Anti-Money Laundering Detection

Thesis by

Shamsiya Salim

In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Data Analytics with Computational Science

SCHOOL OF DIGITAL SCIENCES
KERALA UNIVERSITY OF DIGITAL SCIENCE, INNOVATION AND TECHNOLOGY
Trivandrum, Kerala

Supervisor: Dr. Sanil P Nair

MSc. Data Analytics with Computational Science
Defended: 02 July 2024

ACKNOWLEDGEMENTS

Let me take a moment to acknowledge those who have supported and advised me during this project.

Firstly, I am very grateful to my guide, Mr. Sanil P Nair, Assistant Professor, School of Digital Sciences, Digital University Kerala, Trivandrum. I could not have embarked on this project or finished it without his help, advice, and support. I must thank him for his help and advice during the work; he has been a great mentor to me.

I pay special thanks to Dr. John Eric Stephen, Head of the School, School of Digital Sciences, Digital University Kerala, Trivandrum. He inevitably played the role of creating a good environment and making resources easily accessible, which ensured the performance of this work.

Also, I would like to express my heartfelt thanks to Prof. Saji Gopinath, Vice-Chancellor, Digital University Kerala, Trivandrum for availing the best infrastructure, facilities, and educational standards. The serene environment offered by the university as well as the central facilitation by the educational institution cannot be overemphasized in the conduct of this task.

I would also like to thank friends and family who have inspired and encouraged me in the time demanded by this work. Words cannot express how much I needed their belief in my capabilities and their encouraging words throughout this process.

Lastly, I have to be grateful to everybody who in one or another way has supported me with the accomplishment of this project. Your help and contributions have been very grateful. Thank you all.

ABSTRACT

AML stands for anti-money laundering, which is one of the essential activities of the financial organizations that functions for detecting and preventing of unlawful operations. This project incorporates data science and deep learning methods along with developing Graph Attention Networks (GAT) to detect suspect money laundering. For implementation, the project uses IBM's transaction dataset uploaded in Kaggle For graph construction, GAT is used to identify relevant features from the data source. These embeddings are passed onto several machine learning models for classifying the transactions as either laundering or not. GAT model is created with architectural design that include two GAT convolutional layers and the final linear layer, trained with stochastic gradient descent. The vectors produced by the GAT model are then fed in and used to train and test a variety of machine learning algorithms such as KNN, Decision Trees, Random Forests, GBDT, AdaBoost, SVC, Logistic Regression, and LDA. It has been established from the results that integrating graph-based neural networks with conventional models does a good job of detecting suspicious financial activities. Testing and validation of the models were done using accuracy, precision, recall, F1-score, and confusion matrix measure. This, therefore, offers a rich strategy to improving the AML systems, especially the role that GATs can play on how to detect financial frauds.

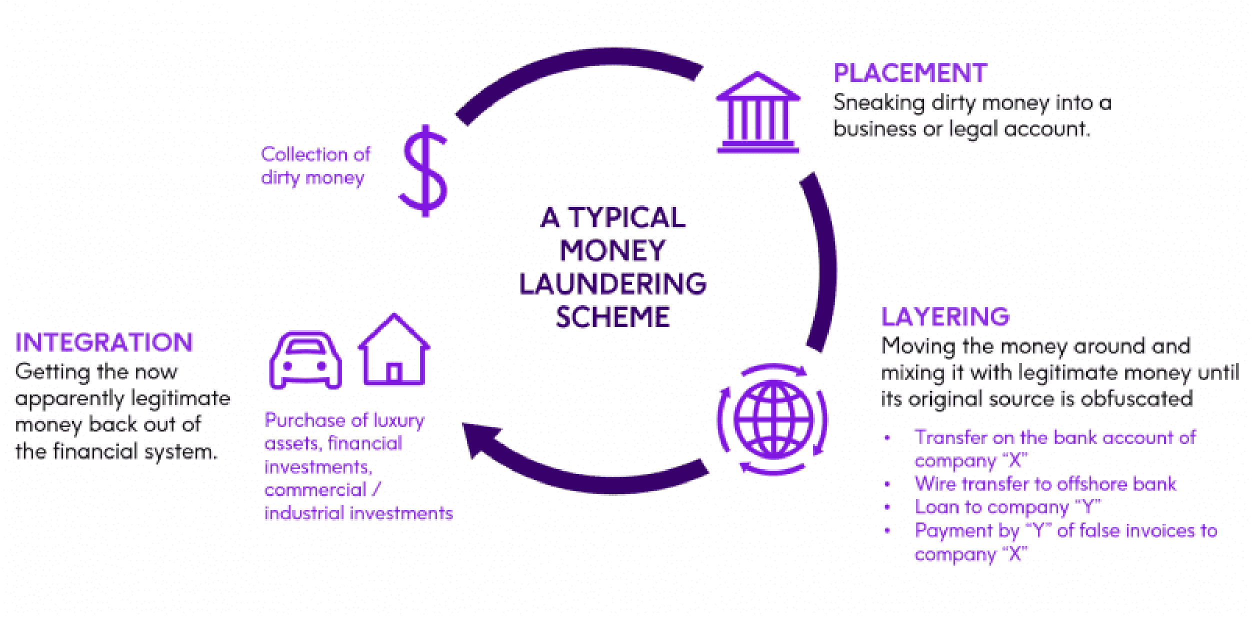
TABLE OF CONTENTS

LIST OF IMAGES

Chapter 1

INTRODUCTION

Money Laundering remains one of the biggest threats to global financial systems and stability. It deals with hiding the source of money that is unlawfully earned, in such a way that looks legit. This illegal activity can then be used to aid a variety of criminal endeavors such as trafficking of drugs, funding terrorism, and fraud. This capability is used by regulatory authorities and financial institutions worldwide to detect and prevent money laundering.


