Knowledge Graph-Enhanced Machine Learning for Financial Crime Intelligence: Linking Fraud, Money Laundering, and Cybersecurity Threats Across Distributed Ledgers

MD SUMSUZOHA, ABHISHEK RAVVA,
REZA E RABBI SHAWON, MD ZAHIDUL ISLAM,
MD ABDULLAH AL HELAL, SANTOSH PANT,
LAXMI PANT

Abstract: *Financial crime in digital asset ecosystems has become a lot harder to track lately. The bad actors are using complicated, overlapping networks of transactions that standard machine learning methods just cannot detect accurately. This paper looks into a graph-based learning setup to identify this kind of illegal activity in cryptocurrency networks by focusing entirely on the connections built into the financial data. The experiment uses the Elliptic Bitcoin Dataset, which is a massive network of labeled transactions split into legal, illegal, and unknown categories. The setup treats the money flows as a directed graph that changes over time, turning individual transactions into points and the actual Bitcoin transfers into the lines connecting them. The approach combines a few different ideas. It starts with basic machine learning models trained on custom transaction details, adds in graph embedding tricks like Node2Vec, and finishes with graph neural networks, specifically GraphSAGE, to handle the semi-supervised sorting of the data points. To keep the testing fair and realistic, the data was split strictly by time. That stops information from leaking out of the future, mimicking a real fraud detection setup where an analyst has to guess what comes next based only on what happened in the past. Success is measured using precision, recall, F1-score, ROC-AUC, and ranking tricks like Precision@K. These metrics are necessary because the dataset is incredibly lopsided, which is always the case with financial fraud, where the bad transactions are buried in a sea of normal ones. On top of that, structural clues from the graph shapes and extra explanation tools were added to make the caught illegal patterns easier to understand, giving financial investigators something useful to actually work with. The main*

Md Sumsuzoha, PhD in Business Management, University of the Cumberlands
Abhishek Ravva, Engineering/Industrial Management, Gannon University, Erie, PA
Reza E Rabbi Shawon, MBA-Business Analytics, Gannon University
Md Zahidul Islam, MBA in Business Analytics, Gannon University, Erie, PA
Md Abdullah Al Helal, Master of Science in Business Analytics, Trine University
Santosh Pant, BBA, Kantipur College of Management and Information Technology, Kathmandu, Nepal
Laxmi pant, PhD in Business Management, University of the Cumberlands
Corresponding Author: Md Sumsuzoha, Email: msumsuzoha56031@ucumberlands.edu

goal here is to show that paying attention to the shape of the graph does a much better job of spotting fraudsters than just looking at standard flat tables. It also helps uncover hidden groups of transactions that point to people working together. The results help push forward better graph-based crime-fighting tools that can handle the structural risks built into decentralized money networks.

Keywords: Financial Crime Detection; Graph Neural Networks; Elliptic Bitcoin Dataset; Fraud Detection in, Cryptocurrency, Node2Vec Embeddings, Temporal Graph Learning

Introduction

Background and Motivation

The rise of decentralized digital currencies has completely changed how money moves around the world. People can now send value directly to each other without needing a bank or any other middleman to approve the transfer. Nakamoto [20] started all of this by introducing Bitcoin as a distributed electronic cash system, which laid down the groundwork for building financial systems on a blockchain using secure, cryptographic ledgers. This setup has made financial systems more accessible and made transactions public for everyone to see, but it has also opened up new gaps that criminals can take advantage of. As more people start using blockchain, tracking down malicious behavior gets trickier because accounts use pseudonyms and there are no standard identity checks like you see at a traditional bank. Financial crime online has become much more clever, ranging from basic fraud to complicated ways of moving dirty money and attacking distributed networks. Ngai et al. [21] pointed out that while data mining has helped fraud detection evolve, older methods usually assume data is neatly structured and independent. That assumption fails to capture how different accounts actually interact with each other. This is an especially big problem on a blockchain, where transactions look more like a massive web of connections than a list of isolated events. Conti et al. [7] also highlighted that Bitcoin presents a unique set of security and privacy headaches, including how anonymity gets abused, the limits of tracking transactions, and flaws in decentralized consensus setups. Dealing with these issues requires smarter analytical tools that can look at both the shape of the network and how users behave.

Recent work in network learning has started using graph-based perspectives to look at financial data. Even with all the progress in machine learning, most standard systems still fail because they cannot factor in relational structures very well. The demand for systems that are easy to understand, scalable, and tough has pushed researchers to combine graph methods with financial intelligence tools. Specifically, using knowledge graphs offers a structured way to encode how different entities link up, which gives a much deeper understanding of how these complex systems function. Hogan et al. [11] described knowledge graphs as structured maps of entities and their relationships that help with reasoning tasks and machine learning, noting how useful they are for fields like finance and cybersecurity. In this light, a network of financial transactions can be viewed as a graph where nodes represent individual transactions and edges show the actual flow of money. This lets machine learning models use those connections to make better predictions. Shifting from analyzing isolated features to learning from relationships is the main reason behind this study.

Financial Crime Intelligence in Distributed Ledgers

Spotting financial crime on a distributed ledger requires tools that can capture both the shape of the network and how things change over time. Bitcoin, as Nakamoto [20] explained, runs on a public ledger where every transaction is visible but the real names of the users are hidden behind pseudonyms. This creates a weird mix of total transparency and total anonymity, making forensic work pretty complicated. Conti et al. [7] went into more detail on this, explaining how this setup makes it tough to track illegal funding since bad actors can use tricks like changing addresses constantly, using mixing services, or chaining transactions together to hide their tracks. These tactics make old-school fraud detection useless because those systems assume they can identify the entities and that the relationships between them stay static. People working on financial intelligence have been paying a lot more attention to graph methods lately because they are great at modeling interconnected systems. Weber et al. [27] showed that graph convolutional networks can work well on Bitcoin transaction graphs to spot anomalies and help with forensics, proving that learning the structure of a network helps identify illegal patterns. Even so, plenty of current methods still run into issues like temporal leakage, uneven datasets, and poor interpretability, which makes it hard to use them in real-world monitoring setups. On top of that, financial crime is rarely just a simple case of fraud. It usually involves multi-layered activities like money laundering or coordinated cyberattacks that show up as interdependent behaviors across the network.

The whole concept of a knowledge graph gives a solid theoretical base for fixing these issues by letting models learn from relationships between entities and their interactions. Hogan et al. [11] noted that knowledge graphs make it easy to combine totally different sources of information into a single structured format, which gives machine learning a big boost in complicated areas. For financial crime tracking, this means you can model how transactions depend on each other, spot similar behaviors, and see how effects ripple through a network, none of which shows up in a standard spreadsheet dataset. It allows a shift from staring at isolated transactions to analyzing the whole system, where bad behavior gets flagged because of structural anomalies or weird relational patterns. Graph representation learning techniques make this even more powerful by turning complex network shapes into low-dimensional embeddings that work well with predictive models. Grover and Leskovec [9] introduced node2vec as a scalable way to learn continuous feature representations for nodes in a graph, catching both local details and the bigger structural picture. Kipf and Welling [16] built on this idea with graph convolutional networks, making semi-supervised learning work on graph data. Hamilton et al. [10] made the framework even more general with inductive representation learning, which lets models handle new, unseen nodes in massive graphs. Together, these developments provide the technical foundation for modern financial intelligence systems that catch illegal activity by looking at structural and relational learning instead of analyzing features in a vacuum.

