Graph Neural Networks for Supply Chain Fraud Detection with Explainability via Saliency Maps and RL-based Feature Masking

Aieh Eissa
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
g00107537@aus.edu

Tsion Regasa
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
g00107807@aus.edu

***Abstract*—Supply chain fraud exhibits complicated relational patterns that traditional machine learning techniques frequently fail to detect. This paper presents a heterogeneous graph neural network (GNN) framework for detecting fraudulent activity across consumers, orders, and items. The study employs a synthetic supply chain dataset, with fraud occurrences identified using a set of predefined rule-based heuristics. The data is used to create a tripartite graph that represents the customer-order-product relationships. A two-layer GraphSAGE model is used for fraud classification, with edge-type-specific message passing. To improve interpretability, the model includes saliency maps for global feature relevance and a reinforcement learning-based strategy to select sparse, local feature subsets associated with specific predictions. The resulting model has a macro-averaged F1-score of 88.28% and shows consistent accuracy across class distributions. The combined predictive performance and interpretability close significant gaps in previous graph-based fraud detection techniques.**

***Index Terms*—Supply chain, Fraud detection, Graph Neural Networks, Heterogeneous Graphs, Explainability, Saliency Maps, Reinforcement Learning,**

I. Introduction

Fraudulent behavior in supply chain operations is often subtle and deeply embedded within complex relationships between entities such as customers, orders, and products. Traditional fraud detection systems, which typically rely on flat, rule-based algorithms or independent transaction-level analysis, struggle to uncover such hidden patterns.

These models treat each data point in isolation, ignoring the rich relational structure that characterizes real-world supply chains and limiting their ability to detect coordinated or indirect fraud.

Recent advancements in Graph Neural Networks (GNNs) offer a promising alternative by enabling learning over structured relationships and multi-entity interactions. However, many existing GNN-based fraud detection systems either simplify heterogeneous graphs into homogeneous formats, leading to loss of critical semantic information, or operate as black-box models, providing limited interpretability. For instance, while models like xFraud incorporate explainability mechanisms, they may not fully capture the complexities inherent in heterogeneous graphs [1]. Indeed, a gap remains in current research at the intersection of heterogeneous graph modeling and interpretable fraud detection, particularly in supply chain contexts.

In this work, we aim to address both structural and interpretability challenges. We formulate fraud detection as a node-level binary classification task on a heterogeneous graph composed of customers, orders, and products, with directed edges capturing transactional relationships. This research uses the DataCo Supply Chain dataset from Kaggle [2], and to overcome the lack of labeled fraud data, we introduce a rule-based simulation mechanism that generates suspicious cases using domain heuristics.

Moreover, our framework emphasizes not only performance but also explainability. We integrate gradient-based saliency analysis and a reinforcement learning-based explainer that attributes predictions to sparse, task-relevant features across all connected node types. This design enables interpretable, entity-aware reasoning within a realistic supply chain setting and help understand which parts of the graph or which features contributed to a specific prediction, making the model more transparent and practically useful. To assess the effectiveness of each method, we also provide a brief comparative analysis, highlighting their differences in sparsity, focus, and interpretability across node types.

This paper contributes a reproducible and interpretable GNN-based pipeline for fraud detection in heterogeneous supply chain graphs, achieving both high predictive accuracy and transparent, feature-level insights into model decisions.

By combining gradient-based saliency and a reinforcement learning-based explainer, our framework provides complementary perspectives on model behavior, addressing a key shortcoming of existing explainability tools, which often struggle with heterogeneous graph structures. This dual-layer approach enables practical interpretability while preserving relational complexity, bridging a critical gap in current fraud detection research.

Explainer	Type	Hetero Support	Key Strengths	Limitations
GNNExplainer [3]	Node-level, Subgraph	No	Simple and provides node and feature attribution	does not scale well to large graphs
PGExplainer [4]	Model-level, Policy-based	No	Learns a probabilistic mask	approximation quality depends on training
RelEx [5]	Path-based, Relational	Yes	Supports heterogeneous graphs; produces symbolic rule-based explanations	Requires symbolic reasoning layer; not widely integrated into major GNN libraries
SubgraphX [6]	Shapley-based, Subgraph	No	Provides precise, high-fidelity explanations using Monte Carlo Shapley values	Computationally expensive; limited to small graphs

TABLE I
COMPARISON OF SELECTED GNN EXPLAINABILITY TOOLS.

II. Related Work

Recent advancements in supply chain fraud detection have increasingly turned toward graph-based models, particularly Graph Neural Networks (GNNs), due to their ability to represent complex, multi-entity relationships. Unlike traditional or deep learning models that treat transactions independently or sequentially, GNNs allow for the modeling of rich interactions between entities such as customers, orders, and products.

This is especially useful in fraud detection, where suspicious activity is often revealed through contextual clues across these interactions rather than in isolated features.

Multiple studies have explored this potential. Gandhi et al. [7] applied a Graph Convolutional Neural Network (GCNN) to model fraud within supplier-transaction networks, outperforming CNN and LSTM baselines with an F1-score of 91.12% and an AUC of 97.35%. This model used a homogeneous graph structure, treating all nodes and edges similarly.

Wasi et al.[8] extended this direction using Temporal GNNs to detect anomalies in FMCG supply chains, outperforming traditional models by up to 30% in scenarios involving time-sensitive delays or abnormal demand spikes. Their model also operated on a heterogeneous graph to capture evolving behaviors across different entity types.

