Midterm Progress Report
Building an Anti-Money Laundering System using Graph Neural Networks

Sophie Wagner (sw3767), Abhitay Shinde (aps2249), Colin Payne-Rogers (cmp2250), Ernest Salim (es4281), Jingyi Zou (jz3867), Tanya Lim (tjl2150)

March 16, 2025

Abstract

This report outlines our midterm progress in developing an Anti-Money Laundering (AML) detection system using Graph Neural Network (GNN) technology. While traditional AML systems focus on flagging suspicious activity through a rules-based approach with fixed thresholds, our approach leverages a GNN's ability to capture complex connections within financial networks to detect suspicious activities at the transaction level. Given the limited literature on this approach, we have explored multiple strategies to optimize model performance, including data preprocessing techniques and feature engineering methods. To benchmark performance and gain insight into the data, we have applied state-of-the-art supervised learning models as points of comparison. Our primary challenges and next steps are identifying the most effective GNN model architecture and accessing GPU compute resources to experiment and train models with.

Introduction

The Anti-Money Laundering (AML) capstone project is provided by TD Bank, a Canadian multinational bank. The project's objective is to use Graph Neural Networks (GNNs) to detect fraudulent transactions in large financial networks.

Money laundering is a pervasive global issue that threatens the integrity of financial systems and economic stability. The United Nations estimates that up to $2 trillion USD, or 5% of global gross domestic product (GDP), is laundered every year (United Nations, 2022). Criminal organizations exploit financial networks to obscure the origins of illicit funds, making it difficult for authorities to track and intercept illegal activities. For the period from 2010 to 2014, Europol found that only 1.1%

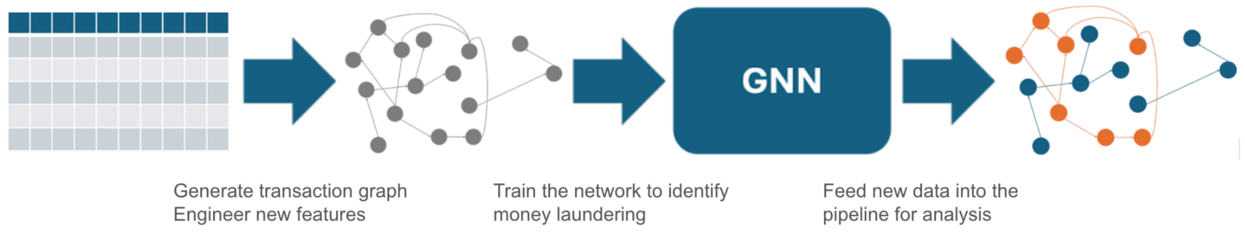
of money laundered in the European Union was identified and confiscated (Europol, 2016). The rapid expansion of digital financial technologies has further complicated detection efforts, which necessitates more sophisticated solutions.

Traditional AML detection methods employed in the industry rely on rule-based approaches by setting fixed thresholds (e.g. amount sent) to identify fraudulent transactions. While these methods capture some fraudulent transactions, they fail to detect different, dynamic, and complex patterns that are used to launder money. Rules-based approaches are not only suboptimal for detection, but are time-consuming and resource intensive. As a result, many institutions have explored machine learning techniques to facilitate detection with high accuracy and flexibility to new money laundering patterns.

Previous data-driven approaches to detect fraudulent transactions include more traditional machine learning (ML) algorithms such as decision trees and logistic regression. While these approaches have shown some success, they often generate high false-positive rates, flagging many transactions as suspicious and burdening investigators with excessive alerts. Further, these approaches treat transactions as individual events, overlooking the relational structure inherent in financial data. This limitation is particularly problematic in detecting money laundering, as illicit financial activities rarely consist of a single transaction. Rather, they often involve complex and multi-layered schemes, moving funds through multiple accounts in order to obscure their origin and evade detection. Effective fraud detection requires models that can capture these dynamic patterns and context, rather than focusing solely on individual transactions.

GNNs address this limitation by capturing relational information within the network, which is vital in identifying complex transaction patterns indicative of money laundering. It can also capture relational dependencies, identify higher-order patterns that are indicative of suspicious behaviour, and improve detection accuracy while reducing false positives by leveraging network-based context.

Concept Slide



Data

Background and Motivation

A ubiquitous challenge in AML research is access to representative financial data. Due to privacy concerns and competitive considerations, financial institutions keep user data to themselves. A 2020 paper by Berg et al. somewhat cheekily shows that weather forecasting apps collecting user location data can better assess user creditworthiness than traditional credit bureaus (Berg et al., 2020). Further, since most money laundering goes undetected, most money laundering transactions in real-world data are unlabeled or incorrectly labeled. These challenges have led to the development of synthetic financial transaction datasets, which allow the application of AML techniques on fully labeled data without the privacy concerns of real-world data.