Problem Statement

Even with all the recent progress in machine learning and graph modeling, the systems used to catch financial crime still hit a wall when it comes to dealing with the messy, time-dependent relationships inside distributed ledgers. A lot of current setups treat transactions as isolated events or rely on static data snapshots. That approach completely misses how blockchain networks actually work, since everything is connected. Because of this, these systems struggle with accuracy, fail to spot new types of scams, and miss coordinated criminal loops that bounce across different parts of the network.

On top of that, there is a bad habit of messing up the evaluation process by accidentally letting future data slip into the training phase. This usually happens when researchers use standard random splits on financial data that actually has a strict timeline. Kapoor and Narayanan [15]

pointed out that this exact kind of data leakage is a huge reason why so many machine learning studies cannot be reproduced. It creates inflated, overly optimistic accuracy scores that fall apart completely the second the model faces real-world data. In the world of financial crime, where the timing of a transaction and who is talking to whom are everything, these structural flaws ruin the credibility of the findings. Right now, there is also a failure to combine solid network data with clear explanations in a way that actually helps someone make a real decision, which keeps these theoretical models from being useful in actual operations.

Research Aim

The main goal here was to build a machine learning framework that uses knowledge graphs to handle financial crime analysis on distributed ledgers. The idea was to do a better job of spotting illegal activity by putting graph representation learning, graph neural networks, and clear explainability features into one single system. Moving past the old way of looking at single transactions meant tapping into the web of connections built right into blockchain networks, making it easier to flag shady patterns. Another big part of the plan was setting up a clean experimental design that avoids regular pitfalls like temporal leakage, uneven data groups, and black-box results. Combining graph feature engineering with smart representation models was intended to create something scalable and adaptable. The final hope was to deliver a system that can actually back up financial intelligence teams working inside these fast-moving, complicated digital networks.

Literature review

Financial Crime Detection

Financial crime detection has changed a lot over the last few decades. It started out with simple systems based on basic rules and has moved toward machine learning setups that can handle massive amounts of financial data. The older research in this area focused on structured ways to spot fraud using set thresholds. Ngai et al. noted that fraud detection used to rely mostly on classification systems that combined data mining methods like clustering, decision trees, and statistical modeling to find odd patterns in neat, organized data [21]. Those older methods built the groundwork for what is used now, but they had a built-in weakness because they depended on people manually creating features. They just couldn't adapt well when criminals changed their tactics.

Lately, the focus has shifted toward making predictive models more stable and adaptable when dealing with messy, unpredictable financial data. Jakir et al. pointed out that modern financial systems have a lot of background noise, making it really hard for AI to separate actual signs of crime from normal market ups and downs [14]. The issue gets worse in fast-moving areas like digital finance, where people change how they act quickly, and bad actors keep tweaking their methods. Because of this, standard machine learning setups like Random Forests became popular. They are tough enough to handle complicated, non-linear connections between different data points. Breiman introduced Random Forests as a way to bundle decision trees together, which helps keep predictions steady and stops the model from overfitting on complex financial data [5]. Even though they work well, these models look at data points as independent pieces, missing the actual connections between entities in a network.

Newer gradient boosting methods have pushed the accuracy even further for structured data. Chen and Guestrin created XGBoost to be a fast, scalable framework that uses regularized learning to balance computing speed with accuracy [6]. Models like that are the standard benchmarks now for spotting fraud, but they still look at data in spreadsheet-style tables. That format makes it hard to see the deep interactions between different financial players. So, while machine learning made things better, current detection systems are still stuck because they leave

out network shapes and timing patterns. This creates a clear need for methods that use graphs and knowledge setups.

Fraud Detection in Cryptocurrency Networks

Cryptocurrency networks bring a whole new set of problems for spotting fraud because they are decentralized, use fake names, and the transaction patterns change constantly. Unlike regular banks, blockchain networks like Bitcoin put all transaction history on a public ledger while keeping the real names of users hidden. Weber et al. showed that graph convolutional networks could be used on Bitcoin transaction history to spot money laundering, showing that graph learning can uncover illegal activity on a blockchain [27]. That study highlighted how cryptocurrency transactions form complex network webs that are useful for finding anomalies and doing financial detective work. Looking past Bitcoin alone, other research has looked into finding hidden networks where people work together to commit fraud. Dola et al. showed that machine learning can spot collusive behavior in corporate settings by looking at how different entities depend on each other and interact [8]. The data showed that fraud usually comes from a coordinated effort across a network rather than just one weird transaction on its own. This matches the broader idea that crypto crime is a systemic issue instead of just a single bad transfer.

Still, catching fraud in crypto stays tough because blockchain data shifts and changes all the time. Bhowmik et al. stressed that forecasting financial risk in crypto markets requires models that can adapt to changing data distributions and new behaviors [4]. This matters because fraud in decentralized setups is never fixed; it changes the moment detection systems get better. Because of that, old-school machine learning models lose their edge over time. Even though graph methods like the ones from Weber et al. beat traditional tools, they still run into issues with scaling up, explaining their logic, and working well across different time periods. These gaps show that the field needs stronger frameworks combining graph learning with time-based tracking and clear explanation tools to handle fraud in crypto ecosystems.

Machine Learning for Financial Crime Intelligence

Machine learning is a big part of how people spot financial crime these days, mostly because it lets computers hunt for weird patterns and flag fraud on their own. For a long time, the go-to tools have been things like logistic regression, decision trees, and ensemble methods. They are fast, and they make sense to human analysts. Breiman really changed things by showing how Random Forests could handle messy, high-dimensional financial data where relationships aren't just simple straight lines, which is why it became a standard baseline for catching fraud [5]. Then Chen and Guestrin came out with XGBoost, making gradient boosting much faster through smart regularization and parallel processing, which pushed the accuracy of these classification tasks even further [6]. Even with all that progress, standard machine learning has a massive blind spot because it assumes every data point is independent and identically distributed. Jakir et al. pointed out that financial data is incredibly noisy, meaning the actual clues about fraud are buried under a mountain of normal market movements and random chaos [14]. This makes it really tough for standard models to tell a smart criminal apart from a regular customer, especially since scammers change their tactics constantly. On top of that, most machine learning tools require flattening the data into a basic spreadsheet format, which completely throws away the connections and network structures that define financial life.

Missing those structural connections is a huge drawback when trying to catch financial criminals. Fraud rarely happens in a vacuum as an isolated transaction; it usually involves a bunch of different accounts working together in a coordinated way, so the math needs to see these complex relationships to be useful. Ensemble methods might be great at looking at individual data points, but they cannot see how different entities are interacting. That makes

them pretty weak at spotting sophisticated operations like money laundering or organized cyberattacks. This exact gap is why researchers started looking into graph-based learning, which actually gives you a way to map out those connections. Some people have tried building hybrid systems that mix old-school statistical learning with some basic structural features, and that helps a bit, but it still doesn't truly bake the network's shape right into the predictions. Because of this, a big gap remains between what traditional machine learning can do and what a modern financial intelligence setup actually needs, since the latter requires both sharp accuracy and a deep understanding of how the whole network is wired.