Attention mechanisms have also been integrated with GNNs to further enhance their effectiveness. For example, Xie et al.[9] introduced DAST-GNN, a spatio-temporal attention model for financial fraud in supply chains, achieving an AUC of 93.64% and F1-score of 89.72%. Suresh et al.[10] combined Heterogeneous Graph Attention Networks (HGAT) with Latent Dirichlet Allocation (LDA), creating a dual-attention pipeline that achieved 96.34% sensitivity and a 93.76% F1-score.

Furthermore, Wu et al. [11] developed MultiFraud, a heterogeneous multitask GNN that includes a built-in explainer, delivering AUC scores above 0.92 across multiple fraud scenarios. All three of these models used heterogeneous graphs to preserve the structural diversity of supply chain data.

Cui et al. [12] integrated reinforcement learning with GNNs in FraudGNN-RL to adaptively explore fraud graphs, reaching a 97.3% F1-score and cutting false positives by 31%. This model also employed a heterogeneous graph structure. However, despite the strong performance of these models, most of them either lack interpretability or are difficult to replicate due to the use of proprietary datasets.

Explainability remains a major challenge in GNN-based fraud detection. Most popular explainability tools were designed for homogeneous graphs and do not support the complexity of real-world supply chain networks as seen in Table I. Tools like GNNExplainer [3] and PGExplainer [4] are widely used but struggle with heterogeneous graphs involving multiple node and edge types.

Newer methods like RelEx [5] are more compatible with heterogeneous graphs but rely on symbolic reasoning components that are not fully integrated into standard libraries such as PyTorch Geometric, limiting their accessibility and practical adoption.

III. Methodology

A. Data Preprocessing

The dataset used is the Kaggle DataCo supply chain dataset [2]. It is a synthetic but realistic dataset that simulates the operational data of a large-scale supply chain. In total, it contains 18,942 rows and originally 53 columns. Each row represents a specific product within a customer's order.

The dataset contains a wide variety of information, including customer details, product names and categories, financial metrics like sales and profit, order and shipping dates, geographic and segmentation data, and pricing-related attributes.

1) Feature Selection and Cleaning: To focus the dataset on transaction-relevant attributes, the preprocessing stage began by eliminating columns deemed irrelevant, redundant, or sensitive. These included personally identifiable fields (e.g., customer email), multimedia content (e.g., product images), and administrative identifiers (e.g., department ID) that offered little modeling value.

Geolocation features were also dropped due to incompleteness and lack of alignment with the graph structure. Overall, this filtering step significantly reduced the number of columns and helped retain only those features directly related to customer behavior, order characteristics, and product metadata.

2) Handling Missing Values: Any rows missing essential identifiers such as `order id`, `customer id`, `product`

name, or product price were completely removed because their absence would break graph connectivity. For missing numerical columns, the values were filled using the median of each column to preserve distribution and reduce skew.

For categorical features, missing values were filled with the placeholder Unknown to ensure that all fields remained usable in modeling and feature encoding.

3) Standardization and Deduplication: To standardize textual categories, key columns such as Customer Segment, Product Name, Shipping Mode, Market, and Order Region were normalized by converting all entries to lowercase and removing any whitespace. This avoided mismatches caused by case or formatting differences (e.g., "late delivery" vs. "Late Delivery"). The dataset was also scanned for exact duplicate rows, but none were found, each row was unique. This verification ensured data integrity before graph construction.

4) Temporal Feature Engineering: The dataset includes both order and shipping dates, found in the Order Date (DateOrders) and Shipping Date columns. These were first converted into datetime format to enable temporal analysis. A new feature called Shipping Delay was created by subtracting the order date from the shipping date, yielding the number of days it took for a product to be shipped after being ordered. This delay is useful for identifying suspicious patterns such as operational irregularities or intentional manipulation.

In addition, further temporal features were extracted from the order date, specifically the day of the week, month, and hour of the order. These variables help capture behavioral patterns, such as whether fraudulent orders tend to occur during weekends, at specific times of day, or near the end of fiscal months.

5) Derived Risk Features: Several new features were engineered to capture subtle risk signals. The Profit Margin was calculated by dividing Order Profit Per Order by Sales, indicating the percentage of profit earned per dollar sold. Unit Discount was created by normalizing the total discount by the number of units ordered.

Two binary flags were also introduced: one for high-discount orders (discount rate $>$ 50%), and another for large orders (quantity $>$ 10). These flags served as simple heuristics helpful for both fraud labeling and downstream modeling.

6) Fraud Labeling Strategy: Since the original dataset lacked ground-truth fraud annotations, a rule-based labeling strategy was implemented to simulate realistic fraudulent transactions. Each row was initially assigned a fraud label of 0, indicating non-fraudulent behavior.

Then, a set of heuristic rules, motivated by common supply chain risk patterns, was applied to identify suspicious cases and update the fraud label to 1. Specifically, transactions were marked as fraudulent if any of the following conditions were met:

- Excessive discounting: The Order Item Discount Rate exceeded 80%.
- Delivery mismatch: Orders flagged with Late_delivery_risk but marked as Delivered on Time.
- Profit-risk inconsistency: Transactions with Order Profit Per Order $<$ 0 while Sales $>$ 500.
- Unreasonable delay: Shipping delays longer than 30 days.
- High-discount bulk orders: Transactions marked with both is_high_discount = 1 and is_large_order = 1.
- Mid-tier discount with quantity: Discount rate over 50% and quantity greater than 3.
- Low margin with high value: Profit margin below 10% and Sales above 400.
- Delayed and flagged: Shipping delay over 20 days and flagged with Late_delivery_risk.
- Expensive discounted items: Discount rate above 30% and Order Item Total above 1000.
- Pending and risky: Orders in Pending status combined with Late_delivery_risk.

