MuleTrack: A Lightweight Temporal Learning Framework for Money Mule Detection in Digital Payments

Ganesh Jambhrunkar, Harsh Sharma, Saurav Singla, and Thirumalai Kailasam

National Payments Corporation of India, Mumbai
{ganesh.jambhrunkar,
harsh.sharma,saurav.singla,Thirumalai.Kailasam}@npci.org.in

Abstract. Money laundering is a substantial danger to financial systems, with money mules playing an important role in hiding the origins of illicit funds. The current paper presents a novel hybrid framework for detecting money mule accounts within the *Unified Payments Interface (UPI)*. The UPI is India's payment gateway, developed by National Payments Corporation of India (NPCI) that processed more than 130 billion transactions worth US$ $\approx$ 2.5 trillion in year 2024, representing 9% of the global digital transaction traffic. In this paper, we present the design and implementation of large-scale data analytics for mule detection at the NPCI. Our approach integrates domain-driven heuristics with probabilistic modeling via Markov Chain, supported by comparisons with supervised machine learning models including logistic regression, XGBoost, Artificial Neural Network and Random Forest. The proposed method captures temporal patterns in account behavior, leading to better identification of money laundering strategies. The Markov chain model reveals superior performance compared to other techniques, making it a great choice for anti-money laundering (AML) applications. This work advances the state-of-the-art in AML by contributing an interpretable and scalable detection methodology.

Keywords: money laundering · money mule · machine learning · markov chain · transition matrix.

1 Introduction

Context. As per the United Nations office on Drugs and Crime, money laundering is estimated to be worth between 2.1% and 4% of the global economic output [1]. Money laundering is a method of cleaning illicit funds, but as Madigner noted in [2], laundered funds are never entirely clean. It is a global challenge that undermines public trust in financial institutions and facilitates serious crimes such as corruption, drug trafficking, and terrorism [3, 4, 5].

A critical link in the money laundering chain is the money mule. According to [6], money mule is an individual who receives funds from an external source into their bank account, later transfers the funds to other account/accounts or

takes it out in cash form to hand over to some other person. Money mules are playing a key role in layering laundered funds. In return, they usually receive an incentive for their help. Some individuals knowingly involve in such activities, but others are misled to become mules under false tricks.

The rise of digital payments has increased the complexity of detecting and preventing money laundering activities.In particular, UPI has revolutionized digital transactions in India by enabling instant, real-time, peer-to-merchant, and peer-to-peer fund transfers across banks and wallets. Having processed approximately billion transactions monthly, UPI has become a dominant financial channel for individuals and small businesses. Fraudsters often exploit UPI accounts by using them for layering transactions or rerouting illicit funds, especially through mule networks.

Motivation. Conventional AML frameworks are largely rule-based and, while useful, these systems are limited in their ability to adapt to rapidly changing laundering strategies. Money mule accounts often display a gradual change in behavior,initially operating normally before suddenly receiving large transfers or rapidly transferring funds elsewhere. Static classification models, even those powered by machine learning, often fail to capture such temporal evolution in account behavior [7, 8]. This limitation is particularly evident in digital payments, such as UPI, where transactions are often low value, high frequency, peer-to-peer, and settled instantly. These characteristics make UPI-based financial systems especially vulnerable to mule operations. Traditional AML models lack the agility to detect such transitions in near-real time. In contrast, Markov Chain provides a powerful mathematical framework to capture such sequences. They model the likelihood of state transitions,such as from 'normal' to 'suspicious' or 'dormant' to 'active',over time [9, 10].

Challenges. Despite advancements in transaction monitoring and fraud analysis, several challenges complicate the detection of mule accounts:

Privacy and Confidentiality. Access to large-scale labeled financial data is restricted, preventing validation. *Behavioural Similarity.* Mule accounts often mimic legitimate user activity, making them difficult to distinguish based on static rules. *Scalability.* AML models must be computationally efficient to operate at large scale like UPI. *Lack of Interpretability.* Deep learning models may provide high accuracy but are difficult to interpret,especially problematic for regulatory compliance.