Knowledge Graphs in Financial Intelligence

Knowledge graphs have become a really popular way to map out messy, interconnected data in fields like banking, cybersecurity, and social networks. Hogan et al. describe a knowledge graph as a structured setup of entities and the links between them, which lets computers do semantic reasoning and run machine learning over connected data [11]. These frameworks let people dump wildly different types of data into one single, unified graph. That is incredibly useful for financial intelligence, where knowing how two things are related is just as important as knowing what those things are in the first place. In the financial world, these graphs can look like a lot of different things, from deep semantic graphs to property graphs, heterogeneous networks, or just simple transaction maps. Semantic graphs use a lot of complex labels and rules, while transaction-centric graphs focus entirely on the direct links created by events like a wire transfer or an interaction. Islam et al. showed that graph neural networks can actually model systemic risk across the whole financial sector by tracking how trouble spreads back and forth between traditional banks and cryptocurrency networks, which proves how useful graph shapes are for keeping tabs on financial stability [13]. It shows why people are paying so much attention to these methods lately.

Still, for all their power, knowledge graphs can get incredibly complicated, hard to scale up, and tough to explain to a human observer. A lot of systems try to use heterogeneous graphs that require a ton of expert industry knowledge just to build, and that kind of detailed data isn't always sitting around in the real world. Because of that, people often fall back on simpler setups like transaction-centric graphs to keep things realistic. This study uses one of those transaction-centric setups, where the nodes are the transactions themselves and the edges show the direction the money moved. This makes it possible to model the network's connections without getting bogged down in full semantic detailing, which keeps the computing time reasonable. A transaction-centric graph doesn't have the rich, conceptual labels of a full knowledge graph, but it still gives machine learning models plenty of structural data to find meaningful patterns. Of course, simplifying things like this means losing some semantic detail and making it harder to interpret exactly why a model made a specific choice. Even so, when dealing with limited data and focusing strictly on catching financial crime, these transaction-centric graphs are a highly practical way to spot shady behavior across distributed ledgers.

Learning Graph Representations

Learning how to represent graphs has become a core part of building machine learning setups that deal with network data. Some of the earliest attempts, like the DeepWalk method from Perozzi and the team, showed that running random walks across a network actually works well for figuring out hidden node traits, basically treating those walks like sentences in a text model [22]. Grover and Leskovec took that concept a step further with Node2Vec, adding some flexible tuning parameters to the walks so the model could look at both close-up neighborhoods and the bigger network picture, which made the resulting embeddings work a lot better for predicting things later on [9]. Graph neural networks came along a bit later as a big step forward, letting systems learn directly from the graph structure itself from start to finish. Kipf and Welling brought out Graph Convolutional Networks, or GCNs, which took regular convolution

ideas and stretched them to fit graphs, making semi-supervised learning possible on network data [16]. These setups pull data in from nearby nodes to build representations that show both what a node looks like on its own and where it sits in the overall shape of the network. The catch is that GCNs usually need to see the whole graph during training, meaning they struggle when they run into totally new nodes later.

To get around that issue, inductive systems like GraphSAGE were built. These methods handle large scales by sampling small pockets of neighbors and blending their information, which means they can handle new nodes and massive networks without breaking a sweat. When it comes to finance, this kind of learning does a great job of picking up on connections for fraud spotting, tracking system-wide risks, and catching weird transactions. Even so, dealing with highly lopsided data, changes over time, and explaining how decisions are made still poses a real challenge in actual financial systems. Even with all the progress, the current ways of learning graph features run into walls when you try using them for financial crime tracking. A lot of setups just ignore the order of events in time, and others get bogged down by scale or become total black boxes when transaction networks get huge. These specific weak spots are exactly why combining graph learning with time-based validation and clear explanation tools is necessary to make these methods actually useful for financial security work.

Methodology

Research Design and Problem Formulation

This research uses a quantitative experimental setup to look at financial crime data through graph-based machine learning. The main problem is set up as a semi-supervised node classification task, using a network built out of blockchain records. Every single node represents a Bitcoin transaction, and the lines connecting them show money moving from one transaction to another. The main goal is sorting these transaction nodes into bad or good buckets, all while keeping a bunch of nodes unlabeled during training to mimic a true semi-supervised setup. This matches up with how real financial investigations work, since people only ever have solid labels for a tiny fraction of transactions, leaving the rest as total question marks. Setting things up this way also makes it possible to fair-test standard machine learning against graph-heavy methods and full graph neural networks, proving whether better results come from the network structure itself or just smart feature tweaks.

Dataset Description and Problem Scope

The project relies on the Elliptic Bitcoin Dataset from Weber and the team, which is a standard choice for blockchain analysis and financial forensic tracking. The data splits into three main parts: transaction features, transaction labels, and an edgelist that maps the flow of Bitcoin between transactions. The feature file includes 166 specific traits calculated for each transaction, capturing basic math properties and timing. The class file contains the actual labels marking things as illicit, licit, or unknown, and the edgelist outlines the actual shape of the network map. For this work, illicit labels point toward fraud and money laundering setups, while licit ones cover legitimate transactions. It is worth noting that specific cyber threats are not explicitly flagged in this data; they just serve as the background reason for wanting better financial intelligence tools rather than being something the model is trying to guess directly. Keeping that boundary clear ensures the models stay focused on what the data can actually support while still mattering to the bigger world of financial security.

Data Integrity and Preprocessing Validation

Before building any models, the dataset goes through a thorough check to make sure everything is consistent and reliable for the machine learning phases. This part involves looking for duplicate transaction IDs, making sure the feature matrix matches up correctly with the

transaction graph, and checking that the edges cover the entire time range of the data. The distribution of labels gets analyzed to measure the massive imbalance between illegal and legal transactions, which is a pretty standard headache in financial fraud datasets. The way labels are spread out over time is also checked to make sure no future info accidentally leaks into the earlier training steps. Doing these preprocessing steps is necessary to keep the structure of the data sound so the final results do not get warped by weird inconsistencies or data leaks.

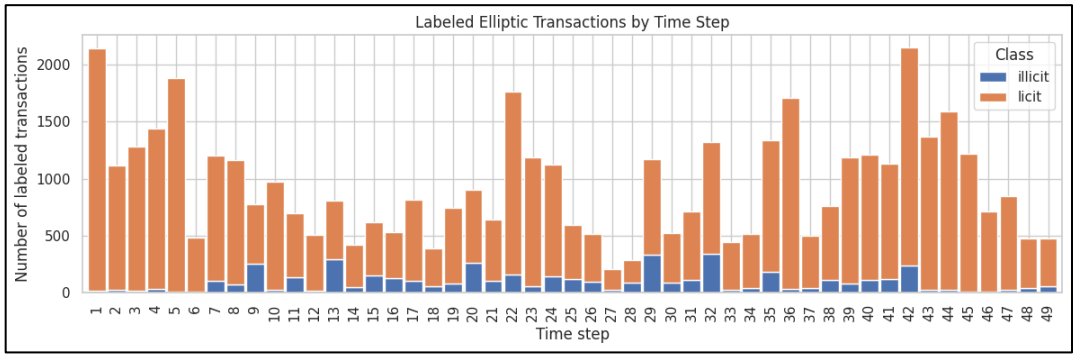

Fig.1: Elliptic transactions by time step

Temporal Data Partitioning Strategy

To keep the evaluation realistic and stop information from leaking, the dataset is split up using a strict chronological approach based on when the transactions happened. The data gets divided into training, validation, and testing sets in order of time. This setup means the model is always tested on future transactions that it didn't get to see during the training period. Choosing this design is incredibly important for financial crime work, where models need to handle future behavior patterns instead of just memorizing old relationships. Unlike splitting things up randomly, this time-based approach stops the performance scores from looking artificially inflated. It gives a much better picture of how the system would actually do if it were deployed in a real financial monitoring setup. This method tackles a major issue in machine learning research, where messy data splitting can lead to totally wrong conclusions about how well a model works.

