xFraud: Explainable Fraud Transaction Detection on Heterogeneous Graphs

Susie Xi Rao
KOF Swiss Economic Institute &
Systems Group
ETH Zurich
raox@inf.ethz.ch

Shuai Zhang
Systems Group
ETH Zurich
shuazhang@inf.ethz.ch

Zhichao Han
eBay China
zhihan@ebay.com

Zitao Zhang
eBay China
zitzhang@ebay.com

Wei Min
eBay China
wmin@ebay.com

Zhiyao Chen
eBay China
zhiyachen@ebay.com

Yinan Shan
eBay China
yshan@ebay.com

Yang Zhao
eBay China
yzhao5@ebay.com

Ce Zhang
Systems Group
ETH Zurich
ce.zhang@inf.ethz.ch

October 16, 2020

Abstract

At online retail platforms, it is crucial to actively detect risks of fraudulent transactions to improve our customer experience, minimize loss, and prevent unauthorized chargebacks. Traditional rule-based methods and simple feature-based models are either inefficient or brittle and uninterpretable. The graph structure that exists among the heterogeneous typed entities of the transaction logs is informative and difficult to fake. To utilize the heterogeneous graph relationships and enrich the explainability, we present **xFraud**, an e**x**plainable **Fraud** transaction prediction system. xFraud is composed of a *predictor* which learns expressive representations for malicious transaction detection from the heterogeneous transaction graph via a self-attentive heterogeneous graph neural network, and an *explainer* that generates meaningful and human understandable explanations from graphs to facilitate further process in business unit. In our experiments with xFraud on two real transaction networks with up to ten millions transactions, we are able to achieve an area under a curve (AUC) score that outperforms baseline models and graph embedding methods. In addition, we show how the explainer could benefit the understanding towards model predictions and enhance model trustworthiness for real-world fraud transaction cases.

***K*eywords** heterogeneous graph, graph neural network, node classification, anomaly detection

1 Introduction

Fraud detection has been a trending topic for e-commerce companies and social media platforms. Online retail industry is reshaping our shopping behaviors and the resulting security risks are not negligible. Common threats in e-commerce include account acquisition, financial information theft, fake chargeback, money laundry, and many more. For instance, malicious attackers might try to steal customer's credit card information; the login credentials can also be acquired by hackers. These criminal activities can bring negative impacts on user experiences, cause financial losses, and seriously degrade the credibility of platforms. As such, it is crucial to identify fraudulent behaviors and take every precaution

to minimize the risk. Yet, the large-scale volume of daily transactions makes manual detection impossible and more advanced automatic detection tools are needed.

This work focuses on automatic fraudulent transaction detection in a real-world e-commerce environment. Existing research on fraudulent transaction can be roughly categorized into rule-based methods and machine learning methods. Rule-based approaches utilize human-created rules to detect suspicious transactions, which is inefficient and heavily relies on expert knowledge. Consequently, machine learning based methods have flourished in recent years. In general, the fraudulent transaction detection task is formulated as a classification problem where features associated with each transaction are fed into models to predict whether it is risky or not. A plethora of classification algorithms such as logistic regression and deep neural networks (DNNs) are applicable in this case. Nevertheless, only transaction level features are leveraged in these models, which may fail to capture the intricate characteristics of more complex scenarios.

Recently, a new paradigm of neural networks for non-Euclidean data structures, graph neural networks (GNNs), is gaining increasing popularity. GNN is a type of neural network which could directly operate on graph structured data with message passing and aggregation. Popular variants include graph convolutional networks (GCN) [1], graph attention networks (GAT) [2], and GraphSAGE [3]. GNNs consider not only instance-level features but also graph-level features by performing message passage to agglomerate information from neighbors when making decisions. Inspired by the successes of GNNs in other fields, some works apply GNNs to anomaly detection tasks such as spam reviews [4, 5], anti-money laundry [6], and malicious accounts detection [7, 8, 9, 10, 11, 12]. Specifically, homogeneous graphs are usually constructed and standard GNNs models are employed.

In this work, we propose **xFraud**, an explainable fraud detection system which consists of a predictor and an explainer. In the predictor, we tackle the fraudulent transaction detection task from the graph perspective. Different from existing works, a heterogeneous graph composed of different types of nodes (e.g., transactions, addresses, payment tokens, etc.) from transaction record is constructed. To capture the heterogeneous relation patterns and learn more expressive node representations, a self-attention based heterogeneous graph neural network is adopted. The predictor can automatically aggregate information from different types of nodes via disparate paths, thus, enriching the node representations. This is important as (1) some entities such as device identifiers and payment tokens are important indicators of risk and (2) it shows how the relationships between different users/transactions are established via different paths. To the best of our knowledge, xFraud is the first heterogeneous GNN system that tackles transaction fraud detection.

Moreover, unlike traditional classification tasks, the claim that a transaction is fraudulent should be made very cautiously as false claim may result in unnecessary trouble for customers, which can degrade customer experience. To improve model trustworthiness and make the predictions more convincing, we integrate an explainer into our framework that can provide intuitive explanation for model predictions. Equipped with valuable explanations, our auditors, regulators, or decision makers know how a transaction is flagged, thus making more sensible decisions.

To summarize, our key contributions are:

- In xFraud, we propose to a heterogeneous GNN model (predictor) in detecting fraud transactions. Our method can be applied to industrial level datasets because it uses efficient sampling (*GraphSage*) and allows weight sharing of various target node types. xFraud predictor provides concrete analytical angles of fraudulent activities.
- We add explainability into xFraud by adding an explainer component. xFraud explainer computes the contributions of its neighboring node types and edges when predicting a node. Thus, it enables explicit case studies of network predictions, which is beneficial for model trustworthiness.
- We have conducted experiments and case studies on two real-world transaction networks to show the efficacy of xFraud predictor and explainer in detecting transaction frauds and in facilitating case analysis of graph structural pattern in frauds.

2 Related Work

In this section, we identify and review several key areas relevant to our work.

2.1 Heterogeneous Graph Neural Networks

Heterogeneous graph neural networks are extension of GNNs that operate on heterogeneous graphs consisting of different node types and edge types. These methods have the capability in processing the heterogeneity of graphs. Most of the work on heterogeneous graphs implement attention mechanisms to aggregate information from various types of nodes, e.g., heterogeneous graph neural network (HetGNN) [13], heterogeneous attention network (HAN) [14], and

heterogeneous graph transformer (HGT) [15]. We refer readers to [16] for an extensive survey on heterogeneous graph representation.

2.2 Fraud Detection

The research community has contributed joint efforts with the industries to understand real-world fraud detection problems and the bottleneck of detection systems. We briefly review the most recent approaches and categorize them from the following two perspectives: studies leveraging graph information with non-GNN methods [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30] and those using GNN methods [5, 10, 11, 12, 4, 6, 7, 8, 9] .

2.2.1 Non-GNN methods for fraud detection

Many existing works use a pipeline of multiple models to detect fraudulent activities (see anti-money laundry [31], spam detection [19, 32, 33], fraudster user detection [17, 24, 34, 20, 27, 22, 28, 29, 30, 25], fraud transaction detection [23, 21, 18, 26]). These models can be event/sequence based [23, 20, 18, 5], or meta path based [24, 19, 17, 26].

