Article | Received 6 March 2026; Revised 13 July 2026; Accepted 20 July 2026; Published 1 September 2026
https://doi.org/10.55092/blockchain20260007

StableAML: machine learning for behavioral wallet detection in stablecoin anti-money laundering on Ethereum

Luciano Juvinski*, Haochen Li and Alessio Brini

Pratt School of Engineering, Duke University, Durham, USA

* Correspondence author; E-mail: luciano.silva@duke.edu.

Highlights:

- Large-scale labeled USDT/USDC Ethereum dataset for supervised stablecoin AML.
- Domain-informed tree ensembles top Macro-F1 0.97, beating GNNs on fragmented graphs.
- Feature importance separates cybercrime multi-hop layering from sanctioned wallets.

Abstract: Over $3.1 trillion in illicit money moves through the global financial system yearly, and much of it now uses stablecoins, which launderers prefer for their liquidity. Decentralized protocols increasingly hide transaction patterns with zero-knowledge proofs, while centralized stablecoins remain visible. To preserve the ability to convert to fiat, they must maintain an auditable record, which makes them natural points of compliance oversight. Using this visibility, we create an Ethereum dataset of Tether USD (USDT) and USD Coin (USDC) wallet transfers and establish a baseline for behavioral anti-money-laundering (AML) detection. We compare linear models, tree ensembles, deep networks, and graph neural networks. Tree ensembles achieve the best Macro-F1 score, while the graph neural networks lose accuracy as the transaction network fragments. The models separate distinct typologies rather than only flagging suspicion: the fast, dispersed movement of cybercrime wallets is distinguished from the constrained, static footprint of sanctioned or frozen wallets. These results align with the industry shift toward deterministic verification and address the auditability and compliance requirements now forming under regulations such as the EU's Markets in Crypto-Assets (MiCA) and the U.S. Guiding and Establishing National Innovation for U.S. Stablecoins Act (GENIUS Act), while limiting unjustified asset freezes. A high-precision behavioral classification of suspicious wallets raises the economic cost of financial misconduct and informs compliance practice under emerging stablecoin rules.

Keywords: anti-money laundering; stablecoins; blockchain analytics; know your transaction; machine learning

1. Introduction

Most organized crime has a financial component. Proceeds from corruption, trafficking, cyber crimes, or terrorist financing must be concealed, moved, and then returned to the legitimate economy, and

that dependence on moving money makes laundering central to criminal enterprise. In 2024, roughly $3.1 trillion in illicit funds moved through the global financial system into a shadow economy that regulators cannot reach [1].

On a blockchain, the same three laundering stages, placement, layering, and integration, happen far faster than in conventional financial crime. In 2024, criminals laundered about $51 billion through cryptocurrencies, roughly 0.14% of the $9 trillion transacted on-chain that year [2]. Launderers inject illicit funds, obscure their origin through long multi-hop paths[1], and reintegrate them into the legitimate economy. On a public ledger every transfer is visible, but each wallet is only a cryptographic address, so activity remains pseudonymous even though its full history is open. Rather than impede illicit usage, the open and permissionless nature has made disguising activities more problematic. This practice is prevalent in decentralized exchanges (DEXs), token swaps, and cross-chain bridges [3,4]. Much of this value moves in stablecoins, cryptocurrencies pegged to a fiat currency or commodity through collateral or an algorithmic rule, which keeps their price steady enough to serve as a medium of exchange and unit of account across centralized and decentralized finance [5]. The features that make stablecoins useful also make them the dominant rail for illicit value, and they accounted for more than 84% of verified crypto fraud volume in 2025 [6], owing to their liquidity, interoperability, and deep integration across exchanges, decentralized finance (DeFi) protocols, and payment rails.

We focus on Ethereum, which dominates Web3[2] activity and has a long record of security incidents. Tether USD (USDT) and USD Coin (USDC), issued by Circle, together settled more than $10 trillion in on-chain transfer volume during 2025 [7], more than Bitcoin and Ethereum settled combined (Figure 1). Most of this activity is legitimate, but the features that make stablecoins useful also make them convenient for laundering. They settle in seconds rather than days, cross borders freely, and reveal only an address [8]. Recent evidence shows stablecoins are now the preferred medium for laundering and sanctions evasion, especially after enforcement acted against obfuscation services such as mixers[3] [4].

Regulation is moving quickly alongside these trends. The EU's Markets in Crypto-Assets (MiCA) Regulation (2023) and the U.S. Guiding and Establishing National Innovation for U.S. Stablecoins Act (GENIUS Act) (2025) require stablecoin issuers to hold reserves that fully back the tokens in circulation, submit to independent audits, and meet anti-money-laundering (AML) obligations, and several Asian jurisdictions have adopted comparable regimes. Singapore tightened oversight through the Payment Services Act and the 2023 to 2024 Stablecoin Regulatory Framework, Hong Kong introduced its Stablecoin Issuer Regulatory Regime by amending the Anti-Money Laundering and Counter-Terrorist Financing Ordinance, and China regulates private stablecoin activity under the People's Bank of China Digital Currency Framework tied to its digital yuan initiative.

[1] A multi-hop transaction routes funds through a sequence of intermediate wallets before they reach a final destination, instead of one direct transfer. Launderers use this to build a complex chain of custody that distances the money from its illicit origin and complicates audit trails.

[2] Web3 refers to a paradigm for web-based applications that uses blockchain technology, smart contracts, and token-based economic mechanisms to enable decentralized execution and coordination. In contrast to Web 2.0 platforms, which typically rely on centralized intermediaries, Web3 applications allow users to interact through decentralized applications (dApps) in which control over digital assets is mediated by cryptographic keys rather than centralized authorities.

[3] Mixers are services that hide the origin of transactions by combining funds from various users and redistributing them. This process disrupts the connection between the sender and the recipient.

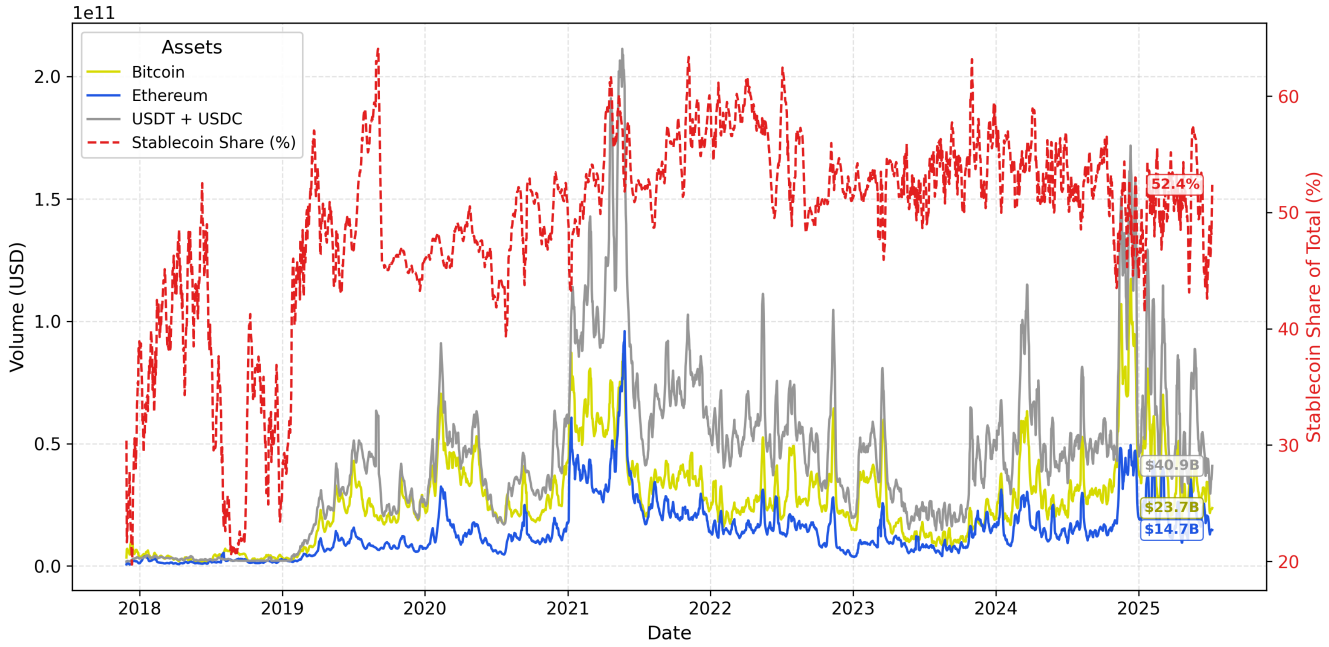


Figure 1. Historical transaction volumes for Bitcoin, Ethereum, and combined stablecoins.

Tether and Circle have functions in their smart contracts that allow them to freeze specific addresses. This action stops transfers and any movement of tokens. As a result, freezes have immobilized over $4 billion in USDT and more than $1 billion in USDC [4]. This increases both the cost and traceability of illicit transfers. These freezes also require offenders to take extra steps, moving value through additional wallets and services to avoid the restrictions. Each extra step lengthens the trail and makes tracking more difficult. When policy and enforcement raise the cost of crime, criminals respond with more complex evasion strategies. Therefore, detection must continually adapt to their behavior on the blockchain.

With millions of activities every day, it is challenging to separate legitimate transfers from illegal ones. Traditional rule-based tools use fixed limits and set patterns that flag transfers exceeding a certain amount, respond to sudden spikes in activity, or block wallets previously identified as risky. While these rules are straightforward, they are also highly inefficient, with false-positive rates exceeding 95% [9]. In Europe, compliance costs around $136.5 billion per year and only captures about 0.1% of criminal funds [10]. The burden is large and still misses the adaptive behavior that defines modern blockchain laundering.

We make three contributions. First, we build StableAML, a large-scale dataset focused only on stablecoin flows. We collect suspicious and malicious Ethereum wallets from Etherscan reports, blockchain-security firms, and official sanctions and enforcement lists, specifically the U.S. Office of Foreign Assets Control (OFAC) Specially Designated Nationals list. We join these labels to the full transfer history of USDT and USDC on Ethereum and pass the data through a feature-engineering pipeline that builds 68 domain-specific attributes in four groups: Interaction Features, Derived Network Features, Transfer-based Features, and Temporal and Direct Features. To the best of our knowledge, this is the first large-scale labeled dataset assembled exclusively for stablecoins. The narrow scope lets us isolate token mechanics specific to stablecoins, smart-contract freezes and blacklist interactions, which native-asset models cannot capture yet which matter for stablecoin laundering typologies. The predicate offenses behind flagged wallets vary widely, from DeFi exploits and phishing to OFAC sanctions and issuer freezes,

but the act of moving, stabilizing, and exiting illicit proceeds tends to converge on a common set of stablecoin rails. It is therefore important to concentrate on the layering and integration phases, where these otherwise dissimilar categories share one behavioral footprint. As privacy layers obscure the native chain, AML procedures must work within deterministic compliance tunnels and depend only on permissioned transfer logs. On the other hand, stablecoin contract events are the ongoing and auditable choke points of an otherwise private ledger, and the risk is therefore determined only by these events. This approach is followed by new tools for compliance introduced in MiCA and the GENIUS Act including audit keys and proof of innocence techniques.

Second, we create an evaluation framework that sorts wallet behavior using supervised machine learning. It focuses on the obfuscation methods that are specific to stablecoins. On-chain detection is different from traditional fraud because illegal actors quickly move funds across many wallets or switch chains through cross-chain bridges and decentralized exchanges to fragment the graph. Privacy technologies compound this, as zero-knowledge shielding protocols such as Railgun and mixers cryptographically sever the traceable link between sender and recipient [11–13] and what a model can observe is a sparse and broken graph. We evaluate Graph Neural Networks (GNNs) on it to reproduce a Compliance Tunnel. As privacy-preserving protocols hide the native graph, stablecoin ledgers become the primary auditable channel, and we test how the models cope when restricted to those fragmented transfer logs.