Building the Transaction-Centric Knowledge Graph

A transaction-centric knowledge graph is put together to map out the structural relationships inside the Bitcoin dataset. The graph is officially set up as a directed graph:

$$G=(V,E)$$

Here, each node $v \in V$ represents a Bitcoin transaction, and each directed edge $e \in E$ shows value moving between those transactions. Unknown nodes stay inside the graph structure to keep the full shape intact while the model learns the representations. This representation does not bother with multiple entity types, complex ontological rules, or extra outside data. Instead, it focuses entirely on how transactions interact with each other, creating a clean structural graph that works well for machine learning and graph neural networks. This choice keeps things running fast computationally while still picking up on the important relational connections between transactions, which helps spot weird patterns through structural anomalies in the network.

Feature Engineering and Baseline Modeling

Feature engineering is used to pull structural and relational details out of the transaction graph. These graph-based features include things like in-degree, out-degree, total degree, PageRank centrality, and core number, which show both local connections and the wider influence a node has in the network. On top of that, neighborhood risk features are built, like the percentage of illegal neighbors and label coverage inside a transaction's local area. These are calculated safely using only training data to avoid any leaks. These new features are mixed in with the original transaction details to give the predictions more punch. Three standard baseline machine learning models are set up so there is something to compare against. Logistic Regression serves as the linear baseline, using class weights to handle the lopsided dataset. Random Forest is brought in as a non-linear ensemble choice that can catch how different features interact, and XGBoost is used because it handles structured tabular data really well. Finally, a graph-augmented XGBoost model is put together by throwing the graph-derived features in with the original transaction attributes to see exactly how much value that structural data adds to the final results.

Graph Representation Learning and Neural Modeling

To dig into the deeper structural patterns of the transaction network, two different graph representation learning methods are brought in. The first one is Node2Vec, which makes low-dimensional embeddings for each transaction node by running biased random walks that map out both the local and global graph shape. These embeddings are trained using adjustable settings like walk length, context size, embedding dimensions, and the total number of walks. After that, they get plugged into an XGBoost classifier as input features to see how much they help with predictions. The second method uses a Directed GraphSAGE model built for semi-supervised node classification. The setup uses separate aggregation branches for incoming and outgoing edges, which lets the model catch the actual direction of the transaction flows. Those aggregated pieces are sent through a merge layer, followed by dropout regularization and a final classification layer. The model trains using a weighted cross-entropy loss to deal with the uneven class balance, and the AdamW optimizer handles the tuning, using early stopping based on how the validation set performs. This setup allows for inductive learning on the graph, meaning the model can handle brand-new nodes while still respecting the overall structural connections.

Explainability, Fraud Intelligence, and Experimental Configuration

To make the outputs easier to understand, model explanations are created using feature importance metrics and SHAP (SHapley Additive exPlanations). These tools give both a broad and a detailed look at why the model made certain predictions, adding some transparency to financial decision-making setups. Moving past simple predictions, a fraud intelligence layer is set up to rank transactions by their risk scores, pull out the top-K most suspicious ones, and generate visual subgraphs of high-risk neighborhoods inside the network. This helps point out weirdly shaped clusters that might mean people are coordinating illegal activity. The whole experimental setup is built to be reproducible and stable. Fixed random seeds are used across every single model to keep the results consistent. Hyperparameters for all the models are clearly set and managed, and the experiments run in a standardized software environment. All the trained models and the files made along the way are saved so anyone can run them again or analyze them more deeply. Finally, potential threats to validity are openly weighed. This includes internal validity worries about class imbalance and model sensitivity, construct validity issues from using proxy labels for illegal acts, external limitations from using just one Bitcoin dataset, and basic reproducibility issues tied to computer performance variance. Weighing these factors keeps the experimental design rigorous and clear.

Experimental results

Overall Model Performance on Temporal Test Set

The evaluation process relied on splitting the Elliptic Bitcoin Dataset strictly by time. This setup was meant to see how well the models could generalize under the kind of conditions you would actually face while monitoring real financial data. To measure performance, several metrics were tracked, including ROC-AUC, PR-AUC, F1-score, macro-F1, plus some ranking metrics like Precision@K and Recall@K. Looking at the final numbers, clear differences showed up between the basic tabular models, the ones boosted with graph features, the embedding methods, and the graph neural networks. Out of every single method tested, the top performance on this temporal test set came from the tabular XGBoost model. It ended up with a PR-AUC of 0.5574. That is a pretty high bar for a baseline, and it shows that if you engineer your transaction-level features carefully, they hold a massive amount of predictive power on their own, even if you do not explicitly plug in the graph structure. Random Forest was right up there too, hitting a competitive PR-AUC of 0.5375. On the flip side, Logistic Regression fell flat with a PR-AUC of only 0.1915. It seems clear that you need non-linear modeling if you want to catch the actual fraud patterns hidden in this data.

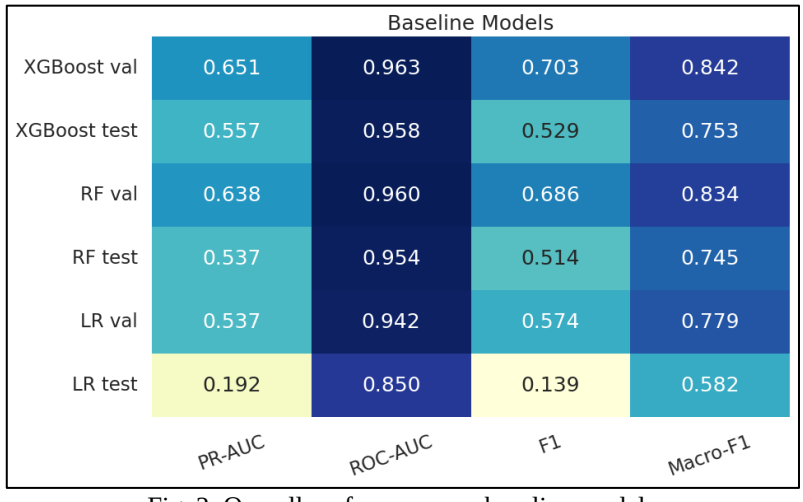

Fig. 2: Overall performance on baseline models

Impact of Graph-Enhanced Feature Engineering

To see whether structural data actually added anything useful, a few graph-based features such as degree centrality, PageRank, core number, and neighborhood-illicit ratios were added to the XGBoost setup. At first, things looked good on the validation set. This hybrid model pushed its PR-AUC up from 0.6508 to 0.6579, making it look like the graph structure was giving the model an extra predictive boost during training. But things took a turn on the temporal test set. The performance actually dropped down to 0.5378, which is lower than the 0.5574 score from the pure tabular baseline. This drop tells us that while graph features do grab hold of some meaningful structural relationships, the way they translate across different periods of time is pretty limited in this current setup. It underscores why testing across time is so critical for systems built to catch financial crime. The structural patterns can change completely from one observation window to the next. Even with that drop in performance, looking at the feature importance logs showed that the neighborhood illicit ratio from training stayed one of the most influential predictors. The local graph layout definitely has valuable clues, but that information seems a bit unstable over time.

