Fraud Detection with Graph Features and GNN

Nikita Iserson

PyData Nights Vol.1

17.11 18:00 - 22:00

JetBrains Munich

PyData

JET BRAINS

Fraud Types in 2021

Social Network

- Fake Reviews
- Social Bots
- Misinformation
- Disinformation
- Fake Accounts
- Social Sybils
- Link Advertising

Finance

- Insurance Fraud
- Loan Defaulter
- Money Laundering
- Malicious Account
- Transaction Fraud
- Cash-out User
- Bitcoin Fraud

Others

- Advertisement
- Mobile Apps
- Ecommerce
- Crowdturfing
- Fake Clicks
- Game
- Account Takeover

Why use Graph for Fraud Detection?

🔷 Class Imbalance, Label Scarcity & Fidelity (Fraud cases are rare events)

🔷 Fraud Camouflage - handle context & feature inconsistency (i.e. fraudsters connecting to regular entities)

🔷 Investigation and Exploration (visual way to connect the dots for the crime case)

🔷 Anomaly Detection - handle point, structural and contextual outliers

🔷 Graph Embeddings - combined with NLP, could be used for scalable fuzzy search and entity resolution

🔷 Explainability & fairness - adding the context and structure for interpretation, rebalancing the data to remove bias.

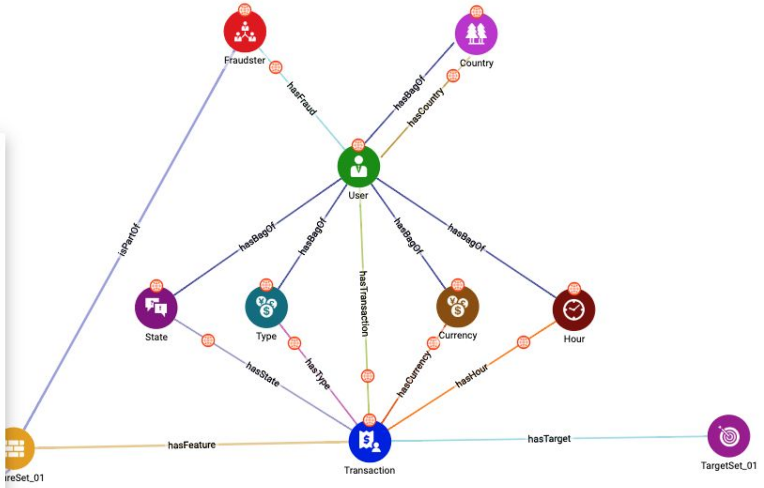
🏆 That's why Facebook, Amazon, Tencent, Alibaba and eBay are using Graph for Fraud Detection 🕵️

Data Model – Transactional Fraud

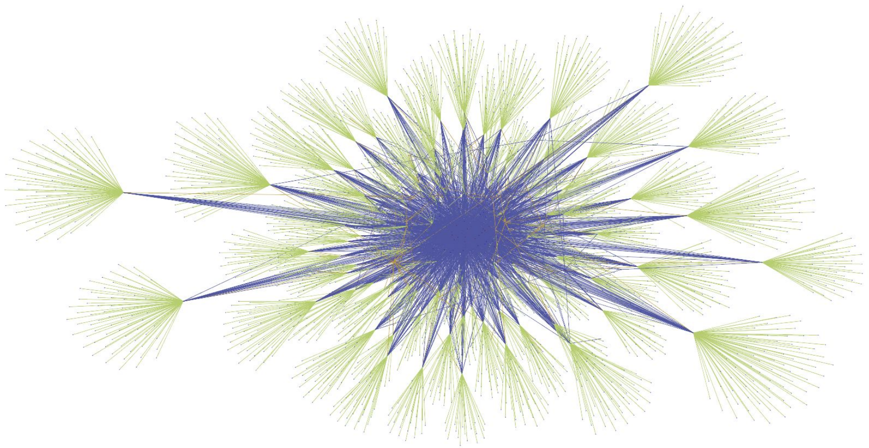
- Account (Country, Fraud)
- Transaction
- State (Failed, Completed)
- Type (Fee, Transfer, ATM)
- Currency (EUR, USD, RON)
- Hour (15:00, 01:00)
- ML DataMart

