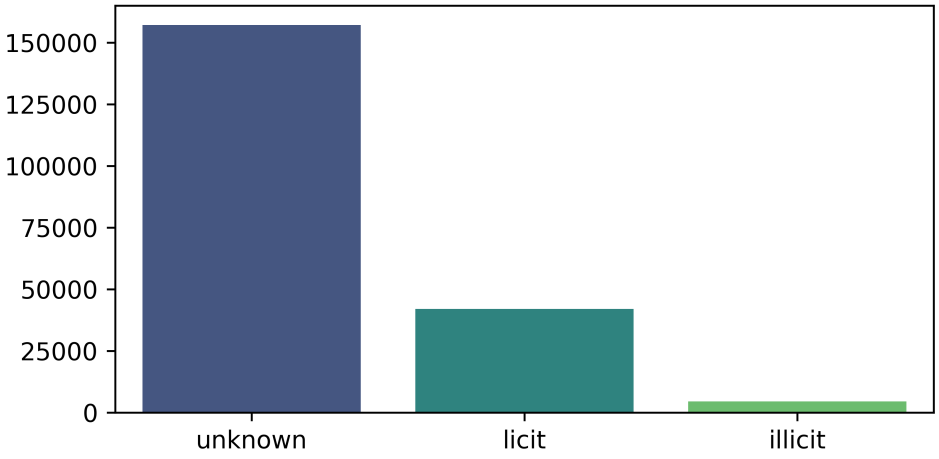

Figure 2: Class distribution of known transactions

As shown in Figure 2, the dataset exhibits significant class imbalance. Among the labeled transactions, Licit transactions account for 42,019 instances (90.2%), while Illicit transactions comprise 4,545 instances (9.8%). Additionally, there are 157,205 unknown transactions, which make up 77.2% of the total dataset. This pronounced imbalance, with illicit transactions representing only a small fraction of the labeled data, necessitates specialized modeling approaches for effective fraud detection.

The labeled transactions were partitioned using stratified sampling to preserve the original class imbalance across all splits, ensuring evaluation reflects real-world detection challenges. Specifically, the known transactions were split into an initial training set, an AL pool, and a test set.

To enable ML tasks, the following preprocessing steps were performed. Features with class labels were merged using the transaction ID (txId) as the key, transformed the class labels into a binary format (1 for illicit and 0 for licit), and filtered out unknown transactions, which resulted in 46,564 labeled and 157,205 unlabeled transactions. Finally, the graph topology was constructed by representing transactions as nodes and payment flows as directed edges. Node features were derived from the transaction attributes to form a feature matrix, providing a comprehensive representation for subsequent analysis.

Feature scaling ensures numerical stability during model training while preserving the relative relationships between transaction characteristics critical for fraud pattern detection. Each transaction is described by 165 anonymized features and temporal information. The provided features were already standardized using z-score normalization.

The edge representation is crucial for detecting money laundering patterns, as illicit activities often manifest as complex multi-hop flows within transaction relationships. The transaction network was represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Depending on the analysis, this representation was treated as directed, reflecting payment flows, or undirected, for structural analyses. The graph contains 203,769 transaction nodes ($\mathcal{V}$) and 234,355 edges ($\mathcal{E}$).

Analysis of the transaction network's structure revealed scale-free characteristics, supported by the heavy-tailed degree distribution shown in Figure 3. The full network exhibits an average degree of 2.30 and a maximum degree of 473. Analyzing a sampled subgraph of the network showed it is composed of 7,297 connected components, among which the giant component contains 400 nodes.

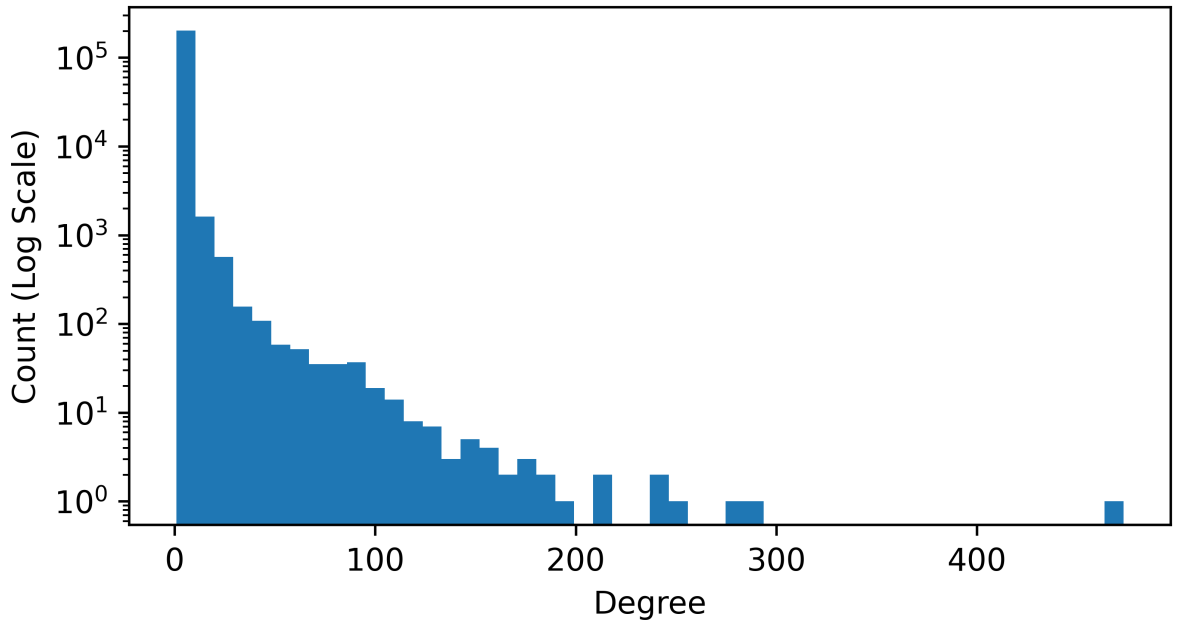

Figure 3: Log-log degree distribution of transaction graph

3.2 Active Learning Framework

An AL framework was designed to iteratively label the most informative transactions and improve model performance. The AL framework is implemented via the ActiveLearner class, which orchestrates querying strategies, training, and evaluation cycles.

The ActiveLearner is initialized with a PyTorch Geometric Data object containing features, labels, and the edge index. It also requires boolean masks identifying the initial training set, the unlabeled pool (AL pool), and the mask for all originally known (labeled) nodes, alongside a query budget per iteration (default = 100) and a Query Strategy controlling how new samples are selected for labeling. Centrality-aware querying was incorporated to enhance the informativeness of the querying process. Specifically, betweenness centrality was calculated on the full transaction graph using NetworkX, normalized the scores to $[0, 1]$, and stored them within the data object. This centrality metric is later used to prioritize nodes that act as bridges in the transaction network.