These conditions collectively capture a broad range of suspicious behaviors, including delivery manipulation, excessive discounting, profitability mismatches, and operational delays. Applying this rule set resulted in 13,057 transactions being labeled as fraudulent out of 180,519, yielding an approximate fraud rate of 7.2%. This binary label (fraud_label) served as the ground truth for supervised learning.

7) Feature Scaling: After labeling, numerical features were scaled using StandardScaler to normalize their distributions. Key columns included Sales, Order Profit Per Order, Shipping Delay, Order Item Discount Rate, Order Item Quantity, Order Item Total, Unit Discount, and Profit Margin. These were first cleaned of infinite or NaN values and then standardized to zero mean and unit variance. The scaled versions were stored as new columns (e.g., sales_scaled), preparing them for use in the GNN model.

B. Graph construction

To accurately represent the supply chain's environment and structure, three distinct types of nodes were used: customers, orders, and products. This tripartite layout closely mirrors real-world supply chain operations, where a customer places one or more orders, and each order includes one or more products.

Rather than flattening these into a homogeneous structure, this method preserves the semantics of the supply chain. From the dataset, 2,072 unique customers, 9,351 unique orders, and 1,859 distinct products were extracted using the Customer Id, Order Id, and Product Name columns.

Two sets of directed edges were created: one from each customer to their orders, and another from each order to its associated products. This structure captures both the transactional flow and the multi-entity relationships present in supply chain behavior.

In total, the graph contained approximately 17,000 transaction-level edges after filtering out records with missing

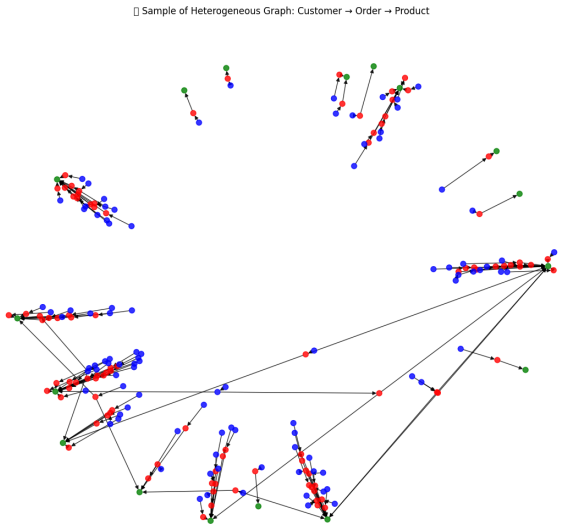

Fig. 1. Heterogeneous graph structure modeling the relationship between customers, orders, and products.

or invalid values. Each edge was enriched with normalized features including quantity, discount rate, sales, profit, shipping delay, and profit margin.

Node features were incorporated based on available structured data. For customers, attributes such as customer segment, market, and region were encoded using one-hot encoding to preserve categorical relationships. For orders, temporal features like order day, hour, and shipping delay were extracted from date fields and standardized.

Product nodes included information like category and subcategory, where categorical values were also processed using one-hot encoding, and numeric features like unit price were scaled. To build the graph, Customer Id, Order Id, and Product Name were all label encoded to assign unique integer identifiers to each node, which are required for efficient graph processing.

While using a heterogeneous graph introduces additional modeling complexity, it preserves critical distinctions between different entity types and allows the model to learn from structurally meaningful relationships across the supply chain. Rather than collapsing all entities into a single node type, this approach maintains the separation between customers, orders, and products, enabling the model to capture the context in which transactions occur.

For example, fraudulent behavior may not be evident when looking only at a product or a customer in isolation, but patterns can emerge when observing how a customer places orders and which products are repeatedly involved. Although this richer structure limits compatibility with most standard explainability tools, which are typically designed for homogeneous graphs, it more faithfully represents the complexity of real-world supply chains.

As a result, the model is better positioned to detect fraud by leveraging patterns that span across multiple entities and transaction layers, rather than relying solely on individual data points.

C. Model Architecture

The model was implemented using PyTorch Geometric and was based on a heterogeneous GraphSAGE architecture designed to operate on a graph composed of three distinct node types: customers, orders, and products.

GraphSAGE was selected because of its ability to perform inductive learning, by aggregating feature information from a node's local neighborhood, which is important in fraud detection where contextual patterns can span across multiple entities. Rather than flattening the graph into a homogeneous structure, this method preserved the natural hierarchy of the supply chain, where customer behavior, order timing, and product characteristics all interact.

The graph followed a heterogeneous format, with edges linking customers to the orders they placed and orders to the products they contained. To support bidirectional message passing, reverse edges were added from products back to orders. These relationships were handled using HeteroConv, a module that applies distinct SAGEConv layers to each edge type.

The architecture consisted of two layers. In the first layer, information was passed from customers to orders and from orders to products. The second layer aggregated messages from products back to orders, allowing the model to capture secondary relationships and feedback from product characteristics. This structure helped the model recognize patterns that wouldn't be visible from a single interaction or isolated transaction.

The fraud label was associated with the order nodes, and the model's task was to classify whether a given order was fraudulent based on information received through its connected customer and product nodes. After pre-processing and filtering, the graph contained around 17,000 transaction-level edges and over 15,000 nodes across the three entity types.

D. Training Setup

For training, the data was split into training, validation, and test sets using a 75-15-15 ratio. Because only 7.24% of transactions were labeled as fraud, stratified sampling was used to maintain consistent class distributions across all splits.

The model was trained using binary cross-entropy loss with class weighting to account for the imbalance. The weights were computed using the ratio of non-fraud to fraud labels in the training set to ensure that the model didn't become biased toward the dominant class.