We discuss two latest works on defaulter detection and transaction fraud detection, where heterogeneous graphs are adopted. Zhong et al. [17] introduced a system *MAHINDER* (Multi-view Attributed Heterogeneous Information Network based financial DEfault useR detection), where pre-defined meta paths in a heterogeneous graph with users and merchants were selected. Each meta path could be viewed based on different relations between two nodes, such as fund, social, family, device. In addition, attention mechanisms were used to measure the importance of nodes, links and meta paths. They used LSTM units to model node and link sequences at different timestamps, finally aggregated all representations and fed them into a *softmax* classifier to identify defaulters. Zhu et al. [18] employed hierarchical explainable network (HEN) to model users' transaction sequences and predict card-stolen transaction frauds. In xFraud, we aim at making more fine-grained predictions by flagging each transaction of a user in various risk scenarios, as a legitimate user does not imply that all its transaction records are legitimate, once its payment token is stolen. We do not need to define meta paths a priori in xFraud.

2.2.2 GNN based fraud detection

GNN studies differ largely in how to define nodes and edges of a network in abstraction and how to quantify node attributes and message passing between them. This can result in two other paradigms when working with networked data: homogeneous graph [1, 35] and heterogeneous graph [15, 14, 36, 13], depending on the number of object types or relation types.

Homogeneous graph has been widely applied in e-commerce applications to analyze various kinds of networks, such as users, products, reviews (see spam review detection [4], anti-money laundry [6], risky/malicious account detection [7, 8, 9]). Recently, people start solving real-world anomaly detection problems using heterogeneous graph (see spam review detection [5], suspicious user detection [10, 11, 12]) or combining homogeneous and heterogeneous graphs [5], because it allows aggregating information propagation through various types of nodes/edges and mimics more closely the authentic data flows in the real-world networks.

Looking at the application of heterogeneous graph in fraudulent detection, there have been few studies on end-to-end fraud detection [12, 10, 5]. For instance, Liu et al. [12] have utilized attention mechanisms in a device-account heterogeneous graph to capture user activity and device embeddings in each subgraph neighborhood. Each device type is one node type in the heterogeneous graph. Their methods were effective to identify malicious accounts in a cashless payment platform and which device types are prone to fraudsters. In xFraud, however, we are interested in transaction level frauds. The methods proposed in [12] do not fit because each transaction only uses one set of attributes (email, shipping address and payment token). Unlike in malicious account detection, where each account can be associated with multiple devices, we do not find multiple associations when looking at a transaction.

2.3 Explainability in GNN

It is equally important in understanding the prediction results in fraud detection applications, namely, why and how certain frauds are flagged as risky and how GNN methods assist us to trace the information propagation in a network.

Recently, how to interpret and explain the predicting behaviors of GNN networks has gained more and more spotlights. There are mainly two levels to explain the GNN models, input level [37, 38, 39, 40, 41], and model level [42].

Liu et al. [37] discussed context, feature and relation inconsistencies in GNN methods and designed a framework *GraphConsis* which built on a heterogeneous graph with multiple relations and alleviated the inconsistency problem in

GNN models. The goal of *GraphConsis* is to make GNN models more interpretable and explainable by changing the aggregation functions of the embedding features calculated in a heterogeneous graph.

Explainability techniques (gradient based vs. decomposition based) are discussed and compared in [38] and the authors showcased in a real-world chemistry task on how GNN networks generated predictions. In the case of node classification, GNNExplainer [39], a GNN model agnostic explanation framework, proposed explaining the GNN predictions by maximizing the mutual information gain of the true node labels and the predicted labels using informative features. GNNExplainer enables a visualization of important subgraph patterns, which assists users to understand the feature contribution and node label propagation. GraphLIME [40] also offered model agnostic explainability to GNN models by computing HSIC Lasso, a nonlinear local fit of top N-hop neighborhood of the node to explain, and then selecting the most important features. Other than focusing on input based explainability of GNNs, XGNN [42] has enabled a model level interpretation, a reinforcement learning framework to explain the graph generation procedure in GCN which can recover the subgraph structures in a specified class with high probabilities.

GNN explainability in the financial domain has been addressed by Li et al. [41]. They have extended GNNExplainer techniques by (1) adding a regularization term that ensures at least one edge connected to each node is selected in the subgraph, (2) adding edge weighted graph attention to calculate the edge weights in the subgraph. They applied the explanation techniques to identify informative graph topological patterns on financial transaction data, such as bitcoin over the counter (OTC), account matching in bank transaction data, when using GCN to perform node classification.

3 Research Questions and Methodologies

In this section, we discuss our research questions, problem formulation of fraudulent detection in real-world transaction graphs, construction of heterogeneous graph (nodes, edges and attributes), and the proposed framework **xFraud** for fraud transaction detection on heterogeneous transaction graphs as well as for explanation generation.

3.1 Research Questions

As we see in many industrial fraudulent detection systems reviewed in Section 2, people have developed numerous systems at industrial level which benefit not only system performance in fraud detection, but also efficiency for business unit (BU) working with qualitative case studies. We in this work want to achieve the following goals: (1) an effective system that flags risky transaction, (2) a better abstraction of entities in transactions using heterogeneous graph and (3) an explainable visualization that assists BU in its daily routine. Concretely, we address two major challenges:

1. **RQ1:** How can we build an effective and efficient fraud transaction detection method in the setting of heterogeneous graph?
2. **RQ2:** How can we automatically generate human understandable explanations for model predictions on the transaction graphs?

3.2 Constructing Heterogeneous Graph

Formally, a heterogeneous network is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, where node type mapping function is $\tau : \mathcal{V} \rightarrow \mathcal{A}$ and link type mapping function is $\phi : \mathcal{E} \rightarrow \mathcal{R}$. Each node $v \in \mathcal{V}$ has only one type $\tau(v) \in \mathcal{A}$, and each edge $e \in \mathcal{E}$ has only one type $\phi(e) \in \mathcal{R}$. Each node or edge can be associated with attributes, denoted by $X_{\tau(v)}$. In addition, each node or edge can be labeled, denoted by $y \in \mathcal{C}$.

Think of the critical entities involved under fraud scenarios. A credit card might be linked to both a legitimate user and a fraudulent user at different stages. The latter happens in a card stolen case. A common shipping address such as a warehouse is sometimes used in frauds. This linkage tends to be stable, compared with stolen financial instruments.

If we formulate fraud detection as a semi-supervised learning problem in an inductive setting [3] in a heterogeneous graph, we have the specification of the problem formulation as follows. In a heterogeneous transaction graph $\mathcal{G}$, $v \in \mathcal{V}$ has a type $\tau(v) \in \mathcal{A}$, where $\mathcal{A} := \{txn, pmt, email, addr, buyer\}$, referring to *transaction, payment token, email, shipping address, buyer*, respectively[1]. If a transaction has relation with another type of node in {*pmt, email, addr, buyer*}, we put an edge between those two nodes in the heterogeneous graph. Each txn node carries node attributes provided by a risk identification system. Each transaction is flagged legit or fraud.