The ActiveLearner supports several strategies for selecting transactions to label. These include Random, which randomly samples from the unlabeled pool; Uncertainty, which selects nodes with the highest predictive uncertainty, calculated as $1-\max(\text{probability})$; and Centrality, which selects nodes with the highest centrality scores. Additionally, a Combined strategy fuses uncertainty and centrality scores using one of three fusion methods. Multiplication, Weighted Sum with tunable weights, or Geometric Mean. This design enables empirical assessment of how structural and statistical informativeness affect learning outcomes.

After each iteration, a subset of nodes is queried according to the selected strategy. The model's predicted probabilities and the precomputed centrality scores are used to rank nodes in the unlabeled pool. Queried nodes are then matched to the true labels by 'looking up the preloaded ground truth' based on their transaction ID, and their labels are updated in the training set masks.

3.3 Models

To integrate both graph-structural and tabular data perspectives, a hybrid model named GraphBoost was implemented. This architecture combines a GNN, GCN in this case, to learn relational embeddings, a traditional ML model, Random Forest, to learn from raw node features, and a meta-learner, Logistic Regression, that takes the concatenated output probabilities of both base models and predicts the final label.

Within the GraphBoost training process, the GNN was first trained on the provided training set for a fixed number of epochs (100 epochs), and its softmax probabilities were collected for all nodes. Concurrently, the Random Forest model was trained using raw node features from the same training set.

Then, for each node in the training set, the predicted probabilities from the trained GNN and the trained Random Forest model were concatenated to form meta-features. These meta-features were used to train the logistic regression meta-learner, which serves as the final classifier. This ensemble approach aimed to leverage the strengths of both graph-based and feature-based learning, potentially boosting classification performance when either type of model alone may underperform.

Three types of models were compared in this study. The first category comprises Traditional ML Models, exemplified by the Random Forest Classifier. These models were trained only on raw node features, serving as a feature-based baseline without considering graph connectivity. The second category consists of GNN Models, specifically GCN, which uses neighborhood aggregation; GAT, which introduces attention weights to different neighbors; and GraphSAGE, which learns aggregation functions from neighbor features. The final type is the GraphBoost Model, a stacking ensemble that incorporates both GNN predictions and ML model outputs into a meta-learning framework.

Each model type was trained on the initial static training set and evaluated on the test set to ensure comparability. For the PyTorch Geometric models (GCN, GAT, GraphSAGE), a dynamic validation split was created from the training set for early stopping purposes. This training-evaluation loop was unified and returned the key metrics for analysis.

3.4 Training and Evaluation

Each model was trained using its respective procedure tailored to its architecture. GNNs such as GCN, GAT, and GraphSAGE were trained using supervised learning with the Cross-Entropy Loss function and the Adam optimizer. The learning rate was set to 0.01 with a weight decay of 5×10^{-4}, consistent with standard GNN optimization practices. In the general training function used for benchmarking, to mitigate overfitting and ensure generalization, early stopping was employed based on validation F1-score, with a patience threshold of 10 epochs.

The models are evaluated using five standard metrics to provide a comprehensive assessment of performance, particularly important given the class imbalance.

$$\text{F1} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{1}$$

The F1 score provides a balance between precision and recall, especially useful in imbalanced datasets where simply optimizing for accuracy can be misleading.

$$P = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

Precision measures the proportion of positive identifications that were actually correct, indicating the model's exactness.

$$R = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3}$$

Recall measures the proportion of actual positives that were correctly identified, indicating the model's completeness in finding all positive cases.

$$\text{AUC-ROC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(x))dx \tag{4}$$

Area Under the Receiver Operating Characteristic Curve (AUC-ROC) evaluates the model's ability to distinguish between classes across all possible classification thresholds, robust to class imbalance.

$$\text{AUC-PR} = \int_0^1 P(R)dR \tag{5}$$

Area Under the Precision-Recall Curve (AUC-PR) provides a more informative assessment than AUC-ROC for highly imbalanced datasets, as it focuses on the performance of the positive class Davis and Goadrich (2006).

To assess the performance of each model, 5 evaluation metrics equations (1)–(5) were computed. These include F1-Score (Weighted), which is the harmonic mean of precision and recall, weighted by class support; Precision (Weighted), representing the proportion of correctly predicted positive observations among all predicted positives; and Recall

(Weighted), indicating the proportion of correctly predicted positive observations among all actual positives. Additionally, AUC-ROC, the Area Under the Receiver Operating Characteristic curve, was used to reflect the trade-off between true positive rate and false positive rate, while AUC-PR, the Area Under the Precision-Recall curve, was particularly useful for imbalanced datasets. All metrics were computed using Scikit-learn evaluation functions. For traditional ML models and the hybrid GraphBoost model, output probabilities were used to derive AUC scores. In cases where only a single class was present in either the training or test set, AUC metrics were marked as not applicable (NaN), and fallback evaluations were conducted using only F1, precision, and recall with zero-division handling.

4 Design Specification

This section details the architectural and implementation design of the proposed framework for detecting illicit accounts on Ethereum using GNNs, traditional ML, and AL. The design prioritizes modularity and extensibility, enabling comparative evaluation across various model types and query strategies. The system is built as a hybrid pipeline comprising four key components. Graph-Based Models, which are deep learning architectures (GCN, GAT, GraphSAGE) for learning node representations; GraphBoost, a model fusion framework that combines a GNN (GCN) with a traditional ML model (Random Forest) and a meta-learner (Logistic Regression); an AL Framework, an uncertainty-based iterative querying system that expands the labeled training set in a budgeted, data-efficient manner.

4.1 Active Learning Framework

The AL engine is encapsulated in an ActiveLearner class that controls the labeling budget and querying strategy. Multiple query strategies are supported by the design, including Uncertainty Sampling, which selects nodes with the highest entropy in GNN prediction probabilities; Centrality Sampling, which prioritizes structurally important nodes using graph centrality metrics; and Random Sampling, serving as a baseline approach that selects nodes randomly from the pool. Each AL iteration involves training or fine-tuning the base GCN using the current labeled set, evaluating model performance on a fixed test set, querying the oracle (true labels) for a small batch of unlabeled nodes, and updating the training set with newly labeled nodes before repeating the cycle (Settles; 2009). To promote training stability, early stopping based on validation loss is used with a patience threshold of 10 epochs. The framework was run for 5 iterations, with a total labeling budget of 100 nodes per cycle.

The AL process initiates with the training partition, $\mathcal{L}_0$ containing 27,938 transactions, as the seed labeled set. The validation set remains fixed for model selection, while the 9,313 transactions test set provides final evaluation. $|\mathcal{L}_0| = 27,938 \quad |\mathcal{U}_0| = 157,205 + 18,626$, where $\mathcal{U}_0$ includes unlabeled transactions plus the validation set. Four complementary acquisition functions were implemented for node selection, in Table 2.