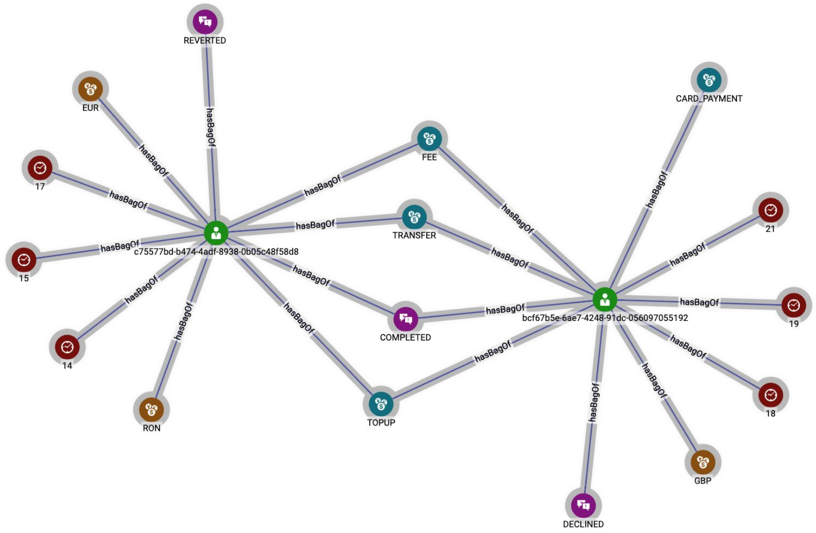
  (TargetSet + FeatureSet)

Vertex	FeatureSet_01
(PRIMARY ID) ID	STRING
first_digit	INT
last_digit	INT
user_cnt_topups	INT
user_mean_topups	INT
user_uniq_states	INT
country	STRING
mean_time_to_tran_min	INT
user_std_topups	INT
louvain_community_tx	INT
hour	INT
louvain_community_u	INT
pagerank_tx	FLOAT
pagerank_u	FLOAT
USER_ID	STRING



Exploration and Investigation





Fraud Detection with Graph Features

🔷 Feature Engineering (Benford law, Tx velocity, Graph)

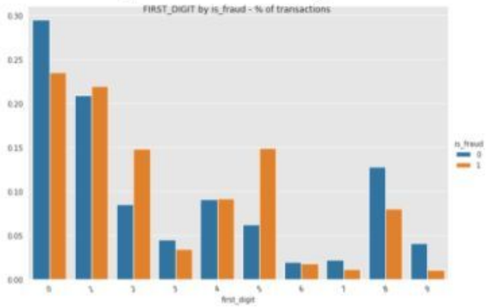
🔷 Exploration Data Analysis (Red Flag Bar Charts)

🔷 Community Detection (Account - Transaction Graph)

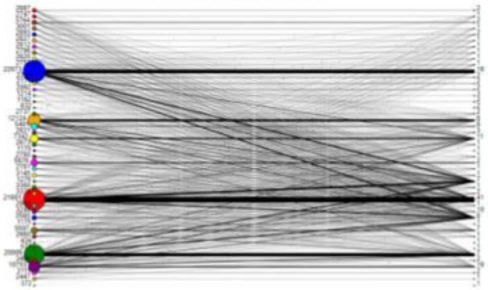
🔷 Model Training and Feature Importance with Graph