[1]For this study, we choose these attributes because they reflect typical patterns of fraudulent activities as reported by BU. Other types of attributes, e.g., device type are incorporated in the in-house risk detection system.

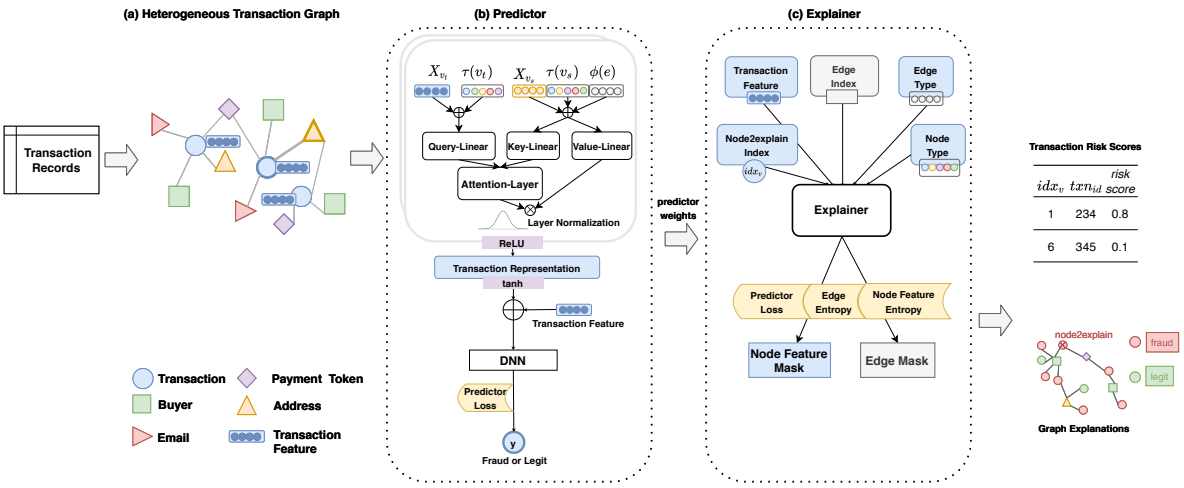

Figure 1: The proposed xFraud system. It consists of a predictor (b) for transaction risky score prediction and an explainer (c) for explanation generation. xFraud takes a heterogeneous graph (a) constructed based on the transaction records and computes the risk score of a transaction and visualizes the subgraph that contributes to the prediction.

3.3 xFraud: Predictor

Here we first discuss the necessity of using heterogeneous GNN in this application, as well as our choices of its architecture.

3.3.1 Why heterogeneous GNN for risky transaction detection?

As we have seen in the literature, it is common to define meta paths when analyzing graph structured data and then to extract corresponding features of nodes and edges on the meta paths before feeding the features into a machine learning or deep learning model. A meta path in graph structured data is defined as a sequence of meta relations. A meta relation is a tuple $\langle \tau(v_s), \phi(e), \tau(v_t) \rangle$, where v_s is the source node, v_t the target node, e the edge from v_s to v_t.

For instance, a buyer uses a financial instrument (*pmt*) to purchase two products, depending on how we perceive nodes and edges in a network, using the definition in Section 3.2, we can have

1. a meta path in a *homogeneous* graph where two transactions share one payment token: $\tau(v_{txn1}) \xleftrightarrow{pmt} \tau(v_{txn2})$;
2. a meta path in a *heterogeneous graph*, where two transactions are decoupled into finer entities:

$$\tau(v_{pmt}) \xleftrightarrow[\text{uses}]{\textit{is used by}} \tau(v_{txn1}) \xleftrightarrow[\textit{has}]{\textit{belongs to}} \tau(v_{buyer}),$$

$$\tau(v_{buyer}) \xleftrightarrow[\textit{belongs to}]{\textit{has}} \tau(v_{txn2}) \xleftrightarrow[\textit{is used by}]{\text{uses}} \tau(v_{pmt}).$$

We see from this simple illustration, how the definitions of meta paths change from homogeneous to heterogeneous graph. Obviously, one can also define the meta paths differently in this simple example. Hence, how to define meta paths is of no doubt influential in many applications [24, 19, 17, 26, 11], which nonetheless also limits the generalization ability of the methods. In a fraud detection scenario, under many circumstances it is by nature impossible to enumerate every possible scenario and their influential meta paths. This is also one of the primary intuitions why heterogeneous GNN method is a desirable choice: it allows a network to learn the importance of meta paths by itself based on the network structure and message passing.

3.3.2 xFraud predictor.

We are inspired by Transformer [35] and HGT [15] and design the xFraud predictor incl. heterogeneous mutual attention and heterogeneous message passing with key, value and query vector operations (self-attention mechanism).

We do not allow target-specific aggregation on different node types, so that we reduce the cost in computing different weights for various node types. We see a better performance in our predictor (see discussion in Section 4.3), when shared weights among different types of nodes are used.

Moreover, we do not adopt relative temporal encoding in HGT when processing transactions with timestamps. Reasons are we would like to keep track of all the transactions a buyer executes, as well as the linking entities a transaction involves. We are also modeling the relation between buyers and transactions. This makes our system adaptable to guest checkouts and their pertaining chargebacks, as those transactions could not be linked to any buyer accounts, but could be linked to suspicious third-party payment accounts or billing email addresses. In this manner, our system is able to capture disguised/missed fraud patterns of guest checkouts that could otherwise be neglected by (1) representing a transaction using buyers and timestamps as in HGT or by (2) representing the transactions in a homogeneous graph.

Figure 1 (b) predictor shows the architecture in detail.

1. xFraud predictor takes a heterogeneous graph (incl. target/source node features X_{v_t}, X_{v_s}; target/source node types $\tau(v_t)$, $\tau(v_s)$; edge types $\phi(e)$) as input. For *txn* nodes, we have node features computed by a company specific risk identifier. For other types of nodes, the initial node features are empty and only get their inputs after the first layer. The typed features are one-hot encoding representations of types.
2. L-connected heterogeneous convolution layers process the graph with self attention mechanism: the input layer $\mathrm{L}^{(0)}$ takes transaction features, node type embeddings (source and target), edge type embeddings as input, which are transformed into query, key and value vectors. Attention scores are calculated for the source and target nodes and then layer-wise normalized, which are then fed into a *ReLU* activation function that emits input for the next convolution layer. In Section 3.3.3, we introduce with formulas how heterogeneous mutual attention and message passing function in one heterogeneous convolution layer.
3. After L-connected layers, a *tanh* activation is applied to the transaction representations generated by GNN. Then these representations are concatenated with the original transaction features and fed into a feedforward connected network. We then apply dropout, layer normalization and *ReLU* transformation before a predicted risk score and a label are calculated (this part is summarized under DNN in Figure 1 (b)).
4. The loss function of xFraud predictor is cross entropy of the true label and the probability score calculated by *softmax* (eq. 11).

3.3.3 Heterogeneous Convolution Layer in xFraud predictor.

We implement a heterogeneous convolution layer as shown in Figure 1 (b).