Gaps. Markov Chain and their variants have been widely used in marketing [11, 12] and credit card fraud detection [13, 14]. Some ensemble models incorporating Graphs and Hidden Markov Models (HMMs) have also been explored in financial anomaly detection [15]. Recent work has explored deep learning, graph-based fraud detection, and ensemble methods for AML applications [16, 17, 18, 19]. Although effective, these methods often lack transparency, require extensive labeled datasets, are computationally expensive, or are not optimized for sequential behavioral modelling [20, 21, 22, 23]

Contributions. To bridge these gaps, this paper proposes a novel approach to detect money mule accounts in a large-scale application like UPI using a

simple and interpretable Markov chain framework. Our key contributions are as follows.

- A temporal modeling strategy that captures weekly transitional patterns in account behavior which helps to flag accounts with unusual state patterns.
- A comparative evaluation of the Markov model with standard machine learning classifiers including Logistic Regression, Artificial Neural Networks, XGBoost, and Random Forest.
- A lightweight and interpretable approach suitable for deployment in production-scale AML systems

2 Related Work

The growing threat of money laundering has driven significant research in automated transaction monitoring and anomaly detection within financial systems. Initial AML systems have been largely rule-based, relying on threshold flags and domain knowledge to define suspicious activity. These systems are limited in their adaptability to new fraud patterns and often produce high false positives [3].

With the emergence of large-scale digital transactions, researchers have applied supervised learning models such as logistic regression, random forests, and gradient-boosting machines to detect fraudulent transactions [24, 25]. Although these models provide predictive power, they often lack interpretability and do not capture the temporal progression of mule activity. Some studies have been extended to deep learning models such as LSTM and CNNs which are capable of handling sequential data [26].

Markov Chain and its variants have been explored in financial anomaly detection due to their ability to model evolving account behavior. [13] used Hidden Markov Models (HMMs) to detect anomalous credit card usage by comparing observed sequences with expected cardholder behavior. [14] improved this by applying ensemble HMMs with custom feature extraction, demonstrating improved fraud detection but with increased complexity.

To date, no known work has applied simple first-order Markov Chain for money mule detection on a real-world dataset. Most studies rely on static features or black-box models that ignore behavioral evolution. Furthermore, reproducibility remains a challenge in AML research, as real-world data is rarely available, and methodologies are often unpublished due to confidentiality. Our proposed method directly models behavioral evolution, capturing subtle shifts indicative of money mule activity.

3 Methodology

This section outlines the various methodologies, dataset, and its preprocessing employed for mule detection.

Dataset and Preprocessing We sampled approximately $1.6M$ accounts for our analysis without considering any Personally Identifiable Information (PII),

all of which have been active for more than six months in the UPI ecosystem. Both the classes, i.e. legitimate and mule, were present in selected accounts. The ground truth labels for mules are reported to NPCI by various banking, regulatory and government agencies of India. UPI raw data is a transaction metadata such as amount sent, time, etc. To prepare the data for Markov chain model, we aggregated weekly data for each of the selected accounts, considering a 6-month window (24 weeks) from the current month (t_m) to five months prior (t_m-5) and weekly features were extracted to summarize transactional behavior. As Markov Chain is a probabilistic model, which identifies suspicious accounts using state transition probabilities, no split needed as training and testing data. We train the other models using account level monthly aggregated data from the previous month (t_m-1) and test them using current month's (t_m) aggregated data. Our analysis relies on a comprehensive feature set, a subset of which is presented in Table 1

Table 1: Sample features used for the models

Feature	**Description**
Inward Amount	Amount received by the account in a week
Inward Transaction Frequency	Count of incoming transactions
Transaction Time	Time at which the transaction occurred
Failure Amount	Total amount involved in failed transactions
Max Amount	Maximum amount received by the account
No. of Beneficiaries	Number of payees of the account