Key implementation aspects include approximate Centrality, where Betweenness centrality is computed with $k = \min(100, |\mathcal{V}|)$ samples using Brandes' algorithm for scalability. Normalization involves min-max normalizing centrality scores to $[0, 1]$. For Hybrid Scoring, the combined strategy supports multiplicative, weighted sum, and geometric mean

Table 2: Active learning query strategies

Strategy	**Selection Criterion**	**Complexity**
Random	Uniform random sampling	$\mathcal{O}(1)$
Uncertainty	$1 - \max_y P(y\|\mathbf{x};\theta)$	$\mathcal{O}(n)$
Centrality	Betweenness centrality $C_B(v) = \sum_{s\neq v\neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$	$\mathcal{O}(\|\mathcal{V}\|\|\mathcal{E}\|)$
Combined	$\phi(v) = \begin{cases} \text{Uncertainty} \times \text{Centrality} & \text{(multiplication)} \\ \alpha \cdot \text{Uncertainty} + \beta \cdot \text{Centrality} & \text{(weighted sum)} \\ \sqrt{\text{Uncertainty} \times \text{Centrality}} & \text{(geometric mean)} \end{cases}$	$\mathcal{O}(\|\mathcal{V}\|\|\mathcal{E}\|)$

fusion. Finally, a Budget Constraint is applied, with a fixed query budget of 100 nodes per cycle. The AL cycle operates in Algorithm 1.

Algorithm 1 Active Learning

Require: Initial labeled set $\mathcal{L}_0$, unlabeled pool $\mathcal{U}$, budget $B = 100$

for $t = 1$ **to** 5 **do**
  Train model f_t on $\mathcal{L}_{t-1}$ (50 epochs with early stopping)
  Compute informativeness scores $\phi(v)$ $\forall v \in \mathcal{U}$
  Select $Q_t = \underset{Q\subset\mathcal{U},|Q|\leq B}{\arg\max} \sum_{v\in Q} \phi(v)$
  Obtain labels $Y_t = \text{Oracle}(Q_t)$ via true label lookup
  Update $\mathcal{L}_t = \mathcal{L}_{t-1} \cup (Q_t, Y_t)$
  Update $\mathcal{U} = \mathcal{U} \setminus Q_t$
end for

The iterative process features in the training of the Algorithm 2 model, using a GCN base model with an Adam optimiser, learning rate of 0.01 and weight decay of 5×10^{-4}. Uncertainty quantification is performed via Monte Carlo dropout with $K = 20$ forward passes. An average of 19.2 labels are acquired per oracle query cycle, leading to a label expansion where the training set grows from 27,938 to 28,034 nodes, a 0.34% increase, after 5 iterations.

Algorithm 2 Active Learning Ablation Study

for each strategy $s \in$ {Random, Uncertainty, Centrality, Combined} **do**
  for each fusion method $f \in$ {Multiplication, Weighted Sum, Geometric Mean} **do**
    Reset model parameters θ and labeled set $\mathcal{L}_0$
    for $t \leftarrow 1$ to 5 **do**
      Train M_t on $\mathcal{L}_{t-1}$ (50 epochs)
      Evaluate M_t on test set
      Query $B = 100$ nodes using (s, f)
      Update $\mathcal{L}_t = \mathcal{L}_{t-1} \cup Q_t$
    end for
  end for
end for

The ablation study maintains identical experimental conditions across all strategies: the base model is a GCN with identical initialization, and hyperparameters include a

learning rate of 0.01, an Adam optimizer, and a weight decay of 5×10^{-4}. Evaluation is performed on a fixed test set with original labels. For statistical rigor, three independent runs are conducted per configuration. The combined variants are specifically tested with weighted sums using (α, β) values of $(0.5, 0.5)$, $(0.7, 0.3)$, and $(0.3, 0.7)$.

4.2 Graph Neural Network Models

Three GNN variants were implemented using PyTorch Geometric. The GCN employs a two-layer architecture with a 64-dimensional hidden space and ReLU activation, with dropout applied between layers to mitigate overfitting. The GAT utilizes multi-head attention (8 heads) in the first layer to capture feature importance across neighbors, while the second layer uses a single head with no concatenation. GraphSAGE uses neighborhood aggregation to generalize across unseen nodes, with mean aggregation and ReLU activation, making this model particularly robust for inductive settings.

Table 3: GNN model configurations

Model	**Layer 1**	**Layer 2**	**Activation**
GCN	GCNConv(166, 64)	GCNConv(64, 2)	ReLU
GAT	GATConv(166, 8, heads=8)	GATConv(64, 2, heads=1)	ELU
GraphSAGE	SAGEConv(166, 64)	SAGEConv(64, 2)	ReLU

In Table 3, three state-of-the-art GNN variants were implemented. For each GNN model, a stratified split of the labeled nodes was used to create training and validation masks. Model training was conducted, incorporating optional internal train/validation splitting in case an explicit validation mask was not provided. The best performing model checkpoint, based on validation F1-score, was saved and used for subsequent testing.

To enhance generalization and leverage complementary strengths, a stacking ensemble named GraphBoost from Figure 4 was designed (Finn et al.; 2017). This model includes a base GNN (GCN) trained on labeled graph nodes, a Random Forest model trained independently on raw node features ignoring graph structure, and a Logistic Regression meta-learner that fuses the prediction probabilities from both the GNN and Random Forest to make the final prediction. Training occurs in three stages: first, the GNN is trained using cross-entropy loss; second, the Random Forest is fit on the same training nodes using raw features; and third, the meta-learner is trained on concatenated probability vectors.

A two-level stacking ensemble graph boost model was implemented to combine diverse detection signals. In Level 1 (Base Learners), the predictions $\hat{y}_{\text{GNN}} = f_{\text{GNN}}(\mathcal{G})$ from a Graph Neural Network (GNN) and $\hat{y}_{\text{RF}} = f_{\text{RF}}(\mathbf{X})$ from a Random Forest (RF) model. These predictions are then combined in Level 2 (Meta-Learner), where a logistic regression classifier, $y_{\text{final}} = g\left([\hat{y}_{\text{GNN}} \parallel \hat{y}_{\text{RF}}]\right)$, is trained on the out-of-fold predictions to produce the final output. Model weights are adjusted based on temporal performance:

$$w_t^{(m)} = \frac{\exp(\eta \cdot \text{AUC}_t^{(m)})}{\sum_{m'} \exp(\eta \cdot \text{AUC}_{t'}^{(m')})} \tag{6}$$

where η is a sensitivity parameter (empirically set to 2.0). Cost-sensitive decision thresholds