This work utilizes the AMLworld datasets provided by IBM and described by Altman et al. (2023). These datasets are created through an agent-based generator, where in a virtual world individuals and companies carry out various financial transactions, some of which involve illicit funds (are money laundering transactions). Offering an improvement over previous synthetic AML datasets, they incorporate all three stages of money laundering: placement, layering, and integration. They model key money-laundering patterns described in the literature, as well as additional patterns. Multiple banks, multiple currencies, and multiple payment types are included in the data. The largest available generation includes over 180 million transactions.

Synthetic Dataset Structure

AMLworld consists of six datasets organized into two groups: HI (high illicit) and LI (low illicit). Each group comes in three sizes: Small, Medium, and Large. This enables dataset users to choose an appropriate dataset based on application or available computing resources. Alongside each dataset is a text file outlining specific laundering patterns for select transactions, helping illustrate how money is laundered into the virtual financial system. Each dataset falls within a relatively short date range, e.g. September 1st to 10th, although some data extends past the defined range and all of these transactions are illicit. An overview of each of the six datasets is included in the **Data Structures** appendix.

Available Features

Each of the six datasets includes the following features for each transaction: timestamp, to and from account, to and from bank, to and from amount paid, to and from currency, and payment type. The account, bank, currencies, and "is laundering" columns are all categorical variables, although bank and account numbers are unique identifiers that aren't included in the data analysis or model training, except to uniquely identify nodes in the graph. The "is laundering" column consists of 1s and 0s, with 1 representing an illicit transaction (money laundering) and 0 representing a licit transaction. The amounts sent and received features are the only numeric columns. Every column, other than the unique identifiers, become edge attributes in the transaction graph.

Data Processing and Feature Engineering

To provide a consistent feature set for each ML model developed in the analysis, a model development pipeline was constructed. At the time of this writing it consists of the following steps:

1. Column renaming
2. Duplicate data dropping
3. Unique ID creation
4. Currency Normalization
5. Time feature extraction
6. Cyclical encoding
7. Weekend encoding
8. Neighborhood context generation
9. Normalization

Each represents a transformation to the input data that helps with model development or adds features to enhance model performance. Refer to the **Model Pipeline: Preprocessing** appendix for more information on the motivation for or computation done at each step. The resultant features, including base features, are described in the following table. Note that features may never be included in model training, e.g. they may only be intermediate to the feature engineering, or they may be pruned during model optimization.

Table: Final Feature Set

Feature	Description	Node?	Edge?
From Bank	The transaction source bank	✓	
From Account	The transaction source account	✓	
Sent Amount	The float amount of money sent		✓
Sent Currency	The currency money was sent in		✓
To Bank	The transaction destination bank	✓	
To Account	The transaction destination account	✓	
Received Amount	The float amount of money received		✓
Received Currency	The currency money was received in		✓
Payment Type	The payment format for the transaction, e.g. ACH, Cheque, etc.		✓
Is Laundering	The transaction label, whether the transaction is money laundering		✓

From Unique	A unique ID for the source entity (account numbers are not unique between banks)	✓	
To Unique	A unique ID for the destination entity	✓	
Sent in USD	Sent amount converted to USD, if applicable		✓
Received in USD	Received amount converted to USD, if applicable		✓
Hour of Day	Categorical hour of day, e.g. 0, 1, 2, ...		✓
Day of Week	Categorical day of week, e.g. 0, 1, 2, ...		✓
Seconds Since Midnight	Categorical second of the day, represented as # of seconds since midnight		✓
Sine-encoded Day	Cyclical encoding of the day of week using sine		✓
Cosine-encoded Day	Cyclical encoding of the day of week using cosine		✓
Sine-encoded Time	Cyclical encoding of the time of day using sine		✓
Cosine-encoded Time	Cyclical encoding of the time of day using cosine		✓
Is Weekend	Binary categorical variable for whether the day of week is the weekend		✓
Degree Centrality	Financial entity importance in the graph, as measured by counting the number of connections or the degree of that entity as a node in the transaction graph	✓	
Page Rank	Financial entity importance in the directed transaction graph, as measured by an adjusted centrality, taking into account incoming and outgoing links separately, as well as the quality (centrality) of connecting nodes	✓	

Methods

Our objective is to utilize transaction attributes and network structure to classify individual transactions as licit or illicit. We will use a non-graph ML model as a baseline comparator to assess performance of graph-based approaches. To achieve this, we will develop the following two models in parallel:

1. Graph Neural Network (GINEConv): Primary model which will use graph architectures and incorporate edge features to capture relational structures and detect illicit transactions.
2. Non-Graph Machine Learning Model (CatBoost): Categorical Boosting (CatBoost) is an extension of Extreme Gradient Boosting (XGBoost) that handles imbalance data effectively. CatBoost employs ordered boosting and regularization techniques to minimize overfitting and increase inference speed.