Third, we make AML detection policy and compliance more transparent. We analyze the static classification of behavioral signatures. This step connects the performance of the model with the regulatory framework prior to deploying any real-time monitoring systems. These are behavioral risk scores and not definitive findings of money laundering. Our labels refer to the laundering activity and are based on verified malicious behaviors such as the OFAC sanctions and known DeFi hacks. A strict legal finding would require off-chain context and evidence of a predicate offense. Our framework is therefore a triaging tool for compliance officers, scoring the behavioral risk of on-chain entities through obfuscation signatures, not a substitute for legal identification.

The rest of the paper is organized as follows. Section 2 reviews work on blockchain AML detection, privacy, and regulatory enforcement. Section 3 describes the dataset and the feature-engineering pipeline. Section 4 presents the modeling methodology and the machine-learning and graph-based approaches we evaluate. Section 5 reports the experiments and their implications, including feature importance, detected typologies, and the binary classification analysis. Section 6 concludes and points to future work.

2. Literature review

Machine learning has a long history in fraud detection within traditional finance. Banks and payment networks routinely use supervised models to detect credit-card fraud and identity theft by learning patterns in user behavior, merchant characteristics, and transaction timing [14].

In that setting, tree ensembles detect fraud comparably to deep neural networks. Recent benchmarks report F1 scores above 0.90 on tabular transaction data, which makes them attractive without the computational cost of graph-based models [15].

The cryptocurrency literature builds on this by applying machine learning and graph analysis to digital-asset laundering. Shojaeinasab [16] studies Bitcoin mixing and trains GNNs to classify links to mixers. Pocher *et al.* [8] detect illicit activity on the Bitcoin network with the Elliptic dataset, classifying transactions as illicit or legitimate. That benchmark was introduced by Weber *et al.* [17], who first showed the feasibility of graph convolutional networks for transaction-level AML on Bitcoin and released the labeled subgraph that has since become the standard reference for this line of work. Extensions to Ethereum-specific AML [18] confirm the same architectural template while noting the extra difficulty introduced by the smart-contract layer.

In the broader Web3 context, Lin *et al.* [3] introduce the EthereumHeist dataset to analyze addresses linked to hacks and scams, and show that programmable obfuscation, in particular token swaps and cross-chain transfers, has become a central mechanism for concealing the origin of illicit funds.

Liu *et al.* [9] add a framework called GTN2vec to detect laundering on Ethereum. GTN2vec shows strong performance for native Ether transactions. However, it needs the involvement of several transaction steps to identify laundering. Temporal architectures like EvolveGCN [19] and the Temporal Graph Network framework [20] update graph embeddings over time and require steady connections. In the stablecoin ecosystem, transaction graphs are broken by the token mechanics and intentional concealment. As a result, both static and dynamic GNNs become less effective on a sparse network. Berentsen *et al.* [21] note how privacy-preserving technologies like Zero-Knowledge Proofs (ZKPs) will reshape access to public data regarding transactions in the future. Cryptographic obfuscation helps protect data, but disrupts the continuous graph connectivity that underpins fraud detection [22,23]. Berentsen *et al.* [21] suggest that due to the economic pressure for privacy, the field will still adopt a form of transparency in which a transaction can be validated without revealing any sensitive data. Chang *et al.* [24] study the impact of blockchain in financial services and, in particular, its adoption, using qualitative data. They identify similar problems with monitoring.

Vatsa [25] provides a large-scale empirical analysis of the strong market concentration among stablecoins. It shows how the EU's MiCA and the U.S. GENIUS Act promote transparency but may also reinforce market dominance, since their compliance costs act as barriers to entry. Fajri *et al.* [26] illustrate how data-protection legislation creates regulatory paradoxes in the use of AI to automate the enforcement of AML regulations. Since privacy frameworks create global blind spots for on-chain analysis, modeling efforts must shift to isolated, deterministic compliance tunnels, such as stablecoins, balancing the tradeoff between the right to privacy and the right to be examined. Griffin *et al.* [4] use illicit transactions to gauge approaches such as OFAC sanctions and balance the tradeoffs of enforcement through the reduction of illicit activity performed in mixers against the shift of criminals to DeFi swaps and cross-chain bridges. Akartuna *et al.* [10] argue against "one-size-fits-all" prevention and believe that different asset classes, such as stablecoins require detection models designed for specific risk patterns and operational contexts.

AML now depends on both analytical models and a regulatory environment that continually shifts. However, most research stays focused on native cryptoassets or relies on continuous graph topologies. This suggests a potential role for a supervised and feature-centric approach to stablecoins instead, which is built around their token mechanics rather than graph-based tracing.

3. The StableAML dataset

Ethereum records billions of transactions, so we keep only the value flow of the two dominant stablecoins, USDT and USDC. We extract the transfer events emitted by their official smart contracts[4] from 2017–11–28 to 2025–08–08. A raw transaction is often merely a contract call with no value transferred, while a transfer event marks an actual token transfer.

Table 1. Example of a token transfer that involved 1,092,761.61 USDT.

Field	Value
Transaction	`0xf8163c3d5ba77186ad4c6c93c4f3f92a88adb1b55cd0da34de2a44d33d4e20bd`
Sender Address	`0x654Fae4aa229d104CAbead47e56703f58b174bE4`
Recipient Address	`0x000000000035B5e5ad9019092C665357240f594e`
Amount	1,092,761.61 USDT
Timestamp	2024-01-31 11:59:59

Transfer events remain visible even when privacy tools hide transaction level detail, and reading them resolves the cases where raw transaction data is ambiguous, such as meta transactions and relayers. When a relayer (Wallet A) submits a transaction on behalf of a user (Wallet B) to move funds to a recipient (Wallet C), the raw ledger lists Wallet A as the initiator and hides the true economic actor. Instead, the transfer event bypasses the intermediary and records the movement directly from B to C.

From these event-level records, we reconstruct value flows between counterparties, aggregate transfers per wallet, and build a behavioral profile for each address, because the tokens' design keeps every transfer on the public record [11,12].

Our baseline is the complete history of USDT and USDC transfers, 334,754,008 transactions across 54,998,597 unique wallets. Within it, 363,887 addresses were flagged as malicious or suspicious. Before any sampling, we computed every behavioral and topological feature (e.g., 2ndWithFlagged, receiveMultipleSameValue) on the full macro-network. We built the full 54.9-million-node transaction graph so that multi-hop metrics are direct for every candidate node. This order keeps each wallet's topological features tied to its true position in the whole Ethereum network, not to a sparse subgraph.

After computing features on the macro-network, we filtered the raw population down to the labeled modeling set. To limit label noise and keep the "Normal" class clean, we applied two strict rules. First, a minimum activity rule drops dormant or near-zero addresses, so trivial cases do not inflate the metrics and the model can focus on distinguishing active legitimate users from laundering operations. Second, a Proximity Exclusion Filter removes any remaining benign candidate within a 2- to 3-hop transaction radius of a known illicit node, so undetected accomplices do not enter the baseline.

We included illicit Blocklisted and Cybercrime addresses by ground-truth certainty, not by activity thresholds. Laundering often runs through single-use or high-churn wallets, so removing the low-activity wallets here would erase the structural signature of layering. We therefore gated illicit labels on deterministic on-chain events such as decoding smart-contract blacklist parameters.

[4]Smart contracts are self-executing programs on a blockchain that run when preset conditions are met. For stablecoins, they manage token issuance, transfers, and balances on their own, with no central intermediary.

The final 16,433 labeled wallets come from this filtered pool through stratified random sampling on the class label. We anchor the sample on the Cybercrime class, the smallest of the three with confirmed stablecoin activity, then draw Blocklisted and Normal addresses to a managed target distribution the model can learn from. Because the two illicit categories together are under 1% of active addresses at the macro level, a sample proportional to true prevalence would leave the model unable to learn the multi-hop layering signatures of interest. Sampling real on-chain addresses, rather than synthetic oversampling such as SMOTE, keeps authentic behavioral and topological signal in every retained wallet. We report the resulting class proportions at the close of the next subsection, and Section 4 describes the 80/20 split into training and held-out test sets.

3.1. Classification groups

We divide addresses into three classes for supervised learning, capturing behavioral and regulatory heterogeneity in the stablecoin ecosystem. Each category's labels come from publicly accessible sources so anyone can reproduce them:

- *Normal*: addresses with no negative risk flags reported by blockchain intelligence providers, specifically Chainalysis, TRM Labs, or Crystal Intelligence[5]. Because these tools use flow-based analytics that follow entire transaction chains rather than only the origin of an offense, the absence of a flag is a strong baseline of operationally benign status, that is, benign under these providers' coverage and the filters below rather than in an absolute sense. Since the providers propagate labels along transaction chains outward from confirmed illicit seeds, their reach extends past individually inspected addresses to the intermediaries and layering wallets connected to known illicit entities. That reach is still bounded by each provider's detection thresholds and can miss intermediaries just beyond them, so a missing flag is a strong first screen rather than proof of complete coverage. We cannot state a precise coverage percentage, because the providers' methods and underlying intelligence are proprietary and no ground-truth set of truly-illicit-but-unflagged addresses exists to measure against. We take these residual observations as error and apply two safeguards, a conservative bound on likely provider false negatives (a Proximity Exclusion Filter with a 2- to 3-hop radius of any illicit cluster) and the behavioral legitimacy markers as a largely independent signal. Below two hops, the filter would reach only the wallets in direct contact with an illicit node and leave the two- and three-hop intermediaries, the layering positions funds pass through, in the Normal pool as undetected accomplices. Beyond 3 hops the neighborhood for a cluster network on close to 55 million addresses grows rapidly, and an increasing portion of the "active" addresses eventually becomes close to some illicit node, therefore an unbounded radius would risk removing legitimate candidates and shrink the Normal class toward zero. A radius of 2 or 3 matches the depth of the layering paths the framework targets while keeping the Normal class adequately populated. Exploratory analysis of the Normal subset also shows it represents regulated retail activity, with frequent use of Know Your Customer (KYC)-compliant Centralized Exchanges

[5]These providers publicly document this multi-hop tracing, which they term "indirect exposure" or "indirect risk," following funds through intermediary addresses outward from known illicit entities: Chainalysis, https://www.chainalysis.com/blog/cryptocurrency-risk-blockchain-analysis-indirect-exposure/; TRM Labs, https://www.trmlabs.com/resources/blog/key-considerations-for-evaluating-indirect-risk-on-the-blockchain; and Crystal, https://crystalintelligence.com/blockchain-analytics-tool/.

(CEX) and frequent long-term holders (both rarer in illicit cases).

- *Cybercrime*: addresses directly involved in malicious on-chain activities. We collect these labels from publicly available security and forensic reports such as SlowMist and PeckShield, as well as community alerts on Etherscan. These addresses represent the direct sources of illicit funds and are divided into two categories:
  - Exploits & Hacks: addresses linked to DeFi protocol exploits, security breaches, or exchange thefts.
  - Scams & Phishing: addresses associated with fraudulent schemes or social engineering attacks.
- *Blocklisted*: addresses under administrative interdiction or legal enforcement. This class is defined by regulatory status, not by behavior like Cybercrime, and it covers both public and private enforcement. We extract these labels by decoding the on-chain `addBlocklist` events of the USDT and USDC contracts from the blockchain logs. This returns the same address set the issuers' compliance teams act on.
  - Sanctioned: addresses designated by government bodies, specifically the OFAC Specially Designated Nationals (SDN) [27].
  - Frozen: addresses blocked at the smart-contract level by stablecoin issuers, so the assets cannot be transferred.

The modeling dataset consists of 48.7% Normal, 36.5% Cybercrime, and 14.8% Blocklisted.

3.2. Feature engineering

Laundering on-chain leaves behavioral and relational traces [16], so we turn each wallet's raw `Transfer` history into a fixed set of engineered features.

For each wallet we store the features as a table row, one column per feature. Table A1 of Appendix A gives the full list. The 68 features fall into four groups.

- Interaction Features: direct wallet-to-protocol activity, e.g. `sentToCex`, `receivedFromSwap`, `sentToMixer`.
- Derived Network Features: multi-hop propagation and indirect ties, e.g. `2ndWithFlagged`, `2ndWithOver10k`, `3rdWithFlagged`.
- Transfer-based Features: direct value flow and volume thresholds, e.g. `transferOver1k/5k/10k`, `sentMultipleSameValue`.
- Temporal and Direct Features: timing patterns and intrinsic account traits, e.g. `highFrequency`, `isLongTermWallet`, `isWallet`.