3.1 Markov Chain

Markov Chain model is a probabilistic technique that describes systems transitioning between different states over time. The model relies on a property called Markov Property, which states that the next state entirely depends on the current state and not on the sequence of previous states [9]. Markov chain concept

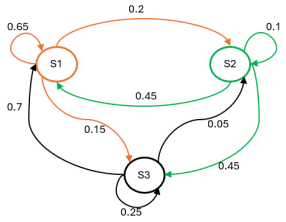

Fig. 1: Markov Chain Process

can be illustrated using a digraph. Fig. 1 represents three states (here, S1, S2 and

Table 2: States inferred from the features

State	Description
S_1 *Normal transaction activity*	Signifies steady, low-to-medium transaction volumes, typical of genuine accounts.
S_2 *Rapid surge in inflows*	Sudden increase in incoming transactions; may be a signal of inflow of illegal funds.
S_3 *Rapid surge in outflows*	Sudden and large outflows of money; usually seen in accounts with suspicious transaction patterns.
S_4 *High transaction velocity*	Indicative of rapid transaction activity, usually associated with illicit fund dispersal.
S_5 *Dormancy*	Periods of inactivity followed by sudden transactions.

S3) Markov approach. The arrows denote the transition from initial state to final state and the number linked with the arrows shows the transition probability.

Transition Probabilities The transition between the various states is represented using a Transition Probability Matrix (TPM) [10],given as $M = [S_{ij}]$ Here, element S_{ij} denotes the probability of transitioning from state i to state j. As we in our analysis we have identified 5 states so M will be a $5X5$ matrix.The summation of each row of M is 1.

Application to Money Mule Detection

Feature Engineering The weekly profile of accounts is categorized into distinct states as given in the Table 2, using domain-specific rules derived from observed patterns. These rules are designed to capture deviations from baseline activity while ensuring operational confidentiality with representative example being:

To identify accounts exhibiting anomalous inflow patterns, historical transaction behaviour is analysed using statistical benchmarks derived from past activity. Weekly inflow metrics are evaluated against dynamically computed thresholds, which are established through empirical analysis of longitudinal trends. A state transition to S_2 ('Sudden Spike in Inflows') is triggered when observed inflows significantly exceed predefined thresholds, calibrated to reflect deviations from established baselines.

Specific parameters and computational details are withheld to prevent adversarial exploitation, as disclosure could enable circumvention of detection mechanisms.

TPM Preparation Historical transaction data was analyzed to estimate the probabilities for state transitions. To construct TPM, the following steps were undertaken:

– *State Transitions Identification:* Weekly state assignments for each account were analyzed to identify transitions from one state to another in successive

weeks. For example, if an account transitioned from 'Regular Transaction Activity (S_1)' in week 1 to 'Sudden Spike in Inflows (S_2)' in week 2, this transition was recorded.
- *Frequency Count:* The frequency of transitions between all possible state pairs ($s_i \rightarrow s_j$) was aggregated for the entire dataset.
- *Probability Calculation:* Transition probabilities were calculated by normalizing the frequencies for each state. For a given state, the probabilities were computed as the ratio of count of transitions from S_i to S_j to total transitions from S_i
- *Matrix Formation:* The calculated probabilities were ordered into n dimensional square matrix, where n corresponds to the number of states. This transition matrix served as the basis for analyzing sequential behavior.

Detection Mechanism We calculated the vector, where the probabilities stabilize regardless of the initial state. This vector is called a steady-state vector. It represents the long-term behavior of the system and is calculated as $\pi = \pi M$. Here, π is the steady state distribution and M is the TPM. This allows us to identify the dominant states over time. If the steady state vector shows a high probability of remaining in the normal state (state S_1) compared to a predefined threshold, the account is considered genuine. However, if the vector indicates a high likelihood than the predefined threshold of staying in other states (e.g. Sudden Inflows or Sudden Outflows etc.), the account is flagged as potentially suspicious. This approach effectively identifies mule accounts by detecting deviations from expected behavior patterns. Mule detection process using markov chain model is illustrated in Fig. 2