Training was performed over 100 epochs using the Adam optimizer with a learning rate of 0.005 and a weight decay of 5e-4. The model was evaluated every five epochs on the validation set to monitor convergence and generalization. A class-weighted binary cross-entropy loss was used to address class imbalance, ensuring that the model remained sensitive to the minority fraud class.

Although no early stopping was applied, the regular validation checks and balanced loss function helped promote robust performance across both classes. If present, dropout was used to further reduce overfitting.

This setup helped ensure that the model stayed generalizable while still detecting subtle and rare fraud cases. Because of the class imbalance and the real-world importance of correctly identifying fraudulent transactions, special attention was given to maintaining sensitivity to false negatives.

E. Evaluation Metrics

Given the imbalanced nature of fraud detection tasks, where fraudulent instances represent a small fraction of all transactions, evaluating model performance requires metrics that go beyond overall accuracy. The following metrics are used to assess the classification quality of the model:

- **Precision**: Measures the proportion of predicted positive (fraudulent) instances that are actually fraud. High precision indicates a low false positive rate, which is important to avoid flagging legitimate transactions incorrectly.
- **Recall**: Measures the proportion of actual fraudulent cases that are correctly identified by the model. High recall ensures that most fraud cases are detected, minimizing false negatives.
- **F1-Score**: The harmonic mean of precision and recall. It provides a balanced evaluation when both false positives and false negatives are important. This is especially useful in fraud detection, where both types of errors carry a cost.
- **Accuracy**: Represents the overall proportion of correct predictions. However, in imbalanced datasets, accuracy can be misleading, as the model might achieve high accuracy simply by favoring the majority (non-fraud) class.
- **Macro-Averaged Metrics**: These compute the metric independently for each class and then take the average, giving equal weight to both fraud and non-fraud classes. Macro averaging is critical in imbalanced settings because it prevents the dominant class from overshadowing the minority class performance. It is particularly emphasized in this work to fairly assess the model's ability to detect fraudulent orders.
- **Weighted-Averaged Metrics**: These average metrics across classes weighted by their support (number of instances). While they provide a more holistic picture of overall performance, they are more influenced by the majority class and may obscure minority class behavior.

This combination of metrics ensures a comprehensive evaluation of the model's performance, with a particular focus on its effectiveness in identifying rare but costly fraud cases.

F. Explainability via saliency maps

To better understand which features contributed most to the model's fraud predictions, saliency-based explainability was incorporated into the pipeline. Traditional explanation tools such as GNNExplainer and PGExplainer were considered but found to be incompatible with the heterogeneous nature of the graph used in this study. These methods are primarily designed for homogeneous graphs and could not adequately handle multiple node types and edge relations without significant architectural simplification.

To preserve the model's structure while still enabling interpretation, a gradient-based saliency approach was adopted. Saliency analysis measures the sensitivity of the model's output with respect to each input feature, providing a transparent, post-hoc explanation of the model's decision-making process. In this case, we computed the gradient of the predicted fraud score with respect to the feature vectors of order, customer, and product nodes.

The analysis focused on order nodes in the validation set that were predicted as fraud. For each such order, gradients were tracked not only on the order features but also on its connected customer and product nodes. By averaging these gradients across 50 high-confidence fraud predictions, we generated saliency maps for each node type.

This method of explanation provided both global and local interpretability. Globally by identifying which features the model relied on across several predictions, and locally by emphasising what mattered in individual cases. The goal here was not to assess model performance, but to gain a deeper understanding of its internal reasoning.

More importantly, saliency analysis was applied without modifying the model architecture or requiring a simplified graph representation, preserving the heterogeneous design while adding transparency. These insights reinforced the evaluation results by showing that the model's decisions were not only accurate but also traceable to meaningful patterns in the data.

G. Explainability via Reinforcement learning

To extend the model's interpretability beyond saliency, we implemented a reinforcement learning–based feature explainer. Unlike gradient-based methods that rely on local sensitivity, this approach directly interacts with the trained GNN model to probe its decision-making process.

The explainer does not operate in isolation; instead, it learns to approximate the internal reasoning of the GNN by observing how the model's output changes when selective information is masked. Specifically, it is trained to identify minimal subsets of input features from each node type, namely, order, customer, and product, that are sufficient for the GNN to maintain its original prediction.

Moreover, the explainer receives as input the averaged feature vectors from an order node and its directly connected customer and product nodes. It outputs a probability distribution over each feature set, which is sampled using Bernoulli distributions to produce binary masks. These masks are then applied to the node features during a forward pass of the original, frozen GNN model.

The reward signal is computed as the negative binary cross-entropy between the GNN's masked prediction and its original prediction, effectively encouraging the explainer to preserve decision fidelity under sparse inputs.

This creates a feedback loop where the explainer continuously queries the GNN and adjusts its masking strategy to minimize information loss. A sparsity regularization term is added to the loss to penalize large masks and promote concise, interpretable explanations. The agent is trained using the Adam optimizer for 100 epochs per node. To generalize insights, the agent was applied to ten high-confidence fraud predictions from the validation set.

This reinforcement-based feature masking framework is fully integrated with the GNN. Furthermore, it does not modify the GNN architecture or retrain it, but instead leverages the trained model's outputs as a supervisory signal. This allows us to derive post hoc explanations that are faithful to the GNN's learned behavior, enhancing transparency in a multi-entity fraud detection setting where trust in automated decisions is critical.

IV. Results and Evaluation

A. GNN Predictive Performance

To evaluate the effectiveness of our heterogeneous GNN model in detecting fraudulent orders, we trained the network on 70% (5947 nodes) of the order nodes, validated on 15% (1274 nodes), and reserved the remaining 15% (1275 nodes) for final testing. The model was optimized using a class-balanced binary cross-entropy loss to account for the natural class imbalance in the dataset, where legitimate transactions significantly outnumber fraudulent ones.