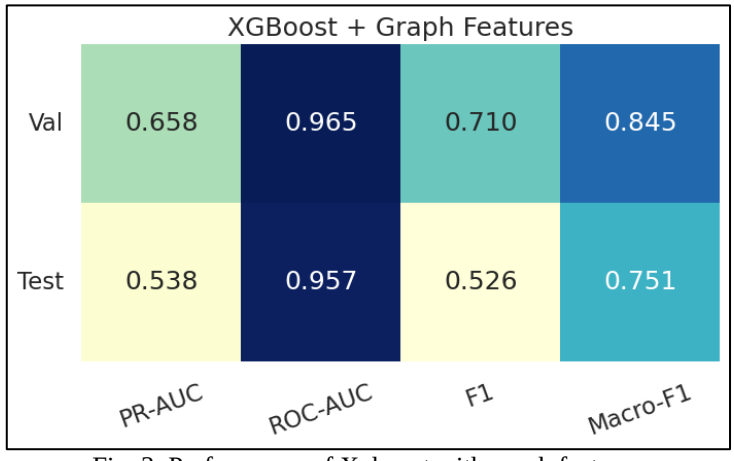

Fig. 3: Performance of Xgboost with graph features

Node2Vec Embedding-Based Representation Learning

For the Node2Vec approach, the training involved building 64-dimensional embeddings by running biased random walks across the transaction graph. After feeding these embeddings into an XGBoost classifier, it pulled off a validation PR-AUC of 0.6482 and a test PR-AUC of 0.5387. That is a tiny bit better than the model with the manual graph features, but it still sits just under the pure tabular baseline. The model did hit a very high test ROC-AUC of 0.9575, but it struggled more with precision and recall when you compare it to the simpler tabular choices. It looks like Node2Vec is great at mapping out who is close to whom and capturing the big-picture graph layout, but those specific representations do not quite line up with the exact markers needed to spot illicit transactions when the data shifts over time. These embedding methods clearly offer a helpful structural signal, but they probably need more tuning or better hybrid setups before they can reliably beat a strong tabular baseline.

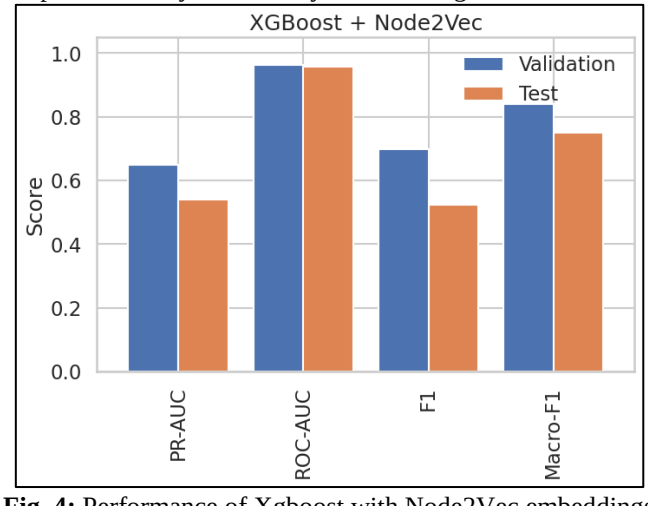

Fig. 4: Performance of Xgboost with Node2Vec embeddings

Directed GraphSAGE Performance Analysis

The Directed GraphSAGE model was built specifically to follow the actual direction of transaction flows. It did this by using completely separate mechanisms to aggregate incoming and outgoing paths. It looked great during training, hitting a solid validation PR-AUC of

0.6384, which indicated that it was learning the graph structure and relationships well. However, the performance took a massive dive on the temporal test set, falling all the way to a PR-AUC of 0.4053. A drop that big points to GraphSAGE being incredibly sensitive to shifts in the data distribution over time. The model proved it had plenty of capacity to learn complex representations during the training phase, but it just could not figure out how to handle future, unseen transaction patterns. This gap between the validation scores and the final test scores highlights how tough it is to use deep graph neural networks on financial data that keeps moving around, especially when the underlying distributions shift from one time step to the next.

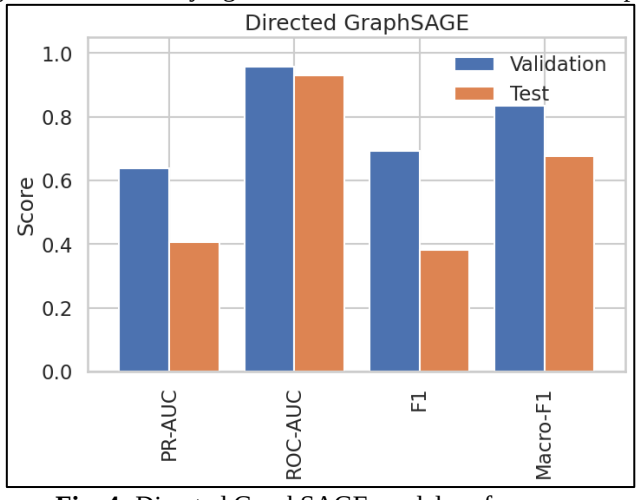

Fig. 4: Directed GraphSAGE model performance

Ablation Study and Comparative Analysis

The ablation study lays out a straightforward comparison of how each different modeling setup stacked up. Looking at the numbers, the top spot went to the tabular XGBoost model. Random Forest and the Node2Vec-enhanced version of XGBoost followed right behind it. On the other end of the scale, GraphSAGE struggled quite a bit, falling way short across every metric used to score it. To get specific about how they ranked on the test PR-AUC scores, tabular XGBoost led at 0.5574. Node2Vec-enhanced XGBoost hit 0.5387, while XGBoost using basic graph features landed at 0.5378. Random Forest was right there too at 0.5375. Further down, GraphSAGE ended up at 0.4053, and Logistic Regression trailed heavily at 0.1915. Seeing things play out this way makes it clear that adding graph elements into the mix failed to give a reliable boost over a solid, standard tabular setup. What these results seem to point out is that throwing in structural data from graphs is tricky. Just because you have that network data doesn't mean it automatically plays nice with tabular features. Sometimes it even messes with performance. The hand-crafted graph features and the learned embeddings didn't really add anything extra once time-based constraints were introduced. That really underscores why testing these things under realistic, chronological conditions matters so much.