🔷 Model Evaluation and Financial Impact with Graph

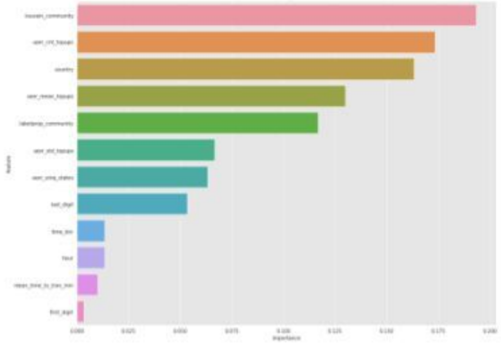
Exploration Data Analysis



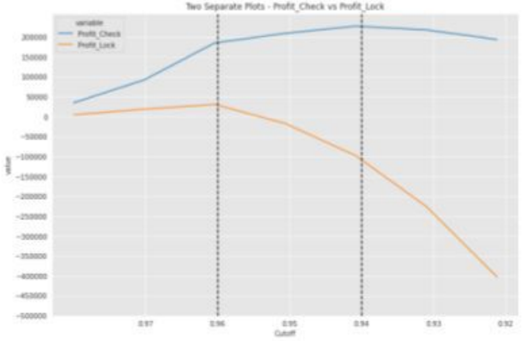
Community Detection



Feature Engineering with Graph



Model Financial Impact (ROI)



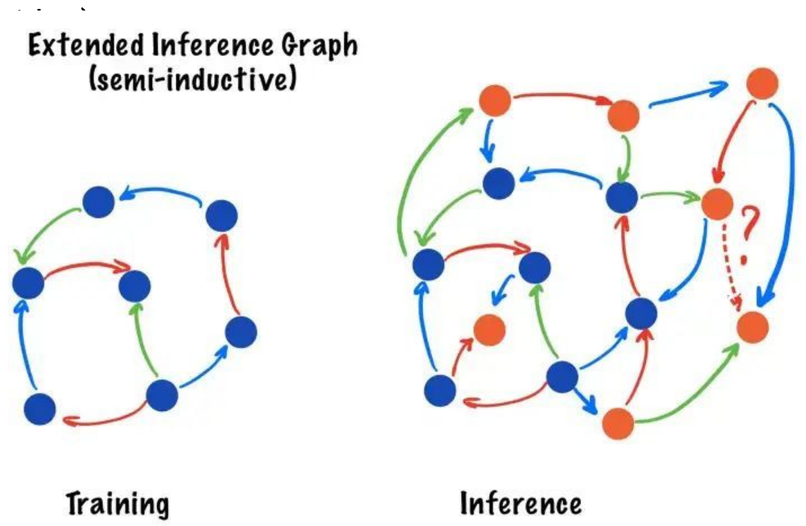
Temporal Graph Features and Inference

Some important practical tips:

🔷 Graph Features - calculate only on Train Subgraph (PageRank, to avoid leakage

🔷 Graph Features - use only Train Labels (k-Hop distances)

🔷 Time-Based Split - use for cross-validation for most of the cases (node classification, link prediction etc.)



Tools for large-scale dynamic predictions: ⚙️

🔷 Sample SubGraph (no need for full neigbourhood, but with tim

🔹 Node classification: ClusterGCN, GraphSAINT

🔹 Link prediction: SEAL, GRAIL

Online Graph Feature Calculation and GNN

Some important practical tips:

🔶 Next two steps could be executed in parallel.

🔷 Calculate Online Graph Features:

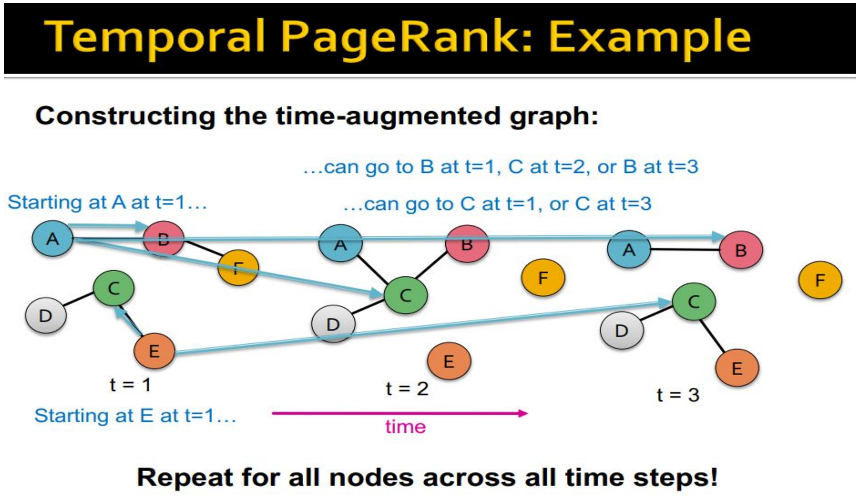
- Temporal PageRank
- Dynamic Community Detection

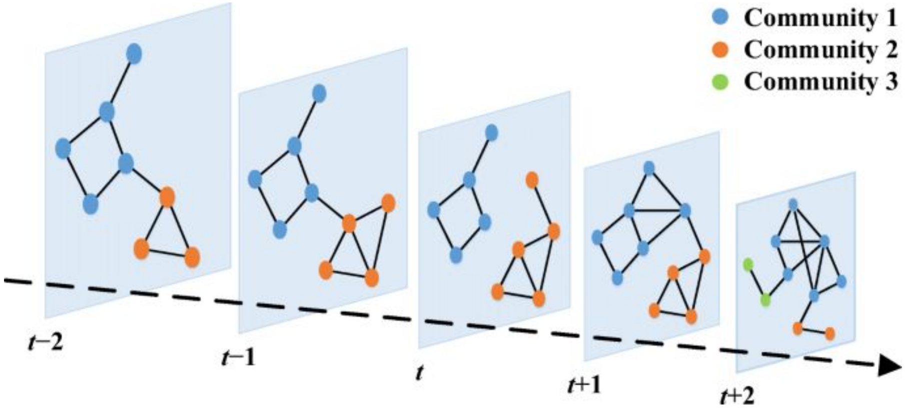
🔷 Add Offline Node/Edge Features

(depending on the data source)

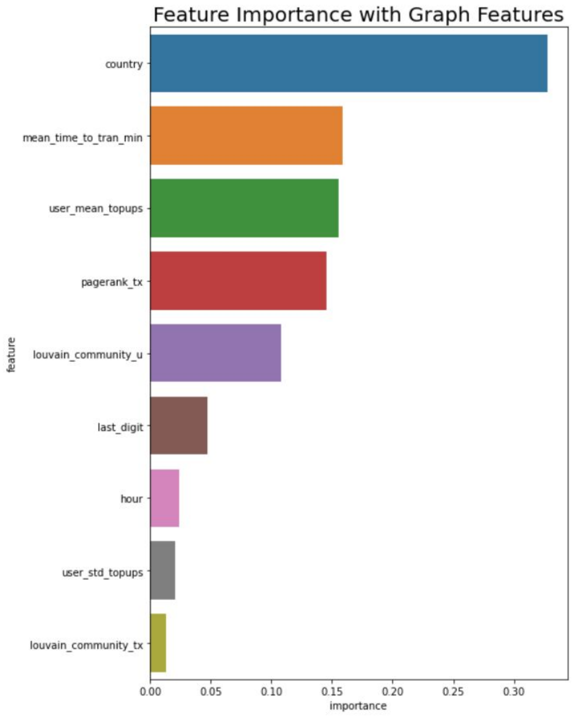
- Stream-table join (cache, DB lookup)
- Stream-stream join (PubSub)