3.2.1. Interaction features

We identify the main on-chain services that a wallet interacts with, such as swap protocols, lending platforms, and staking pools. We label each service and link every transaction to its protocol. As a result, each feature captures how a wallet routes funds through intermediaries.

Table 2 lists three of these service categories, and its "Wallets" column reports how many addresses activate each feature. The counts sum to more than the number of unique wallets, since one address can use several service types (e.g., both Uniswap and Aave).

Table 2. Service interaction labels based on the protocol types with which each wallet interacts, allowing a wallet to belong to multiple categories.

Service Label	Description	Features	Wallets
Swap Protocols	Permissionless token-exchange platforms such as Uniswap or Curve.	`sentToSwap`, `receivedFromSwap`	44,312
Lending Platforms	Protocols enabling borrowing and collateralized lending.	`sentToLending`, `receivedFromLending`	28,904
Staking Pools	Protocols that allow users to lock tokens to earn rewards or participate in activities.	`sentToStake`, `receivedFromStake`	12,587

3.2.2. Derived network features

Interaction features capture direct relationships. Derived network features track how illicit funds move after the first transaction, over several hops, including second- and third-degree connections to flagged wallets.

We label these proximity features as known-bad-proximity screening signals. They reproduce the initial screening step that operational compliance systems run in practice. The intelligence sources behind these signals draw on the same labeling universe as our ground-truth dataset.

Figure 2 follows one real laundering pattern from the dataset. It begins at 04:44 UTC with a withdrawal of about 90 USDT from the Tornado Cash mixer (`0x16...D52B`) to an intermediary wallet (`0xe434...f30c`). A portion is diverted or paid as fees, and twenty minutes later the majority (82.416 USDT) moves to a second intermediary wallet (`0xCC6C...d5FB`). At 05:10 UTC, less than 30 minutes after the initial withdrawal, the funds reach a Binance wallet (`0x01...49e`) for off-ramping[6]. This rapid multi-hop transfer[7], carried out during off-peak hours, is the "hop" pattern our derived features are built to detect.

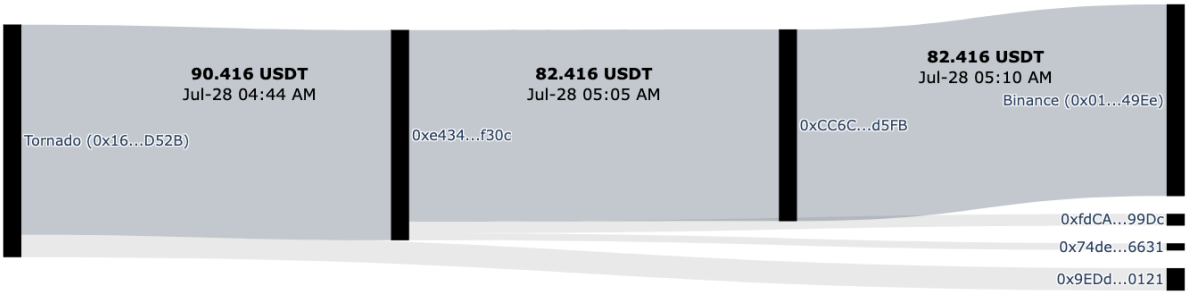


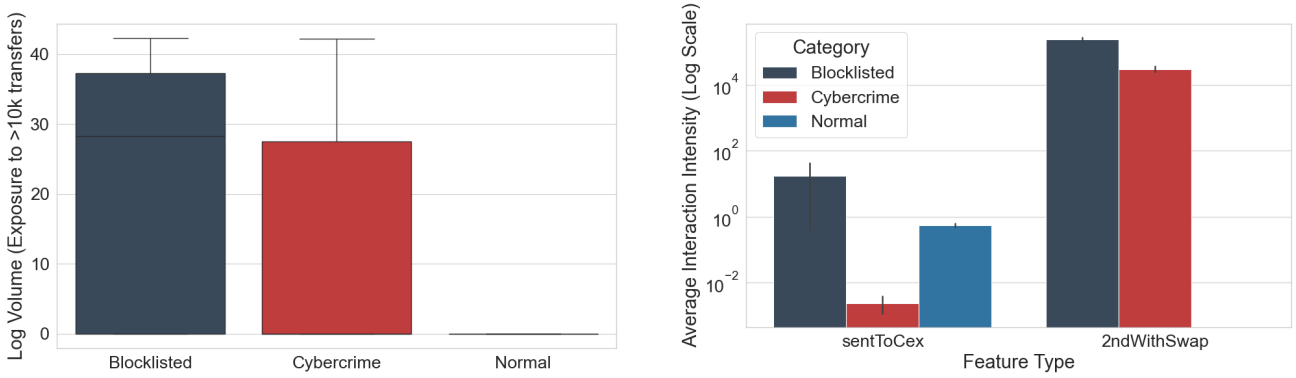
Figure 2. Sankey diagram illustrating a multi-hop transfer where funds move from Tornado through intermediary wallets before reaching Binance within 26 minutes.

Figure 3 plots key interaction and derived-network indicators. In the left panel, legitimate users have no exposure to high-value laundering flows, while Blocklisted and Cybercrime wallets carry many

[6]Off-ramping refers to the process of converting digital assets back into fiat currency or tangible goods, serving as the exit point from the blockchain ecosystem.

[7]Full transaction hashes: `0x5f35d27d5d17f1249f6245544f0426aeb091fdaafb16db956afa636c4ca0111d`, representing the withdrawal from the Tornado Cash router to the first intermediate wallet; `0x41c2e200e8d078883f88aceaf117acb9230a25e7ddce46378f4128e50b74abf1`, recording the transfer from the first intermediate wallet to a downstream recipient approximately twenty minutes after the mixer withdrawal; and `0x4a5be4a5df2099cc53eedaba8b9185c465d0d8dc55591475d971cf1ea46fcca`, representing the subsequent transfer from the downstream wallet to the final destination address.

second-degree connections to transfers above \$10k (`2ndWithOver10k`). In the right panel, legitimate users favor CEX (`sentToCex`), whereas illicit actors mostly route funds through decentralized swap protocols.



(a) (b)

Figure 3. Exploratory plots showing the distribution of `2ndWithOver10k` across classes and interactions with CEXs (`sentToCex`) and swap protocols across classes. **(a)** Indirect High-Value Exposure (Log Scale); **(b)** Protocol Interaction Profile.

3.2.3. Transfer-based features

Transfer-based features capture the direct flow of value into and out of a wallet, including threshold indicators for transfers above \$1000, \$5000, and \$10,000. Their design follows the classic AML literature, where illicit funds move through identifiable entry and exit channels during the placement and integration stages of laundering. Rather than assuming these stages *a priori*, the features record transactional patterns consistent with them and let the model learn whether such movements signal elevated risk.

3.2.4. Temporal and direct features

Temporal features track time-based patterns such as daily transfer bursts, repeated identical-value movements, and rapid reciprocals, which flag abnormal rhythms often tied to laundering, such as sudden high-frequency inflows. For example, the address `0x03d09ec664f9241b23223240a06079300de1b14d` sends a series of repeated 50,000 USDT transfers to the recipient `0xeFd2fd5c18093030E15a08fF8799BEC9c612Ec4f`. On 2025-08-08 the recipient receives four such transfers in about sixteen minutes, totaling 200,000 USDT. Table 3 gives a concrete example of the temporal anomalies these features are designed to capture.

Direct features capture intrinsic wallet properties that do not depend on other addresses, such as whether the address is an externally owned account (EOA)[8], whether its contract code is verified[9], and whether it shows prolonged activity[10]. Together, these properties help distinguish ordinary user wallets from the ephemeral or synthetic addresses common in laundering.

[8] An EOA refers to a standard user-controlled wallet managed by a private key, in contrast to a smart contract account whose behavior is defined by on-chain code.

[9] A verified smart contract publishes its source code and metadata on block explorers such as Etherscan, enabling public inspection and confirmation that the deployed bytecode matches the published source.

[10] Prolonged activity means a wallet stays active over a long span, unlike the short-lived or disposable addresses often used for obfuscation or layering.

Table 3. Temporal excerpt showing repeated inflows of 50,000 USDT from `0x03...` to `0xeF...`.

Transaction	Timestamp	USDT
`0xeb073c65c481114289e8aa63d86ca36ae79854eb270b58ce12961c4d41666124`	2025-08-08 03:04:47 UTC	50,000
`0x16a3e2bb4c73bc50324b3ba24ae391ae3ead13e90deed2ea835f3c1fb4e780cb`	2025-08-08 03:04:59 UTC	50,000
`0x74b8321b5c8817e0dc00c21907e8a85673f1762caa21a3d52328ca6421db6e59`	2025-08-08 03:20:11 UTC	50,000
`0xb4a930b2e5aaf8d09055fea6eda1a134cb1524ba6be86418119eb1714d4329a1`	2025-08-08 03:20:23 UTC	50,000

3.3. Graph construction

The feature-based classifiers above flag suspicious behavior wallet by wallet, so they overlook the links that connect wallets in a laundering network. We add a graph model of the transaction network to capture how illicit funds move between accounts.

To use the relational structure of the ecosystem, we model the data as a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

- Nodes ($\mathcal{V}$): The set of unique wallet addresses identified in the StableAML dataset.
- Edges ($\mathcal{E}$): For each ordered pair of wallets (A, B) with at least one observed transfer, we define a directed edge $e_{A\to B}$. The edge weight $w_{A\to B}$ equals the total stablecoin volume transferred from A to B, aggregated over the full temporal span of the dataset.

We frame detection as supervised multiclass node classification, mapping each node v to a risk category $y \in \{\text{Normal}, \text{Cybercrime}, \text{Blocklisted}\}$. To incorporate behavioral information, we initialize each node $v \in \mathcal{V}$ with a feature vector $\mathbf{x}_v \in \mathbb{R}^{68}$ from the engineered attributes of Section 3.2. As in the tabular setting, we build the graph over the curated labeled wallets and split at the wallet level on this fully labeled node set, as described in the next section. The splits are disjoint, so no wallet appears in both training and test. Because both the graph structure and the node features are cumulative aggregates over the full observation window, the task is static multiclass node classification.

To mirror the restriction on operations for this problem, we restrict the graph. In Web3, native transactions may appear fractured across privacy-preserving transaction protocols, mixers and bridges. Stablecoin transaction ledgers remain comparatively intact, the most persistent and auditable layer available to compliance authorities. By keeping only stablecoin edges over the labeled wallets, we test the message-passing models inside this realistic Compliance Tunnel, not on an idealized and fully observed network. The resulting sparsity, edge density below 0.01, is a structural property of the environment stablecoin AML operates in, and not an artifact of our curation. Thus, the graph-versus-feature comparison in Section 5 measures relative resilience to topological fragmentation, not a general ranking of one architecture family over another.

4. Modeling methodology

The task is multiclass wallet classification. Each wallet that transfers USDT or USDC receives one of the three labels from Section 3 ($K = 3$). We also run a binary version ($K = 2$) that merges the two illicit classes, to assess how well the models separate suspicious from normal.

Since the classes are strongly imbalanced, with Cybercrime and Blocklisted wallets a small share of the raw ecosystem, we evaluate every model with the Macro-F1 score ($F1_{Macro}$). It weights every class equally and rewards a balance of precision and recall, which matters more than raw accuracy when the classes are skewed. We also report the Area Under the Receiver Operating Characteristic (AUROC), but treat it as secondary, since it tends to appear optimistic on imbalanced data where the negative and positive classes are unevenly sampled. This balance matters in an AML setting, where missing a suspicious case, and the fines that follow, costs far more than flagging an innocent transaction, as stressed by the Financial Action Task Force risk-based framework [12]. Existing studies note that recall becomes crucial when the target is a rare but high-risk outcome [16].