During training, the model demonstrated steady improvements in validation performance. Early epochs showed poor recall and F1-score for the fraud class, reflecting the difficulty of learning subtle fraud signals in the presence of dominant non-fraud examples. However, by epoch 100, the validation F1-score for fraudulent orders had improved to 0.6982 from 0.0467 at epoch 1, and the overall validation accuracy reached 96.00%. This indicates that the model successfully learned both structural and feature-based fraud indicators over time.

The final evaluation on the test set confirms the model's ability to generalize to unseen data. The quantitative evaluation of the model's performance is summarized in Table II and Table III.

Table II presents class-wise metrics, showing that the model achieves a high F1-score of 78.11% for the fraud class and 98.45% for the non-fraud class. This indicates that the model is capable of correctly identifying the majority of fraudulent orders while maintaining excellent performance on legitimate transactions.

Table III reports overall metrics, with an accuracy of 97.10%, a macro-averaged F1-score of 88.28%, and a weighted F1-score of 96.96%. These results confirm the model's ability to maintain strong prediction quality across an imbalanced dataset and validate the effectiveness of using a heterogeneous graph structure for fraud detection.

These results reflect the model's ability to correctly identify a high proportion of fraudulent orders, despite their rarity. The recall of 70.97% for the fraud class means that the model successfully detected nearly 71% of all fraudulent transactions in the test set. This is particularly significant in fraud detection, where false negatives (missed frauds) can have high real-world consequences. The precision of 86.84% indicates that the majority of flagged fraudulent cases are indeed correct, reducing the burden of false alarms.

TABLE II
Class-wise Performance on Test Set

Class	Precision	Recall	F1-score	Support
Non-Fraud (0)	97.75%	99.15%	98.45%	1,182
Fraud (1)	86.84%	70.97%	78.11%	93

TABLE III
Aggregate Metrics on Test Set

Metric	Score
Accuracy	97.10%
Macro Avg F1	88.28%
Weighted Avg F1	96.96%

The F1-score, defined as the harmonic mean of precision and recall, is particularly important in fraud detection scenarios where the cost of false positives and false negatives is high and class imbalance is severe.

A model with high precision but low recall might miss many actual fraud cases (false negatives), while one with high recall but low precision may raise too many false alarms (false positives). The fraud-class F1-score of 78.11% indicates that the model strikes a strong balance, accurately identifying a big proportion of fraudulent transactions while maintaining relatively few false positives.

The macro-averaged F1-score of 88.28% further confirms that the model performs well across both classes, not just the majority class. This metric averages the F1-score of each class equally, regardless of class size, making it a reliable indicator in imbalanced classification tasks like this one. Meanwhile, the weighted F1-score of 96.96%, which accounts for class proportions, shows that overall prediction quality remains high across the full dataset.

These results validate our modeling approach and demonstrate that representing the supply chain as a heterogeneous graph allows the model to effectively reason over customer–order–product relationships. The model successfully learns not only from local node features but also from relational context, resulting in robust fraud detection performance that exceeds what is typically achievable using traditional tabular or flat graph-based approaches.

Crucially, these performance levels were achieved without compromising interpretability, which is addressed in the following sections through our two-layer explanation framework.

B. Saliency-Based Feature Attribution

To gain insight into which features influenced the GNN's predictions, we conducted a gradient-based saliency analysis. As shown in Figure 2, the most salient order-level feature was `shipping_delay_scaled`, followed

by Order Profit Per Order_scaled and Order Item Discount_scaled. These results suggest that fraudulent transactions are often associated with unusual delivery timelines or inflated profit margins. A high shipping delay may indicate logistical irregularities common in fraudulent fulfillment, while high per-order profits might reflect pricing manipulations or fake large-volume purchases. The prominence of discount-related features also implies that fraudsters may exploit discount mechanisms, either through coupon abuse or internal price overrides.

Notably, more conventional features like Order Item Quantity_scaled, Order Month, and Order Hour were among the least salient. This suggests that temporal and volume patterns, though commonly used in rule-based systems, were less informative for the learned GNN model. Instead, the model focused on subtle behavioral traits not easily detectable by static thresholds.

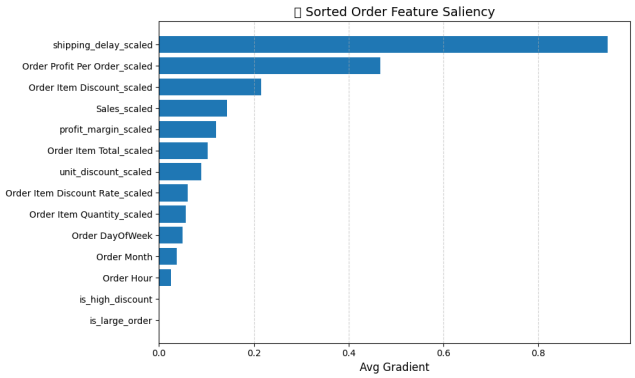

Fig. 2. Sorted feature saliency for order nodes. Delivery delay and profit features dominated.

Customer node saliency, visualized in Figure 3, shows that regional identifiers were the most influential. Features like Market_latam, Market_pacific asia, and Market_usca ranked highest, along with country-specific attributes such as Customer Country_ee. uu. and Customer Country_puerto rico. This pattern suggests that fraudulent behavior may correlate with specific geographic clusters, either due to region-specific fraud schemes, regulatory loopholes, or differing consumer behaviors.