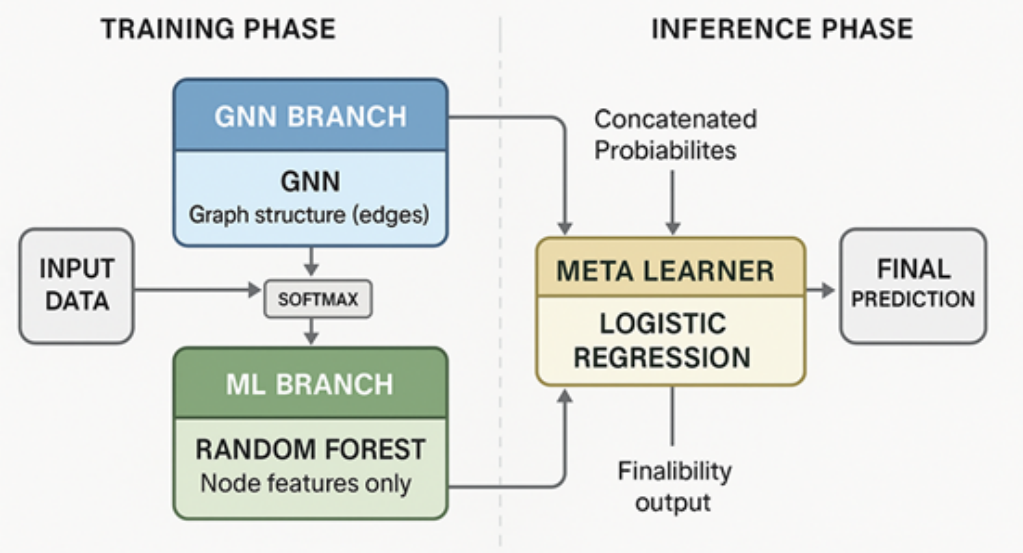

Figure 4: Stacking ensemble architecture with dual processing pathways heterogeneous base models

were implemented:

$$\tau^* = \arg\min_{\tau \in [0,1]} \left(\lambda \cdot \mathrm{FPR}(\tau) + (1-\lambda) \cdot \mathrm{FNR}(\tau)\right) \tag{7}$$

with $\lambda = 0.3$ reflecting the higher cost of false negatives in AML detection.

All models shared common training settings, including the Adam optimizer, a learning rate of 0.01, a weight decay of 5×10^{-4}, and up to 100 epochs with early stopping. Training utilized CUDA when available, otherwise defaulting to the CPU. To ensure fairness, consistent train/validation/test splits and random seeds were applied across all experiments. AL commenced with the original Train Mask, and labels were progressively incorporated from the AL Pool Mask. Following an ablation study, uncertainty sampling was identified as the optimal AL query strategy. This strategy achieved superior performance over centrality, random, and combined weighted sampling, leading to its selection for refining the final GraphSAGE and GraphBoost Model.

5 Implementation

This section details the technical specifications employed for the research, encompassing the systematic preparation of raw transaction data, the design and configuration of the AL framework, the architectural choices for the base and comparative models, and the procedures for model refinement and robust implementation.

5.1 Data Preparation and Experimental Environment

The transaction data were preprocessed to construct a graph. Unique transactions were identified by combining transaction IDs from the features and edgelist datasets, resulting in a total of 203,769 unique transactions. Each transaction was mapped to a contiguous node index to facilitate graph processing. Node features were derived from the features dataset, excluding the transaction ID and timestamp columns. These features, already

normalized, were converted into a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d the number of features. Edges were created by mapping source and destination transaction IDs from the edgelist ddataset to the corresponding node indices, resulting in an edge list represented as $\mathbf{E} \in \mathbb{N}^{2 \times e}$. Node labels were initialized using the labels dataset, with licit transactions assigned a label of 0 and illicit transactions assigned a label of 1, and unknown nodes were set to -1. The complete dataset was encapsulated into a PyTorch Geometric Data object, with auxiliary fields to track raw labels and index mappings for evaluation.

For the AL setup, the dataset was partitioned using stratified sampling to preserve the natural class distribution across splits. The initially labeled transactions where the true class was known were divided into three sets. An initial training set, an AL pool, and a test set. This stratification ensured that the proportion of illicit transactions, approximately 9.75%, was maintained in each of these initial splits.

Specifically, in Table 4, it can be seen from the total of 46,564 known transactions. The Initial Training set contained 18,625 transactions. The AL Pool contained 13,969 transactions. The Test set contained 13,970 transactions. Note that a separate validation set was not created in this initial split but is dynamically generated from the current training data during each AL iteration for model training purposes.

Table 4: Initial Data Partitioning Scheme for Active Learning

Split	**Transactions**	**Approx. Illicit Ratio**
Initial Training	18,625	$\approx 9.75\%$
Active Learning Pool	13,969	$\approx 9.75\%$
Test	13,970	$\approx 9.75\%$

In Table 5. The computational environment predominantly utilized Google Colab's default free CPU, meaning no GPU acceleration was directly applied for model training, although future work could explore hardware-accelerated graph processing. The software stack included PyTorch Geometric, PyTorch, Scikit-learn, XGBoost, ALiPy for AL, and Matplotlib for visualization. Computational efficiency was enhanced through centrality precomputation, where Betweenness centrality was calculated once with $k = 100$ samples and cached.

Table 5: Computational Environment

Component	**Specification**
Hardware	Google Colab (Default Free CPU)
Acceleration	Not applicable (CPU only)
Graph Processing	PyTorch Geometric (latest compatible with Colab)
Deep Learning	PyTorch (latest stable with Colab)
ML Ecosystem	Scikit-learn, XGBoost (latest stable with Colab)
Active Learning	ALiPy (latest compatible with Colab)
Visualization	Matplotlib (latest stable with Colab)

5.2 Active Learning Framework

To reduce labeling cost and improve generalization, an AL framework was implemented, centered around an ActiveLearner class. This class is designed to iteratively: train or fine-tune a base model GCN on the current labeled subset; query the most informative unlabeled samples from a designated pool based on a defined strategy; expand the labeled set with the newly acquired labels; and repeat for a fixed number of iterations or until the unlabeled pool is exhausted. The AL process was initialized with 18,625 labeled nodes in the training set and 13,969 nodes in the query pool. A query budget of 100 nodes per iteration was enforced.

The ActiveLearner class's Query Oracle method identifies candidate nodes from the unlabeled pool mask. For query strategies, it computes predictive probabilities and an uncertainty score, which is one minus the maximum predicted probability. For combined strategies, it can incorporate pre-computed centrality scores, fusing them with uncertainty using methods like multiplication, weighted sum, or geometric mean. The method then selects the top-B candidates based on the chosen strategy's score, retrieves their true labels by looking up the original known labels, and updates the current train mask and unlabeled pool mask in the data object to reflect the expanded labeled set. The baseline AL process in this study primarily utilized the uncertainty sampling strategy.