For a tuple $\langle \tau(v_s), \phi(e), \tau(v_t) \rangle$, where $e = (v_s, v_t)$, we initialize node type embeddings $\tau(v)^{emb}$ and edge type embeddings $\phi(e)^{emb}$ with zero weights, attention weight matrices of source node $W^{att}_{\tau(v_s)}$ and that of target node $W^{att}_{\tau(v_t)}$ with weights subject to uniform distributions, as well as weight matrices for key vector, query vector and value vector, denoted by W^K, W^Q, W^V, respectively. Their weights are also subject to uniform distributions. In a nutshell, the general attention-based heterogeneous convolution layer of the node v_t has three components, attention, message and aggregate as shown in:

$$H^l[v_t] \leftarrow Aggregate(Attention(v_s, v_t) \cdot Message(v_s)). \tag{1}$$

For each target node v_t, we create query, key, value vector representations for self-attention mechanism with multi-heads.

To construct the ith query vector for target node v_t, we start with input to the first layer by taking transaction features of target node $X^{txn}_{\tau(v_t)}$ and its node type embedding $\tau(v_t)^{emb}$:

$$Q^i(v_t) = \text{Q-Linear}^i_{\tau(v_t)}\Big(X^{txn}_{\tau(v_t)} + \tau(v_t)^{emb}\Big), \tag{2}$$

and then for $H^{(l-1)}$, where $l \neq 1$, by:

$$Q^i(v_t) = \text{Q-Linear}^i_{\tau(v_t)}\Big(H^{(l-1)}[v_t]\Big), \tag{3}$$

where $H^{(l-1)}[v_t]$ is the node representation of the node v_t on the $H^{(l-1)}$ layer.

To construct the ith key vector for source node v_s, we start with input to the first layer by taking transaction features of source node $X^{txn}_{\tau(v_s)}$, its node type embedding $\tau(v_s)^{emb}$ and edge type embedding $\phi(e)^{emb}$:

$$K^i(v_s) = \text{K-Linear}^i_{\tau(v_s)}\Big(X^{txn}_{\tau(v_s)} + \tau(v_s)^{emb} + \phi(e)^{emb}\Big), \tag{4}$$

and then for $H^{(l-1)}$, where $l \neq 1$, by:

$$K^i(v_s) = \text{K-Linear}^i_{\tau(v_s)}\Big(H^{(l-1)}[v_s]\Big), \tag{5}$$

where $H^{(l-1)}[s]$ is the node representation of the node v_s on the $H^{(l-1)}$ layer.

To construct the ith value vector for source node v_s, we start with input to the first layer by taking transaction features of source node $X^{txn}_{\tau(v_s)}$, its node type embedding $\tau(v_s)^{emb}$ and edge type embedding $\phi(e)^{emb}$:

$$V^i(v_s) = \text{V-Linear}^i_{\tau(v_s)}\Big(X^{txn}_{\tau(v_s)} + \tau(v_s)^{emb} + \phi(e)^{emb}\Big), \tag{6}$$

and then for $H^{(l-1)}$, where $l \neq 1$, by:

$$V^i(v_s) = \text{V-Linear}^i_{\tau(v_s)}\Big(H^{(l-1)}[v_s]\Big). \tag{7}$$

Adopting the multi-headed attention to control the randomness of initial weights, we first compute the attention output $\alpha\text{-head}^i(v_s, e, v_t)$ of one attention head by:

$$\alpha\text{-head}^i(v_s, e, v_t) = \frac{\Big(K^i(v_s)W^{att}_{\tau(v_s)} + Q^i(v_t)W^{att}_{\tau(v_t)}\Big)}{\sqrt{d_k}}, \tag{8}$$

where d_k is the square root of the dimension of the key vectors.

The heterogeneous mutual attention of target node query vector $Q^i(v_t)$ and source node key vector $K^i(v_s)$ is then computed by:

$$\alpha(v_s, e, v_t) = \underset{\forall v_s \in N(v_t)}{\text{softmax}}\Big(\Big\|_{i\in[1,h]} \alpha\text{-head}^i(v_s, e, v_t)\Big), \tag{9}$$

where $N(v_t)$ represents the neighbors of v_t, h the number of attention heads, $\|$ indicating vector concatenation.

Finally, the message passing between $H^{(l)}$ and $H^{(l-1)}$ is computed by:

$$\text{msg}(v_s, e, v_t) = \Big\|_{i\in[1,h]} \Big(V^i(v_s)\text{dropout}\Big(\alpha\text{-head}^i(v_s, e, v_t)\Big)\Big), \tag{10}$$

where msg-head^i is the message passing of one attention head at the ith query vector.

3.4 xFraud: Explainer

We build xFraud explainer based on the prototype of GNNExplainer, in order to analyze feature contributions in each node prediction and the subgraph structure that a GNN leverages to compute the prediction. The prototype of GNNExplainer as discussed in Section 2.3 provides a node feature mask of the node to explain, when emitting the local structure of subgraph that contributes important features and edges to the prediction. We extend the node feature mask from the node to explain to nodes in the subgraph. Our extension is built on the *pytorch-geometric* implementation of GNNExplainer[2].

The training of explainer is initialized with a random edge mask $1 \times |\mathcal{E}|$ and a random node feature mask $|\mathcal{V}| \times F$ (F the size of node features), as well as four random coefficients of edge size β_{es}, node feature size β_{nfs}, edge entropy β_{ee}, and node feature entropy β_{nfe}. It then takes the node index of node to explain, transaction features, node types, edge types and edge indices as input. The explainer takes the weights trained in the predictor and uses only the predictor model in the evaluation mode, as Figure 1 (c) demonstrates. The loss function of explainer has to include edge entropy and node feature entropy so that the explainer knows which nodes, node features and edges are important in predicting the current node. The meaning of explainer loss is to jointly minimize the predictor loss with the entropy of node features and edges.

To compute the explainer loss, we first compute the edge mask and node feature mask of the subgraph S in G that contributes to the node to explain. We use M_{E_S} to denote the edge mask in the subgraph S ($S \subseteq \mathcal{V}$, S a subgraph of G), we have $M_{E_S} = \sigma(E_S)$, E_S a random initialization of edge parameters. Likewise, we can have the node feature

[2]pytorch-geometric GNNExplainer implementation (last accessed: July 5, 2020).

Table 1: Dataset summary. *The ratio of frauds is only reported on the sampled datasets.

Dataset	Features	#Classes	Train/Val/Test%	Graph type	#Nodes	#Edges	Fraud%*
company-small	114	2	70/10/20	homogeneous heterogeneous	207,749 288,853	2,102,863 612,904	4.3%
company-large	480	2	70/10/20	heterogeneous	8,857,866	13,158,984	3.57%

Table 2: Node type counts $|\mathcal{V}_t|$ in heterogeneous graphs.