3.2 Logistic Regression

Logistic Regression [27] is a fundamental model used for binary classification tasks. It models the log-odds of the dependent variable as a linear function of the input features and outputs a probability score through the logistic sigmoid function. In our case, this model was used to classify accounts as either mule or non-mule based on transactional behavior. Z-score normalization was applied to bring all features to a common scale. To address class imbalance, inverse class frequency weighting was incorporated into the loss function. The model was trained using the `liblinear` solver with L2 regularization, and hyperparameter tuning was performed via grid search over the regularization strength parameter $C \in \{0.01, 0.1, 1, 10\}$. Five-fold cross-validation was used during tuning to select the optimal configuration.

Despite its linearity assumption, Logistic Regression was selected as baseline model for three reasons: Interpretability, high training speed, serve as benchmark model to evaluate the added value of temporal models like markov Chain.

3.3 Artificial Neural Network(ANN)

To capture non-linear relationships in account behavior, we implemented a feed-forward neural network [28]. The network consisted of an input layer matching

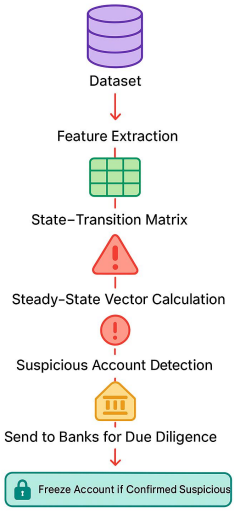

Fig. 2: Markov Chain Model pipeline for mule detection

the number of standardized features, followed by two hidden layers with 64 and 32 neurons, respectively. The ReLU activation function was used in the hidden layers to introduce nonlinearity, while the output layer used a sigmoid activation function to predict class probabilities. Binary cross-entropy was employed as the loss function, optimized using the Adam optimizer. The network was trained for 50 epochs with a batch size of 128. Early stopping was applied to avoid overfitting, and dropout regularization with a rate of 0.2 was inserted between hidden layers. This model architecture was designed to be computationally efficient while maintaining sufficient depth to capture complex patterns indicative of mule activity.

3.4 Extreme Gradient Boosting (XGBoost)

XGBoost [29] is an ensemble learning algorithm based on decision tree boosting. It incrementally builds an additive model by optimizing a regularized loss function, making it well-suited for structured data with complex feature interactions. We used the `binary:logistic` objective function to perform binary classification. The model was trained using 100 estimators, with each tree having a maximum depth of 5. A learning rate of 0.1 was used to control step size, while subsampling and column sampling rates were both set to 0.8 to reduce overfitting. To tackle with unbalanced classes scale_pos_weight was used. Unlike models that require normalization, XGBoost internally handles raw numeric features and missing values, simplifying preprocessing. Feature indicators derived from categorical attributes were one-hot encoded. Hyperparameter tuning

was conducted via randomized search, and five-fold cross-validation was used to validate performance during training.

3.5 Random Forest

Random Forest [30] is a bagging-based ensemble method that constructs a multitude of decision trees during training and outputs the class prediction based on majority voting across all trees. It is robust to overfitting, especially when applied to high-dimensional or noisy datasets. Our implementation included 100 trees, each trained on a bootstrap sample of the training data. At each split, a random subset of features was selected to introduce diversity, with the number of features per split set to the square root of the total number of features. Gini impurity was used as the splitting criterion. Although tree-based models are generally insensitive to feature scaling, we applied Z-score normalization to maintain consistency across experiments. Random Forest is capable of capturing complex decision boundaries and provides useful insights into feature importance based on mean decrease in impurity, which can assist in understanding contributing behavioral signals in mule detection.

4 Results and Analysis

This section presents the performance evaluation of our proposed Markov Chain-based framework for money mule detection in comparison with existing machine learning models. Given the operational requirements of NPCI and the regulatory sensitivity involved in money laundering investigations, we focus our evaluation on two primary aspects: *precision* and *computational efficiency*.