A TYPICAL MONEY LAUNDERING SCHEME
PLACEMENT
Sneaking dirty money into a business or legal account.
Collection of dirty money
LAYERING
Moving the money around and mixing it with legitimate money until its original source is obfuscated
• Transfer on the bank account of company "X"
• Wire transfer to offshore bank
• Loan to company "Y"
• Payment by "Y" of false invoices to company "X"
INTEGRATION
Getting the now apparently legitimate money back out of the financial system.
Purchase of luxury assets, financial investments, commercial / industrial investments

AML diagram

It is known that conventional techniques for identifying the same involve the use of rules and some other approaches based on manual analysis, and therefore, they suffer from some key problems, such as inefficiency and unsuitability for combating new strategies of money laundering. ineffective at handling large volumes of transactional data that is processed daily, thus causing delay and possible omission of critical aspects.

Concerning these challenges, this project aims to propose the improved use of complex Machine Learning algorithms, which focus on Graph Attention Networks (GATs), to improve the identification of suspicious financial transactions. GATs are a form of neural network that can specifically work with graph-based data, which should be appropriate for modeling the financial transactions' features.

Therefore, a primary goal of this project is to create a dependable system for the identification of money laundering activities using GATs node embedding and other proven classification algorithms. The work employs the IBM Anti-Money Laundering dataset on Kaggle containing a set of anonymized transactions.

1.1 Graph Attention Networks (GATs)

Although basic ML models accomplish the tasks, there could be complex relations in transactional data that may not be captured. This is where Graph Attention Networks (GATs) are used. Taking inspiration from graph convolution operation in GNNs, the Graph Attention mechanism is introduced. It stands for Graph Neural Networks – a special sort of neural net constructed for processing graph-based data, which is highly suitable for the relationships between entities in financial networks.

In this project, we utilize Graph Attention Networks to:

- Capture complex relationships: GATs work with the relations between the entities (accounts, transactions) to identify complex money laundering schemes.
- Improve detection accuracy: Therefore, through working directly on the graph and strategically selecting the nodes and edges, GNNs improve the system's performance in identifying suspicious activities.
- Adapt to evolving threats: GATs can learn from new data, thus making the Anti-Money Laundering (AML) system known to the emerging money laundering techniques.

1.2 Machine Learning (ML) Models

ML models have been used in classification and anomaly detection problems and due to these features can be applied to detect suspicious actions in financial operations. In this project, we employ a variety of simple ML models, including:

- K-Nearest Neighbors (KNN): An example of nonparametric techniques applied in classification due to features' similarity.
- Decision Tree: A model that divides input data with the help of the most important feature of the resulting subsets of data at every node.
- Random Forest: An ensemble of decision trees that improves classification accuracy and reduces overfitting.
- Gradient Boosting: A technique that builds an ensemble of weak learners to create a strong predictive model.
- AdaBoost: An adaptive boosting algorithm that combines multiple weak classifiers to form a robust classifier.
- Support Vector Classifier (SVC): A model that finds the optimal hyperplane to separate different classes.
- Logistic Regression: A linear model used for binary classification tasks.
- Linear Discriminant Analysis (LDA): A method used to find the linear combinations of

features that best separate classes.

The procedure of this study entails generating a graph from the transactional information, with the accounts being nodes and the transactions being edges. They are then used to train the GAT model to learn representations of the transactional behavior and the relation among the accounts. These embeddings are then used as feature inputs to multiple conventional machine learning models that are trained for identifying and categorizing as fraudulent or genuine transactions later.

The expected output of this project is developing a very efficient and effective system that can identify money laundering. In this way, using the proposed approach based on GATs and other classical ML models, it is possible to build a scalable and efficient solution for the AML problem and generally, to strengthen the fight against financial crimes and increase the effectiveness of regulation in this area.

Chapter 2

LITERATURE REVIEW

Money laundering is a major global problem that bears threats to the soundness and quality of the systems that underlie economies. This is the act of integrating black money with white in an attempt to cover the source of the money and make it seem clean. A lot of the conventional techniques of money laundering analysis used in the past entail a rule-based system, which is essentially a form of filtering that is fairly slow and often incapable of providing the kind of detection that is required anymore due to more complex attempts at laundering money. These rule-based systems are recognized by their weakness to rely on some given patterns that are rather predictable and can be easily fooled by fairly new techniques of money laundering. This is because the volume and types of financial transactions are growing steadily and can engage a more effective solution.

The Financial Action Task Force (FATF) and other regulatory bodies have acknowledged that these constraints arise from the existing systems and have encouraged the use of improved methods hinged on technological advancements. Earlier systems of AML screening return highly significant percentages of false positives, which can negatively affect the efficiency of work in financial companies and increase their expenditures dramatically. That inefficiency is multiplied by the constant changes on the side of the money launderers in their approach to the process since it has become a constantly shifting game of finding the newest ways in which certain financial systems can be exploited.

Artificial Intelligence (AI) has become one of the significant methodologies in combating money laundering. Due to the large datasets and the complex formulas, the Machine Learning (ML) models can detect patterns and discrepancies that other systems can miss. In recent works, the primary focus is on how the application of ML also improves the identification of emergent unlawful actions. For instance, McKinsey’s article titled "The fight against money laundering: Machine learning is a game changer" [1] regarding AML notes that ML models enhance transaction monitoring since they entail detailed data sets, which signal behaviors to help identify suspicious activities. Algorithms like Artificial Neural Networks, decision trees, and ensemble methods are extensible in learning new patterns from the historical data discerningly and improving the detection ratio of suspicious transactions.

Indeed, amongst generative models, graph-based models such as Graph Neural Networks (GNNs) and Graph Attention Networks (GATs) have proved to be effective in dealing with the inherent structural relations in financial transactions. Such models depict data as graphs in which nodes are such things as bank accounts, and edges are transactions between them. The important information stored in these graphs is relational information that helps in supporting the detection of anomalies.