Dataset	#Node type t	#Count	%in $\|\mathcal{V}\|$	Dataset	#Node type t	#Count	%in $\|\mathcal{V}\|$
company-small	*txn*	207,749	71.9%	*company-large*	*txn*	3,752,225	42.4%
	pmt	22,273	7.7%		*pmt*	1,180,114	13.3%
	email	25,878	9.0%		*email*	1,307,179	14.8%
	addr	7,138	2.4%		*addr*	1,316,251	14.9%
	buyer	25,815	9.0%		*buyer*	1,302,097	14.6%

mask as $M_{V_S} = \sigma(V_S)$, V_S a random initialization of node feature parameters. Now we calculate the explainer loss in three parts, predictor loss (eq. 11), edge entropy (eq. 12) and node feature entropy (eq. 13). We finally sum them up to form the explainer loss:

$$\sum_i C_i \log(S_i), \tag{11}$$

$$\beta_{\text{es}} \sum M_{E_S} + \beta_{\text{ee}} \text{mean}\Big(- M_{E_S} \log(M_{E_S} + \epsilon) - (1 - M_{E_S}) \log(1 - M_{E_S} + \epsilon)\Big), \tag{12}$$

$$\beta_{\text{nfs}} \frac{\sum M_{V_S}}{M_{V_S}.\text{size(0)}} + \beta_{\text{nfe}} \text{mean}\Big(- M_{V_S} \log(M_{V_S} + \epsilon) - (1 - M_{V_S}) \log(1 - M_{V_S} + \epsilon)\Big), \tag{13}$$

, where $i \in \{\text{labeled transactions}\}$, C_i is the true label of a transaction, S_i the output of the *softmax* layer in DNN, ϵ a small constant to prevent $\log(0)$.

Via back propagation, the new loss is used to compute the new network weights in explainer and thus the network has the capacity to learn the subgraph nodes and their features that most importantly contribute to certain predictions made by the predictor.

4 Experiments of xFraud Predictor

In this section, we conduct experiments on two real-world datasets to verify the effectiveness of xFraud predictor on fraud transaction detection.

4.1 Data and Preprocessing

From transaction records, we have a rich set of relations, for example, two transactions can be linked by the same linkage entities, such as purchasing from the same buyer, using the same payment instrument, shipping to the same address. Based on these linkage strategies, we have two graph data models to choose: homogeneous and heterogeneous graph. Our experiments are also set up to compare the efficacy of homogeneous and heterogeneous graphs in risk scenario. We implement these two models as follows:

Homogeneous graph: We only keep the transactions as nodes, and the linkage entities as edges, so that we have only one node type. Transaction features are constructed to reflect the individual transaction riskiness.

Heterogeneous graph: Both transaction and linkage entities are treated as nodes. If an entity is used in a transaction, we create an edge between the transaction node and that entity. The setting of node feature is the same as in the homogeneous graph: only transaction node has input features to indicate riskiness.

We construct two datasets from historical transactions of a real-world e-commerce platform. Historical transaction records were sampled to generate the following two graph datasets with different scale, so that different methods could

Table 3: Hyperparameters in xFraud predictor and its baselines.

Model	n_batch	n_layer	n_hid	dropout	optimizer	learning rate	beta	weight decay	clip	max_epochs	patience	n_heads	walk length
LR	32	-	-	-	adamw	0.001	(0.9,0.999)	0.01	0.25	128	32	-	
DNN	32	2	400	-	adamw	0.001	(0.9,0.999)	0.01	0.25	128	32	-	-
node2vec	32	-	-	-	-	-	-	-	-	-	-	-	12
GCN	32	6	400	0.2	adamw	0.001	(0.9,0.999)	0.01	0.25	128	32	-	-
GAT	32	3	400	0.2	adamw	0.001	(0.9,0.999)	0.01	0.25	128	32	8	-
HGT	32	6	400	0.2	adamw	0.001	(0.9,0.999)	0.01	0.25	128	32	8	-
xFraud predictor	32	6	400	0.2	adamw	0.001	(0.9,0.999)	0.01	0.25	128	32	8	-

be implemented and fairly compared. Note that the ratio of fraudulent transactions is usually much smaller than that of the legitimate ones; hence, we try to sample more fraudulent transactions than legitimate transactions to mitigate the class imbalance problem.

Dataset *company-large*. All the historical transaction records spanning a given period are used for experiments. The dimension of individual risk features for each transaction is 480. The linking entities whose transaction numbers below a predefined threshold are removed to maintain graph connectivity, which means that there is no isolated transaction in the graph.

Dataset *company-small*. Compared with the large dataset, the smaller dataset shrinks the transaction spanning period and the dimension of individual risk features is 114. To further reduce graph size while preserving graph connectivity, we adopt a graph sampling strategy: (1) All fraudulent transactions and random sampled normal transactions are selected as seeds. (2) Each seed is expanded to its 3-hop neighbors, and at each hop, no more than 32 neighbors are picked. (3) Those groups with transaction numbers less than five are filtered out.

The ratio of transaction frauds is 3.57% and 4.3% in *company-large* and *company-small*, respectively. In Table 1 we summarize the datasets in terms of number of vertices, edges, features, split ratio of train/validation/test. We split these sets in a chronological order. We further summarize the distribution of each node type in Table 2.

4.2 Implementing Predictor and its Baselines

We discuss in Section 2.2.2 why previous work such as Liu et al. [12] does not fit our scenario. We, hence, choose several general baselines for predictor, incl. conventional machine learning models, such as logistic regression (LR) and deep neural network (DNN), graph embeddings methods *node2vec* [43] and simple GNN models such as GCN and GAT[3]. We report the detailed settings of each model and its hyperparameters based on the model performance on the validation set. The hyperparameters in xFraud predictor and its baselines are listed in Table 3.

- LR.
- DNN. A feedforward neural network with only transaction features X^{txn} as input. DNN is implemented as in the DNN component of xFraud predictor, see Section 3.3.2. The NN uses *Relu* as activation function.
- *node2vec*. *node2vec* is a node embedding learning framework which preserves the diverse neighborhood of nodes when simulating a random walk of certain length. We implement *node2vec* on homogeneous transaction graphs. The *walk_length* is set to 12. The output of *node2vec* is used as input features to a binary DNN classifier with the same set of hyperparameters as mentioned in the DNN baseline.
- GCN. We implement GCN on both homogeneous and heterogeneous graphs to compare their performances and see if methods like GCN and GAT which were originally developed for homogeneous graphs can be outperformed by their heterogeneous versions. We test GCN with up to six convolution layers.
- GAT. We also implement GAT in both homogeneous and heterogeneous settings. When constructing the transaction graph as a homogeneous graph on the dataset *company-small*, we use three hidden layers of convolution in GAT, which are less than the layers in GCN. This is to prevent memory blow up.
- HGT. We use the original implementation of HGT[4] with this set of hyperparameters: $\textit{width} = 16, \textit{depth} = 6, \textit{max\_epochbdscanhs} = 10$. We tested HGT only on the *company-small* dataset and observed that xFraud outperformed HGT in this task (see Table 5 in Section 4.3). Due to efficiency problem in *HGSampling* adopted by HGT, it would take much longer than the remaining methods to finish.
- xFraud predictor. We report the performance of predictor with the hyperparameters reported in Table 3.

[3]We did not report the performances of HAN and HetGNN due to the limitation of their current implementations in larger graphs as in our case. However, we tested HAN with 5% of the dataset *company-small* and it did not outperform the baseline DNN.

[4]The original HGT implementation: https://github.com/acbull/pyHGT (last accessed: April 24, 2020).