Graph Construction

To use tabular transaction data for network analysis, we define the *transaction graph* as a collection of nodes and edges:

- Nodes**:** Each node represents an account (individual, merchant, or institution).
- Edges: Each directed edge ($u \rightarrow v$) represents a transaction sent from account u to account v.
- Node features: Attributes associated with each account (bank, PageRank, degree centrality).
- Edge features: Attributes associated with each transaction, including payment type, currency, temporal features, and transaction amount.

For the CatBoost model, we exclude account and bank identifiers, as these are only needed for constructing the transaction graph in the GNN. Including them introduces noise and risks data leakage, where the model may memorize accounts or banks rather than learning to identify laundering based on transaction features and their relationships.

Train-Test Splits and Validation

To ensure fair model evaluation and improve generalizability, we implement two different train-test splitting strategies and compare their effectiveness:

1. Random stratified split (baseline approach): Transactions are randomly split into training and testing sets while preserving licit-to-illicit transaction ratios. This ensures class balance, which is particularly important due to the highly imbalanced nature of AML datasets.

2. Temporal split (more realistic, real-world approach): Transactions are sorted by timestamp before splitting such that older transactions are used for training and newer transactions are used for testing. We split the data at the 70th percentile (t_1) and 85th percentile (t_2) timestamps. The training set includes all transactions to timepoint t_1. The validation set includes all transactions up to timepoint t_2, but performance is only evaluated on transactions between t_1 and t_2. The test set includes all transactions, but performance is only evaluated on transactions following timestamp t_2. While this approach introduces a risk of data leakage, it better reflects real-world AML detection where models process transactions in batches and rely on historical data for decision-making.

We use a 70/15/15 split for training, validation, and testing. Both XGboost and GNN models use the same splits, ensuring that comparisons of model performance are meaningful and consistent.

Model Architecture and Training

Graph Neural Network (GNN) Component

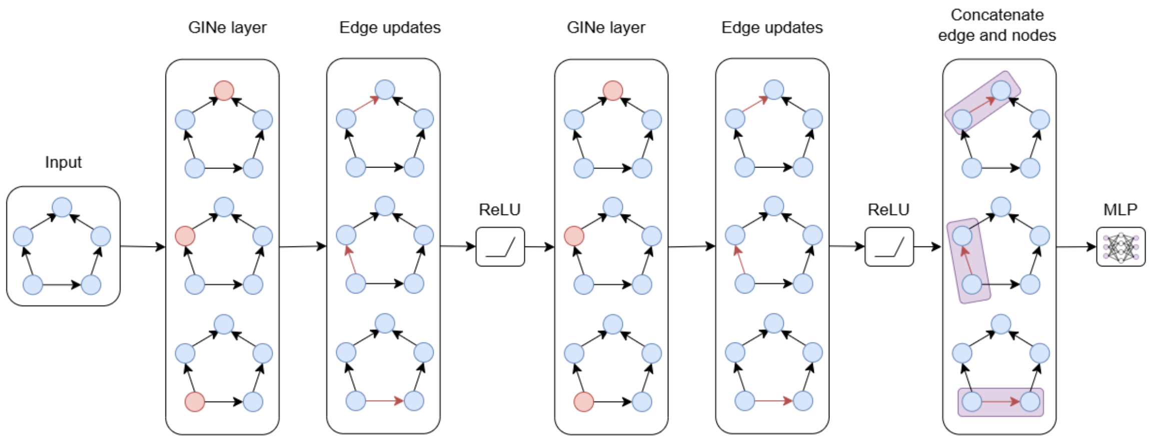
In traditional GNNs, message passing occurs at the node level, where edges primarily serve as connectivity structures or are used as weights in an attention mechanism. This is problematic in our task because the transaction (edge) itself carries the most important information – thus, standard node-based aggregation is not sufficient. Further, most GNNs perform classification of node embeddings, and often don't generate edge embeddings at all.