One of the practical cases of using graph-based models in AML is described in the paper "Anti-Money Laundering Alert Optimization Using Machine Learning with Graphs" [2]. The paper also provides empirical evidence of how graph-based features complement the catalog of ML models applied to AML alerts for the improvement of the identification of suspicious transactions. Graph-based models are especially useful in capturing transactional structures of the parties involved and transactional structures that trigger suspicious activities like transaction loops or accounts with extremely high transaction density.

Previous research for the AML system has been done in the form of a literature review to identify the existing methodologies utilized for the evaluation. One such systematic literature review is presented in the paper "Anti-Money Laundering Systems: A Systematic Literature Review" [3]. This review, in turn, divides the available solutions by the algorithms used, data, methods of evaluation, tools used for implementation, and sampling techniques. This review considers 27 documents that were published in the period 2015 to 2020 to gain thick descriptions of the dynamism and efficiency of several AML systems. It makes sense to assume that such reviews play a crucial role in outlining the weaknesses of the current research and the topic areas that are still left unaddressed, including the application of real-time data processing and the enhancement of accurate, reliable evaluation indicators.

Hence, the challenges experienced in implementing the techniques of ML and graph-based models for AML systems are as follows. These are the data quality and availability, the requirement of high computational power to implement the models, and the challenges of incorporating these models in the already existing financial systems. However, there is another major challenge which is relating to the interpretability of ML models because to pass through regulations, organizations still have to explain how their decisions are made.

The application of Machine learning combined with graph-based structures is the next-generation fight against money laundering. These technologies provide the opportunity to enhance the detection of suspicious activities by modeling relationships and capturing some other features in the transactional data. But as the developments in this area concern, it is crucial to consider the issues that come with such effective AML tools for these global systems to be optimally effective.

Chapter 3

OBJECTIVES

The primary objectives of this project are outlined as follows:

1. Develop a Graph-Based Representation of Financial Transactions:

Build a graph from IBM dataset in which graph vertices correspond to the entities (e. g. bank accounts) and graph edges correspond to transactions. This representation will try to capture the interactions and trends of the financial activities.

2. Implement a Graph Attention Network (GAT):

All the components will be used in a specifically designed GAT model optimized for the AML task. Withing the proposed model, it should be possible to learn high-quality embeddings that capture the complex nature of financial transactions as well as relationship between entities.

3. Generate Node Embeddings for Classification:

Employ the trained GAT model to obtain the embedding that is applied to generate embeddings of nodes in a given graph. These embeddings should compress features of the related transactions and the involved entities.

4. Evaluate Machine Learning Models Using GAT Embeddings:

Take the salient embeddings computed by the GAT model as inputs to many other machine learning models. After developing the above models, assess and compare them in order to determine which model yields the most accurate results concerning the classification of the transactions into either the legitimate or suspicious category.

5. Assess Model Performance:

Assess the results of the machine learning models' performance using various measures such as accuracy score, precision, recall, F1 score, and the confusion matrices. This assessment seeks to analyse and compare the efficiency of the above proposed strategy in checking possible money laundering activities.

6. Enhance Anti-Money Laundering Systems:

Explain how the integration of the existing graph-based neural networks with the conventional machine learning approaches is able to enhance the detection of illicit behaviors in the financial sector. Discuss how these techniques can be incorporated into currently used AML systems and offer recommendations on the integration.

Through achieving these objectives, it is the intention of the project to advance the establishment of better and enhanced AML solutions through the use of graph-based learning systems and machine learning classification.

Chapter 4

MATERIALS AND METHODS

It is important for entities to be able to identify AML activities due to their relevance in integrating the financial systems. This project involves applying advanced Machine Learning to detect fraudulent financial transactions and model the transactions as graphs. We employ a GAT where we map entities to nodes and transactions to edges in an attempt to capture complex relations of the given data. It includes, therefore, the data preprocessing stage, the training targeted for models, and the evaluation stage that uses both DL and traditional ML methodologies to improve the detection rate of the project. This has made adopting this innovative methodology an efficient and scalable solution for tackling AML appeals to organizations, as it's going to enable accurate identification of fraudulent activities.

4.1 Dataset

The primary dataset utilized in this project is from the set of IBM Anti-Money Laundering datasets, available on Kaggle. It has 6 datasets divided into two groups of three:

- Group HI has a relatively higher illicit ratio (more laundering).
- Group LI has a relatively lower illicit ratio (less laundering).

Both HI and LI internally have three sets of data: small, medium, and large. In this project, the HI-Small_Transactions dataset is used. This dataset includes a large collection of financial transactions and all the records are anonymized to mimic real-life banking situations. Some of the attributes are transaction amount, transaction time, details of sender and recipient, and type of transaction. The features included in the dataset are:

- Timestamp: The time of the transaction in 'Year/Month/Day Hour: Minute' format
- From Bank: Numeric code for the bank where the transaction originates
- Account: Hexadecimal code for the account where the transaction originates
- To Bank: Numeric code for the bank where the transaction ends
- Account.1: Hexadecimal code for the account where the transaction ends
- Amount Received: Monetary amount received from Account
- Receiving Currency: Currency such as dollars, euros, etc. of from Account
- Amount Paid: Monetary amount paid to Account.1
- Payment Currency: Currency such as dollars, euros, etc. of from Account
- Payment Format: How transaction was conducted, e.g. cheque, ACH, wire, credit cards, etc.
- Is Laundering: 0/1 value with 1 = Transaction is Laundering, 0 = Not

The dataset is structured to aid in the development and testing of machine learning models aimed at detecting suspicious activities indicative of money laundering.