Once an account is flagged as a potential mule account by NPCI's detection system, an alert is generated and forwarded to the respective issuing bank for further investigation. The bank is responsible for conducting a detailed review using its internal policies, customer due diligence records, and transaction analysis. If the bank confirms the involvement of the account in mule activity, it initiates appropriate remedial actions, which may include freezing the account and reporting the case to regulatory authorities. This escalation process highlights the importance of maintaining high precision in NPCI's detection,thus minimizing disruption to legitimate users and reducing the burden on partner banks.

Table 3 summarizes the precision values for each of the five models. The Markov Chain model achieved the highest precision at 0.67, followed by XGBoost at 0.51 and ANN at 0.42. Logistic Regression and Random Forest achieved lower precision values at 0.33 and 0.39 respectively. The superior precision of the Markov Chain model can be attributed to its explicit modeling of sequential behavioral transitions, which allowed it to identify accounts that deviate from normal activity patterns over time. While traditional classifiers like Logistic Regression evaluate feature snapshots in isolation, the Markov model captures dynamic patterns such as the evolution from dormant to active inflow states, a

signature trait of mule behavior. In addition to detection performance, computational efficiency was a key considering the UPI scale. Table 3 presents a comparison of the average training time and inference time across models. The Markov Chain model exhibited the lowest computational footprint among all methods. Unlike traditional machine learning models, it does not involve an explicit training phase with parameter optimization. As a first-order probabilistic model, it relies solely on the computation of state transition probabilities and the steady-state distribution derived from observed behavioral sequences. This lightweight nature allows the model to perform inference in 28 minutes, making it highly suitable for operational deployment in batch settings. In contrast, models like XGBoost and ANN required significantly more training time—52 minutes and 57 minutes respectively—due to their iterative and parameter-rich architectures. The Logistic Regression model provided a good balance but still lagged behind the Markov in terms of overall resource demand. The minimal runtime of the Markov Chain approach makes it highly suitable for operational integration, especially in systems like UPI where rapid evaluation of accounts is critical. Fig. 3 illustrates a comparative analysis of weekly behavioral state

Table 3: Computational efficiency of models

Model	**Precision**	**Training Time**	**Inference Time**
Logistic Regression	0.33	31 minutes	32 minutes
Artificial Neural Network(ANN)	0.42	57 minutes	42 minutes
XGBoost	0.51	52 minutes	49 minutes
Random Forest	0.39	39 minutes	38 minutes
Markov Chain (Proposed)	0.67	–	**28 minutes**

transitions for a representative mule account and a genuine account. The mule account exhibits a highly dynamic and atypical behavior, with multiple abrupt transitions between 'Dormant', 'High Inflow', 'High Outflow' and 'High Velocity' states. These sharp fluctuations are indicative of mule activity—where accounts. In contrast, the genuine account demonstrates a relatively stable behavioral pattern, primarily remaining in the "Normal" state with occasional, low-risk deviations into "Dormant" or "High Inflow" states due to benign usage variations. This visualization reinforces the strength of the Markov Chain model in capturing temporal deviations in account behavior, enabling it to distinguish between suspicious and legitimate transactional profiles effectively. We also examined the steady-state distributions for both a suspicious and a regular account (Fig. 4). The suspicious account exhibited high probability mass in "High Inflow", "High Outflow", and "High Velocity" states, while the normal account remained stable in the "Regular" state. This further highlights the interpretability of the Markov model, which produces actionable signals that are explainable to both fraud analysts and regulators.