Within each family we pick widely used architectures for tabular and graph learning, so any performance gap reflects the family's inductive bias, not an implementation artifact. The linear family is Logistic Regression with ℓ_1/ℓ_2 penalty chosen by grid search, the standard baseline for high-dimensional tabular classification. The tree-ensemble family covers four canonical implementations, Random Forest, XGBoost, CatBoost, and LightGBM, spanning the bagging and gradient-boosting designs used in industry-grade fraud detection. For deep learning we use a fully connected network with Rectified Linear Unit (ReLU) activations and dropout, a representative tabular configuration. For graph learning we evaluate a static model (GraphSAGE) and dynamic ones (EvolveGCN, in both -H and -O variants), which cover the inductive-static and temporal-dynamic axes of the GNN literature most relevant to financial transaction networks. This selection is representative rather than exhaustive; a full ablation across every published variant is beyond our scope and left to future work in Section 6.

We split the dataset 80/20 into training and test sets, giving 13,146 training and 3287 held-out test wallets. The test portion is a fully isolated holdout used once, for the final assessment of generalization.

Inside each split the classes stay balanced, so the majority class does not overwhelm the rare ones and we need no synthetic oversampling. After selecting the best configuration, we retrain on the full training split and evaluate once on the holdout. Appendix B lists the optimized hyperparameters for every model.

4.1. Logistic Regression

We use Logistic Regression (LR) as a clear baseline that shows how well the engineered features distinguish the classes using a linear decision boundary. We take the multinomial form, which expands Logistic Regression to include multiple classes by modeling the joint probability distribution for all K classes at the same time.

For a wallet with feature vector $\mathbf{x} \in \mathbb{R}^{68}$, the probability of class k follows the softmax function:

$$P_k(\mathbf{x}) = \frac{\exp(\beta_k^\top \mathbf{x} + \beta_{0,k})}{\sum_{j=1}^{K} \exp(\beta_j^\top \mathbf{x} + \beta_{0,j})}, \quad k \in \{0,1,2\}$$

where $\beta_k \in \mathbb{R}^{68}$ denotes the class-specific coefficient vector, $\beta_{0,k}$ represents the intercept for class k, and $K = 3$ corresponds to the total number of risk categories. The predicted class is the one with the largest probability mass:

$$\hat{y}(\mathbf{x}) = \operatorname*{argmax}_k P_k(\mathbf{x})$$

We estimate the parameters by minimizing the categorical cross-entropy loss. We use the Stochastic Average Gradient Augmented (SAGA) solver, which performs well on high-dimensional feature spaces and supports the L_1 regularization penalties that we examined in our grid search.

4.2. Tree ensembles

Ensemble methods combine many base estimators to generalize better than a single model, which suits the nonlinear feature interactions typical of laundering. We benchmark four tree ensembles chosen for their established performance on heterogeneous behavioral signals and sparse transaction data [14,16]: Random Forest (RF) and three Gradient Boosting Machines (GBMs), namely XGBoost, LightGBM, and CatBoost.

RF uses bagging to cut variance by averaging an ensemble of independent decision trees. For a wallet with features $\mathbf{x}$, the probability of class k is the average of the empirical class proportions produced by each of the M trees, where $P_m(y = k \mid \mathbf{x})$ denotes the fraction of class-k training samples in the terminal leaf of tree m. We split nodes by the Gini impurity index, and assign the class by:

$$\hat{y}(\mathbf{x}) = \operatorname*{argmax}_k P_k(\mathbf{x}).$$

GBMs, by contrast, build an additive model in a forward stage-wise way. Each new tree is fit to the negative gradient of the loss at the current prediction, which amounts to gradient descent in function space against a global objective.

We use the native multiclass formulation in XGBoost, LightGBM, and CatBoost. The ensemble produces a vector of class-specific scores (logits) $F_k(\mathbf{x})$ from the M trees:

$$F_k(\mathbf{x}) = \sum_{m=1}^{M} f_{m,k}(\mathbf{x}).$$

The softmax turns these into class probabilities:

$$P_k(\mathbf{x}) = \frac{\exp(F_k(\mathbf{x}))}{\sum_{j=1}^{K} \exp(F_j(\mathbf{x}))},$$

and the predicted label is

$$\hat{y}(\mathbf{x}) = \operatorname*{argmax}_k P_k(\mathbf{x}).$$

We fit the parameters by minimizing the multinomial cross-entropy loss:

$$\mathscr{L} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{i,k} \log P_k(\mathbf{x}_i).$$

Appendix B gives the search grids and the chosen hyperparameters.

4.3. Deep neural network

We use a Deep Neural Network (DNN) with two fully connected hidden layers to capture higher-order nonlinear feature interactions. ReLU activations, $f(x) = \max(0,x)$, provide the nonlinearity. Dropout masks a fraction of connections during training to limit overfitting on the high-dimensional features.

We project linearly the last hidden representation to a vector of unnormalized logits $\mathbf{z} \in \mathbb{R}^K$ and normalize them with the softmax function:

$$P_k(\mathbf{x}) = \frac{\exp(z_k(\mathbf{x}))}{\sum_{j=1}^{K} \exp(z_j(\mathbf{x}))}. \tag{1}$$

The predicted label $\hat{y}$ is the class with the largest probability mass:

$$\hat{y}(\mathbf{x}) = \operatorname*{argmax}_k P_k(\mathbf{x}).$$

We optimize the parameters with the Adam algorithm, minimizing the categorical cross-entropy loss over the n training samples:

$$\mathscr{L} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{i,k} \log P_k(\mathbf{x}_i),$$

where $y_{i,k} \in \{0,1\}$ is the binary indicator for the true class membership of sample i, and $P_k(\mathbf{x}_i)$ denotes the predicted class probability.

4.4. Graph neural network architecture

To evaluate graph-based approaches on fragmented stablecoin networks, we use both static and dynamic message-passing architectures.

We adopt the GraphSAGE architecture [28], which learns neighborhood aggregation functions rather than node-specific embeddings. The inductive design generalizes to large and evolving transaction networks by aggregating local neighborhood features. In line with the Compliance Tunnel setting of Section 3.3, our primary results confine GNN message passing to the labeled wallet subgraph. As a robustness check, we also evaluate an extended variant that routes messages through the first- and second-degree neighborhood of the labeled wallets; because the architecture uses two message-passing layers, this expansion supplies the full two-hop receptive field a two-layer model can exploit. The extended-scope results appear by class in Table C1 of Appendix C and are summarized in Section 5.2. Widening the scope reduces the difference from the tree ensembles for the Normal and Cybercrime classes but leaves the Blocklisted metrics unchanged, in line with the short, linear paths of frozen or sanctioned wallets.

The architecture consists of two spatial convolution layers with neighbor sampling to handle graph sparsity. At layer l, the model updates the representation of node v by concatenating its current embedding with an aggregated summary of its local neighborhood. We use the mean aggregator for its efficiency and permutation invariance. The update rule is:

$$h_v^{(l)} = \sigma\left(W^{(l)} \cdot \left[h_v^{(l-1)} \,\middle\|\, \frac{1}{|\mathscr{N}(v)|} \sum_{u \in \mathscr{N}(v)} h_u^{(l-1)} \right] \right),$$

where $h_v^{(0)} = \mathbf{x}_v$ corresponds to the initial feature vector described in Section 3.2, $\|$ denotes the concatenation operation, and $W^{(l)}$ is a learnable weight matrix. The term within the summation computes the element-wise mean of the neighbor embeddings, the average behavioral profile of a wallet's direct transactors. σ is the non-linear activation function, typically a ReLU.

After $L = 2$ layers of aggregation, we project the final node embeddings $h_v^{(L)}$ to the output space of the three target classes via a linear transformation, and the class probabilities follow from the softmax function:

$$P_k(h_v^{(L)}) = \frac{\exp(z_{v,k})}{\sum_{j=1}^{K} \exp(z_{v,j})},$$

with predicted label $\hat{y}_v = \operatorname{argmax}_k P_k(h_v^{(L)})$.

Since GraphSAGE is static, we add EvolveGCN [19] to capture temporal network dynamics. At time step t, the node representations for layer l are updated with a standard graph convolution:

$$H_t^{(l)} = \sigma\left(\widehat{A}_t H_t^{(l-1)} W_t^{(l)}\right)$$

where $\widehat{A}_t$ is the normalized adjacency matrix at time t, $H_t^{(l-1)}$ represents the node features from the previous layer, and $W_t^{(l)}$ is the weight matrix.

EvolveGCN is distinguished by a weight matrix $W_t^{(l)}$ that evolves over time through a recurrent neural network. Within this dynamic family, we evaluate both the -H and -O variants to gauge the effect of feature-based noise. EvolveGCN-H updates the weights with a Gated Recurrent Unit (GRU) driven by the current node features $H_t^{(l-1)}$:

$$W_t^{(l)} = \mathrm{GRU}\left(H_t^{(l-1)}, W_{t-1}^{(l)}\right)$$

EvolveGCN-O instead uses a Long Short-Term Memory (LSTM) network that evolves the weights from past weights alone, bypassing the node feature representations during the temporal update:

$$W_t^{(l)} = \mathrm{LSTM}\left(W_{t-1}^{(l)}\right)$$

The -O variant's independence from node hidden representations during weight updates may make it more resilient to the extreme topological fragmentation in our dataset, which otherwise injects noise into feature-based message passing. Together, the static and dynamic architectures let us test whether the bottleneck in stablecoin AML is architectural or comes from a reliance on persistent edge connectivity in intentionally obfuscated environments.

5. Results and discussion

This section reports the empirical results and their implications for detecting suspicious activity in stablecoin ecosystems. We first evaluate the full three-class setting, then examine feature importance to identify which behavioral and relational attributes drive predictive performance. We then relate these features to known money-laundering typologies and to the structural differences between illicit and restricted wallets, and we close with a binary configuration that merges the suspicious classes for comparison with prior work.

5.1. *Per-class performance analysis*

We break the results down by class, Normal, Cybercrime, and Blocklisted. Table 4 reports precision, recall, F1-score, and OvR AUROC for each category.

The gap between AUROC and F1-score is large for the linear baseline. LR exceeds AUROC 0.93 on every class, but its Blocklisted recall is only 0.49. AUROC masks those false negatives under imbalance, so we prioritize F1-score and recall in our evaluation.

The three classes differ sharply in how detectable they are.

- Normal & Cybercrime: Tree ensembles are near-optimal on both, with F1-score above 0.98. The engineered features capture the consistent, often automated patterns of legitimate high-frequency users and of professionalized cybercrime operations.
- Blocklisted: This category is much harder. Ensemble methods still outperform the LR baseline by a wide margin ($F1 \approx 0.95$ *vs.* 0.63), but the precision-recall gap is widest here. The class is heterogeneous, its wallets frozen for reasons as varied as sanctions evasion and theft recovery, so its transactional footprint is less uniform than the active Cybercrime wallets'.

Deep learning (DNN and GNN) is unstable on this minority class. The GNN recalls Blocklisted entities especially poorly ($Recall \approx 0.58$), far below CatBoost ($Recall \approx 0.94$). Without a dense transaction graph, the deep models miss the signals that tree ensembles extract from the tabular features.

Table 4. Per-class performance metrics on the test set (AUROC, precision, recall, and F1) for each model and class.

Model	Class	AUROC	Precision	Recall	F1
Logistic Regression	Normal	0.9453	0.7877	0.9988	0.8807
	Cybercrime	0.9351	0.9375	0.7614	0.8403
	Blocklisted	0.9353	0.8773	0.4896	0.6285
CatBoost (Best)	Normal	0.9990	0.9927	1.0000	0.9963
	Cybercrime	0.9991	0.9873	0.9831	0.9852
	Blocklisted	0.9943	0.9579	0.9440	0.9509
EvolveGCN-O	Normal	0.9922	0.9075	0.9968	0.9501
	Cybercrime	0.9865	0.9207	0.9419	0.9312
	Blocklisted	0.9602	0.8912	0.5782	0.7014

We provide the full per-class breakdown across all models in Table C1 of Appendix C.

5.2. *Overall model comparison*

Table 5 reports the macro-averaged metrics across all categories. Tree ensembles outperform the others by a clear margin, ahead of the linear baseline and the deep learning models.

GBMs discriminate strongly across all measures. The average AUROC is above 0.997, Macro-F1 is between 0.975 and 0.978, and these averages continue across classes. The performance for Normal is close to perfect precision and recall. For Cybercrime, it is around 0.98 F1, for even the disparate Blocklisted class it reaches about 0.95 F1.

Table 5. Overall multiclass classification performance (macro-averaged metrics) for each model, with the best model in bold.