4.2 Data Preprocessing

Data preprocessing is a crucial step in preparing the dataset for machine learning models. The preprocessing steps include:

- Data Cleaning: The dataset had no missing values and no significant outliers or anomalies.
- Feature Engineering: There were two columns representing the paid and received amount of each transaction. After analyzing, it seems that there are transaction fees between different currencies, we cannot combine/drop the amount columns.
- A new feature unique ID is created for each account by adding the bank code with the account number.
- Label Encoding: The categorical features such as 'Payment Format', 'Payment Currency', and 'Receiving Currency' are converted to numerical formats using the sklearn LabelEncoder.
- Normalization: The Timestamp column was converted to numerical features and scaled using min-max normalization to ensure consistency across the dataset.
- Grouping and Splitting: Created two new datasets (i) receiving_df with the information of receiving accounts, received amount, and currency, and (ii) paying_df with the information of payer accounts, paid amount, and currency and a list of currency used among all transactions.
- Create a list of currency used among all transactions.

4.3 Graph Construction

Since the relational information is inherent in the transactions, a graph-based model of the data is created for representation. In this graph:

- Nodes: The nodes represent the individual entities within the network. All the distinct accounts either in the payer or the receiver are extracted as the nodes of the graph. It comprises the unique account ID, the Bank's code, and the label, 'Is Laundering'. Both the payer and the receiver of an illicit transaction, for which (`Is Laundering == 1`), are considered suspicious accounts. The mean of the paid and received amount variables with different types of currencies are aggregated to become the new features of the nodes. The node attributes are specified by the bank code as well as the means of the paid and received amounts with different types of currency.

- Edges: The relations between the modes are determined by the concept of edges used in graph theory. That is, the transactions between the accounts are considered as the edges. In the case of the edge index, all the accounts were eliminated and replaced by an index and stacked up in a list. The features chosen as edge attributes were 'Timestamp', ‘Amount Received’, ‘Receiving Currency’, ‘Amount Paid’, ‘Payment Currency’, and ‘Payment Format’.

This graph-based enables the application of Graph Neural Networks (GNNs) to learn the relational structure and interactions between different accounts.

4.4 Graph Attention Network (GAT)

The core model adopted for this project is known as the Graph Attention Network (GAT). GATs are based on Graph Neural Networks (GNNs) but introduce an attention component that enables the model to pay the most attention to the relevant parts of the particular graph to perform a specific task. This attention mechanism allows the model to put more weight on some of the edges that are used in transactions compared to others. The model was constructed using two GATConv layers followed by a linear layer with a sigmoid activation function for classification.

GAT Architecture:

It is noteworthy to explain the architecture of the GAT used in this project implemented with several layers each with an essential function in the processing of the graph data. The layers are:

1. Input Layer:
   Features: The input to the GAT is the node features which in this context are some characteristics of the accounts the actual transactions involve.

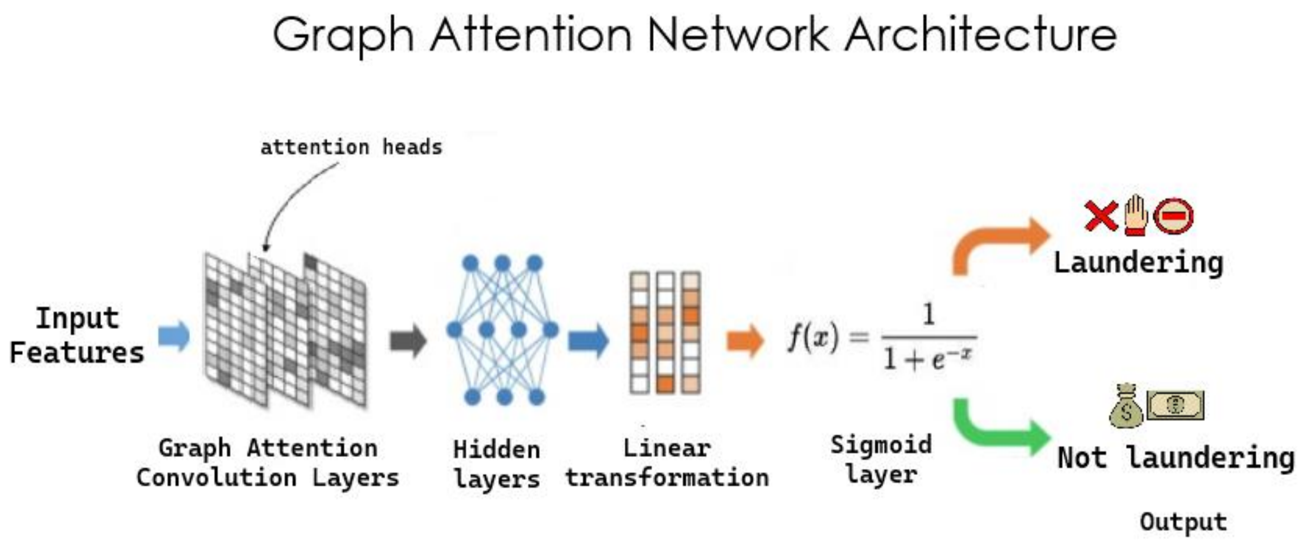
2. Attention Layers:
   - GATConv Layers: The GAT uses Graph Attention Convolution layers, GATConv, that calculate attention coefficients for neighbors, which determines the significance of each neighbor’s input. This makes it possible for the network to attend to the most important nodes.
   - Heads: Multi-head attention mechanisms are utilized, with 8 heads in the first layer. This allows the model to learn from different subspaces and provides robustness.