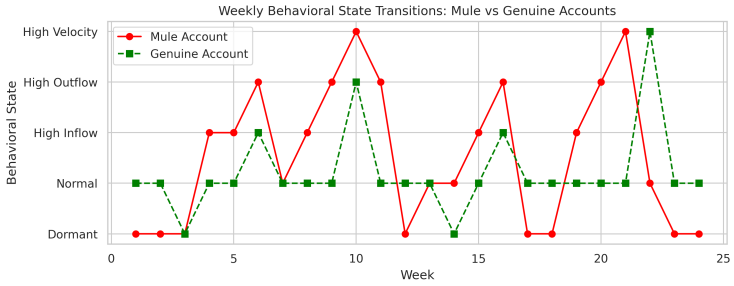

Fig. 3: Weekly behavioral state transitions for a flagged mule account over 24 weeks

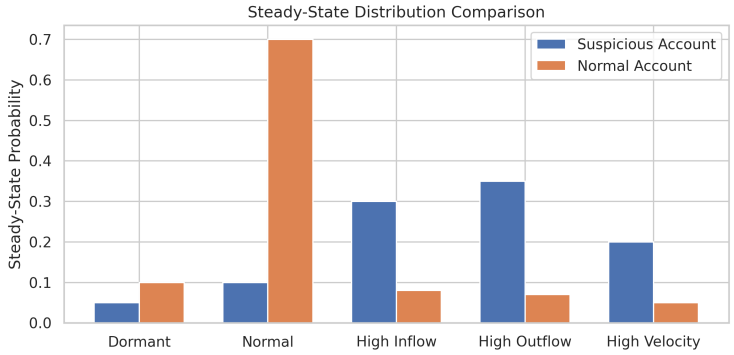

Fig. 4: Steady-state vector comparison for a suspicious and a normal account

5 Conclusion

In this work, we presented a probabilistic approach for detecting money mule accounts using a first-order Markov Chain model, tailored to the behavioral dynamics observed in UPI-based transactions. Unlike traditional machine learning classifiers, our method emphasizes temporal transitions and steady-state behaviors to identify anomalous patterns characteristic of mule operations. The model demonstrated high precision and computational efficiency, making it highly suitable for operational deployment in batch settings. Comparisons with baseline models such as Logistic Regression, ANN, XGBoost, and Random Forest validated the effectiveness of our approach, especially in minimizing false positives—an operational priority for financial institutions. The interpretability of the Markov Chain framework, supported by empirical transition analysis and steady-state vectors, further improves its utility in detection usecases. Future work will explore higher-order Markov models, integration with graph-based transaction networks, and adaptive state modeling to further enhance detection capabilities.

References

[1] United Nations Office on Drugs and Crime. *Overview - Money Laundering*. 2025. URL: https://www.unodc.org/unodc/en/money-laundering/overview.html.

[2] John Madinger. *Money Laundering: A Guide for Criminal Investigators*. CRC Press, 2011.

[3] John McDowell and Gary Novis. "The consequences of money laundering and financial crime". In: *Economic Perspectives* 6.2 (2001), pp. 6–10.

[4] Brigitte Unger and Daan Van der Linde, eds. *Research Handbook on Money Laundering*. Edward Elgar Publishing, 2013.

[5] Brent L. Bartlett. *The Negative Effects of Money Laundering on Economic Development*. Tech. rep. Asian Development Bank Regional Technical Assistance Project No 5967, 2002.

[6] Caribbean Financial Action Task Force. *Role of Money Mules in Money Laundering*. https://www.cfatf-gafic.org/documents/research-corner/22220-role_of_money-mules_ml_november23-pdf/file. Accessed: 2025-04-11. 2023.

[7] Jason Wittenbach, Brian d'Alessandro, and C. Bayan Bruss. "Machine Learning for Temporal Data in Finance: Challenges and Opportunities". In: *arXiv preprint arXiv:2009.05636* (2020). URL: https://arxiv.org/abs/2009.05636.

[8] Ravi Lingarkar. "Why Traditional ML Fails in the Presence of Temporal Relationships". In: *LinkedIn Pulse* (2023). URL: https://www.linkedin.com/pulse/why-traditional-ml-fails-presence-temporal-case-study-ravi-lingarkar-kl8pc.

[9] James R. Norris. *Markov Chains*. Vol. 2. Cambridge University Press, 1998.

[10] Kai Lai Chung. *Markov Chains*. New York: Springer-Verlag, 1967.