Model	AUROC	Accuracy	F1	Recall
Logistic Regression	0.9385	0.8387	0.7831	0.7499
Random Forest	0.9976	0.9824	0.9714	0.9708
LightGBM	0.9976	0.9845	0.9755	0.9742
CatBoost	**0.9974**	**0.9857**	**0.9775**	**0.9757**
XGBoost	0.9974	0.9851	0.9766	0.9760
DNN	0.9617	0.9192	0.8708	0.8504
GraphSAGE	0.9683	0.8290	0.8048	0.7699
EvolveGCN-O	0.9796	0.9107	0.8608	0.8389

Tree ensembles perform effectively on blockchain transaction data. In tabular data that mainly uses binary or dummy-encoded indicators, they often perform better than deep neural networks [29]. This is because they divide the feature space based on clear signals, avoiding heavy smoothing or complex learning. Our feature set includes many of these flags, like `isVerifiedContract` and `hasProxyBehaviour`. The ensembles use these flags to identify high-risk typologies.

The deep learning models are less competitive. Both DNN and GNN reach AUROC above 0.96, but their Macro-F1 scores plateau around 0.80 to 0.87. GraphSAGE and EvolveGCN do not outperform the DNN baseline ($F1 \approx 0.86$ *vs.* 0.87).

To assess whether this limitation was an artifact of static modeling, we evaluated the dynamic EvolveGCN architecture. As anticipated in Section 4.4, the EvolveGCN-O variant, which evolves network weights independently of the noisy node features, outperformed both GraphSAGE and its feature-dependent-H counterpart (Macro-F1 0.8608 *vs.* 0.8048 and 0.8349, respectively; see Table 5 and Table C1). Even with this temporal adaptation, EvolveGCN-O reaches at most an F1-score of 0.9312 for the Cybercrime class, far below CatBoost (0.9852). As a further robustness check, we re-ran the GNN suite with message passing extended to the first- and second-degree neighborhood beyond the labeled set. Under this expanded scope, GraphSAGE attains the highest Macro-F1 among the three architectures (0.8670), above EvolveGCN-H (0.8629) and EvolveGCN-O (0.8548). Between the two EvolveGCN variants, EvolveGCN-H slightly outperforms EvolveGCN-O, which suggests that once enough topological context is restored, conditioning weight updates directly on node features provides a small advantage over feature-independent evolution. Even then, the extended EvolveGCN-H reaches at most an F1-score of 0.9265 for the Cybercrime class, still below CatBoost by a wide margin.

This points to a data constraint rather than an architectural one. The transaction graph is very sparse (density < 0.01, where density is the fraction of observed edges relative to the maximum number of possible directed edges $|\mathcal{V}|(|\mathcal{V}|-1)$, excluding self-loops), primarily because the analysis is restricted to USDT and USDC transfers. Laundering typically converts stablecoins into volatile assets such as Wrapped Bitcoin (WBTC) or Ether (ETH) on a DEX to break the chain, so these cross-asset hops fall outside a stablecoin-only graph and the message-passing connectivity the models need is broken. The GNN therefore cannot recover the layering patterns that the ensembles capture through aggregate node features such as `2ndWithSwap`.

Figure 4 plots the per-class curves, where the models separate by how they treat each risk class. For the Normal and Cybercrime classes (Panels (a) and (b)), all tree ensembles are near-optimal, with precision and recall closely aligned. The engineered features capture the consistent, often automated patterns of large-scale cybercrime and hacking.

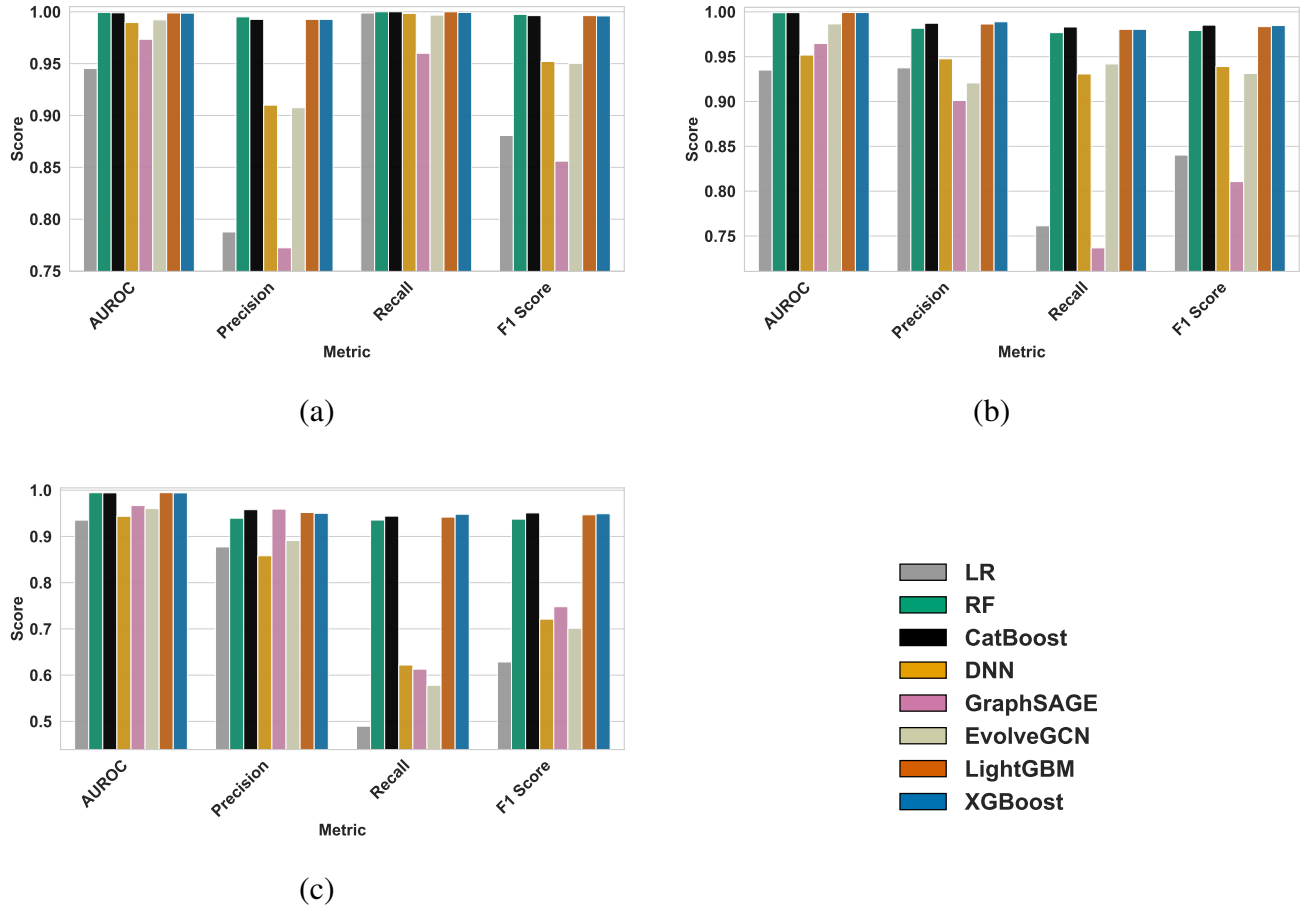


Figure 4. Per-class OvR performance curves for multiclass wallet classification for the Normal, Cybercrime, and Blocklisted classes. **(a)** Normal; **(b)** Cybercrime; **(c)** Blocklisted.

The Blocklisted class (Panel (c)) is more difficult. Compared with the linear baseline, the ensemble models show results on F1 up to 0.32 points higher. However, this is the category with the largest deviation. The restriction-wallet type has a wide variety, and the behavior is more affected by post-restriction circumstances than continued criminal behavior. The DNN and GNN are particularly unstable here, especially in recall, which shows their sensitivity to sparse relational structure. This reinforces our conclusion that tree ensembles provide the best balance between precision and recall for detecting rare, heterogeneous risk events.

5.3. Feature importance

To ensure our findings are reliable and avoid the bias of any single interpretability method, we run a multi-stage consensus ranking. It identifies the most stable predictors of illicit activity by combining three complementary importance measures:

(1) Model-Specific Importance (Built-in): native importance scores from each model. For tree ensembles this is Gini Importance or Mean Information Gain, measuring each feature's contribution to reducing node impurity across all trees. For LR we use the absolute values of the normalized coefficients (β).

(2) Model-Agnostic Permutation Importance: to gauge the functional impact of features on unseen data, we run permutation importance on the full holdout test set, measuring the drop in the model's Macro-F1 score when a feature's values are randomly shuffled. We use 10 shuffling repetitions per feature.

(3) SHAP Values: for the ensemble models we compute SHAP [30] values with the `TreeExplainer` framework to capture complex non-linear interactions. Because Shapley value estimation is costly, we use a representative subsample of 3000 test observations to bound memory requirements while keeping a confidence interval on the importance estimates.

For each model-importance method pair, features are sorted in descending order of importance and assigned ordinal ranks (1 = most important). For each feature we average its ranks across all models and methods, then re-rank on that average to obtain the consensus rank.

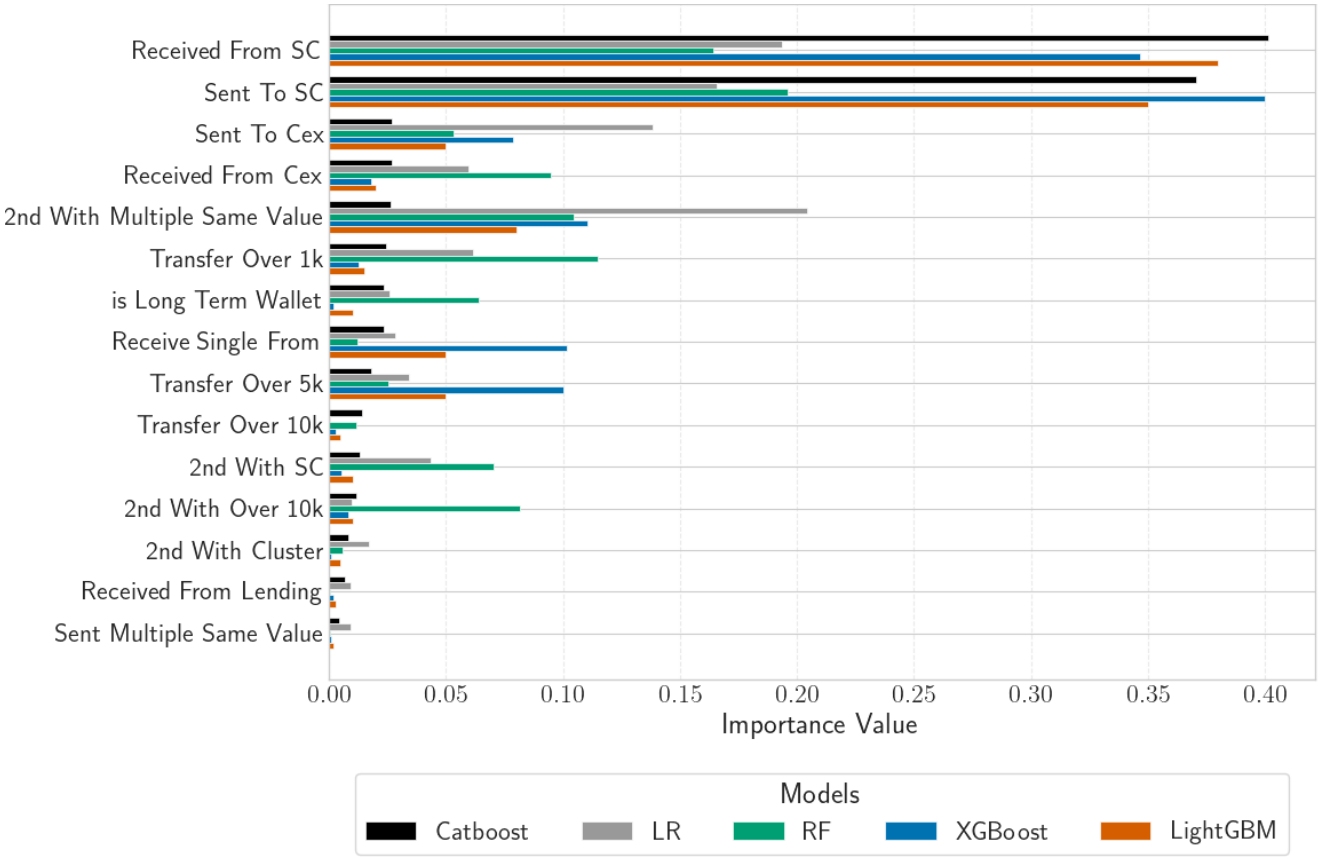


Figure 5. Consensus ranking of the top 15 features across primary models. The reported "Value" represents the normalized importance derived from the multi-stage pipeline, aggregating: **(i)** model-specific impurity/gain for tree ensembles; **(ii)** normalized coefficients (β) for LR; **(iii)** model-agnostic permutation importance on the test set; **(iv)** SHAP values. Ranks are averaged across the linear and non-linear models.