3. Graph Attention Convolution Layers:
   - First GATConv layer: This layer has a specified number of attention heads and dropouts for regularization.
   - Second GATConv Layer: This layer further processes the aggregated information, condensing it into a lower-dimensional space to form meaningful embeddings.
   - Hidden Layers: The first GATConv layer transforms the input features into a higher-dimensional space (hidden_channels), using multiple attention heads to aggregate information from neighboring nodes.

4. Output Layer:
   - Linear Transformation: A linear layer reduces the dimensionality of the node

embeddings to the desired output size.

- Activation Function: A sigmoid activation function is applied to produce the final output, which represents the probability of a node (account) being involved in money laundering.



The forward pass method defines how the input data is to be passed through the network:

- Input Dropout: In the training process, dropout is employed for the input features to reduce overfitting.

- First GAT Layer: The input node attributes, and the graph connectivity relation in the form of edge indices and edge attributes are passed through the first GAT layer. The last dense layer of neurons uses the Exponential Linear Unit (ELU) activation function to provide the output.

- Intermediate Dropout: Dropout is applied again to the output of the first GAT layer.

- Second GAT Layer: The output is passed through the second GAT layer and again the ELU activation function is applied. The output of this layer would be the learned node embeddings.

- Linear Transformation: These embeddings are passed through a linear layer to get the final output to be used in the desired task of the model.

- Sigmoid Activation: The sigmoid function is then used in generating the probability score of the nodes to define the chances of the nodes being involved with money laundering.

The output of the model is the final output probabilities of the sequence along with the node embeddings. Overall, the transaction network relationships implemented within the GAT model make it possible to capture the details likely to contain money laundering activities.

Hyperparameters

The hyperparameters are somewhat like the knobs that you set before training a model in the field of machine learning. These are related to the learning process and not something the model learns about the data set. To ensure that AML detection obtains the best possible performance, a hyperparameter tuning process was conducted. This included tweaking parameters within the model architecture such as the number of layers, optimizers, loss functions, and number of neurons as well as training parameters such as the learning rate, batch size, and epochs for the best performance on the given dataset and task. The following are summarized from the description of the final optimal settings:

- in_channels: This hyperparameter, which is set based on the data, describes the number of features that are to be linked to every node (account) in the graph. These features indicate numerous aspects of the accounts like the amounts of the transactions, the type of the currency, and the details of the account holders.
- hidden_channels: It defines the size of the hidden layers in the GNN. It affects the model's capacity – higher values enable incorporating more complex relations but may cause over-learning. It is set to 16.
- out_channels: With regards to the final layer, it is set to 1 because the model is binary, as it either involves laundering or does not. It defines the dimension of the output layer of the neural network.
- heads (GAT layers): This setting (8 for the first layer) determines how many attention heads are used in the Graph Attention Network (GAT) layers. The target layers contain more heads, increasing the model's scope of vision to the information from various subspaces of the graph.
- dropout: Set to 0. 6, dropout has a guard over training by randomly erasing some neurons to avoid overfitting. This hyperparameter helps in the determination of the probability a neuron will be dropped.
- activation functions: The ELU (Exponential Linear Unit) activation function is only used before the final classification while introducing the ELU after each of the GAT convolutional layers adds non-linearity. This enables the model to capture higher degree polynomials in data thus making it possible for the model to capture complex relationships in the data.
- epochs: The model is trained for 100 epochs, with early stopping based on validation loss to prevent overfitting.
- loss Function: Using Binary Cross-Entropy Loss (BCELoss) to measure the difference between the predicted probabilities of laundering generated by the model and the actual labels from the data. This loss value helps the training process to minimize the discrepancy between predictions and reality.
- optimizer: Employing Stochastic Gradient Descent (SGD) to update the model's internal

parameters iteratively. It represents the knowledge the model learned from the data.

- learning rate: This value is the step size taken by the optimizer while the model updates the weights. The learning rate for the optimizer (SGD) is set to a smaller value of 0.0001, ensuring the model is not stuck on suboptimal solutions.

4.5 GAT Training Process

The training of the GAT involves the following steps:

1. Data Loading and Preparation:

   The dataset is split into training and validation sets, and loaders are created to handle batching and shuffling. The T. RandomNodeSplit function divides the data into the prescribed portions. The training set is used to train the model, and the validation set is used to check the performance of the final trained model on unseen data.

   The data loaders, train_loader, and test_loader, are made using NeighborLoader to facilitate the feeding of data into the model as the model is being trained and evaluated. Neighbor loaders make sure that the model not only gets the information associated with a particular node (account) but also information about its neighboring nodes in the graph so that the model has to consider the information in the context of the surrounding accounts in this case.

2. Model Training:

   In the training mode, the model conducts calculations tailored to enhance the model's ability to make improvements based on the examples. This includes enabling techniques, such as dropout, which randomly drops out neurons during training to reduce overfitting as well as linear transformations. The training loop iterates for mini-batches of data which is part of the train_loader. Mini-batches enhance training effectiveness because they allow processing some amount of data at a time. This is beneficial for several reasons:

   - It reduces memory usage compared to processing in one pass, which is useful for large datasets or where training is carried out on limited hardware.
   - Hence, it can lead to enhancements in the optimization process since the model can be updated more often with the help of the portions fashioned from the entire dataset.

   Inside each pass, the gradients are cleared, so that gradients for the current pass can be calculated. Gradients inform about the degree of the contribution of each parameter in the model to the particular error (loss) fixed in the current batch. Since the next big data can be placed into a different batch, the gradients must be reset to zero before new gradients from this batch are calculated.

   The current batch of data is then passed on to the selected computation device or unit either the CPU or GPU for enhanced efficiency. Initial studies on GNN training imply the use of calculations in graph manipulation and computation during the training process of the model. Otherwise, the data must be copied to another location such as a GPU if the authorizes it to speed up the learning process in response to the GPU's parallelism.