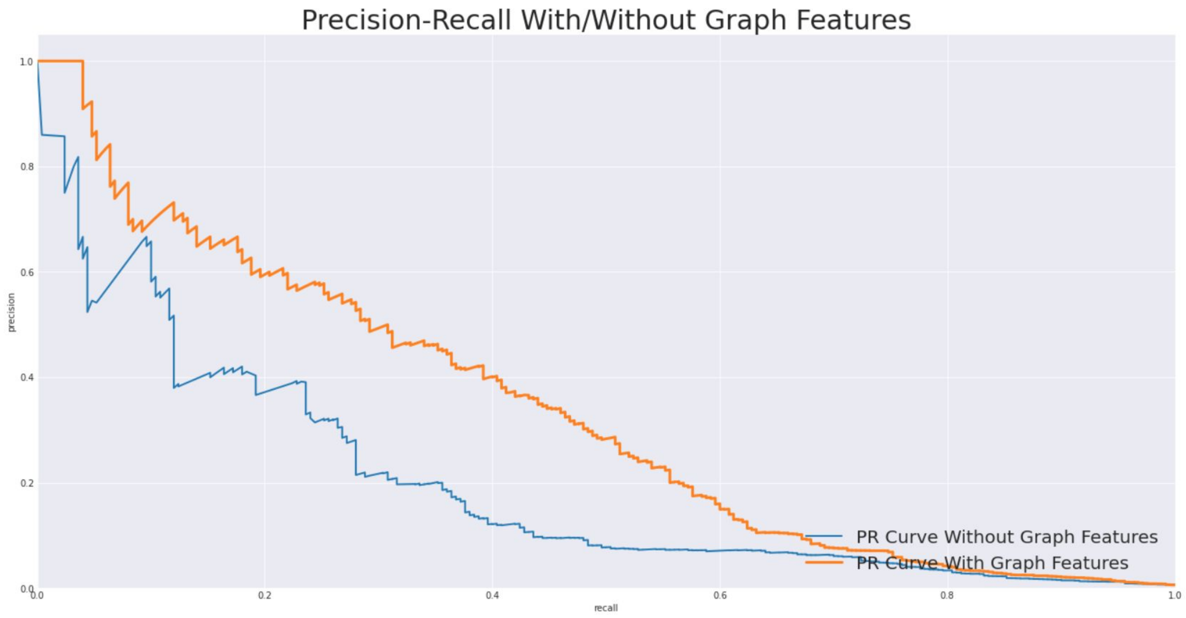
🔷 Use GNN model for predictions (GraphSAGE, GAT, GCN etc.), if necessary - with hardware acceleration (GPU





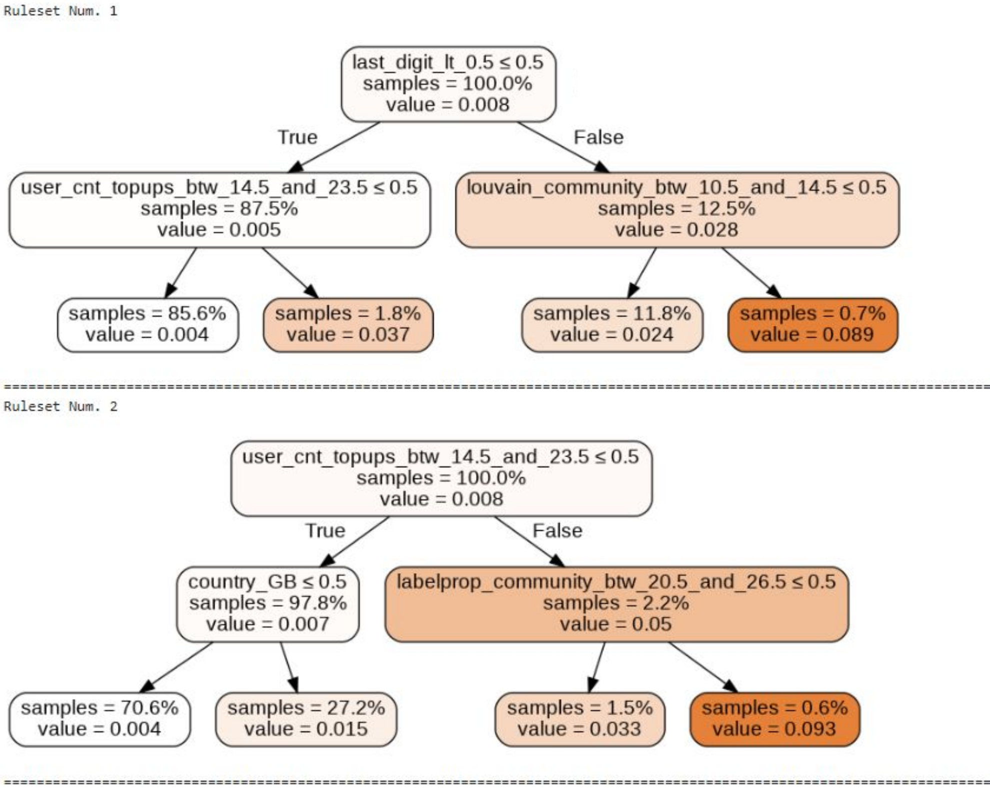
Impact of Graph Features – Same XGBoost Model