Across all models, structural and volumetric indicators dominate. The top features include direct smart-contract interactions, `receivedFromSC` and `sentToSC`, along with volume thresholds like `transferOver1k` and `transferOver5k`. These serve as immediate behavioral indicators. Second-degree exposure features also rank high. These include `2ndWithMultipleSameValue` (which tracks exposure to wallets making repeated identical-value transfers) and `2ndWithOver10k` (which shows connections to high-value flows). The models extend their focus beyond immediate neighbors to the "hop" patterns of the layering stage.

5.4. Class-specific behavioral signatures

We split feature importance by output class to see how the signals differ across categories. Figure 6 shows the result as a per-class heatmap of feature influence.

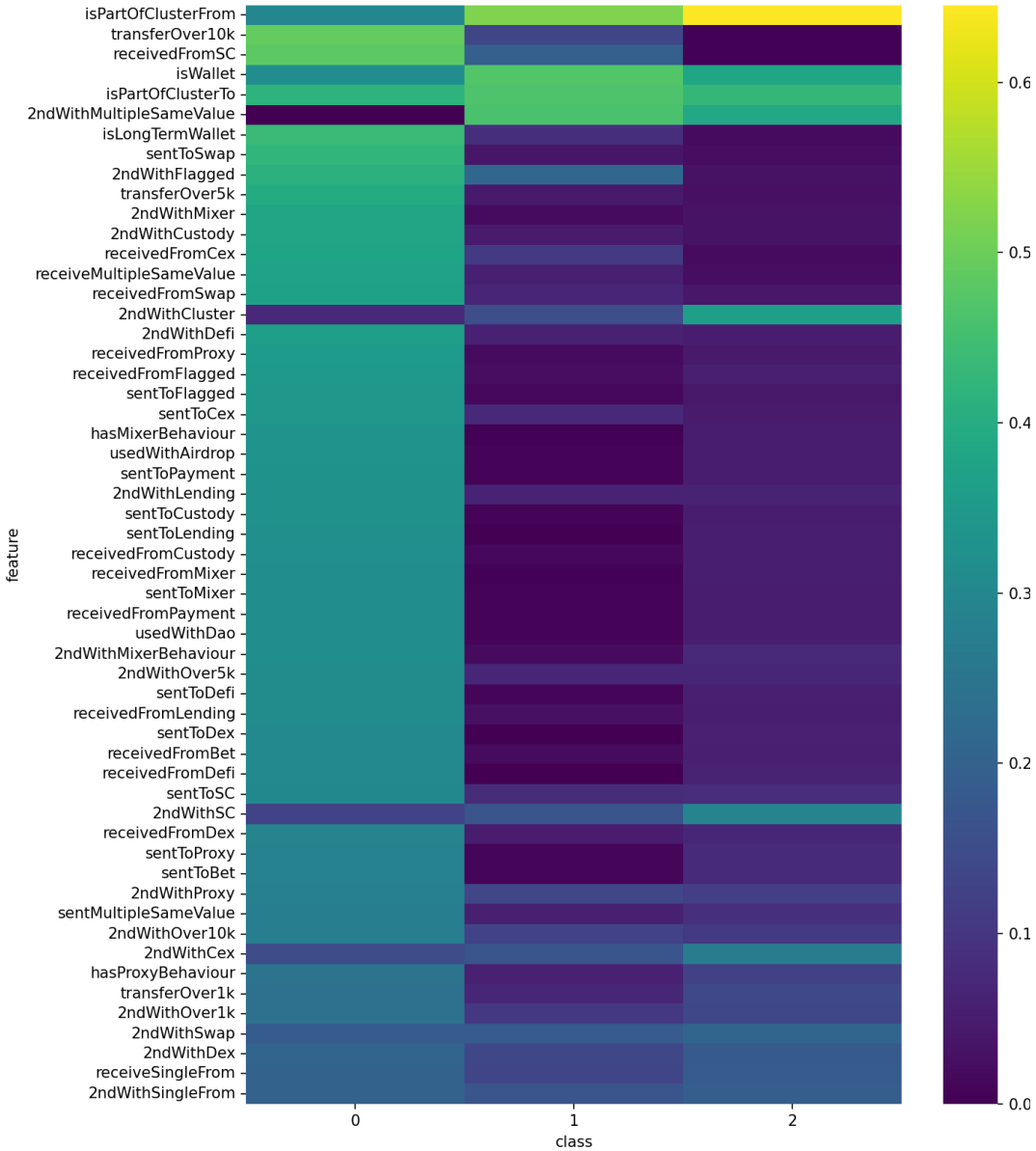


Figure 6. Class-specific feature importance heatmap. Color intensity is the relative influence (normalized aggregated importance) of each feature on the predictions, by output class. Values are averaged across the tree ensembles.

The heatmap separates the risk categories. Features that predict Normal often remain relevant for Blocklisted, and vice versa. Each class has a unique profile that corresponds to the traditional stages of money laundering [3].

(1) Normal: this class shows consistently low importance on the illicit-specific indicators. The model interprets benign activity partly from the lack of strong, coordinated signals seen in the suspicious classes. It does not depend solely on absence, and it identifies benign complexity directly. Legitimate high-frequency actors such as arbitrage bots are identified through interaction with public protocols (`sentToSC`) and account longevity (`activeMonths`), not generic transaction volume. By weighing these benign markers against the obfuscation signatures of the illicit classes (`hasProxyBehaviour`, `2ndWithFlagged`), the framework avoids penalizing ordinary Web3 complexity and targets genuine money-laundering typologies.

(2) Cybercrime: this class shows complex network features, including cluster-membership derivatives (`isPartOfClusterFrom`, `isPartOfClusterTo`) and indirect relational exposure (`2ndWithMultipleSameValue`). This deep, structured signal fits the automated layering stage. Professionalized groups route funds through many intermediary wallets to obscure origins, and under enforcement pressure, they add hops and shift to decentralized bridges [4].

(3) Blocklisted: this class relies on direct interactions (`sentToSC`, `receivedFromCex`) and simpler network connections. This points to placement or constrained post-enforcement activity rather than complex layering. Sanctioned wallets show reduced mobility and simpler paths, and often move reactively into non-freezable tokens such as DAI [4].

Table 6. Principal features mapped to money-laundering typologies.

Feature Group	AML Stage	Observed Behavioral Pattern
SC & CEX Flows	Placement	Direct interaction with smart contracts and centralized endpoints for initial fund entry.
Clustering & Hops	Layering	High relevance of `2ndWithMultipleSameValue` and multi-hop flows for obfuscation.
Long-Term Retention	Integration	Use of `isLongTermWallet` and large-value transfers to identify re-integration points.

The recovered patterns track the classical stages of money laundering.

- Placement: funds enter through CEXs or initial SC interactions (`receivedFromCex`, `sentToSC`), where mixers often weaken the direct link to the origin [3].
- Layering: obfuscation shows up in `transferOver1k/5k/10k` and `2ndWithMultipleSameValue`, signs of value fragmentation and proxy behavior through pass-through wallets [3]. These signals go together with cross-chain bridges and DEXs that lack KYC and enable asset swaps.
- Integration: re-integration shows up in `isLongTermWallet`, which flags wallets that retain funds before redistribution, and in continued mixer interaction for re-anonymization [8].

We ran a qualitative analysis of the errors the best model (CatBoost) makes. Residual false positives come from legitimate high-frequency traders and arbitrage bots. These actors move capital rapidly and in high volume through centralized exchanges and DeFi protocols, which structurally mimics the placement and layering stages. In the multiclass setting, residual confusion arises between the Cybercrime and Blocklisted categories. Both operate under adversarial constraints and often share obfuscation mechanisms

(e.g., CEX pass-throughs and rapid swaps) before a specific enforcement action, so their transactions overlap enough to challenge even non-linear boundaries.

These patterns follow the typical stages of money laundering, but the features generate an AML risk score based on structural characteristics. That score indicates a high likelihood of layering or obfuscation, which strongly suggests illegal activity. However, it does not serve as a legal conclusion of money laundering without off-chain verification.

5.5. Legitimate complexity vs. illicit obfuscation

Distinguishing legitimate high-frequency traders and arbitrage bots from laundering wallets is one of the most difficult aspects of blockchain AML. These benign actors move funds rapidly and use many protocols, much like illicit ones. To assess whether the model relies on laundering-specific behavior and not general complexity, Table 7 compares three CatBoost false-positive (misclassified) arbitrage-bot wallets with three confirmed laundering addresses from the test set, feature by feature.

Table 7. Comparison between three CatBoost false-positive arbitrage-bot wallets and three confirmed laundering-related addresses.

Feature	Arbitrage Bot Examples (FP)	Laundering Examples
`sentToSC`	1 / 0 / 1	0 / 0 / 1
`activeMonths`	2.1 / 1.6 / 1.9	1.5 / 0.1 / 0.0
`isPartOfClusterFrom`	0 / 0 / 0	0 / 0 / 8
`2ndWithFlagged`	0 / 0 / 0	562 / 421 / 0
`sentMultipleSameValue`	0 / 0 / 0	2 / 0 / 2
`sentToCex`	0 / 0 / 0	4 / 0 / 0
`receivedFromCex`	6 / 28 / 4	6 / 1 / 0
`receivedFromSwap`	3 / 0 / 1	0 / 0 / 2
`2ndWithSwap`	0 / 0 / 0	11 / 1 / 1
`2ndWithProxy`	0 / 0 / 0	0 / 0 / 30
`hasProxyBehaviour`	0 / 0 / 0	5 / 3 / 3
`transferOver10k`	0 / 0 / 0	11 / 9 / 16

Table 8 extends this to every false-positive arbitrage-bot wallet in the test set, with feature averages across the group. Arbitrage-bot wallets are zero on all layering markers but retain positive CEX-centric averages (`sentToCex` = 2.48, `receivedFromCex` = 3.07) and longer account longevity (`activeMonths` = 0.89). The laundering wallets show the reverse, with high second-degree exposure to flagged entities (`2ndWithFlagged` = 23.39), indirect swap connectivity (`2ndWithSwap` = 43.80), and large-value transfers (`transferOver10k` = 338.29), and near-zero values on the CEX features. Thus, the residual misclassifications arise from shared volumetric features, not from the framework missing the structural signatures of laundering.

Benign complexity and illicit obfuscation split along structural, not volumetric, lines. The arbitrage-bot wallets have longer account longevity (`activeMonths` = 1.6 to 2.1) and CEX-centric funding but zero on layering markers. The laundering wallets carry the structural signal instead, with non-zero proxy behavior in all three cases and large-transfer volume (`transferOver10k`). The model focuses on layering-stage

topology (e.g., multi-hop proxy patterns, indirect swap exposure), not raw transaction volume. The residual false positives come from shared volumetric overlap, so the framework relies on AML-specific risk signals.

Table 8. Comparison of average values over all false-positive arbitrage-bot wallets identified in the test set ($N = 9$).

Feature	Arbitrage Bot (Average)	Laundering (Average)
`sentToSC`	0	0
`activeMonths`	0.89	0.03
`isPartOfClusterFrom`	0	0
`2ndWithFlagged`	0	23.39
`sentMultipleSameValue`	0	0
`sentToCex`	2.48	0
`receivedFromCex`	3.07	0
`receivedFromSwap`	1.13	0
`2ndWithSwap`	0	43.80
`2ndWithProxy`	0	3.57
`2ndWithMixer`	0	0.01
`hasProxyBehaviour`	0	1.05
`transferOver10k`	0	338.29

5.6. Ablation study

We ran an ablation to check how much the models lean on the proximity signals above. Table 9 reports one model from each family, under the labeled-scope configuration of our main results, with and without these known-bad-proximity screening signals (e.g., `sentToFlagged`, `receivedFromFlagged`).