The model performs a forward pass to the network, the data gets through the layers, and an output is generated. This output corresponds to the model's initial prediction for the nodes of the current batch. The data is passed through the layers of the GNN, the final output represents the model's current state as to what it currently knows of the relationships and patterns within the graph data for this specific batch.

The loss is computed using BCEloss function. The loss value shows the accuracy of the model's predictions comparing the actual labels in the data (laundering or not laundering) of the current batch of data. A lower loss means that the model is closer to predicting the outcome as observed in the original data.

The backward pass is executed for gradient computations. Gradients give information about to which extent each parameter in the model is responsible for the error (loss) in the current batch. The gradients of these values would indicate how the internal parameters of the model need to be manipulated to minimize future loss.

Using the gradients calculated, the optimizer adjusts the parameters of the model in such a way as to reduce the loss. This is simply the model 'updating' what it has learned based on the errors that was made in the current batch. The optimizer, for example, Stochastic Gradient Descent (SGD) used in this work, moves the model's neurons' weights and biases in the direction indicated by the gradients to better fit the predicted results to the true labels.

In the case of each specified batch, the loss is accumulated so that one can monitor the training progress throughout the epoch. The total training loss allows us to evaluate the model's ability to learn and recognize possible problems, such as getting trapped in local minima.

3. Validation:

   The model has been set to evaluation mode, disabling some operations like dropout which is utilized during training to avoid over-fitting. The evaluation mode is used to restrict the model to produce only predictions while the internal parameters of the model are not changed. Passages like dropout, which causes some neurons to drop out during training, are not used during testing because all that is of interest is the model's performance on the validation set.

   The model goes through the batches of test_loader in a loop. This is a new set of data different from the training data to which the model has never been exposed to. It is applied to a model during training to help detect possible overfitting on the current training data. If the model is trained with the given data to learn the concepts, then the generalization test can be used to determine the ability of the model to work on other data not used in training the model.

Gradient calculation is disabled since we do not want to train the model during the evaluation, but to get the output. Gradients are used to change the parameters of the model during the training phase. Since we are not changing the parameters when evaluating, the gradient computing is arguably costly. Disabling it improves efficiency.

The loss measure is calculated and built up for the validation dataset so that its generalization ability for other instances may be checked. As with the training loss, the validation loss measures how well the learned model estimates their corresponding labels in the validation set. A lower validation loss indicates that the model is doing well and can learn and apply a patterning that holds for unseen data.

Class labels are determined from the model's scores of the validation data. The accuracy, and prediction results for correct responses are kept to check the model on the outcome of the validation set. Here, the working of the model for the particular set of data is evaluated right by checking how many times the proposed model was right on the validation data. That is, whether or not discovering money laundering activities are conducted.

4. Logging and Early Stopping:

   The training and validation losses and the validation accuracy of the present epoch are displayed. This offers information on the learning phase of the model. These metrics enable us to determine how well the model performs on both the training data and the unseen validation data.

   The training and validation metrics are then stored in a CSV file for post-processing and comparison with the other models. It also saves these results to be able to later observe the training process and compare them to the other configurations of the model or the other values of the training parameters.

   The early stopping technique is used to overcome the problem of overfitting. The code saves the best validation loss encountered until this point. If the current validation loss is smaller, it means that the model is better at reducing losses on unseen data, this state of the model is saved as the best model. This stands for the epoch in terms of the model concerning validation data. Based on a predefined value of patience, training stops if there's no variation in the validation loss implying that the model has overly concentrated on the training data. The problem of overfitting is prevented by early stopping whereby training is stopped when the model begins to have a higher loss in the validation data.

This evaluation phase is therefore crucial for monitoring the training process and avoiding overfitting. Through the model evaluation with cross-validation and applying early stopping techniques, we can optimize the efficiency of the model's training on the data and the effectiveness of its subsequent application to detect money laundering activities in practice.

4.6 Embedding Extraction and Machine Learning

Graph embedding is a technique of converting the graph-structured data into a lower-dimension while maintaining the original proximity of the graph data. In this project, the embeddings produced by the Graph Attention Network (GAT) are passed to other classical machine learning models to achieve better performance in identifying money laundering.

1. Embedding Generation:

   - The GAT model is loaded with the best-performing weights that were created during the training phase.
   - Embedding Extraction: With the help of the trained GAT model, embeddings are given to each node in the graph. This also involves a forward pass through the network in which the node features, the indices of edge, and their attributes are passed through the network to obtain the node embeddings.
   - Conversion to Numpy: The extracted embeddings are in the PyTorch tensor format. So, they are converted to a more traditional NumPy array format required for traditional machine learning.

2. Dataset Preparation:

   - Feature Matrix (X): The generated embeddings serve as the feature matrix.
   - Target Variable (Y): The labels from the original graph data, indicating whether a transaction is classified as money laundering or not.
   - Train-Test Split: Cross-validation is applied to the data to divide the dataset into the training and testing sets and, thus, to determine the effectiveness of the used ML algorithms.

3. Machine Learning Models:

   The embeddings obtained from the GAT act as input features for various classifiers that belong to traditional machine learning methods. It focuses on classifying nodes (accounts) with the help of learned representations about them. The list of models traverses through data for classification purpose one after the other:

   - K-Nearest Neighbors (KNN)

- Decision Tree Classifier
- Random Forest Classifier
- Gradient Boosting Classifier
- AdaBoost Classifier
- Support Vector Classifier (SVC)
- Logistic Regression
- Linear Discriminant Analysis (LDA)

These models are trained on the embeddings and evaluated to assess their performance in detecting suspicious transactions.

4. Evaluation Loop:

Each machine-learning model is iterated through the data. The model is trained on the training data (X_train, y_train) and makes predictions on the testing data (X_test).

- Accuracy: The accuracy of the model is calculated using accuracy_score.