Table 4: AUC model performances in the dataset *company-large*.

Dataset	Model	AUC	Graph type
company-large	LR	0.7527 ± 0.0007	-
	DNN	0.8370 ± 0.0013	
	GAT	0.8391 ± 0.0021	heterogeneous
	GCN	0.8364 ± 0.0051	
	xFraud predictor	$\mathbf{0.8690 \pm 0.0076}$	

Table 5: AUC model performances in the dataset *company-small*.

Dataset	Model	AUC	Graph type
company-small	LR	0.6925 ± 0.0045	-
	DNN	0.7153 ± 0.0028	
	node2vec	0.6753 ± 0.0066	homogeneous
	GAT	0.7084 ± 0.0039	
	GCN	0.7132 ± 0.0009	
	GAT	0.7256 ± 0.0039	heterogeneous
	GCN	0.7238 ± 0.0122	
	HGT	0.7248 ± 0.0035	
	xFraud predictor	$\mathbf{0.7262 \pm 0.0059}$	

We adopt batch training for models on homogeneous graphs, while on heterogeneous graphs we adopt mini-batch training strategies with sampling, e.g., *GraphSAGE*, *HGSampling*. In *GraphSAGE* sampling, the algorithm first samples k-hop neighborhood of a node and then aggregates feature information from neighbors and finally allows GNN to predict the label using aggregated information. In *HGSampling*, it is able to maintain a similar number of different types of $\tau(v)$ and $\phi(e)$ after sampling and minimize the information loss in the subgraph after sampling. We implement GCN and GAT with both *GraphSAGE* and *HGSampling* in heterogeneous graphs. In all comparisons, models with *GraphSAGE* outperform those with *HGSampling*. Since HGSampling is much costly than *GraphSAGE* because it requires all types of nodes and edges to be of similar size in the sampled subgraph. We only adopt *GraphSAGE* as the mini-batch sampling strategy when implementing xFraud predictor.

4.3 Predictor Result Analysis

We use area under the curve (AUC) as performance metric as the classes are highly imbalanced in our datasets. For each model, we run with five random seeds and report the mean and standard deviation of AUC in those five runs. We also tune xFraud predictor and its baselines and here we only report the best performing models.

Noting the difference in time span and data selection in two datasets (see Section 4.1), we implement xFraud predictor experiments on both datasets and report the AUC performances in Tables 4 and 5.

We first discuss the results on *company-large* in Table 4. Due to excessive computation resources requirement of *node2vec*, GCN and GAT on homogeneous graph, we only report the heterogeneous versions[5]. We make the following observations: (1) Our xFraud predictor outperforms all the baselines. (2) A naive implementation of GCN and GAT on heterogeneous graph does not suit our scenario because those methods were originally developed for homogeneous graphs. (3) Interestingly, a DNN classifier without GNN embeddings as input can achieve the performance on par with that in graph based approaches. This might be due to the fact that GAT and GCN lose the capability in discriminating the types of nodes when different nodes are regarded as an identical type when the algorithm processes the graph.

To show the disparity of GAT/GCN performances on homogeneous and heterogeneous graphs, we further discuss the results on the *company-small* dataset. From Table 5 we make observations as follows. (1) xFraud predictor remains to be the best performer. (2) Models on homogeneous graph are largely outperformed by the model on heterogeneous graphs. Even if one simply changes the input graph from homogeneous to heterogeneous, the same model (e.g., GAT

[5]Our current implementation of GCN and GAT uses batch training, which leads memory explosion when being training with the dataset *company-large*. However, if one rewrites the frameworks using mini-batch training, the memory problem should vanish. We have not yet done that because the observation in Table 5 has confirmed the effectiveness of GAT and GCN when we change the input network from a homogeneous graph to a heterogeneous one. This means that how we construct the graph has a positive impact on how effective a model is.

and GCN) will perform better. Our predictor learns richer network messages propagated through a heterogeneous graph, while a homogeneous graph is weaker in learning messages passed by linking edges representing shared payment tokens, or shipping addresses, because the features carried by those entities are aggregated up to be transaction features. This argument is further substantiated by the comparison of DNN and GCN, the best performing GNN on homogeneous graph: DNN is even slightly better than GCN. (3) xFraud predictor outperforms the baselines on heterogeneous graphs (GAT, GCN and HGT). Our predictor also runs faster than HGT, because HGT has two bottlenecks (sampling and weight sharing among node types) which hamper its performance on larger datasets. In Section 4.2 we have discussed the efficiency of *HGSampling* which is more costly compared to *GraphSAGE* sampling. Our predictor is more efficient by using *GraphSAGE* sampling and allowing weight sharing among different node types.

5 Experiments of xFraud Explainer

In this section, we discuss how we build xFraud explainer on top of the predictor. The main contribution of the explainer is to compute node features and edge masks of important features and nodes. As we will see in case studies (Section 5.2), our extension of GNNExplainer from one node prediction to all nodes in the subgraph provides interesting insights for risk experts when analyzing label propagation in a local heterogeneous graph.

5.1 Implementation Details

Due to graph size, we implement only xFraud explainer on the dataset *company-small* using the trained xFraud predictor, transaction features, node index (of node to explain), edge indices, edge types, and node types as input. As output, we obtain node feature masks of the subgraph that are important to the node to explain, as well as the edge masks indicating the strength of connection between nodes. The hyperparameters of explainer are: *epochs* $= 100$, *lr* $= 0.01$, $\beta_{\text{edge\_size}} = 0.005$, $\beta_{\text{node\_feature\_size}} = 1$, $\beta_{\text{edge\_entropy}} = 1$, $\beta_{\text{node\_feature\_size}} = 0.1$. The xFraud predictor is not retrained during the explanation process.

The output of explainer carries the following meaning: node feature masks give high weights to the feature dimensions of all the nodes in the subgraph that are influential in predicting a certain node; edge masks are the weights of edges in the subgraph, which indicate the strength of connectedness between pairs of nodes.

To visualize the subgraph for a certain node, we use the node index, edge indices and their masks, true labels of nodes as input, where the threshold for visualizing important edges is set to 0.15. The thicker an edge is, the stronger the connection is. Although the threshold is arbitrary, we show in the following cases studies that the visualization with this threshold suits our business cases and highlights the crucial information for fraud analysis. We visualize the connections with bilateral edges. Note that to focus on the node to explain in the case studies, we use the ground truth labels for transactions.

5.2 Case Studies

We demonstrate here with two case studies that xFraud is able to grasp the important contributions of node features and edges when detecting transaction frauds in heterogeneous graphs in risk scenarios. Our risk experts are able to utilize the outputs and make fine-grained and informed decisions. Our xFraud explainer enables a graph based explanation which a traditional rule based system cannot offer. Currently the BU is only using a rule based system[6] to filter the suspicious transactions stored in the tabular format. xFraud explainer is innovative to a traditional BU annotation routine, because it allows experts to combine graph level and feature level information[7].