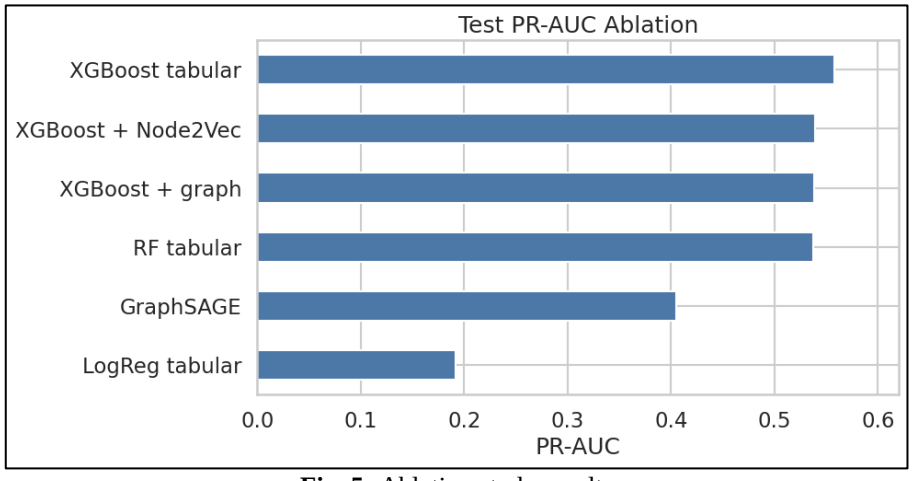

Fig. 5: Ablation study results

Temporal Robustness and Generalization Behavior

One thing that happened across the board was a noticeable drop-off in scores when switching from validation data to the actual test periods. To give an idea of the gap, the strongest validation setup was hitting PR-AUC scores north of 0.65. By the time those same models ran on the test data, those numbers dipped down to around 0.54. A slide like that is a clear sign of a temporal distribution shift happening inside the dataset. It highlights how tough it is to build financial crime detection models that hold up well against future transactions they haven't seen yet. These outcomes really prove why splitting data chronologically is the right move for this kind of financial research. Models that looked great during the validation phase lost some of their edge on future data. If the data had been split randomly or without keeping time in mind, the performance estimates would have looked way too optimistic. Using strict time-based validation rules is pretty much essential if you want to know how a system will actually behave in a real-world financial monitoring environment.

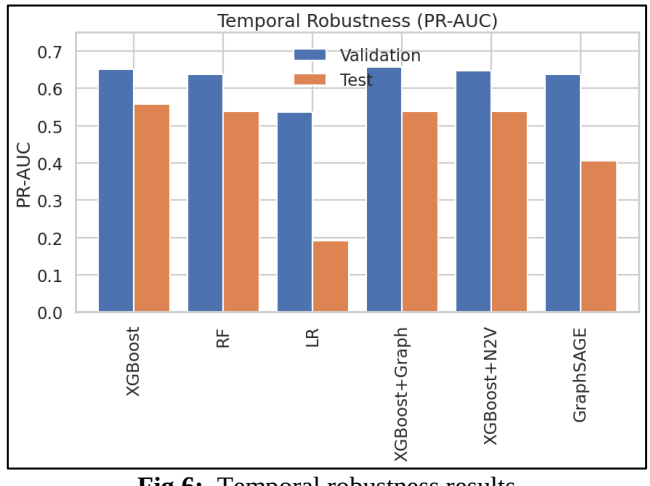

Fig.6: Temporal robustness results

Explainability and Fraud Intelligence Insights

Peeking inside the models using feature importance and SHAP values showed that a mix of basic transaction details and graph-derived metrics drove the predictions. Looking at the graph side of things, the illicit ratio in the training neighborhood turned out to be one of the heaviest hitters. This tells us that shady transactions have a strong tendency to bunch up inside specific pockets of the network graph. From a practical, day-to-day security perspective, sorting these transactions by their risk scores worked well for flagging high-risk groups that held a massive chunk of the illicit activity. Checking out the top-K predictions showed that a major portion of the bad transactions wound up packed into a pretty small window of alerts. For an engineering team trying to figure out which alerts an investigator should look at first, that kind of concentration is incredibly useful.

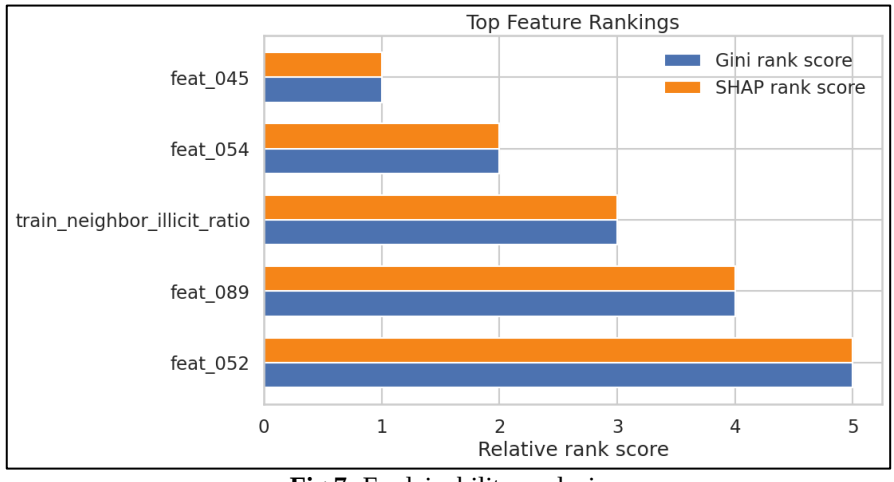

Fig.7: Explainability analysis

Digging into the subgraphs also brought to light some dense clusters of suspicious transactions linked together by directed edges, which looks a lot like people coordinating their actions. It shows that even if global performance jumps around depending on which architecture is running, the intelligence pulled from the graph structure is still highly valuable for spotting local fraud setups and helping teams make better decisions during financial monitoring.

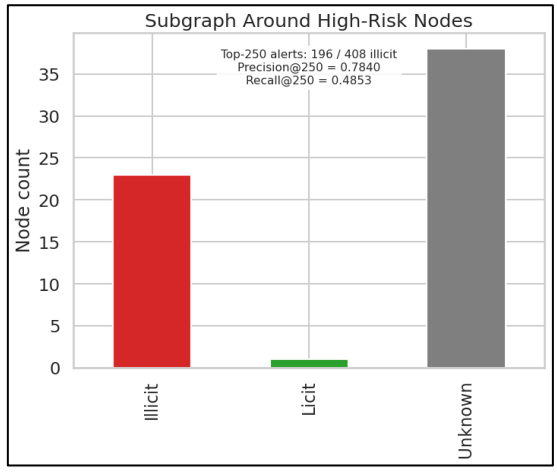

Fig.8: Fraud intelligence analysis

Discussion

Value of knowledge graph enhancement

Bringing knowledge graph structures into financial crime intelligence marks a big shift away from old-school spreadsheet-style machine learning toward models that actually understand connections and network shapes. Using graphs to represent data lets models spot links between transactions that completely disappear when looking at individual data points in isolation. It has been shown that inductive representation learning on big graphs helps models handle completely new nodes while keeping structural connections intact, which fits perfectly with how fast financial systems change [10]. Other work demonstrates that Node2Vec embeddings can capture both close-up and big-picture network features by using biased random walks, offering highly adaptable features for making predictions [9]. Further research proves that graph convolutional networks allow deep learning to work outside standard grids by pulling together info from neighboring nodes, making semi-supervised learning possible on graph data [16]. For tracking financial crime, these breakthroughs mean moving past checking single transactions on their own. Instead, systems can use relational logic to spot illegal activity through weird patterns in the network structure rather than just looking at basic account details. Even so, the actual data from this study shows that while adding graph data brings in great structural insights, the whole thing heavily depends on things staying stable over time and features lining up properly. Graph upgrades help a lot, but they do not automatically beat out a fine-tuned standard tabular setup.