To address these limitations, we utilize a lesser-known architecture, GINe (Graph Isomorphism Network with Edge Features, implemented with GINEConv), which explicitly incorporates edge attributes into message passing updates rather than treating them as secondary weights (for further information, refer to **Message Passing in GINe** in the appendix). Additionally, we modify the structure to create edge embeddings, allow embeddings to be dynamically updated, and construct meaningful and relevant output that relates to an individual transaction for classification (see the following table for model components, and subsequent figure for the simplified model schematic). We expect GINe with edge updates to capture complex transaction patterns across multiple layers of accounts. This will be the primary model architecture we use moving forward.

Table: Graph Neural Network Architecture Components

Component	**Description**
Input	Transaction graph with edge attributes and limited node features.
Initial embeddings	Edge and node features are each passed through a fully connected linear layer, resulting in embeddings of size hidden_dim (default=64).
Layers	Two GINe layers with multi-layer perceptron (MLP)-based aggregation update node embeddings.
Edge updates	Edges are updated through concatenation of edge and node embeddings, followed by a nonlinearity function.
Output	Each edge embedding is concatenated with its source (paying account) and destination (receiving account) node embeddings, then passed through a final MLP to output logits for transaction classification.
Loss function	Binary Cross Entropy Loss (BCE Loss) is used to compare logits with ground truth labels.
Backpropagation	Loss is backpropagated to update weights for all layers, including initial embeddings, GINE layers, edge MLPs, and final edge-node concatenation MLP.
Hyperparameters	Batch size = 1 (entire graph processed as one structure; memory constraints require optimization). Optimizer = Adam. Learning rate = 0.01.

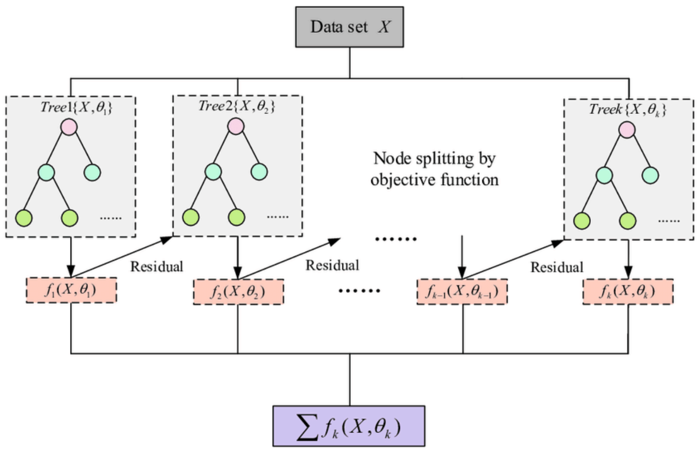
Figure. Graph neural network model schematic.



Traditional machine learning (CatBoost) component

CatBoost serves as our baseline reference model for comparison of GNN performance. We optimized our hyperparameters to balance overfitting and improve generalizability by maximizing PR-AUC. Data balance is addressed by setting class weights (e.g. {0: 1, 1: 12}) to ensure the minority class (illicit transactions) receives sufficient emphasis during training. The loss function used is "Logloss", which penalizes misclassifications logarithmically and provides a smooth gradient for optimization. We set maximum tree depth to 8 to minimize overfitting. We used a learning rate of 0.05. Overall, the model is designed to extract feature-based insights from the engineered tabular data without solely relying on the graph topology, thereby providing a robust baseline. An overview figure of CatBoost, taken from (Emami et al., 2023), is provided below.