Additionally, features like Order Item Quantity_scaled and Sales_scaled surfaced among the top attributes, hinting at a behavioral signal related to purchase intensity or spending habits.

Interestingly, some customer segment tags like Customer Segment_consumer and Customer Segment_corporate also contributed meaningfully, potentially reflecting differing risk profiles between individual buyers and organizational accounts, with consumers being prevalent in fraudulent orders.

For product nodes, Figure 4 presents the top 15 most salient features. Category Name_accessories and Category Name_electronics topped the list, followed closely by Category Name_golf balls, Category

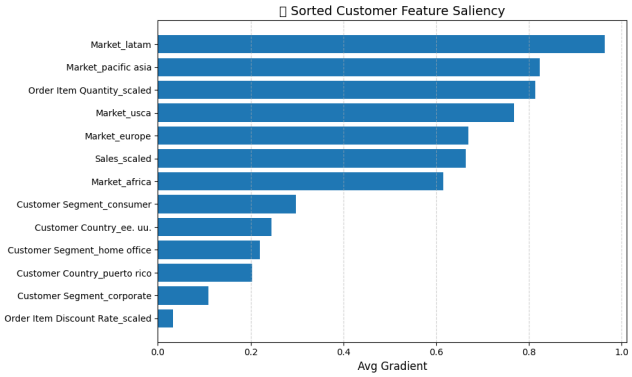

Fig. 3. Sorted feature saliency for customer nodes. Region and segment information had the strongest influence.

Name_books, and Category Name_children's clothing. These results suggest that fraud is often concentrated in specific categories, possibly due to high resale value, frequent promotional cycles, or ease of disguising fraudulent behavior among common purchases.

Interestingly though, many product category features exhibited moderately high saliency scores without extreme separation. This pattern suggests that while product type contributes to fraud detection, it may not serve as a strong discriminator on its own. Instead, most categories carry some predictive weight, likely due to the model learning generalizable patterns of fraud risk across diverse product types.

Notably, the inclusion of Order Item Total_scaled within the top product features reaffirms that transaction value remains a key factor in the model's assessment, especially when linked to high-risk item types.

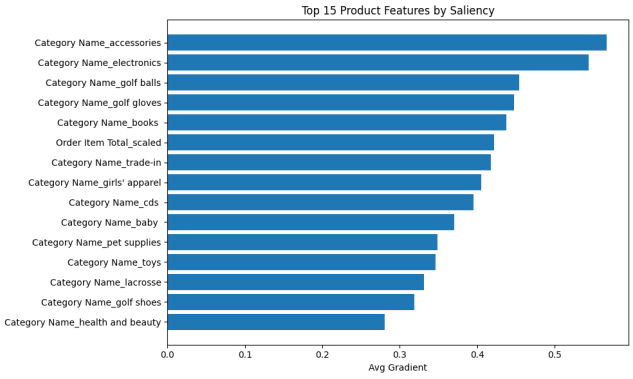

Fig. 4. Top 15 product features by average saliency score. Accessory and electronics categories were especially indicative of fraud.

Collectively, these findings indicate that the GNN does not rely on obvious indicators like time or item count alone. Instead, it picks up nuanced interactions between region, category, pricing, and logistics. This saliency-based analysis highlights the value of explainable AI in surfacing actionable fraud insights, showing that the model is capturing meaningful signals aligned with real-world fraud typologies, while avoiding overreliance on generic or noisy attributes.

C. RL-Based Explainability Agent

To complement gradient-based saliency with a more discrete and interpretable explanation mechanism, we applied a reinforcement learning (RL)-based explainer that learns sparse feature masks for the fraud prediction task. The explainer agent is trained to identify the minimal set of features across the *order*, *customer*, and *product* nodes that can preserve the GNN's prediction for high-confidence fraudulent orders in the validation set. Importantly, it does so without modifying the underlying GNN architecture.

Across ten selected validation nodes, the agent was trained using REINFORCE with a reward function composed of prediction fidelity and a sparsity regularization term. During training, the average total number of active features per node type steadily decreased, demonstrating the agent's ability to isolate relevant features. Final mask sums ranged between 3–11 for orders, 2–11 for customers, and 18–33 for products, showing meaningful compression across diverse node types.

The averaged feature importances reveal the features the RL agent deemed most influential for fraud detection. For the **order** node type, the most important features included `Order Hour`, `is_large_order`, and `shipping_delay_scaled`, all consistent with plausible fraud indicators such as off-hour purchases or large but delayed orders.

For **customer** nodes, the RL explainer highlighted `Customer Country_puerto rico`, `Market_pacific asia`, and `Order Item Discount Rate_scaled`, suggesting strong regional or behavioral signals in fraudulent patterns.

Lastly, for **product** nodes, the agent assigned highest importance to product categories like `Category Name_electronics`, `Category Name_golf gloves`, and `Category Name_fishing`, which may reflect unusual or targeted high-value purchases.

These results are visualized in Figures 5, 6, and 7, which show the top seven features per node type. Overall, the RL explainer complements saliency analysis by providing sparse, entity-specific explanations aligned with domain intuition, thereby strengthening the transparency and interpretability of the model.