Implications for Financial Crime Intelligence

The results here matter directly for building financial crime systems that look at blockchains and ledger networks. Modeling things with graphs matches up well with how real financial crime happens, since illegal schemes usually come from people working together rather than totally separate actions. Past studies show that applying graph convolutional networks to Bitcoin transaction graphs makes finding money laundering easier because the model uses the shape of the connections between transfers [27]. It is also clear that machine learning can uncover hidden collusion networks by digging into how different entities depend on one another, which underscores why network-focused modeling is so useful for spotting fraud [8]. This idea goes even further when looking at how early warning setups for financial crises get a boost from mixing macro-financial indicators with machine learning that handles systemic connections [23]. Looking at the big picture, all this research points to the need for crime intelligence systems to drop static classification habits. They need to start using relational setups that can see coordinated fraud spreading across a network. Still, the data in this paper shows that graph-based upgrades do not automatically guarantee better accuracy when things change over time. Putting these tools to work in the real world requires keeping a close eye on data drift, how stable the model stays, and whether the network structure holds up as time goes on.

Explainability and Analyst Trust

Being able to explain how a model works is incredibly important when putting machine learning into financial crime systems, especially since these choices have huge consequences and need to be fully auditable. The SHAP framework offers a unified way to figure out predictions using cooperative game theory, giving a reliable look at which features matter most inside complicated models [17]. Using explainable AI in legal and decision-heavy areas builds a lot of trust and clarity because it gives clear reasons behind what the algorithm decided, which is vital for high-stakes tasks like sentencing or risk tracking [18]. In financial crime work, this kind of transparency lets human analysts and regulators double-check the predictions coming out of the system. Using SHAP explanations in this research sheds light on both the specific data points and the structural network factors that make a transaction look illegal, which pulls

back the curtain on some very complex models. However, even the best explanation methods can only show how a model thinks; they cannot fix bad data or hidden biases baked into the dataset itself. Understanding the decisions has to go hand-in-hand with smart modeling and tough validation to actually give analysts data they can rely on.

Knowledge Graph Interpretation

Knowledge graphs are a useful way to organize complicated networks of information, especially when the connections between different things are what matter most for making predictions. A helpful definition from Hogan et al. describes them as structured webs of objects and relationships that make it possible for machines to reason and learn across different fields [11]. Looking at financial systems, Islam et al. showed that graph neural networks can actually model risk by tracking how trouble spreads back and forth between regular banks and crypto markets [13]. That same research team also pointed out that decision systems powered by AI can make financial operations run a lot better by using transaction data to build relational structures in simulated setups [12]. For this specific project, the transaction graph gets treated like a simplified version of a knowledge graph. Every node is a single financial transaction, and the edges show the direct transfer of money from one point to another. It is true that this setup leaves out a lot of the extra labels and varied types of data you might normally see. It still manages to keep the core structural connections that graph-based learning needs to work. The actual results indicate that even a stripped-down graph like this offers plenty of useful clues for spotting financial crime. Building richer graphs with more detailed descriptions could definitely make the results easier to interpret and analyze deeply down the road.

Computational Considerations

When you look at the technical side of things, using graphs to sniff out financial crime adds a lot of extra heavy lifting compared to standard machine learning approaches. Methods like Node2Vec and GraphSAGE do a great job of capturing how the network is shaped, but they demand a ton of computing power because they constantly have to crawl through the graph, map out embeddings, and look at what neighboring nodes are doing. Hamilton et al. noted that while inductive graph learning is built to handle massive networks, the computing overhead is still a major factor when dealing with rapid, high-frequency transactions [10]. Grover and Leskovec mentioned something similar about Node2Vec, explaining that its reliance on endless random walks gets incredibly expensive on a larger scale [9]. This issue pops up a lot in industrial settings too. Alam et al. showed that hybrid deep learning models usually need a lot of fine-tuning to keep a good balance between accurate predictions and computing efficiency, especially when resources are tight or things need to happen in real time [3]. All of this matters immensely for fraud detection systems where speed and scalability dictate whether a tool can actually be used in the real world. The data here suggests that even though graph models give you great insights into structural patterns, the computing costs and the constant need to tweak hyperparameters have to be managed carefully before trying to run these tools inside a massive financial monitoring network.

Limitations

Even though the knowledge graph-enhanced machine learning framework works well for financial crime intelligence, it has several limitations that need to be laid out for transparency and proper context. For starters, the study relies entirely on a single dataset, the Elliptic Bitcoin Dataset. This makes it hard to say if the findings apply to other blockchain ecosystems or traditional banking systems. The dataset is a staple in financial forensics research, but it only looks at Bitcoin transactions. It misses the wider variety of financial crime patterns seen in multi-asset or institutional setups. On top of that, looking only at Bitcoin means the study

misses cross-chain interactions, exchange behaviors, and financial flows that jump across different platforms. Because of this, the model cannot fully mirror real-world financial crime networks, which usually stretch across multiple blockchains and banking setups. The modeling framework also builds everything around individual transactions. Each node represents a single transaction instead of bigger financial entities like actual users, wallets, or institutions. This keeps the graph structure simple, but it strips away semantic detail and makes it harder to understand what individual actors are doing.

Another issue is that the study does not use a semantic ontology or model different kinds of entities. The graph focuses strictly on transactions and leaves out multiple entity types or deep relational meanings that you would normally find in a full-scale knowledge graph. Consequently, the representation learning captures structural shapes but misses deeper meaningful connections between financial players. Cybersecurity threats are also handled indirectly. The model only sees them through the illicit transaction labels already in the dataset, meaning it does not actually learn from specific cybersecurity signals like malware activity, phishing setups, or stolen credentials. Money laundering gets treated the same way. It is inferred through those same illicit labels rather than being mapped out through the actual stages of laundering. Lastly, the framework cannot handle real-time deployment or streaming graph analysis. Everything runs in an offline batch setup. This limits how useful the system would be for live financial monitoring, where models need to learn continuously and catch fraud as it happens. These gaps show what needs to be fixed down the road to make this approach more scalable, realistic, and ready for actual operations.

Future work

Moving forward, research can go in a few different directions to make these knowledge graph-enhanced financial crime intelligence systems more expressive, scalable, and practical. The first big step is building heterogeneous knowledge graphs that include more than just transactions. Adding wallets, users, devices, IP addresses, exchanges, and smart contracts would give a much clearer picture of the financial ecosystem. This change would let models pick up on richer relationships and make results easier to interpret at the actor level instead of just looking at transactions. It will also be important to bring in dynamic graph neural networks that can handle time and show how financial networks change over days or months. Models like Temporal Graph Attention Networks (TGAT), Temporal Graph Networks (TGN), and EvolveGCN offer ways to learn from graph structures that never stop shifting. Using these tools would help catch time-based patterns in illegal behavior, making the system sturdier when data shifts and more reliable in real-world settings.

Another area worth exploring is explainable graph neural network architectures to make financial decisions easier to see through. Tools like GNNExplainer and GraphSHAP can pull out local and global explanations for why a model made a certain prediction. This helps financial analysts understand the exact structural or feature-based reasons behind a fraud alert, which goes a long way in building trust and meeting regulatory standards. Blending cyber-financial intelligence signals right into the modeling framework is another promising path. This means pulling in external threat feeds, malware indicators, phishing reports, and compromised wallet data. Mixing blockchain data with cybersecurity intelligence gives a single, unified view of both financial and cyber threats, allowing for quicker detection of coordinated attacks across digital platforms.