Figure: Catboost Model Schematic



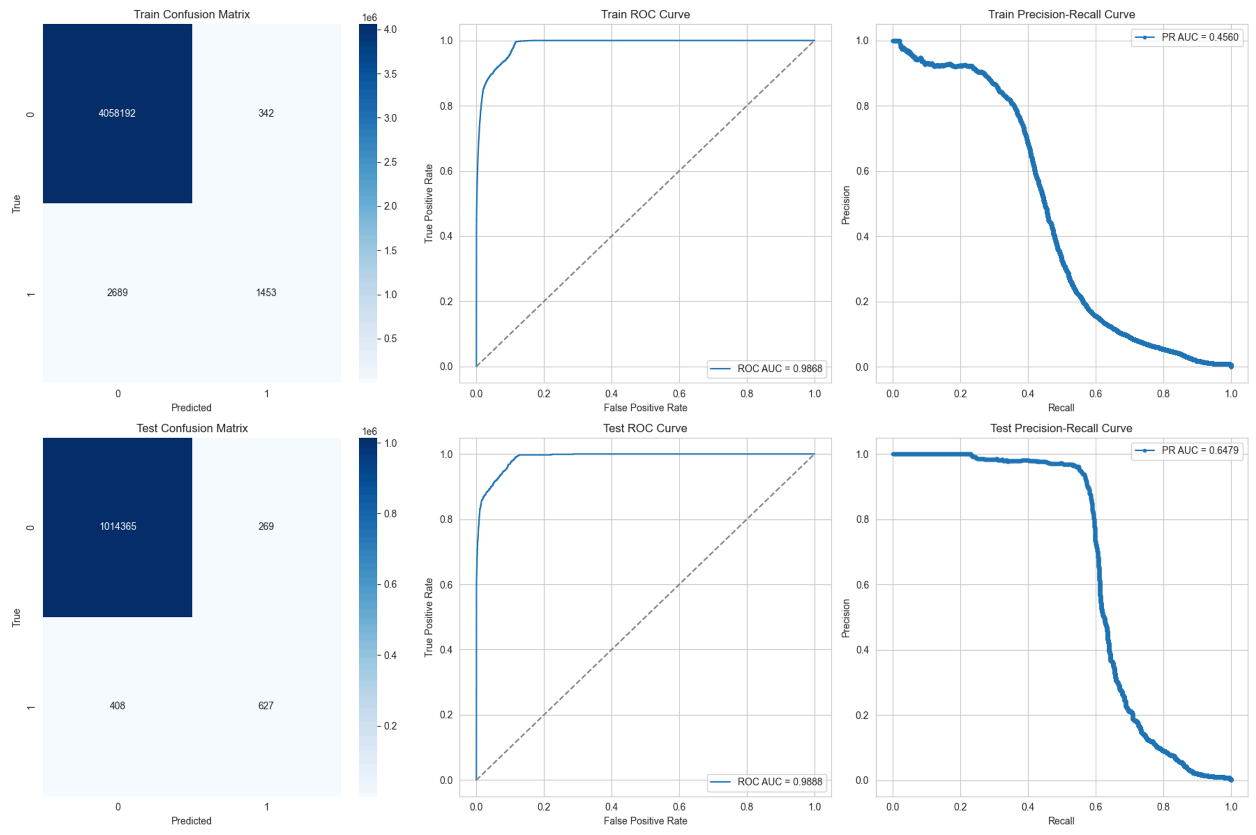
Evaluation metrics

This work uses Precision-Recall Area Under the Curve (PR-AUC) as the primary metric for evaluating model performance. PR-AUC measures the trade-off between precision and recall and thus focuses on the minority class (fraud), making it well-suited for highly imbalanced data. Secondary metrics include ROC-AUC, F1 score, precision, and recall, along with MLDR (Money Laundering Detection Rate) – a custom AML-specific classification metric used by Xu (2024) that evaluates detection within the top 1% of computed probabilities.

Results

At this stage in the term, we have preliminary results for the CatBoost model performance in detecting illicit transactions, though baseline results for the graph-based model are still pending. We initially trained Catboost on all available predictors and then carried out backward selection, finally achieving a PR-AUC of 0.6597 and ROC-AUC of 0.9888. While the ROC-AUC is high, this may misrepresent model performance as it is influenced by the overwhelming majority of licit transactions. On the other hand, the moderate PR-AUC suggests room for improvement in handling the minority (fraud) class. A more detailed performance breakdown is provided in the following figure.

Figure: CatBoost Performance Curves



Feature Importance

Feature importance was analyzed using Shapley values, which provide a more reliable interpretation than weight-based or gain-based methods. Rather than looking at frequency or average contribution, Shapley values quantify the direct impact of a feature on each prediction relative to a baseline, allowing for more granular and robust identification of feature importance.

The Shapley bar plot suggests that Payment Format is on average the most influential feature (Appendix **Figure: Shapley's Feature Importance**). Graph-based features (degree centrality and PageRank) also rank highly, emphasizing the significance of network connectivity in detecting suspicious activity. This may further support adoption of a graph-based model. Other important features include transaction time (cyclically encoded) and transaction amount (amount sent/received in USD). Temporal features (day of week, hour of day, and is_weekend) have a smaller impact on prediction.

While we have some preliminary insights from CatBoost, we plan to expand feature importance analysis to graph-based model moving forward. This will allow us to compare and contrast prioritization of key transaction features in each method.