Ablating these features drops the tabular models only marginally (for CatBoost the Macro-F1 changes negligibly). The GNN family shows a larger relative drop (EvolveGCN-O: -0.0197 Macro-F1, -0.0286 Recall), in line with its reliance on these features as a compact substitute for the structural context its message passing cannot otherwise reconstruct under the labeled-scope setting of Section 4.4. The proximity signals reflect true compliance screening. However, most of the models' predictive ability derives from other behavioral and structural features.

Table 9. Performance with and without label-proximity features.

Model	Metric	With proximity	Without	Δ
CatBoost	Macro-F1	0.9775	0.9678	−0.0097
	Accuracy	0.9857	0.9802	−0.0055
	AUROC	0.9974	0.9974	0.0000
Random Forest	Macro-F1	0.9714	0.9690	−0.0024
	Accuracy	0.9824	0.9808	−0.0016
Logistic Regression	Macro-F1	0.7831	0.7818	−0.0013
DNN	Macro-F1	0.8708	0.8656	−0.0052
EvolveGCN-O	Macro-F1	0.8608	0.8411	−0.0197
	Accuracy	0.9107	0.9032	−0.0075
	Recall	0.8389	0.8103	−0.0286
	AUROC	0.9796	0.9796	0.0000

5.7. Binary classification results

We also run a simplified binary setting (Normal *vs.* Suspicious) to benchmark against established baselines. Here the Cybercrime and Blocklisted classes merge into one Suspicious category.

We compare the result against the Elliptic dataset [8,17], the standard Bitcoin benchmark for illicit-activity detection. The settings and the data differ, since the Elliptic dataset models the Bitcoin graph while ours is the Ethereum account model restricted to stablecoin flows. Both still separate licit from illicit entities in noisy transaction networks, so the comparison holds.

On this reduced task the models are near-perfect, which indicates the expressiveness of the feature set once the problem reduces to benign versus suspicious. The best tree ensemble reaches an F1 of 0.999 (Table 10), well above earlier graph-based benchmarks. Studies using the Elliptic datasets [8] show consistent recall trade-offs. This is true even with complex subgraph classifiers like RevClassify, which uses local graph structure to handle limited data. In our study of account-based stablecoin networks, our results suggest that domain-specific feature engineering can be more effective than simple topological subgraph-based approaches. However, these results and the improvement over the benchmark models should be interpreted with caution. The modeling dataset is constructed using the stratified sampling and filtering procedure described in Section 3, with a controlled class distribution and a Normal class cleaned based on proximity to illicit addresses. Therefore, the reported results mainly reflect the separability of the constructed classes rather than the performance expected under realistic class imbalance, where illicit addresses may represent less than 1% of active wallets, or with the noisier labels typically found in operational data. Section 6 returns to this caution and to the negative-class construction.

Mapping feature importance onto the financial-crime typologies shows that domain-informed feature engineering makes AML detection explainable and ties model performance to regulatory interpretability.

Table 10. Binary classification performance (Normal *vs.* Suspicious) comparing prior benchmark results from the Elliptic dataset with models evaluated in this study.

Model	AUROC	Accuracy	F1-Score	Recall
GCN (Elliptic1)	–	0.8440	0.9730	–
GAT (Elliptic1)	–	0.7230	0.9710	–
Random Forest (Elliptic1)	–	0.7820	0.9770	–
RevClassifyDS (Elliptic2)	0.9740	–	0.9530	–
Logistic Regression	0.9997	0.9953	0.9918	0.9959
Random Forest	1.0000	0.9997	0.9994	0.9988
CatBoost	1.0000	0.9995	0.9991	1.0000
LightGBM	1.0000	0.9998	0.9997	1.0000
XGBoost	1.0000	0.9993	0.9988	0.9994
DNN	0.9994	0.9998	0.9997	0.9994
GraphSAGE	0.9974	0.9929	0.9886	0.9897
EvolveGCN-O	0.9999	0.9970	0.9965	0.9962

6. Conclusion

The evolution of privacy for blockchain introduces a distinct challenge. Decentralized protocols are increasingly adopting ZKPs to obscure transactions. Meanwhile, to keep fiat convertibility and acceptance by regulated entities, stablecoins like USDT and USDC must guarantee auditable provenance.

The industry is shifting from probabilistic risk scoring toward deterministic verification, in which users must mathematically prove that their funds never contacted a blocklisted address. In that setting, stablecoins remain the "transparent choke points" of the crypto economy. This study shows that these transparent footprints alone are enough to flag illicit actors, even as they attempt to exit through privacy-preserving protocols. The decisive factor was domain-informed feature engineering. Tree ensembles outperformed every deep learning approach evaluated, exceeding a Macro-F1 of 0.97, because they split the feature space along the discrete behavioral signals we engineered rather than smoothing over them. They also separate the typologies, distinguishing the complex multi-hop layering of Cybercrime syndicates from the more constrained movement of Blocklisted entities. Aligning these predictions with the placement, layering, and integration stages provides a blueprint for stablecoin AML detection, a baseline that should remain valid as the ledger economy becomes more private. Our framework contains some limitations. It uses static data and does not include dynamic monitoring or adaptive labeling. The mapping between predicted risk classes and the placement, layering, and integration stages is analytical rather than prescriptive. It shows which typologies can be identified using the engineered features, but the results should not be treated as a unique basis for compliance decisions.

The Normal class is treated as benign based on the available intelligence and screening sources, and is not assumed to be entirely free of illicit activity. Some illicit wallets may remain in this class if current sources have not identified or flagged them. In addition, the Proximity Exclusion Filter removes the benign candidates closest to the illicit frontier, which gives the model a cleaner separation between licit and illicit wallets than a real deployment would encounter. The near-perfect binary results therefore reflect the separability of the constructed dataset more than the accuracy to expect once real-world imbalance and label noise are present. These constraints temper the headline numbers, but they do not affect the comparative result, since every model was evaluated under identical conditions. There are several areas for future work. These include using streaming data for real-time scoring, updating labels as new intelligence becomes available, and validating the framework against the audit and reporting requirements of MiCA and the GENIUS Act. Before the framework is used in practice, its external validity should also be evaluated using temporal splits, incident- or cluster-level splits, and more realistic class imbalance. In these settings, rank-based metrics such as PR-AUC and Precision@K would provide a more meaningful evaluation than metrics obtained from a balanced test set.

Data availability statement

The StableAML dataset and the code supporting the findings of this study are available in the OpenAML Project repository (Linux Foundation FINOS initiative) at https://github.com/finos-labs/dtcch-2025-OpenAML. The underlying USDT and USDC events are publicly retrievable from Ethereum blockchain.

Acknowledgments

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Declaration of generative AI and AI-assisted technologies

During the preparation of this manuscript, the authors used Grammarly and Claude only for language polishing and revising. The authors take full responsibility for the content of the manuscript.

Authors' contribution

Conceptualization, L.J., H.L. and A.B.; methodology, L.J., H.L. and A.B.; software, L.J. and H.L.; validation, L.J., H.L. and A.B.; formal analysis, L.J. and A.B.; data curation, L.J.; writing—original draft preparation, L.J.; writing—review and editing, H.L. and A.B.; supervision, A.B. All authors have read and agreed to the published version of the manuscript.

Conflicts of interest

The authors declare no conflicts of interest.

Appendix A. Full feature list

This appendix provides the complete list of 68 engineered features, categorized by their extraction logic and data type.

Table A1. Engineered features capturing wallet behaviors and interactions.

Feature Name	Description	Type	Source
`hasKYC`	Interacted with a protocol requiring KYC	Boolean	Interaction
`receivedFromPayment`	Received funds from a payment service	Numeric	Interaction
`receivedFromBet`	Received funds from a betting protocol	Numeric	Interaction
`receivedFromCex`	Received funds from a centralized exchange	Numeric	Interaction
`receivedFromCustody`	Received funds from a cold wallet or treasury	Numeric	Interaction
`receivedFromDefi`	Received funds from a DeFi protocol	Numeric	Interaction
`receivedFromDex`	Received funds from a decentralized exchange	Numeric	Interaction
`receivedFromFlagged`	Received funds from a flagged address	Numeric	Interaction
`receivedFromLending`	Received funds from a lending protocol	Numeric	Interaction
`receivedFromMixer`	Received funds from a mixer service	Numeric	Interaction
`receivedFromSC`	Received funds from a smart contract	Numeric	Interaction
`receivedFromStake`	Received funds from a staking protocol	Numeric	Interaction
`receivedFromSwap`	Received funds from a swap protocol	Numeric	Interaction
`sentToBet`	Sent funds to a betting protocol	Numeric	Interaction
`sentToCex`	Sent funds to a centralized exchange	Numeric	Interaction
`sentToCustody`	Sent funds to a cold wallet or treasury	Numeric	Interaction
`sentToDefi`	Sent funds to a DeFi protocol	Numeric	Interaction
`sentToDex`	Sent funds to a decentralized exchange	Numeric	Interaction
`sentToFlagged`	Sent funds to a flagged address	Numeric	Interaction

Table A1. *Cont.*

Feature Name	Description	Type	Source
sentToLending	Sent funds to a lending protocol	Numeric	Interaction
sentToMixer	Sent funds to a mixer service	Numeric	Interaction
sentToPayment	Sent funds to a payment service	Numeric	Interaction
sentToSC	Sent funds to a smart contract	Numeric	Interaction
sentToStake	Sent funds to a staking protocol	Numeric	Interaction
sentToSwap	Sent funds to a swap protocol	Numeric	Interaction
usedWithAirdrop	Interacted with an airdrop participant	Numeric	Interaction
usedWithDao	Interacted with a DAO address	Numeric	Interaction
2ndWithMixBehaviour	Second-degree connection with mixer behavior	Numeric	Derived
2ndWithMultipleSameValue	Second-degree connection with repetitive same-value transfers	Numeric	Derived
2ndWithBet	Second-degree connection with betting wallet	Numeric	Derived
2ndWithCex	Second-degree connection with centralized exchange	Numeric	Derived
2ndWithCluster	Second-degree connection with clustered wallets	Numeric	Derived
2ndWithCustody	Second-degree connection with cold wallet	Numeric	Derived
2ndWithDefi	Second-degree connection with DeFi protocol	Numeric	Derived
2ndWithDex	Second-degree connection with decentralized exchange	Numeric	Derived
2ndWithFlagged	Second-degree connection with flagged address	Numeric	Derived
2ndWithLending	Second-degree connection with lending protocol	Numeric	Derived
2ndWithMixer	Second-degree connection with mixer	Numeric	Derived
2ndWithOver1k	Second-degree connection with $>$ 1k transfer	Numeric	Derived
2ndWithOver5k	Second-degree connection with $>$ 5k transfer	Numeric	Derived
2ndWithOver10k	Second-degree connection with $>$ 10k transfer	Numeric	Derived
2ndWithPayment	Second-degree connection with payment wallet	Numeric	Derived
2ndWithProxy	Second-degree connection with proxy wallet	Numeric	Derived
2ndWithSC	Second-degree connection with smart contract	Numeric	Derived
2ndWithSingleFrom	Second-degree connection with single outgoing wallet	Numeric	Derived
2ndWithSingleTo	Second-degree connection with single receiving wallet	Numeric	Derived
2ndWithStaking	Second-degree connection with staking protocol	Numeric	Derived
2ndWithSwap	Second-degree connection with swap protocol	Numeric	Derived
3rdWithFlagged	Third-degree connection with flagged wallet	Numeric	Derived
circleDetected	Reciprocal transfers within 24 hours	Numeric	Derived
clusterScore	Participation in closed send/receive groups	Numeric	Derived
hasMixerBehaviour	Imbalance between sent and received	Numeric	Derived
hasProxyBehaviour	Received and sent same amount in 24 hours	Numeric	Derived
isPartOfClusterFrom	Sends funds with same value and destination	Boolean	Derived
isPartOfClusterTo	Receives funds with same value and origin	Boolean	Derived
receivedFromProxy	Received funds from proxy wallet	Numeric	Derived
sentToProxy	Sent funds to proxy wallet	Numeric	Derived
receiveMultipleSameValue	Received multiple transfers with same value	Numeric	Transfer
receiveSingleFrom	Multiple transfers from a single address	Numeric	Transfer
sentMultipleSameValue	Sent multiple transfers with the same value	Numeric	Transfer
sentToSingleAddress	Multiple transfers to a single address	Numeric	Transfer
transferOver1k	Transfer over 1000 USD-equivalent	Numeric	Transfer
transferOver5k	Transfer over 5000 USD-equivalent	Numeric	Transfer
transferOver10k	Transfer over 10,000 USD-equivalent	Numeric	Transfer
highFrequency	More than 10 transfers in the same day	Numeric	Temporal
isLongTermWallet	Active months	Numeric	Temporal
isVerifiedContract	Smart contract with verified source code	Boolean	Direct
isWallet	Address has no bytecode (EOA)	Boolean	Direct

Appendix B. Hyperparameter optimization

To ensure fair comparison and reproducibility, we performed a full Grid Search with 5-fold cross-validation on the training set to identify the optimal hyperparameters for each model. The optimization objective was to maximize the Macro-F1 score, ensuring a balanced trade-off between precision and recall across the imbalanced classes.