5.2.1 Case study 1: Leverage Graph Topology

In Figure 2, we present a potential stolen financial case with pertaining nodes. In the subgraph generated by xFraud explainer, the target transaction to explain (node 0), was linked to a financial instrument (node 2) together with a number of other transactions. Explained by the subgraph structure, transaction node 0 was suspicious as it was linked to the suspicious nodes 1, 2 & 4, which were linked to a group of fraudulent transactions (the orange group on the left). With edge mask weights computed by xFraud explainer, it is further identified that the buyer account plays the most important role in risk propagation than the other entity nodes. Moreover, fraud transactions (nodes 8, 9, 14 & 18), as indicated by the edge thickness, have contributed largely to the prediction of node 0. Meanwhile, as shown in Figure 2 (a), the

[6] *skope-rules* that performs rule mining on tabular data, https://github.com/scikit-learn-contrib/skope-rules (last accessed: Oct 18, 2020).

[7] In the future, it would be certainly of interest to evaluate the respective contributions of xFraud explainer and the rule based system the BU currently uses.

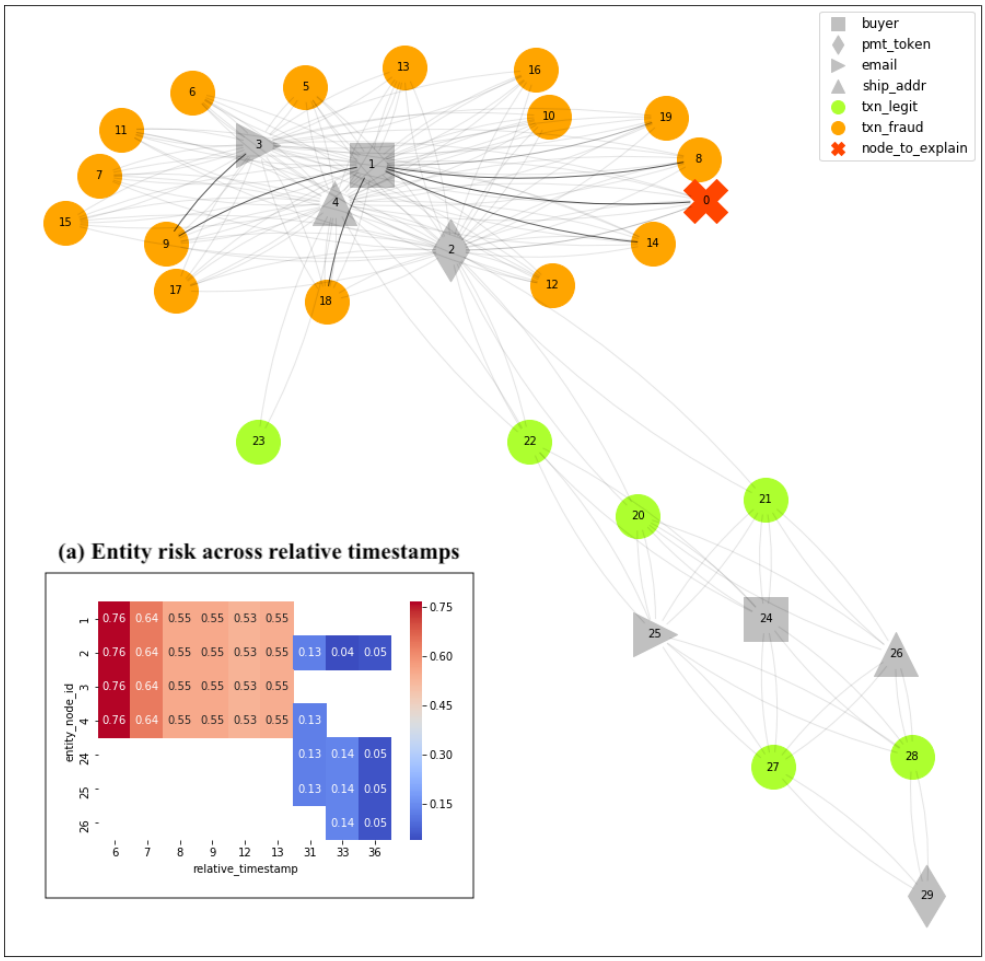

Figure 2: Case study 1: We show the patterns of fraudulent and legitimate transactions, as well their connection strength with linking entities. Transaction risk scores across relative timestamps are shown in the subplot (a).

risk scores on transactions differ significantly between the two time periods. By utilizing the information from xFraud explainer, our risk expert determines that the case is possibly a stolen financial. The financial instrument was initially stolen and the fraudster used it to complete a number of transactions. After a period of time, the owner of the financial instrument reported unauthorized chargebacks to the card issuer and claimed it back. Then the owner also conducted legitimate transactions using the same financial instrument with different entities (nodes 24, 25 & 26).

5.2.2 Case study 2: Leverage Node Features

In the case described above, most of the transactions are linked to the same suspicious entity nodes, which can be prevented by velocity rules control. In contrast, in Figure 3 we present an account takeover (ATO) case where the velocity rules control might not be an effective risk control strategy. For example, there are both legitimate and fraudulent transactions associated to node 4 in this case. If we enforced a limit control regarding with number of transactions associating with one entity, all the legitimate transactions would be affected. However, by leveraging important features generated from xFraud explainer, we see most of the influential features are from user behavior domain. When projecting these features to a 2D plot using TSNE [44], we can see very clearly that the feature space of target transaction to explain is close to that of the fraud transactions. It also shows that fraudulent and legitimate transactions have two different purchasing patterns. Together with the subgraph patterns and the influential features, risk expert diagnoses the case as ATO qualitatively. It is possible that the account (node 4) was taken over by fraudster for subsequent transactions with different entities like devices and shipping addresses.