- Confusion matrix: A confusion matrix is generated to visualize the distribution of correct and incorrect predictions for each class (laundering/not laundering).

- Classification Report: A detailed classification report is generated using classification_report to provide insights into precision, recall, F1-score, and support for each class.

The evaluation metrics for the each of the models are as follows: The model name, accuracy, confusion matrix, precision, recall, F1-score, and support are written to a CSV file named gml_results.csv for future analysis. Thus, the above stated comprehensive evaluation process enables the comparison of performance of other machine learning models using the embeddings that are developed from the GNN model. It then becomes possible, by comparing the obtained results, to determine the model which optimises, with regard to the graph data, the learned relations within the learning process for AML.

Chapter 5

RESULTS AND EVALUATION

In this part of the paper, we discuss the results and evaluation of the Anti-Money Laundering (AML) detection project. The evaluation concerns the identification of the Graph Attention Network (GAT) model and some machine learning (ML) models based on the GAT model's embeddings.

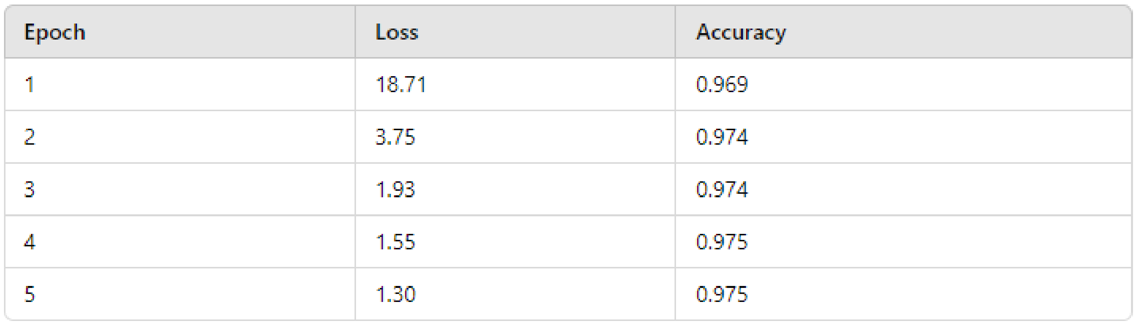
5.1 Graph Attention Network Results

The Graph Attention Network (GAT) model was trained for 100 epochs and Training loss, and validation accuracy of each epoch were noted. The GAT model's architecture and hyperparameters are as follows:

- Input Channels: 16
- Hidden Channels: 16
- Output Channels: 1
- Heads: 8
- Dropout Rate: 0.6
- Activation Function: ELU (Exponential Linear Unit)
- Optimizer: SGD (Stochastic Gradient Descent)
- Learning Rate: 0.0001
- Loss Function: Binary Cross-Entropy Loss

The results are evaluated and stored in the file gnn_results.csv. From this, it is clear that there is a decreasing nature of training loss and high validation accuracy. Here are some important findings from training:

- Training Loss: In the first epoch, the training loss was high at 18.71. By the second epoch, it lowers to 3.75, indicating quick initial learning. After a continuous decrease, the training loss falls to 0.85 after the 13th epoch.
- Validation Accuracy: Beginning with a good value at 0.969 in the first epoch, the accuracy increased much more, reaching about 0.974 in later epochs. So, we can infer that the GAT model was able to learn to recognize patterns related to money laundering activities.
- Here are the detailed results for the first few epochs:



Epoch	Loss	Accuracy
1	18.71	0.969
2	3.75	0.974
3	1.93	0.974
4	1.55	0.975
5	1.30	0.975

Results 1

Overall, the work done in this paper shows that the proposed GAT model performs well in the scenario of anti-money laundering detection. The loss values are low and decreasing while the model achieves high validation accuracy due to its ability to generalize well from the data.

Early Stopping and Model Selection

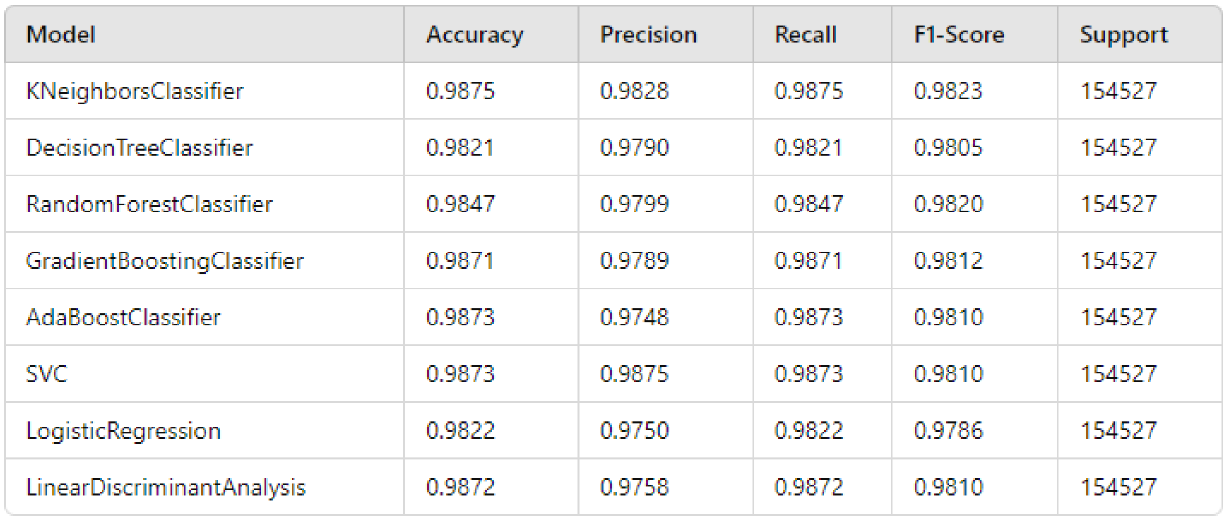
To avoid overfitting, the technique of early stopping was used regarding the validation loss. The training was truncated at 18 epochs because of early stopping. The best model according to the lowest validation loss was chosen which will subsequently be used for generating graph embeddings from the input graphs. This made it possible to achieve a near-perfect generalization from the model evidenced through the accuracy figures with epochs of training having a high figure very early into the training phase with the use of early stopping.