For Logistic Regression, we adjusted the regularization strength (C) and penalty type, using the `saga` solver for its fast convergence on large datasets, especially with L_1 penalties. For tree ensemble methods, both bagging (Random Forest) and boosting algorithms (XGBoost, LightGBM, CatBoost), we fine-tuned parameters related to tree complexity, such as depth and number of splits, as well as ensemble size, including the number of estimators and learning rate.

Tables B1, B2, and B3 give the search grids, the rationale for each parameter, and the selected values.

Table B1. Logistic Regression optimization settings. The `saga` solver was fixed for all experiments.

Parameter	Search Grid	Best Value	Design Rationale
C (Inverse Regularization)	$[10^{-3}, 10^{1}]$	10	Controls model complexity; higher values correspond to weaker regularization.
Penalty	$\{L_1, L_2\}$	L_1	L_1 promotes feature sparsity, whereas L_2 discourages excessively large weights.

Table B2. Random Forest (Bagging) optimization settings.

Parameter	Search Grid	Best Value	Design Rationale
$n_{\texttt{estimators}}$	$[100, 500]$	400	Number of trees; increasing the number generally improves ensemble stability.
`min_samples_split`	$[2, 10]$	5	Minimum number of samples required to split an internal node.
`min_samples_leaf`	$[2, 5]$	2	Minimum number of samples required at a leaf node, providing smoothing.
`max_features`	$\{\text{sqrt}, \text{log2}\}$	sqrt	Controls the number of features considered at each split and helps decorrelate trees.

Table B3. Gradient Boosting optimization settings. Best values are reported in the order XGBoost / LightGBM / CatBoost.

Parameter	Search Grid	Best Values	Rationale
$n_{\texttt{estimators}}$	$[100, 500]$	400	Number of trees; increasing the number generally improves ensemble stability.
Learning Rate	$[0.01, 0.3]$	0.1 / 0.01 / 0.25	Controls the step size; lower values generally require more estimators.
Max Depth	$[3, 10]$	9 / 10 / 7	Controls interaction complexity and helps regulate overfitting.
Subsample	$[0.8, 1.0]$	0.8 / 0.8 / –	Fraction of training data used in each boosting iteration.
Colsample	$[0.8, 1.0]$	0.8 / 0.8 / –	Fraction of features considered for each tree.

Appendix C. Full three-class classification performance results

Detailed multiclass metrics are reported for Normal (0), Cybercrime (1), and Blocklisted (2). The breakdown shows where tree ensembles perform better than linear baselines.

Table C1. Detailed multiclass performance metrics for each model, with labels representing Normal (0), Cybercrime (1), and Blocklisted (2) entities.

Model	Class	AUROC	Precision	Recall	F1	Support
Logistic Regression	0	0.9453	0.7877	0.9988	0.8807	1623
	1	0.9351	0.9375	0.7614	0.8403	1182
	2	0.9353	0.8773	0.4896	0.6285	482
Random Forest	0	0.9994	0.9951	1.0000	0.9975	1623
	1	0.9990	0.9817	0.9768	0.9792	1182
	2	0.9946	0.9395	0.9355	0.9375	482
CatBoost	0	0.9990	0.9927	1.0000	0.9963	1623
	1	0.9991	0.9873	0.9831	0.9852	1182
	2	0.9943	0.9579	0.9440	0.9509	482
LightGBM	0	0.9989	0.9927	1.0000	0.9963	1623
	1	0.9993	0.9864	0.9805	0.9835	1182
	2	0.9947	0.9518	0.9419	0.9468	482
XGBoost	0	0.9987	0.9927	0.9994	0.9960	1623
	1	0.9992	0.9889	0.9805	0.9847	1182
	2	0.9942	0.9501	0.9481	0.9491	482
DNN	0	0.9896	0.9100	0.9986	0.9522	1623
	1	0.9518	0.9477	0.9319	0.9390	1182
	2	0.9436	0.8581	0.6208	0.7211	482
GraphSAGE (labeled)	0	0.9735	0.7725	0.9601	0.8557	1623
	1	0.9647	0.9010	0.7366	0.8109	1182
	2	0.9667	0.9592	0.6131	0.7481	482
GraphSAGE (extended)	0	0.9776	0.8702	0.9943	0.9281	1623
	1	0.9755	0.9626	0.9056	0.9332	1182
	2	0.9605	0.8827	0.6365	0.7397	482
EvolveGCN-O (labeled)	0	0.9922	0.9075	0.9968	0.9501	1623
	1	0.9865	0.9207	0.9419	0.9312	1182
	2	0.9602	0.8912	0.5782	0.7014	482
EvolveGCN-O (extended)	0	0.9919	0.8619	0.9968	0.9245	1623
	1	0.9870	0.9631	0.8729	0.9158	1182
	2	0.9526	0.8469	0.6324	0.7241	482
EvolveGCN-H (labeled)	0	0.9886	0.8412	0.9931	0.9108	1623
	1	0.9652	0.9574	0.8600	0.9061	1182
	2	0.9534	0.8402	0.5821	0.6877	482
EvolveGCN-H (extended)	0	0.9935	0.8943	0.9880	0.9388	1623
	1	0.9844	0.9433	0.9102	0.9265	1182
	2	0.9517	0.8337	0.6388	0.7234	482

References

[1] Verafin N. Global financial crime report. 2024. Available: https://nd.nasdaq.com/rs/303-QKM-463/images/2024-Global-Financial-Crime-Report-Nasdaq-Verafin-20240115.pdf (accessed on 15 October 2025).

[2] Chainalysis. The 2025 crypto crime report. 2025. Available: https://go.chainalysis.com/2025-Crypto-Crime-Report (accessed on 12 December 2025).

[3] Lin D, Wu J, Fu Q, Yu Y, Lin K, *et al.* Towards understanding crypto money laundering in Web3 through the lenses of Ethereum heists. *arXiv* 2023, arXiv:2305.14748.

[4] Griffin J, Mei K, Wang Z. Are crypto anti-money laundering policies effective? *SSRN* 2025.

[5] Clark J, Demirag D, Moosavi M. Demystifying stablecoins. *Commun. ACM* 2020, 63(7):40–46.

[6] TRM Labs. 2026 crypto crime report. 2026. Available: https://www.trmlabs.com/reports-and-whitepapers/2026-crypto-crime-report (accessed on 5 March 2026).

[7] Visa. Visa onchain analytics dashboard. 2026. Available: https://visaonchainanalytics.com/transactions (accessed on 5 March 2026).

[8] Pocher N, Zichichi M, Merizzi F, Shafiq MZ, Ferretti S. Detecting anomalous cryptocurrency transactions: an AML/CFT application of machine learning-based forensics. *Electron. Mark.* 2023, 33(1):37.

[9] Liu J, Yin C, Wang H, Wu X, Lan D, *et al.* Graph embedding-based money laundering detection for Ethereum. *Electronics* 2023, 12(14):3180.

[10] Akartuna EA, Johnson SD, Thornton A. Preventing the money laundering and terrorist financing risks of emerging technologies: an international policy Delphi study. *Technol. Forecasting Social Change* 2022, 179:121632.

[11] Europol. Internet organised crime threat assessment. 2020. Available: https://www.europol.europa.eu/publications-events/main-reports/internet-organised-crime-threat-assessment-iocta-2020 (accessed on 2 November 2025).

[12] Force FAT. Updated guidance for a risk-based approach to virtual assets and VASPs. 2021. Available: https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Guidance-rba-virtual-assets-2021.htm (accessed on 23 September 2025).

[13] Manning M, Akartuna EA, Johnson S. Opportunities to future crime: scoping the future of money laundering and terrorist financing through cryptoassets. *Technol. Forecasting Social Change.* 2025, 210:123894.

[14] Chen Z, Van Khoa LD, Teoh EN, Nazir A, Karuppiah EK, *et al.* Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review. *Knowl. Inf. Syst.* 2018, 57(2):245–285.

[15] Gorte AS, Mohod SW, Keole RR, Mahore TR, Pande S. Credit card fraud detection using various machine learning and deep learning approaches. In *Proceedings of International Conference on Innovative Computing and Communications*, Delhi, India, February 17–18, 2023, pp. 621–628.

[16] Shojaeinasab A. Decoding illicit bitcoin transactions: a multi-methodological approach for anti-money laundering and fraud detection in cryptocurrencies. Doctoral Thesis, University of Victoria, 2024.

[17] Weber M, Domeniconi G, Chen J, Weidele DKI, Bellei C, *et al.* Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics. *arXiv* 2019, arXiv:1908.02591.

[18] Lo WW, Layeghy S, Portmann M. Inspection-L: practical GNN-based money laundering detection system for bitcoin. *arXiv* 2022, arXiv:2203.10465.

[19] Pareja A, Domeniconi G, Chen J, Ma T, Suzumura T, *et al.* EvolveGCN: evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, New York, USA, February 7–12, 2020, pp. 5363–5370.

[20] Rossi E, Chamberlain B, Frasca F, Eynard D, Monti F, *et al.* Temporal graph networks for deep learning on dynamic graphs. *arXiv* 2020, arXiv:2006.10637.

[21] Berentsen A, Lenheimer J, Nyffenegger R. An introduction to zero-knowledge proofs in blockchains and economics. *Review* 2023, 105(4):1.

[22] Xue Y, Wang J. Design of a blockchain-based traceability system with a privacy-preserving scheme of zero-knowledge proof. *Secur. Commun. Netw.* 2022, 2022(1):5842371.

[23] Solunke H, Bhaladhare P. Blockchain approaches for privacy preservation: a review. In *Proceedings of 2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU)*, Bhubaneswar, India, March 1–2, 2024, pp. 1–6.

[24] Chang V, Baudier P, Zhang H, Xu Q, Zhang J, *et al.* How blockchain can impact financial services—the overview, challenges and recommendations from expert interviewees. *Technol. Forecasting Social Change* 2020, 158:120166.

[25] Vatsa VR. Stablecoin growth and market dynamics. *SSRN* 2025.

[26] Fajri KF, Fachrezzi BR, Urumsah D. AI-driven RegTech and crypto laundering: the dilemmas between financial crime prevention and privacy law. *J. Interdiscip. Econ.* 2026.

[27] US Department of the Treasury, Office of Foreign Assets Control. Specially Designated Nationals and Blocked Persons List (SDN List). 2024. Available: https://sanctionssearch.ofac.treas.gov/ (accessed on 13 October 2025).

[28] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. In *Proceedings of Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Long Beach, USA, December 4–9, 2017.

[29] Grinsztajn L, Oyallon E, Varoquaux G. Why do tree-based models still outperform deep learning on tabular data? *Adv. Neural Inf. Process. Syst.* 2022, 35:507–520.

[30] Lundberg SM, Lee SI. A unified approach to interpreting model predictions. In *Proceedings of Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Long Beach, USA, December 4–9, 2017.