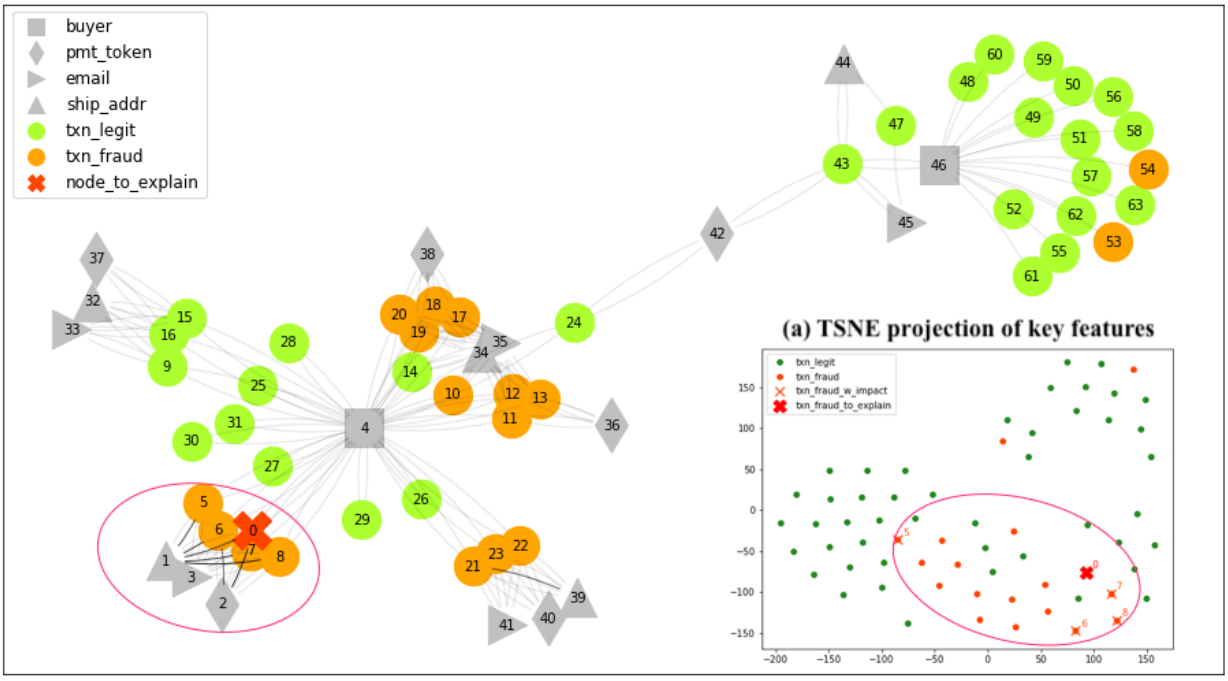

Figure 3: Case study 2: We show the feature linkage patterns of commonly used entities. TSNE projection of important features is shown in the subplot (a).

6 Conclusion

In this paper, we propose xFraud, a system for both fraud transaction detection and explanation generation on model prediction. Specifically, heterogeneous graphs are constructed and a self-attentive heterogeneous graph neural network is leveraged for risky transaction scoring. To generate explanations, we design a revised graph explainer that could directly provide explanations on heterogeneous graphs. We conduct experiments on two real-world transaction datasets and show that our predictor is the best performer compared with other strong baselines. We show by real-world case studies that, with the xFraud explainer, we can generate convincing explanations at both feature-level and graph-level to assist further decision-making of business units.

References

[1] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[2] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[4] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 310–316, 2019.

[5] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2703–2711, 2019.

[6] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

[7] Jun Ma, Danqing Zhang, Yun Wang, Yan Zhang, and Alexey Pozdnoukhov. Graphrad: A graph-based risky account detection system. 2018.

[8] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4424–4431, 2019.

[9] Chen Liang, Ziqi Liu, Bin Liu, Jun Zhou, Xiaolong Li, Shuang Yang, and Yuan Qi. Uncovering insurance fraud conspiracy with network learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1181–1184, 2019.

[10] Rui Wen, Jianyu Wang, Chunming Wu, and Jian Xiong. Asa: Adversary situation awareness via heterogeneous graph convolutional networks. In *Companion Proceedings of the Web Conference 2020*, pages 674–678, 2020.

[11] Yiming Zhang, Yujie Fan, Yanfang Ye, Liang Zhao, and Chuan Shi. Key player identification in underground forums over attributed heterogeneous information network embedding framework. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 549–558, 2019.

[12] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2077–2085, 2018.

[13] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 793–803, 2019.

[14] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032, 2019.

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710, 2020.

[16] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond. *arXiv preprint arXiv:2004.00216*, 2020.

[17] Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. Financial defaulter detection on online credit payment via multi-view attributed heterogeneous information network. In *Proceedings of The Web Conference 2020*, pages 785–795, 2020.

[18] Yongchun Zhu, Dongbo Xi, Bowen Song, Fuzhen Zhuang, Shuai Chen, Xi Gu, and Qing He. Modeling users' behavior sequences with hierarchical explainable network for cross-domain fraud detection. In *Proceedings of The Web Conference 2020*, pages 928–938, 2020.

[19] Kai Shu, Deepak Mahudeswaran, Suhang Wang, and Huan Liu. Hierarchical propagation networks for fake news detection: Investigation and exploitation. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 626–637, 2020.

[20] Adam Breuer, Roee Eilat, and Udi Weinsberg. Friend or faux: Graph-based early detection of fake accounts on social networks. In *Proceedings of The Web Conference 2020*, pages 1287–1297, 2020.

[21] Yuxiang Ren, Hao Zhu, Jiawei ZHang, Peng Dai, and Liefeng Bo. Ensemfdet: An ensemble approach to fraud detection based on bipartite graph. *arXiv preprint arXiv:1912.11113*, 2019.

[22] Hamed Nilforoshan and Neil Shah. Slicendice: Mining suspicious multi-attribute entity groups with multi-view graphs. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 351–363. IEEE, 2019.

[23] Shaosheng Cao, XinXing Yang, Cen Chen, Jun Zhou, Xiaolong Li, and Yuan Qi. Titant: online real-time transaction fraud detection in ant financial. *arXiv preprint arXiv:1906.07407*, 2019.

[24] Binbin Hu, Zhiqiang Zhang, Chuan Shi, Jun Zhou, Xiaolong Li, and Yuan Qi. Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 946–953, 2019.

[25] Haibo Wang, Chuan Zhou, Jia Wu, Weizhen Dang, Xingquan Zhu, and Jilong Wang. Deep structure learning for fraud detection. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 567–576. IEEE, 2018.

[26] Bokai Cao, Mia Mao, Siim Viidu, and S Yu Philip. Hitfraud: a broad learning approach for collective fraud detection in heterogeneous information networks. In *2017 IEEE international conference on data mining (ICDM)*, pages 769–774. IEEE, 2017.

[27] Dhivya Eswaran, Stephan Günnemann, Christos Faloutsos, Disha Makhija, and Mohit Kumar. Zoobp: Belief propagation for heterogeneous networks. *Proceedings of the VLDB Endowment*, 10(5):625–636, 2017.

[28] Shenghua Liu, Bryan Hooi, and Christos Faloutsos. Holoscope: Topology-and-spike aware fraud detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1539–1548, 2017.

[29] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 895–904, 2016.

[30] Xiang Li, Wen Zhang, Jiuzhou Xi, and Hao Zhu. Hgsuspector: Scalable collective fraud detection in heterogeneous graphs. 2018.

[31] Paul Emmerich, Maximilian Pudelko, Sebastian Gallenmüller, and Georg Carle. Flowscope: Efficient packet capture and storage in 100 gbit/s networks. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–9. IEEE, 2017.

[32] Sarthika Dhawan, Siva Charan Reddy Gangireddy, Shiv Kumar, and Tanmoy Chakraborty. Spotting collective behaviour of online frauds in customer reviews. *arXiv preprint arXiv:1905.13649*, 2019.

[33] Parisa Kaghazgaran, James Caverlee, and Anna Squicciarini. Combating crowdsourced review manipulators: A neighborhood-based approach. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 306–314, 2018.

[34] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341, 2018.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[36] Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. An attention-based graph neural network for heterogeneous structural learning. In *AAAI*, pages 4132–4139, 2020.

[37] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. *arXiv preprint arXiv:2005.00625*, 2020.

[38] Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*, 2019.

[39] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in neural information processing systems*, pages 9244–9255, 2019.

[40] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *arXiv preprint arXiv:2001.06216*, 2020.

[41] P. Reddy X. Li, J. Saude and M. Veloso. Classifying and understanding financial data using graph neural network. In *AAAI*, 2020.

[42] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. *arXiv preprint arXiv:2006.02587*, 2020.

[43] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[44] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.