Challenges

The primary challenge faced when experimenting with GNN models is access to compute resources. The research team behind the AMLworld synthetic dataset used in this work provided estimates for training its GNN models with an Nvidia Tesla V100 GPU: 7-10 hours for Small datasets, 24 hours for Large datasets, and around 1,000 total GPU hours for all experimentation and training (Altman et al., 2023). The GPU architecture provided by Google Colab uses Nvidia Tesla T4 GPUs (Google, n.d.). Dell suggests that V100 GPUs outperform T4 GPUs by a factor of 2-4x (Dell Technologies, 2019). This means that for us to train our GNN models on Google Colab, we can expect a single training run on a small dataset to take one to two days. This is a limitation to experimenting with model architectures and at the time of this writing we have not achieved a successful GNN training run.

Planned analysis and expected results

Our future results will include a comparative evaluation of the GNN and XGBoost models on the same datasets, focusing on key performance metrics described above (PR-AUC, recall, MLDR), as well as computational cost.

Discussion

Over the past eight weeks, we have refined our dataset understanding, reviewed existing literature, constructed a model development pipeline, and explored both traditional and graph-based machine learning approaches for money laundering detection. We conducted exploratory data analysis (EDA), addressing challenges like class imbalance and engineered features to enhance model performance. Stakeholder feedback emphasized model interpretability, data drift, and real-world application, which has guided our approach.

Key Takeaways

1. GNNs capture relational structures. GNNs outperformed traditional ML models (XGBoost, CatBoost) in identifying intricate money laundering patterns. Focusing on transaction-level (edge-level) predictions, rather than account-level (node-level) predictions, aligns better with real-world AML requirements and provides more granular insights.

2. Temporal features are essential. Incorporating time differences between transactions and cyclical encoding improves model performance.
3. Class imbalance is a major challenge. Underrepresentation of illicit transactions in the dataset impacted performance metrics. Techniques like stratified sampling, attention mechanisms, and weighted loss functions helped improve model performance, but further work is needed to fully address this issue.
4. Feature engineering drives performance. Backward selection revealed transaction amounts, currency, temporal patterns, and graph-based features (node centrality, time differences) were most impactful.
5. Real-world considerations matter. Stakeholders emphasized risks like data drift, the need for model interpretability, and prioritizing precision-recall curves over ROC-AUC for imbalance datasets.

Future Work

1. Feature selection. Use insights from feature importance analysis to refine model inputs.
2. Improve model pipeline. Model pipeline must contain sufficient documentation for replicability and scalability, and must be adapted for both GNN and XGBoost models.
3. Class imbalance solutions. Experiment with advanced sampling techniques, focal loss, and metric refinements to improve model performance.
4. GNN architecture. Train and test a baseline GNN, and experiment with varying the GNN architecture and node and edge attributes.
5. Computational requirements. Identify a strategy for experimenting with GNNs over long training runs, or obtain access to more powerful GPU architectures

Ethical Considerations

While our project uses synthetic data, applying the model to real-world transactions introduces ethical considerations. Privacy is a concern and data protection regulations such as GDPR and CCPA should be complied with. Although the models aren't trained on individual characteristics, results against real-world data should be audited to ensure equitable treatment. The balance between false positives (legitimate transactions falsely flagged as suspicious) and false negatives (missed money laundering) must be carefully considered so as not to burden financial institutions or allow criminal activity to go undetected. Model predictions must be interpretable, and an accountability framework should be established to assess and justify outcomes; ensuring that financial institutions can understand and validate model reasoning will be key to responsible deployment in real-world AML systems.

References

Altman, E., Blanuša, J., von Niederhäusern, L., Egressy, B., Anghel, A., & Atasu, K. (2023). Realistic Synthetic Financial Transactions for Anti-Money Laundering Models. *Advances in Neural Information Processing Systems*, *36*, 29851--29874.

Berg, T., Burg, V., Gombović, A., & Puri, M. (2020). On the rise of fintechs: Credit scoring using digital footprints. *The Review of Financial Studies*, *33*(7), 2845--2897. https://academic.oup.com/rfs/article-abstract/33/7/2845/5568311

Dell Technologies. (2019, March). *Deep Learning Performance on T4 GPUs with MLPerf Benchmarks*. Retrieved March 16, 2025, from https://www.dell.com/support/kbdoc/en-us/000132094/deep-learning-performance-on-t4-gpus-with-mlperf-benchmarks

Emami, S., Emami, H., & Parsa, J. (2023). LXGB: a machine learning algorithm for estimating the discharge coefficient of pseudo-cosine labyrinth weir. *Nature: Scientific Reports*, *13*.