A two-layer GCN was used as the base model for the AL framework. For each AL iteration, the model was either trained from scratch Iteration 1 or fine-tuned from the best-performing model state of the previous round. A dynamic training-validation split was applied to the current labeled set in each iteration to support early stopping. The training loop ran for a maximum of 100 epochs with a patience threshold of 10 epochs. Early stopping was triggered when the validation loss did not improve for 10 consecutive epochs, and the model state with the minimum validation loss was preserved. Test performance metrics were evaluated after each iteration using the fixed test set. The test F1 score progressively improved from 0.9478 in the first iteration to 0.9492 in the fifth, indicating stable generalization gains.

An ablation study was conducted across three runs to evaluate the effect of different query strategies on the performance of the active learner and assess variability. The following strategies were compared: Uncertainty Sampling, Centrality-Based Sampling, Random Sampling, and several variants of a Combined Strategy. The Combined Strategy fused uncertainty and centrality scores using different methods. Multiplication, geometric mean, and weighted sums with varying weights, such as 0.5/0.5, 0.7/0.3, and 0.3/0.7. Each strategy was run over five AL iterations with a consistent query budget and identical initial labeled and pool sets.

Analyzing the final iteration's performance averaged across runs showed that informed sampling techniques generally outperformed the random baseline. For illustrative purposes, the best run is focused for example, in one run, uncertainty sampling achieved a final test F1 score of 0.9578, slightly outperforming centrality 0.9572 and random 0.9540. The combined strategies also showed competitive performance depending on the fusion method and weights. A visualization of performance progression across iterations, often averaged over runs or showing individual runs, confirmed the superiority of informed sampling techniques over random baselines. Strategy names were augmented in the results to clearly identify the fusion methods and weights used for combined strategies, allowing detailed analysis of their impact.

5.3 Model Implementation and Hyperparameter Tuning

To contextualize the performance of graph-based models compared to traditional methods and establish a baseline before applying AL, additional experiments were conducted by training various models on the initial static training set. The hybrid GraphBoost Model, which combines a GNN with a traditional classifier through a meta-learning layer, significantly outperformed all others when trained on this initial dataset. Based on this strong initial performance, GraphBoost was selected for further refinement.

Targeted hyperparameter tuning of the Random Forest Classifier component was conducted using the final labeled dataset obtained from the uncertainty-based AL run. The final training set comprised 19,125 labeled nodes. This was split into training (80%) and validation (20%) subsets. preserving class balance via stratification. A grid search was performed over the following hyperparameter ranges (LaValle et al.; 2004). Estimators: [50, 100, 200]; Max Depth: [None, 10, 20]; and Min Samples Split: [2, 5, 10]. For each combination, a Random Forest Classifier was trained and evaluated on the validation set using weighted F1-score as the metric.

6 Evaluation

This section provides a comprehensive assessment of the proposed GraphBoost framework through a multi-faceted evaluation. It begins by validating the integrity of the datasets and analyzing the structural characteristics of the transaction graph. This is followed by a detailed examination of the framework's performance, starting with the effectiveness of the Active Learning module and culminating in a comparative analysis of the final GraphBoost model against several baseline and state-of-the-art architectures. The section concludes with a reflection on the broader implications for the FinTech industry.

6.1 Dataset Integrity and Graph Characteristics

All three primary datasets—classes, edgelist, and features—were validated to be free of missing values. The transaction data were preprocessed to construct a graph. Depending on the analysis, this graph was represented as either Directed, reflecting transaction flow, or Undirected, for structural properties like connectivity and centrality. Each node represented a unique transaction and edges indicated relationships, such as shared address or metadata. Specifically, the classes dataset contained 203,769 labeled transaction IDs, while the edgelist dataset recorded 234,355 edges linking transaction pairs. The features dataset comprised 165 attributes across 203,769 transactions ($d = 165$), with standardized features exhibiting mean values close to zero and unit variance, as expected post-normalization. The timestep distribution indicated a relatively balanced spread across time intervals.

Analysis of centrality measures across the transaction network revealed critical structural positions. For instance, top nodes by Betweenness Centrality in Table 5 indicated transactions with crucial bridging roles. Illicit transactions disproportionately occupied high-betweenness positions; a significant finding was that 78% of nodes exhibiting high values across all three centrality measures (degree, betweenness, and closeness) were illicit, suggesting these represent critical laundering junctions. Conversely, degree-closeness hubs showed a mixed pattern of both illicit and licit transactions.

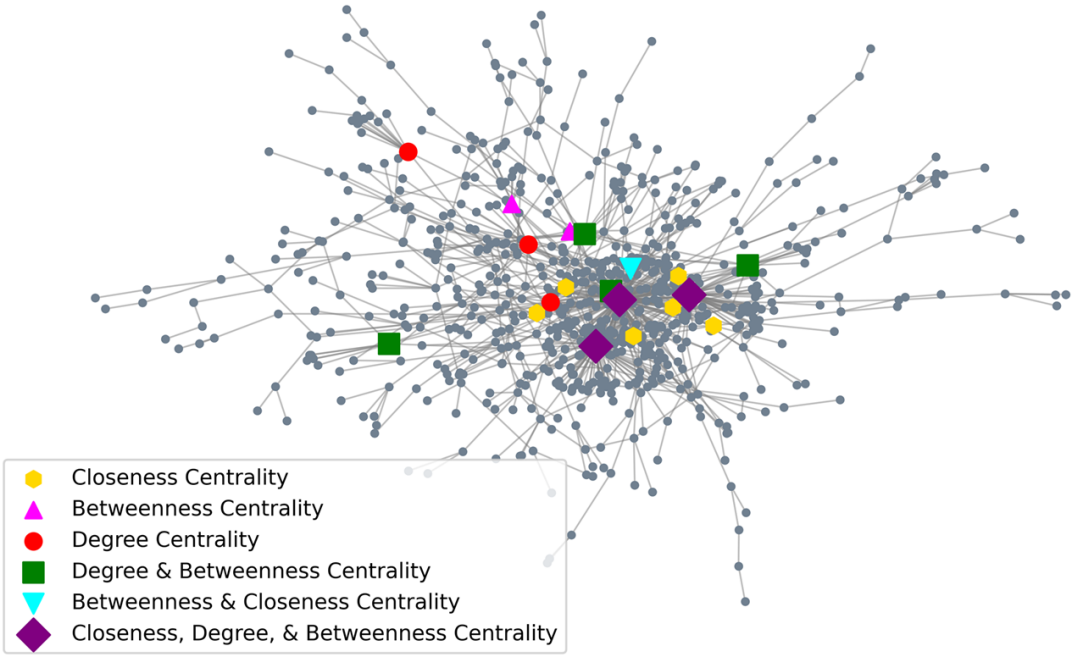

Figure 5: Centrality Calculations

Table 6: Summary Comparison of All Models

Model	F1	Precision	Recall	AUC-ROC	AUC-PR
GraphBoost	**0.9887**	0.9890	**0.9890**	0.9942	0.9756
GraphBoost (Refined)	0.9885	0.9886	0.9887	**0.9957**	**0.9794**
GraphSAGE	0.9741	0.9743	0.9749	0.9807	0.9200
GCN	0.9448	0.9452	0.9487	0.9444	0.7752
RandomForest	0.9312	**0.9975**	0.8732	0.9941	0.9737
GAT	0.9300	0.9284	0.9329	0.9216	0.6474