5.2 Machine Learning Model Results

By using the embeddings generated by the GAT model, several machine-learning models are trained to classify the transactions as suspicious or not. The trained and evaluated ML models are:

- KNeighborsClassifier
- DecisionTreeClassifier
- RandomForestClassifier
- GradientBoostingClassifier
- AdaBoostClassifier
- Support Vector Classifier (SVC)
- Logistic Regression
- Linear Discriminant Analysis

The results of the evaluation metrics for each machine-learning model are summarized in the form of a table:



Model	Accuracy	Precision	Recall	F1-Score	Support
KNeighborsClassifier	0.9875	0.9828	0.9875	0.9823	154527
DecisionTreeClassifier	0.9821	0.9790	0.9821	0.9805	154527
RandomForestClassifier	0.9847	0.9799	0.9847	0.9820	154527
GradientBoostingClassifier	0.9871	0.9789	0.9871	0.9812	154527
AdaBoostClassifier	0.9873	0.9748	0.9873	0.9810	154527
SVC	0.9873	0.9875	0.9873	0.9810	154527
LogisticRegression	0.9822	0.9750	0.9822	0.9786	154527
LinearDiscriminantAnalysis	0.9872	0.9758	0.9872	0.9810	154527

Results 2

Also, the confusion matrices for each model give a picture of the prediction classes and better idea about the accuracy values:

1. KNeighborsClassifier: [[152494, 71], [1862, 100]]
2. DecisionTreeClassifier: [[151493, 1072], [1689, 273]]
3. RandomForestClassifier: [[151905, 660], [1703, 259]]
4. GradientBoostingClassifier: [[152521, 44], [1942, 20]]
5. AdaBoostClassifier: [[152565, 0], [1962, 0]]
6. Support Vector Classifier (SVC) [[152565, 0], [1961, 1]]
7. Logistic Regression [[151765, 800], [1947, 15]]
8. Linear Discriminant Analysis [[152554, 11], [1961, 1]]

The results of the evaluation show that all the implemented ML models have high accuracy, it is 98%, hence, it is recommended for use in this AML detection task. This surprisingly shows that the employed ML models of GAT embeddings have more accuracy values than the GAT model though the difference is so small.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Going by what Graph Attention Network (GAT) has been able to do in the fight against financial crimes through the AML project, it has been significant that it was able to execute its duties well. By focusing on the inherent relationships and dependencies within financial transaction data, we have achieved important advancements:

1. Effective Feature Extraction: This paper has shown how, by using the GAT model, it is possible to obtain useful embeddings from the transaction graphs and how the details and interdependencies can be extracted which was not visible through other methods.

2. Improved Detection Accuracy: As a result of these embeddings incorporated as attributes of the features used in our ML models the detection accuracy of suspicious activities concerning money laundering and terrorist financing has improved significantly.

3. Scalability and Efficiency: Using the GAT framework, we have been able to process large amounts of transactional data and perform real-time or near-real-time analysis on them.

4. Interpretability: Due to the high interpretability of the GAT embeddings, the structure of financial networks has been recognized thus helping the compliance officers/ investigators to make better decisions.

5. Compliance and Risk Management: Thus, the integration of GAT-based embeddings into an AML framework improved compliance with the requirements of legislation and other regulation acts, as well as overall risk management.

Finally, I would like to emphasize that the application of the Graph Attention Network has not only strengthened our AML capacities but has also placed us at the forefront of employing complex graph-based methods for financial crime identification and prevention.

6.2 Future Work

Looking forward, there are several avenues for further exploration and enhancement of our AML framework leveraging GAT and embeddings:

- Refinement of GAT Parameters: Fine-tuning of the parameters of the GAT model to enhance on the quality of an embedding as well as the model’s ability to capture weak but important transactional features.

- Integration with Additional Data Sources: The use of other related data types such as textual information from the descriptions of transactions, scorecards of the external risk indicators, and customer behavior analytics to augment the GAT embeddings and improve the detector’s performance.

- Real-Time Monitoring Enhancements: Developing technologies and methods to utilize GAT embeddings of continuously updated, from which alerts can be generated in real-time so that threats remain responded to proactively.

- Privacy-Preserving Techniques: Exploring privacy preservation methodologies with which the GAT embeddings can be developed to conform to the principles of data protection while at the same time retaining the efficacy of the model.

- Deployment of Explainable AI (XAI): Introducing concepts of explainable AI to improve understanding of the finalized results relying on GAT embeddings to make modifications in the system and gain credibility among partners.

In these areas, there is more work to be done to enhance the institution’s AML expertise and fortify financial crime safeguards in an ongoing quest to reinforce global security and compliance using graph-based approaches. This way, the conclusion gives credit to the obtained accomplishments and advantages based on the GAT deployment, and the future work section suggests how to fine-tune and develop this AML framework in the future.

REFERENCES

[1] McKinsey & Company. (2022). The Fight Against Money Laundering: Machine Learning is a Game Changer. Retrieved from McKinsey.

[2] ResearchGate. (Year). Anti-Money Laundering Alert Optimization Using Machine Learning with Graphs. Retrieved from ResearchGate.

[3] Emerald Insight. (2020). Anti-Money Laundering Systems: A Systematic Literature Review. Journal of Money Laundering Control, 23(4), 833-848. Retrieved from Emerald Insight.