Building real-time financial intelligence systems is just as critical. This requires setting up streaming analytics and online learning systems so the models can update on the fly as new transactions roll in. Systems like this are necessary for fast-paced financial environments where

waiting too long to catch fraud leads to major losses. Finally, expanding the framework to cover cross-chain intelligence is a major goal. Future setups need to pull in multiple blockchain ecosystems, like Ethereum, stablecoins, and decentralized finance platforms, to track crime across chains. Doing this would expose illegal patterns that jump between platforms, which stay completely hidden in a single-chain analysis, ultimately making financial crime intelligence much more effective.

Conclusion

This study presented a knowledge graph-enhanced machine learning framework for financial crime intelligence in distributed ledger environments, with the objective of improving the detection of illicit transactions through the integration of graph-based representations and advanced learning techniques. Using the Elliptic Bitcoin Dataset, a transaction-centric knowledge graph was constructed to capture structural relationships among Bitcoin transactions, enabling the investigation of conventional machine learning models, graph-derived feature engineering, Node2Vec embeddings, and a directed GraphSAGE architecture within a unified experimental framework. To ensure realistic evaluation, a strict temporal validation strategy was adopted, thereby reducing information leakage and providing a more reliable assessment of model generalization under future transaction scenarios.

The experimental findings demonstrate that graph-based information contains valuable structural signals for financial crime intelligence, particularly through neighborhood characteristics and transaction connectivity patterns. However, the results also reveal that graph-enhanced methods do not automatically outperform strong tabular baselines under temporally shifted conditions. Among the evaluated approaches, the tabular XGBoost model achieved the strongest overall predictive performance, while Node2Vec and graph-derived features provided competitive results and contributed additional structural insights. Although the Directed GraphSAGE model successfully captured relational dependencies during training, its reduced temporal generalization highlights the challenges posed by distribution shifts in evolving financial systems. These findings emphasize that the value of knowledge graph enhancement lies not solely in predictive accuracy but also in its ability to expose relational structures and support intelligence-driven analysis.

Beyond classification performance, the study demonstrated the importance of explainability and graph-based intelligence mechanisms in supporting operational financial investigations. Feature attribution and suspicious subgraph analysis provided interpretable insights into transaction behavior, enabling risk prioritization and facilitating analyst understanding of complex transaction networks. Consequently, the proposed framework contributes not only to illicit transaction detection but also to the broader objective of transforming financial crime analytics from isolated prediction tasks into relationship-aware intelligence systems. Overall, this research provides a reproducible and practically oriented framework that bridges conventional machine learning and graph representation learning for financial crime intelligence. The findings highlight both the opportunities and challenges associated with knowledge graph-enhanced analytics and underscore the importance of temporal robustness, explainability, and structural reasoning in next-generation financial monitoring systems. As digital financial ecosystems continue to expand and criminal activities become increasingly interconnected, graph-based intelligence frameworks are expected to play a central role in the development of scalable, transparent, and proactive solutions for combating financial crime across distributed ledger infrastructures.

References

Aashish, K. C., Zamil, M. Z. H., Mridul, M. S. I., Akter, L., Sharmin, F., Ayon, E. H., ... & Malla, S. (2025). Towards eco-friendly cybersecurity: Machine learning-based anomaly detection with carbon and energy metrics. International Journal of Applied Mathematics, 38(9s).

Al Wahid, S. A., Ghos, T. C., Sajal, A., Akter, A., Faysal, A. H. M., Hasan, A., ... & Shawon, R. E. R. (2026). AI-driven dynamic reorder point prediction for inventory management in supply chains: A machine learning approach. International Journal of Special Education, 41(4s), 441–461.

Alam, M., Shil, S. K., Sharmin, F., KC, A., Md, A. H., Ali, M., ... & Malla, S. (2026). Hybrid deep learning models for equipment failure prediction in US industrial systems. International Journal of Applied Mathematics, 39(1s).

Bhowmik, P. K., Subha, D. T., Rahim, A., Mohammed, A. A., Begum, M., Chowdhury, R., ... & Shati, M. A. Self-adaptive machine learning models for financial risk forecasting: Handling non-stationarity in banking and cryptocurrency time series.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

Conti, M., Kumar, E. S., Lal, C., & Ruj, S. (2018). A survey on security and privacy issues of Bitcoin. IEEE Communications Surveys & Tutorials, 20(4), 3416–3452.

Dola, A., Begum, S., Antara, U. K., Islam, M. R., Sultana, T., & Zabin, N. (2024). Machine learning models for detecting hidden collusion networks in US corporate finance. Journal of Economics, Finance and Accounting Studies, 6(1), 143–154.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 855–864.

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in Neural Information Processing Systems, 30, 1024–1034.

Hogan, A., Blomqvist, E., Cochez, M., et al. (2021). Knowledge graphs. ACM Computing Surveys, 54(4), 1–37.

Islam, M. R., Subha, D. T., Pramanik, M. T., Akter, M., Sweet, M. M. R., Robbani, M. S., ... & Zeeshan, M. A. F. AI-driven decision support systems for optimizing working capital and customer experience in the US: A transaction-based simulation framework for SMEs.

Islam, M. Z., Sumsuzoha, M., Islam, M. R., Kawsar, M., Mithu, M. F. H., Pant, S., ... & Al Helal, M. A. Graph neural networks for systemic financial risk forecasting: Modeling cross-market contagion between banking systems and cryptocurrency markets.

Jakir, T. (2025). Signal-to-noise analysis of crisis indicators in global finance using artificial intelligence. International Journal of Applied Mathematics, 38(10s), 1815–1836.

Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. Patterns, 4(9), 100804.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. International Conference on Learning Representations.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 4765–4774.

Miah, M. N. I., Uddin, M. J., & Kakumani, M. (2026a). Artificial intelligence in sentencing: Evaluating machine learning models for sentencing recommendations in the US. Frontiers in Computer Science and Artificial Intelligence, 5(4), 30–43.

Miah, M. N. I., Uddin, M. J., & Kakumani, M. (2026b). Artificial intelligence in crime scene reconstruction: Using machine learning for predictive analysis and scenario simulation. Frontiers in Computer Science and Artificial Intelligence, 5(8), 43–57.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. Decision Support Systems, 50(3), 559–569.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online learning of social representations. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 701–710.

Rahman, M. K., Hossain, S. U., Haque, S. U., Jahed, K. A., Robbani, S. M., Shati, M. A., & Pramanik, M. T. Machine learning models for early warning of financial crises in the US economy using macro-financial indicators.

Rahman, M. S. (2025). Machine learning–enabled early warning system for detecting micro-inflation clusters in the US economy. International Journal of Applied Mathematics, 38(12s), 2743–2769.

Reza, S. A., Rahman, M. K., Rahman, M. D., Sharmin, S., Mithu, M. F. H., Hasnain, K. N., & Kabir, R. (2025). Machine learning enabled early warning system for financial distress using real-time digital signals. arXiv preprint arXiv:2510.22287.

Shawon, R. E. R., et al. (2025). Detecting illicit cross-chain fund movement: Behavioral machine learning models for bridge-based laundering patterns. International Journal of Applied Mathematics, 38(12s).

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., & Leiserson, C. E. (2019). Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics. KDD Workshop on Anomaly Detection in Finance.