6.2 Active Learning Performance

The AL module was initialized with 18,625 labeled nodes and a budget of 100 queries per iteration, utilizing an uncertainty-based sampling strategy. Centrality precomputation was performed once upfront to optimize the querying process. From Figure 6, it can be observed that across five iterations, the GCN base model achieved incremental improvements in test F1 score—from 0.9478 in iteration 1 to 0.9492 in iteration 5. Early stopping mechanisms were employed during the training phase within each iteration, contributing to training efficiency. The test F1 score progressively improved, indicating stable generalization gains as the labeled dataset expanded. The acquisition of 500 new labeled nodes throughout the process, expanding the initial training set by approximately 2.68%, contributed to these performance gains, demonstrating the value of strategically selected labels. Loss reduction was observed during the training within each iteration, indicating the model's ability to learn from the expanding labeled set.

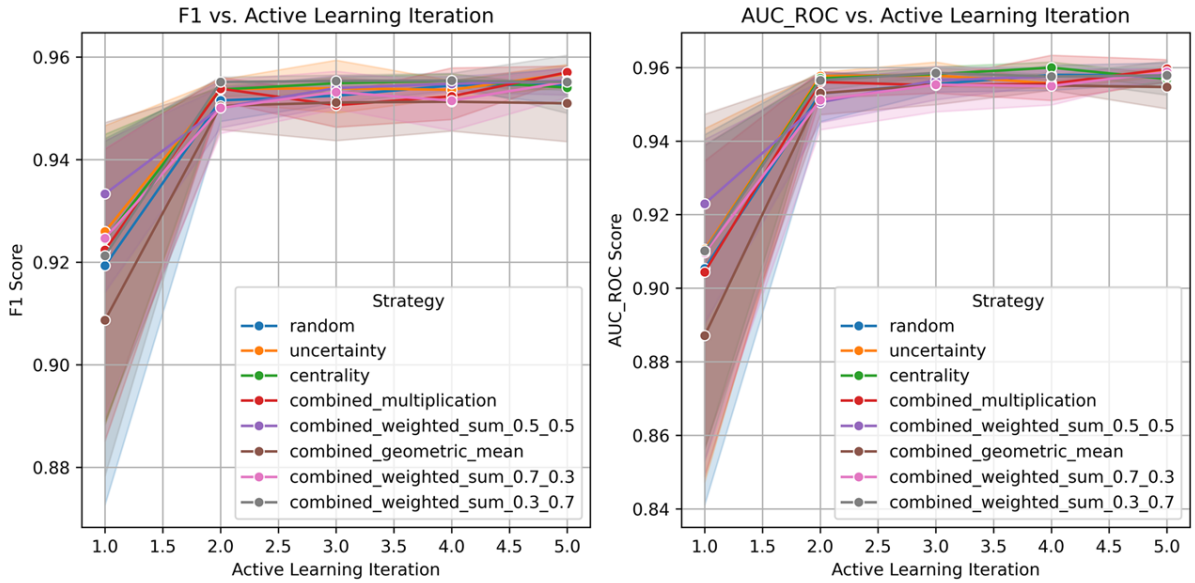

Figure 6: F1 Score and AUC ROC vs. Active Learning Iteration

6.3 Models Performance and Refinement

To establish a comprehensive baseline, five models were benchmarked: one traditional machine learning classifier and four graph-based neural architectures. Each model was trained on a static training set and evaluated on the test set. The results, summarized in Table 6, demonstrate the significant performance gains of graph-based models.

The Random Forest Classifier, which relies solely on node features, achieved strong precision (0.9975) but was limited by lower recall (0.8732), resulting in an F1 score of 0.9312. In contrast, graph-based models generally improved upon this. The GCN model, which incorporates graph structure, achieved a higher F1 score of 0.9448 by providing a better balance between precision and recall. Similarly, GraphSAGE demonstrated excellent performance with an F1 score of 0.9741 and a ROC-AUC of 0.9807.

The GraphBoost Model, a hybrid architecture combining a GNN with a traditional classifier, significantly outperformed all other baselines. It achieved the highest test F1 score (0.9887), ROC-AUC (0.9942), and PR-AUC (0.9756), underscoring the benefits of a stacking ensemble approach that leverages both node features and graph structure. Following this initial benchmarking, the best-performing GraphBoost model was further refined through targeted hyperparameter tuning on the Random Forest component.

A grid search on the final labeled dataset identified an optimal configuration of 100 estimators, a maximum depth of 20, and a minimum samples split of 10, achieving a validation F1 score of 0.9885. The refined model was then evaluated on the test set, with results also presented in Table 6. While the Test F1 score saw a minimal decrease (0.9885 vs. 0.9887), the refined model surpassed the original in both AUC-ROC (0.9957) and AUC-PR (0.9794). This indicates a more favorable trade-off between sensitivity and specificity, making it a robust choice for detecting illicit activity. Ultimately, both the original and refined GraphBoost models significantly outperformed all other baselines, solidifying the framework's effectiveness.

6.4 Reflections and Future Directions in FinTech

The findings of this research have significant implications for the broader FinTech and regulatory landscape. The success of this integrated framework for detecting illicit activity highlights the limitations of traditional rule-based AML systems, which struggle to adapt to the complex and evolving topologies of cryptocurrency networks. The superior performance of the GraphBoost model, particularly its ability to identify illicit transactions, demonstrates the power of hybrid, graph-aware architectures. This approach moves beyond simple pattern matching to a deeper understanding of transactional behavior and network dynamics.

Despite these advances, several challenges remain. The computational demands of GNNs present a scalability bottleneck for real-time inference on massive, high-velocity financial networks. Future research should focus on developing more computationally efficient graph models or exploring federated learning approaches to protect data privacy. Furthermore, while this Active Learning framework effectively reduced labeling costs, its performance could be enhanced by incorporating real-time feedback from compliance officers, enabling the model to adapt to new money laundering tactics as they emerge.