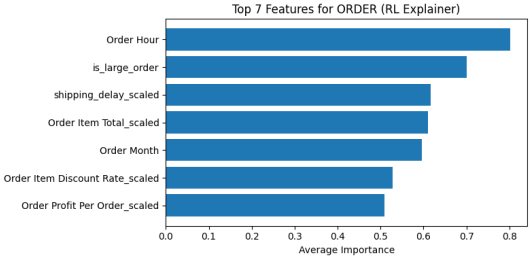

Fig. 5. Top 5 Features for ORDER Node (RL Explainer)

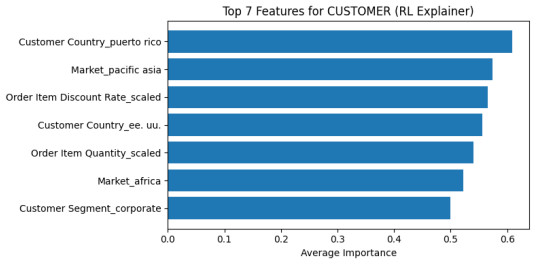

Fig. 6. Top 5 Features for CUSTOMER Node (RL Explainer)

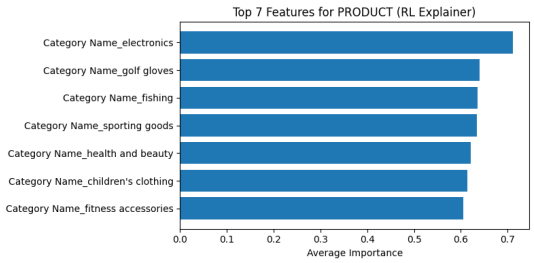

Fig. 7. Top 5 Features for PRODUCT Node (RL Explainer)

D. Comparison Between Explainability Methods

While both the saliency-based and RL-based explainers aimed to attribute importance to input features contributing to fraud predictions, they did so through fundamentally different paradigms. Therefore, resulting in both overlapping insights and notable divergences.

The saliency-based method produced a smooth, continuous importance distribution by computing gradients with respect to input features. This technique surfaced general trends across the dataset and allowed for holistic interpretation of what the model had globally learned.

For example, it emphasized profit-related features such as `Order Profit Per Order_scaled` and `Order Item Discount_scaled`, as well as regional identifiers like `Market_latam` and `Market_pacific asia`. These results suggest the model captured widespread fraud-related behaviors tied to transaction profitability and region-specific patterns.

In contrast, the RL-based explainer operated locally on individual high-confidence fraud predictions, learning sparse binary masks that retain only the most essential features per node. This resulted in more concise, selective interpretations. The RL explainer favored features like `Order Hour`, `is_large_order`, and `Order Item Total_scaled`, which were not among the top saliency scores.

This discrepancy suggests that while these features might not exhibit strong average gradients globally, they are crucial for specific fraud scenarios, perhaps capturing episodic behaviors like off-hour bulk purchases or unusually high-value transactions.

Despite methodological differences, both explainers converged on the significance of certain pricing-related features, most notably `Order Item Discount Rate_scaled`

and `Order Profit Per Order_scaled`. This agreement reinforces the idea that pricing anomalies are consistently informative across both global and local levels of model interpretation.

On the customer side, both methods acknowledged the importance of regional and segment-level signals, but with different emphases. While the saliency-based method elevated geographic markets like `Market_latam` and `Market_pacific asia`, the RL explainer assigned higher importance to more specific identifiers, such as `Customer Country_puerto rico` and `Customer Segment_corporate`.

Interestingly, these features ranked relatively low in the saliency analysis, suggesting that while they may not drive widespread gradients, they are critical in isolated high-risk patterns. This could reflect fraud mechanisms that are uniquely prevalent in certain business segments or regulatory environments, particularly for corporate accounts or regions with less fraud monitoring.

Ultimately, the two explainers not only corroborate each other in several areas, such as the relevance of pricing features and some customer-level behaviors, but also diverge in scope and granularity. The saliency method provides a global overview of the model's feature sensitivities, ideal for general auditing and pattern discovery, while the RL-based explainer delivers sparse, high-precision rationales suitable for case-specific fraud forensics.

Used together, they offer a layered and interpretable view of the model's decision-making, empowering domain experts to understand both the broad logic and specific triggers of fraud predictions.

Key takeaway: The divergence between the two methods should not be seen as a conflict, but rather as a complementary strength. Saliency offers trust in the overall direction of the model's learning, while RL attribution exposes minimal, sufficient feature subsets critical for decisions.

When interpretability is needed at scale, for instance, in model auditing or pattern monitoring, saliency may be more suitable. On the other hand, when explainability is needed for individual transactions, especially in high-risk or legal contexts, the RL method provides sharper, actionable insights. In practice, relying on both together forms a robust interpretability strategy: using saliency to build confidence in model behavior and RL explanations to justify decisions at the instance level.

V. Discussion

This project demonstrated how graph-based modeling can reveal patterns in supply chain fraud that would be difficult to detect through flat, row-by-row analysis. By structuring the data as a heterogeneous graph the model was able to learn from relationships that span multiple entities.

One of the key strengths of this setup was its ability to capture indirect but meaningful interactions. For instance, certain product categories consistently appeared across fraud-labeled orders placed by specific customer segments, especially when large discounts or abnormal shipping delays were involved. These cross-entity signals were only detectable because the model could reason over the graph structure, rather than treating each transaction in isolation.

The explainability tools added transparency to the model's predictions and highlighted which features mattered most across node types. The gradient-based saliency method revealed that features like profit margin, discount rate, and order quantity played a significant role in identifying suspicious transactions.

Meanwhile, the reinforcement learning-based explainer was able to distill more selective and sparse explanations. These explanations helped validate the model's behavior and made its decisions easier to interpret in the context of supply chain operations.

However, several limitations should be acknowledged. Most importantly, the dataset itself is synthetic and was not collected from a real-world supply chain. While its structure reflects plausible relationships between customers, orders, and products, it was likely designed for educational or analytical use rather than operational fraud detection.