[11] George PH Styan and Harry Smith Jr. "Markov chains applied to marketing". In: *Journal of Marketing Research* 1.1 (1964), pp. 50–55.

[12] Codruţa Dura. "The use of Markov chains in marketing forecasting". In: *Scientific Bulletin of the Politehnica University* 6 (2006), pp. 69–76.

[13] V. Bhusari and S. Patil. "Application of hidden markov model in credit card fraud detection". In: *International Journal of Distributed and Parallel Systems* 2.6 (2011), p. 203.

[14] Olayinka Ogundile et al. "Credit card fraud: analysis of feature extraction techniques for ensemble hidden markov model prediction approach". In: *Applied Sciences* 14.16 (2024), p. 7389.

[15] S. Khosravi et al. "GHM: an ensemble approach to fraud detection with a graph-based HMM method". In: *2024 10th International Conference on Web Research (ICWR)*. Tehran, Iran: IEEE, 2024, pp. 99–104. DOI: 10.1109/ICWR61162.2024.10533348.

[16] Henrique S Assumpção, André RS da Silva, and Alan L Oliveira. "DELATOR: Money Laundering Detection via Multi-Task Learning on Large Transaction Graphs". In: *arXiv preprint arXiv:2205.10293* (2022). URL: https://arxiv.org/abs/2205.10293.

[17] Ruofan Wu et al. "GRANDE: A Neural Model over Directed Multigraphs with Application to Anti-Money Laundering". In: *arXiv preprint arXiv:2302.02101* (2023). URL: https://arxiv.org/abs/2302.02101.

[18] Mark Weber. *Graph Deep Learning for Anti-Money Laundering*. https://www.markrweber.com/graph-deep-learning. Accessed: 2025-04-11. 2023.

[19] Bruno Deprez, Flavio Chierichetti, and Mark Weber. "Advances in Continual Graph Learning for Anti-Money Laundering Systems: A Comprehensive Review". In: *arXiv preprint arXiv:2503.24259* (2025). URL: https://arxiv.org/abs/2503.24259.

[20] Antonio Longa et al. "Graph Neural Networks for Temporal Graphs: State of the Art, Open Challenges, and Opportunities". In: *arXiv preprint arXiv:2302.01018* (2023). URL: https://arxiv.org/abs/2302.01018.

[21] Salv. "Why Machine Learning Won't Save AML". In: *Salv Blog* (2024). Accessed: 2025-04-11. URL: https://salv.com/blog/anti-money-laundering-machine-learning-artificial-intelligence.

[22] Oracle. "Anti–Money Laundering AI Explained". In: *Oracle Financial Services* (2024). Accessed: 2025-04-11. URL: https://www.oracle.com/financial-services/aml-ai/.

[23] BytePlus. "Challenges with Graph Neural Networks (GNN)". In: *BytePlus Blog* (2025). Accessed: 2025-04-11. URL: https://www.byteplus.com/en/topic/400475.

[24] Rasmus Ingemann Tuffveson Jensen and Alexandros Iosifidis. "Fighting Money Laundering With Statistics and Machine Learning". In: *IEEE Access* 11 (2023), pp. 8889–8903. DOI: 10.1109/ACCESS.2023.3239549.

[25] David Savage et al. *Detection of money laundering groups using supervised learning in networks*. 2016. arXiv: 1608.00708 [cs.SI]. URL: https://arxiv.org/abs/1608.00708.

[26] Jiani Fan et al. *Deep Learning Approaches for Anti-Money Laundering on Mobile Transactions: Review, Framework, and Directions*. 2025. arXiv: 2503.10058 [cs.LG]. URL: https://arxiv.org/abs/2503.10058.

[27] David W. Hosmer, Stanley Lemeshow, and Rodney X. Sturdivant. *Applied Logistic Regression*. 3rd ed. Wiley, 2013. ISBN: 9780470582473.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. https://www.deeplearningbook.org. MIT Press, 2016. ISBN: 9780262035613.

[29] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[30] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.