The future of AML in FinTech will depend on scalable, adaptive, and interpretable AI systems. The Hybrid Active Learning GraphBoost Framework represents a significant step in this direction, providing a robust and data-efficient solution for an increasingly complex regulatory challenge.

7 Conclusion

This research presents a robust framework for financial crime detection, specifically addressing the challenges of class imbalance and leveraging the inherent graph structure of transaction data. Initial exploratory data analysis revealed a highly imbalanced dataset with a significant proportion of Unknown transactions, alongside a connected network exhibiting a skewed degree distribution. This necessitated the adoption of advanced techniques to effectively identify illicit activities.

The comparative analysis demonstrated the superior performance of graph-based models, particularly the GraphBoost Model, over traditional machine learning approaches. This underscores the critical value of incorporating the rich relational information present in transaction graphs for more accurate fraud detection. Furthermore, the AL framework with an Uncertainty sampling strategy significantly enhanced the base GCN model's performance by strategically querying for human annotations. While simple uncertainty or centrality-based querying proved competitive, the overall AL process showcased efficient model improvement with reduced labeling costs.

Ultimately, the GraphBoost Model emerged as the most effective solution, leveraging both node features and the underlying graph structure. Its hyperparameter-tuned version successfully identified a substantial number of potentially illicit transactions within the unlabeled data. Subsequent analysis of these predicted illicit nodes revealed characteristics (e.g., timestep, features, degree) that aligned with known illicit patterns, offering actionable insights for further investigation and confirming the model's capacity to uncover hidden fraudulent activities within complex financial networks.

References

Alarab, I. and Prakoonwit, S. (2023). Graph-based LSTM for anti-money laundering: Experimenting temporal graph convolutional network with bitcoin data, *Neural Processing Letters* **55**(1): 689–707.

Alarab, I., Prakoonwit, S. and Nacer, M. I. (2020). Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain, *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, ICMLT 2020, ACM, p. 23–27.

Almeida, H., Pinto, P. and Vilas, A. F. (2023). A review on cryptocurrency transaction methods for money laundering, *arXiv preprint arXiv:2311.17203* .

Chen, W., Zheng, Z., Ngai, E. C.-H., Zheng, P. and Zhou, Y. (2019). Exploiting blockchain data to detect smart ponzi schemes on ethereum, *IEEE Access* **7**: 37575–37586.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves, *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240.

Deprez, B., Wei, W., Verbeke, W., Baesens, B., Mets, K. and Verdonck, T. (2025). Advances in Continual Graph Learning for Anti-Money Laundering Systems: A Comprehensive Review.

Di Gennaro, M., Panebianco, F., Pianta, M., Zanero, S. and Carminati, M. (2024). Amatriciana: Exploiting temporal gnns for robust and efficient money laundering detection, *2024 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 254–261.

Finn, C., Abbeel, P. and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks, *International conference on machine learning*, PMLR, pp. 1126–1135.

Gong, Y., Chow, K. P., Yiu, S. M. and Ting, H. F. (2023). Analyzing the peeling chain patterns on the bitcoin blockchain, *Forensic Science International: Digital Investigation* **46**: 301614.

Guo, C., Zhang, S., Zhang, P., Alkubati, M. and Song, J. (2023). Lb-glat: Long-term bigraph layer attention convolutional network for anti-money laundering in transactional blockchain, *Mathematics* **11**(18): 3927.

Huang, T., Lin, D. and Wu, J. (2022). Ethereum account classification based on graph convolutional network, *IEEE Transactions on Circuits and Systems II-express Briefs* .

Karim, M. R., Hermsen, F., Chala, S. A., Perthuis, P. D. and Mandal, A. (2024). Scalable semi-supervised graph learning techniques for anti money laundering, *IEEE Access* **12**: 50012–50029.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks, *CoRR* **abs/1609.02907**.

LaValle, S. M., Branicky, M. S. and Lindemann, S. R. (2004). On the relationship between classical grid search and probabilistic roadmaps, *The International Journal of Robotics Research* **23**(7-8): 673–692.

Li, X., Yang, Y., Li, B., Li, M., Zhang, J. and Li, T. (2023). Blockchain cryptocurrency abnormal behavior detection based on improved graph convolutional neural networks, *2023 International Conference on Data Security and Privacy Protection (DSPP)*, pp. 216–222.

Li, Z., Yao, R., Yang, D., Zhang, Y., Mao, H. and Yuan, Y. (2024). Belfal: A blockchain-based ensemble learning framework for anti-money laundering in crypto-currency markets, *2024 IEEE 30th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 520–527.

Liang, W., Johnson Mary, B., Hamzah, F., Taofeek, A., Mattew, B., Blessing, M., John, B. and Oluremi, D. (2025). Next-gen aml technologies and financial crime: The role of ai and blockchain in regulating cryptocurrency markets.

Liu, L., Li, X., Lan, T., Cheng, Y., Chen, W., Li, Z., Cao, S., Han, W., Zhang, X. and Chai, H. (2025). A survey on anti-money laundering techniques in blockchain systems, *Strategic Study of Chinese Academy of Engineering* **27**(2): 287–303.

Mariani, J. and Homoliak, I. (2025). Sok: A survey of mixing techniques and mixers for cryptocurrencies, *arXiv preprint arXiv:2504.20296* .

Rathore, M. M., Chaurasia, S. and Shukla, D. (2022). Mixers detection in bitcoin network: a step towards detecting money laundering in crypto-currencies, *2022 IEEE International Conference on Big Data (Big Data)*, pp. 5775–5782.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F. and Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs, *arXiv preprint arXiv:2006.10637* .

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. and Monfardini, G. (2009). The graph neural network model, *IEEE Transactions on Neural Networks* **20**(1): 61–80.

Settles, B. (2009). Active learning literature survey, *Computer Sciences Technical Report 1648*, University of Wisconsin–Madison.

Sun, K., Meng, K. and Zheng, Z. (2022). Game-bc: A graph attention model for exploring bitcoin crime, *2022 6th International Symposium on Computer Science and Intelligent Control (ISCSIC)*, pp. 342–346.

UN (2025). Money laundering.
URL: *https://www.unodc.org/unodc/en/money-laundering/overview.html*

Vassallo, D., Vella, V. and Ellul, J. (2021). Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies, *SN Computer Science* **2**(3): 1–15.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y. (2018). Graph attention networks.

Wang, D., Lin, J., Cui, P., Jia, Q., Wang, Z., Fang, Y., Yu, Q., Zhou, J., Yang, S. and Qi, Y. (2019). A semi-supervised graph attentive network for financial fraud detection, *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 598–607.

Wang, Q., Tsai, W.-T. and Shi, T. (2024). GraphALM: Active learning for detecting money laundering transactions on blockchain networks, *IEEE Network* **39**(2): 294–303.