Furthermore, the fraud labels used for the training were manually defined based on heuristics, not based on confirmed fraudulent transactions. This means the model was learning to detect simulated patterns of fraud rather than behavior observed in practice, which limits the ability to generalize these results to live systems.

Additionally, the dataset had a severe class imbalance, with only 7.24% of the transactions labeled as fraud. While class weighting and stratified sampling were used to mitigate this, the small number of positive samples can still make learning unstable and limit generalization.

From an explainability perspective, most tools are designed for homogeneous graphs or assume simplified model structures, which makes adaptation challenging. For example, GNNExplainer [3], one of the earliest general-purpose GNN explanation tools, is built to extract subgraphs and features relevant to individual predictions but assumes a single-node-type, single-edge-type graph.

Similarly, PGExplainer [4] learns a distribution over important edges to explain predictions, but it is also tailored for homogeneous settings and primarily focuses on edge-level explanations.

In our case, simplifying the supply chain graph into a homogeneous structure would undermine the core advantage of representing entity-specific interactions. Since the environment we modeled depends on multi-type nodes and edges, such as customer–order and order–product relations, these existing explainers do not apply cleanly.

Our saliency- and reinforcement-based feature masking approach served as a workaround to this limitation, allowing us to produce interpretable explanations within a truly heterogeneous graph without collapsing its structure.

VI. Conclusion and Future Work

This study presents a robust and interpretable pipeline for fraud detection in supply chains using heterogeneous Graph Neural Networks (GNNs). By modeling the data as a heterogeneous graph comprising customers, orders, and products, the proposed system captures multi-entity interactions that are often overlooked in traditional tabular approaches.

This structure enables the model to detect fraud not just through isolated features but through contextual patterns that span across multiple entities and relationships.

The GraphSAGE-based architecture was effective in classifying fraudulent orders, achieving high predictive performance across multiple metrics. Particularly, a macro-averaged F1-score of 88.28%, which highlights the model's balanced treatment of both minority (fraud) and majority (non-fraud) classes.

Beyond predictive performance, this work addresses a critical gap in GNN-based fraud detection: the lack of interpretability. While GNNs have demonstrated strong performance, they often operate as black-box models, limiting their transparency in decision-making.

To overcome this, two complementary explainability approaches were integrated. In particular, gradient-based saliency for global insights and reinforcement learning (RL)-based feature masking for sparse, instance-specific attributions. Together, these methods reveal that the model not only captures broad signals related to pricing, delay, and regional behaviors, but also responds to subtle transactional anomalies, offering a more transparent and trustworthy fraud detection framework.

The comparative analysis showed convergence on core fraud indicators, such as discount rate and profit per order, but divergence in granularity. Saliency surfaced smooth, dataset-wide trends, whereas the RL agent provided focused, case-specific rationales. This dual-layer explanation framework strengthens trust in the model's decisions by offering both global pattern discovery and local justifications.

However, several limitations must be acknowledged. The dataset is synthetic, and the fraud labels were generated using rule-based heuristics, which may not fully represent real-world fraud complexities. Moreover, existing GNN explainability tools were not compatible with the graph's heterogeneous structure, necessitating custom solutions. These challenges highlight the need for more standardized and flexible interpretability techniques tailored to heterogeneous graph data.

Overall, this research demonstrates that interpretable, graph-based fraud detection is both feasible and effective, providing a solid foundation for real-world deployment in complex transactional systems.

References

[1] S. X. Rao, Z. Liu, X. Cheng, L. Liu, S. Wang, W. Wu, and V. W. Zheng, "xfraud: Explainable fraud transaction detection on heterogeneous graphs," *Proceedings of the VLDB Endowment*, vol. 15, no. 2, pp. 427–439, 2022.

[2] R. D. Rajamani, "DataCo Smart Supply Chain for Big Data Analysis," https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis, 2020, accessed: 2024-04-29.

[3] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[4] H. Luo, R. Y. Cheng, and J. Zhang, "Pgexplainer: Probabilistic graph explainer," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 12 277–12 287.

[5] X. Wang, Y. Liu, J. He, H. Wang, and J. Han, "Relex: Symbolic relational explanations for graph neural networks," *arXiv preprint arXiv:2301.13478*, 2023.

[6] H. Yuan, H. Yu, J. Wang, X. Li, and S. Zhang, "Subgraphx: Explaining graph neural networks by subgraph extraction," in *Proceedings of the 38th International Conference on Machine Learning*, vol. 139, 2021, pp. 12 218–12 227.

[7] R. Gandhi and A. Kumar, "Graph convolutional neural networks for supplier fraud detection," *Expert Systems with Applications*, vol. 207, p. 117936, 2025.

[8] A. Wasiac, L. Chen, and R. Kumar, "Graph-based machine learning for anomaly detection in supply chain networks," *Journal of Supply Chain Analytics*, vol. 9, no. 1, pp. 55–74, 2024.

[9] L. Xie, Z. Liu, W. Huang, and Y. Yang, "Dast-gnn: Dual-attention spatial-temporal graph neural network for financial fraud detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 781–789.

[10] A. Suresh, K. Iyer, and R. Verma, "Hierarchical graph attention with topic modeling for supply chain financial fraud detection," in *Proceedings of the 2025 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2025.

[11] X. Wu, Y. Zhang, H. Li, and J. Wang, "Multifraud: A multitask framework for fraud detection and explanation based on heterogeneous graph neural networks," *Information Systems*, vol. 115, p. 102335, 2024.

[12] Z. Cui, L. Wang, L. Chen, and X. Zhang, "Fraudgnn-rl: A graph neural network with reinforcement learning for adaptive financial fraud detection," *Applied Intelligence*, vol. 55, no. 1, p. 132–148, 2025.