Europol. (2016). *Criminal asset recovery in the EU, survey of statistical information in 2010-2014*. Does crime still pay? Retrieved March 16, 2025, from https://www.europol.europa.eu/media-press/newsroom/news/does-crime-still-pay

Google. (n.d.). *GPU Architecture*. Retrieved March 16, 2025, from https://colab.research.google.com/github/d2l-ai/d2l-tvm-colab/blob/master/chapter_gpu_schedules/arch.ipynb

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, 448--456. https://arxiv.org/pdf/1502.03167

Open Source. (2025). *pandas* [open source data analysis and manipulation tool]. Retrieved March 16, 2025, from https://pandas.pydata.org/

United Nations. (2022). *Money Laundering*. Office on Drugs and Crime. Retrieved March 16, 2025, from https://www.unodc.org/unodc/en/money-laundering/overview.html

Xu, H., Yu, K., Wei, M., Zhu, Y., & Liu, Y. (2024). Intelligent Anti-Money Laundering Transaction Pattern Recognition System Based on Graph Neural Networks. *Authorea Preprints*.

Appendix

Data Structure

Table: Dataset Overview

The data in this table are taken from the appendix of (Altman et al., 2023).

	Small		**Medium**		**Large**	
Statistic	**HI**	**LI**	**HI**	**LI**	**HI**	**LI**
# of Days Spanned	10	10	16	16	97	97
# of Bank Accounts	515K	705K	2077K	2028K	2116K	2064K
# of Transactions	5M	7M	32M	31M	180M	176M
# of Laundering Transactions	3.6K	4.0K	35K	16K	223K	100K
Laundering Rate	1/981	1/1942	1/905	1/1948	1/807	1/1750

Table: Feature Overview, Numerical

	Received Amount	**Sent Amount**	**Is Laundering?**
count	6.924e+06	6.924e+06	6.924e+06
mean	6.324e+06	4.676e+06	5.149e-04
std	2.105e+09	1.544e+09	2.268e-02
min	1.000e-06	1.000e-06	0.000e+00
25%	1.742e+02	1.754e+02	0.000e+00
50%	1.398e+03	1.399e+03	0.000e+00
75%	1.230e+04	1.223e+04	0.000e+00
max	3.645e+12	3.645e+12	1.000e+00

Table: Feature Overview, Categorical

	From Account	**To Account**	**Received Currency**	**Sent Currency**	**Payment Type**
count	6924049	6924049	6924049	6924049	6924049
unique	681281	576176	15	15	7
top	10042B660	10042B660	US Dollar	US Dollar	Cheque
freq	222037	1553	2537242	2553887	2503158

Model Pipeline: Preprocessing

The following table describes each step, and the motivation behind it, in our model preprocessing pipeline. The result of running the pipeline is a Pandas (Open Source, 2025) data frame that can be used as the input to a machine learning model.

Table: Model Pipeline Preprocessing Steps

Step	Description
Column renaming	The column names in the original dataset can be unclear. They are renamed to reduce the likelihood of error. A representative example is renaming `Account` and `Account.1` to `from_account` and `to_account`.
Duplicate data dropping	Duplicate rows in the Pandas data frame are dropped
Unique ID creation	The account identifiers in AMLworld datasets are randomly generated and were shown not to be distinct between banks within a given dataset, so a unique account ID is generated for each bank, account pair
Currency Normalization	To be able to aggregate some transaction data at the node level, like total sent or total received, the sent and received amounts need to be normalized to the same currency, in this case to USD
Time feature extraction	Each transaction provides a timestamp, e.g. the date and time of the transaction. This is linearly increasing and unlikely to be a meaningful input feature, so features like day of week and time of day are extracted from it
Cyclical encoding	Encoding time as day of week, hour of day, or seconds since midnight fails to capture the cyclical nature of the data, e.g. that the day of week 6 is as close to the day of week 0 as it is to day of week 5; cyclical encoding helps represent features in a way that preserves this
Weekend encoding	Subjectively, or culturally, we consider weekend days of the week distinct from week days, information that is not necessarily captured by categorically or cyclically encoding the day of the week
Neighborhood context generation	Neighborhood context is a concept in graph theory of information one might ascribe to a node based on its position in the graph, e.g. degree centrality or page rank
Normalization	Features are normalized; gradient descent converges much faster with feature scaling (Ioffe & Szegedy, 2015)

Methods: Message Passing in GINe

GINe explicitly propagates edge attributes in its message passing functions. Unlike traditional GIN, which only learns node embeddings, GINe extends the architecture to update node features using edge embeddings. The message passing function is as follows:

$$h_v^{(l+1)} = MLP((1 + \epsilon)h_v^{(l)} + \sum_{u \in N(v)} MLP(h_u^{(l)} + e_{uv}^{(l)})$$