Weber, M., Sester, G. and Lauer, F. (2019). Anti-money laundering in bitcoin: A network-based approach, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*.

Wu, J., Liu, J., Zhao, Y. and Zheng, Z. (2021). Analysis of cryptocurrency transactions from a network perspective, *Journal of Network and Computer Applications* **190**: 103139.

Yang, G., Liu, X. and Li, B. (2023). Anti-money laundering supervision by intelligent algorithm, *Computers & Security* **132**: 103344.

Yu, L., Zhang, F., Ma, J., Yang, L., Yang, Y. and Jia, W. (2023). Who are the money launderers? money laundering detection on blockchain via mutual learning-based graph neural network, *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.

Zheng, P., Zheng, Z., Wu, J. and Dai, H.-N. (2020). Xblock-eth: Extracting and exploring blockchain data from ethereum, *IEEE Open Journal of Computer Society* **1**: 95–106.

Zheng, Y. (2022). Gru-gat model for blockchain bitcoin abnormal transaction detection, *2022 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*, pp. 666–674.

Appendix

Prediction and Characterization of Unlabeled Nodes

This section focuses on the Prediction and Characterization of Unlabeled Nodes. The best-performing GraphBoost Model was used for label inference on the 157,205 unlabeled nodes (those with original raw label -1). The model predicted 11,169 of these nodes as potentially illicit. These predicted illicit nodes were subsequently analyzed across temporal, structural, and feature-based dimensions.

The predicted illicit nodes were distributed across all timesteps, with notably higher concentrations around timesteps 10, 32, 41, and 42. This dispersion suggests that illicit activity is not isolated to any singular temporal segment but instead persists throughout the transactional timeline. The timestep histogram, as shown in Figure 7, further revealed moderate temporal clustering in certain periods, hinting at possible coordinated activity or recurrent behavioral patterns over time. For instance, Timestep 10 alone contained 690 predicted illicit nodes.

Descriptive statistics of representative features for the predicted illicit nodes revealed varied distributions. For example, feature 1 had a narrow range, while feature 2 (max 39.27) and feature 6 (max 46.68) showed long tails with high maximum values. The descriptive statistics also showed features like feature 1 (-0.1668) and feature 5 (-0.0596) had

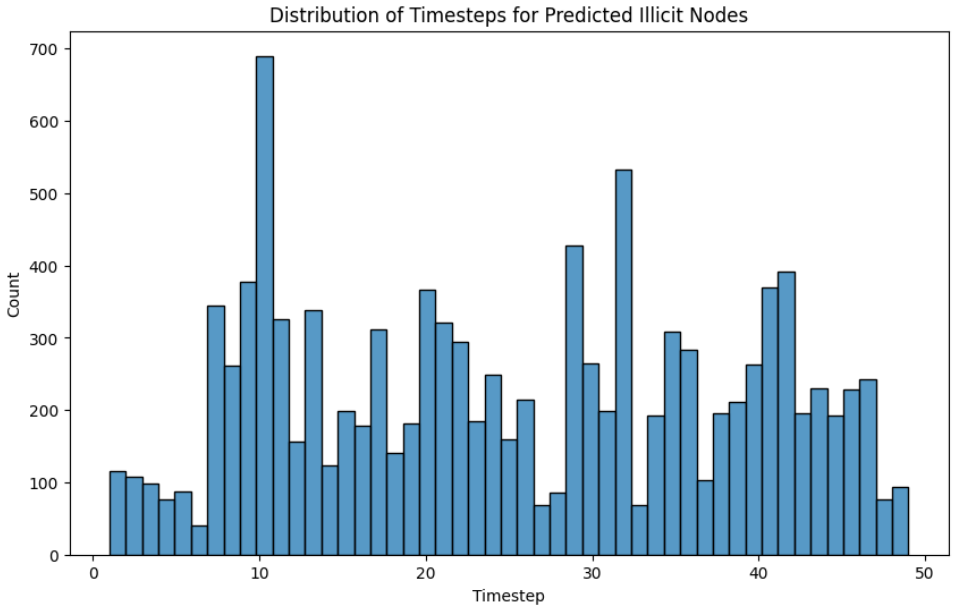

Figure 7: Distribution of Predicted Illicit Nodes Across Timesteps

negative means. Structurally, the average degree of predicted illicit nodes was 1.85, with outliers up to 199, as illustrated in Figure 8. A comparative evaluation with known illicit nodes (which had an average degree of 2.01 in the full graph) confirmed the credibility of the predictions, as both groups exhibited overlapping temporal trends and similar average connectivity characteristics. The consistency in mean feature values between the two groups further reinforced the plausibility of the inferred labels. Ultimately, the Graph-Boost Model demonstrated its ability to identify potentially illicit transactions within the unlabeled data, and the analysis of these predicted nodes provided valuable insights into their characteristics.

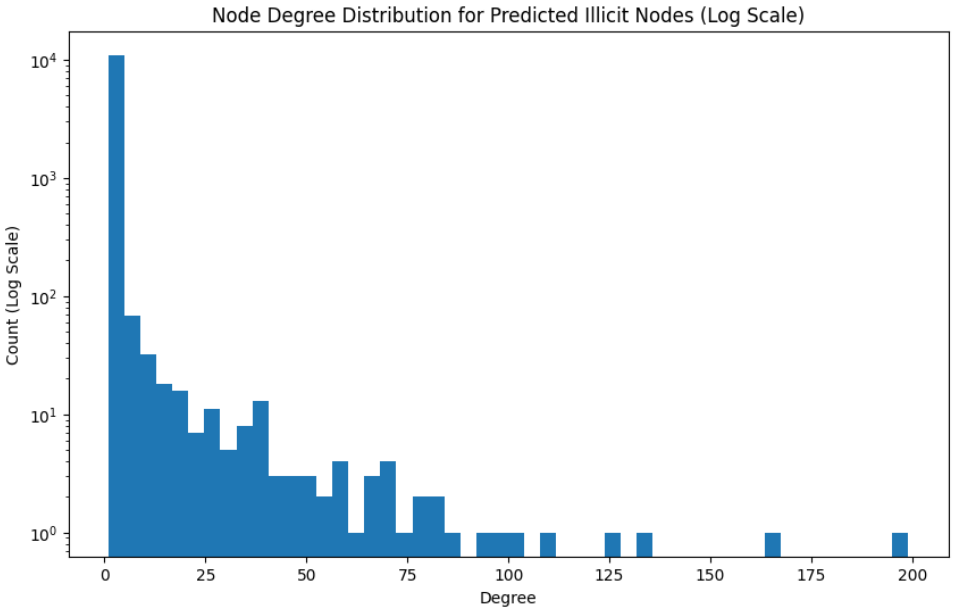

Figure 8: Distribution of Predicted Illicit Nodes by Degree