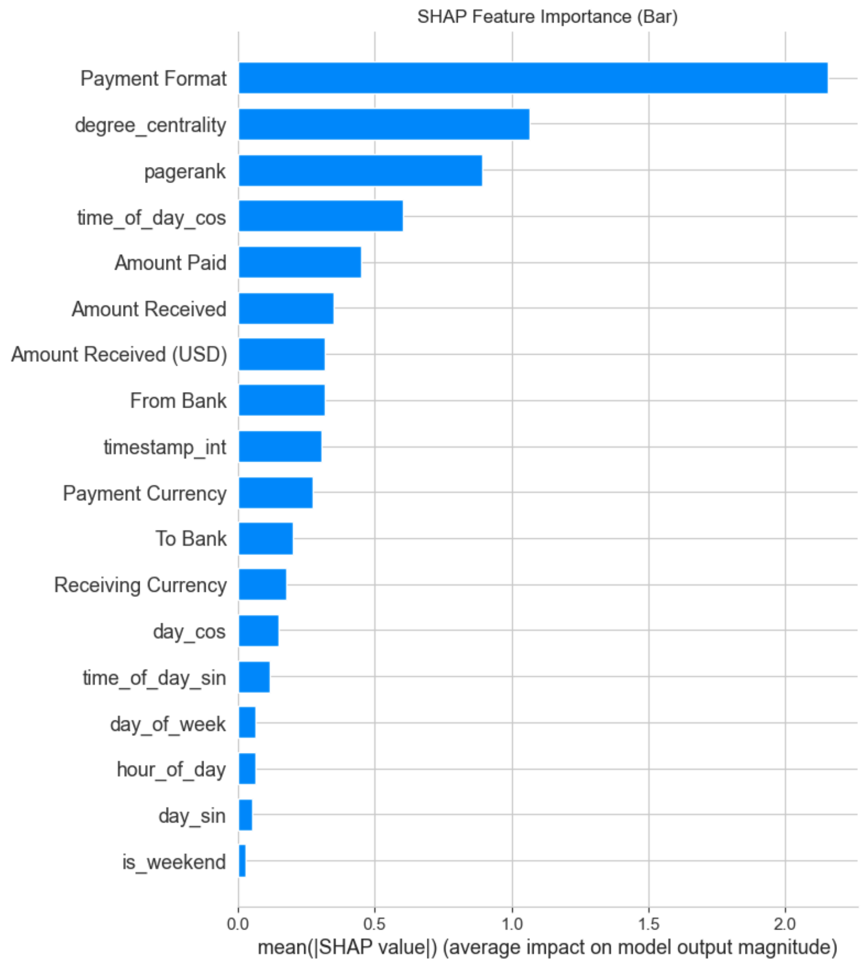
where $h_v^{(l)}$ and $e_{uv}^{(l)}$ represent the node and edge embeddings, respectively, at layer l, and ϵ is a learned parameter that determines retention of the previous node embedding. After one layer (or round of message passing) each node embedding h_v contains an aggregate representation of its immediate transactions. After a second message passing round, each node contains information expanding one transaction removed. While this initial function does not incorporate edge updates, we modify the graph structure such that edges are updated after each layer, comprising a concatenation of the edge embedding with source (paying account) and destination (receiving account) embeddings.

Methods: Further GNN research

For further reference, we had also experimented with several other state-of-the-art GNN architectures such as GCN (Graph Convolutional Network), GAT (Graph Attention Network), and RGCN (Relational Graph Convolutional Network). None of these are suitable in the context of our data and objective as these models don't have the ability to incorporate edge features into the message passing. However, GATe (Graph Attention Network with Edge Embeddings) does support edge embeddings, the attention mechanism can prove itself to be quite of a technical jump which may reduce the interpretability nature of our models; therefore, we are shelving GAT for future use. Finally, even though our review of RGCN was not fruitful, it opened up the possibility of integrating one of the most distinguishable aspects of RGCN: heterogeneous graph structure, to our current implementation as this specific graph could help our model better capture diverse transaction patterns.

Results

Figure: Shapley's Feature Importance



SHAP Feature Importance (Bar)
Payment Format
degree_centrality
pagerank
time_of_day_cos
Amount Paid
Amount Received
Amount Received (USD)
From Bank
timestamp_int
Payment Currency
To Bank
Receiving Currency
day_cos
time_of_day_sin
day_of_week
hour_of_day
day_sin
is_weekend
0.0
0.5
1.0
1.5
2.0
mean(|SHAP value|) (average impact on model output magnitude)