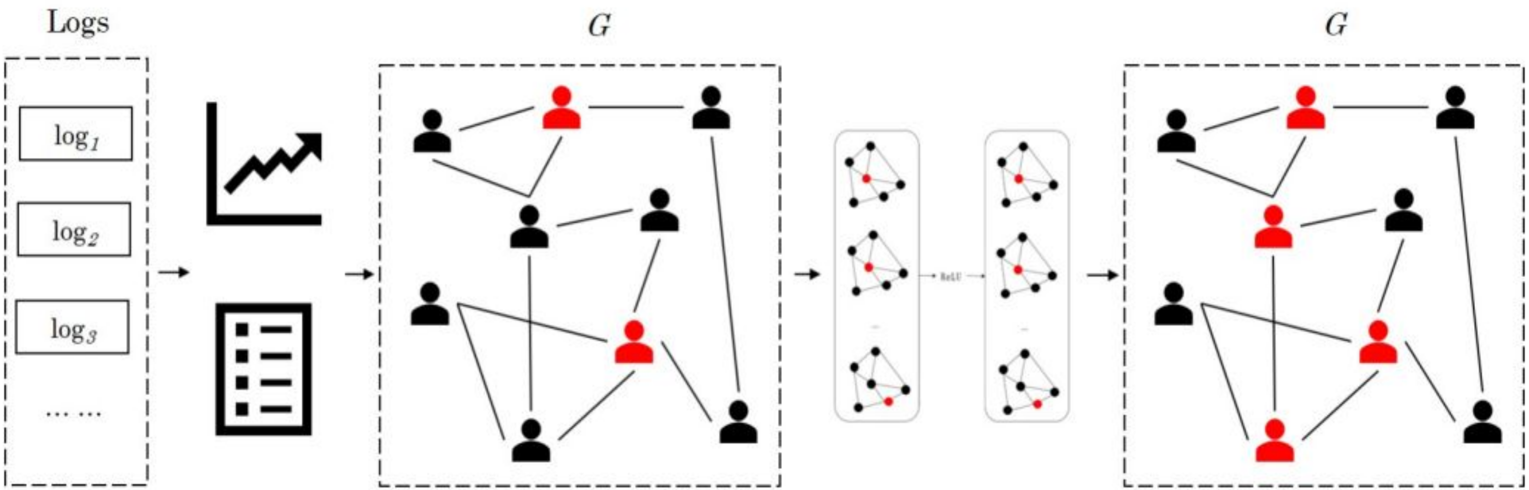


Fraud Detection with Graph Features

- 🔷 Run Supervised Discretization of Numerical Features (MDL, DecisionTree)
- 🔷 Apply One-Hot Encoding to get Binary Features
- 🔷 Use XGBFir library to generate useful Feature Interactions (with Monotonicity Constraints)
- 🔷 Add them as Binary Features back to OHE dataset
- 🔷 Train Online Classifer with SGD (or FTRL)



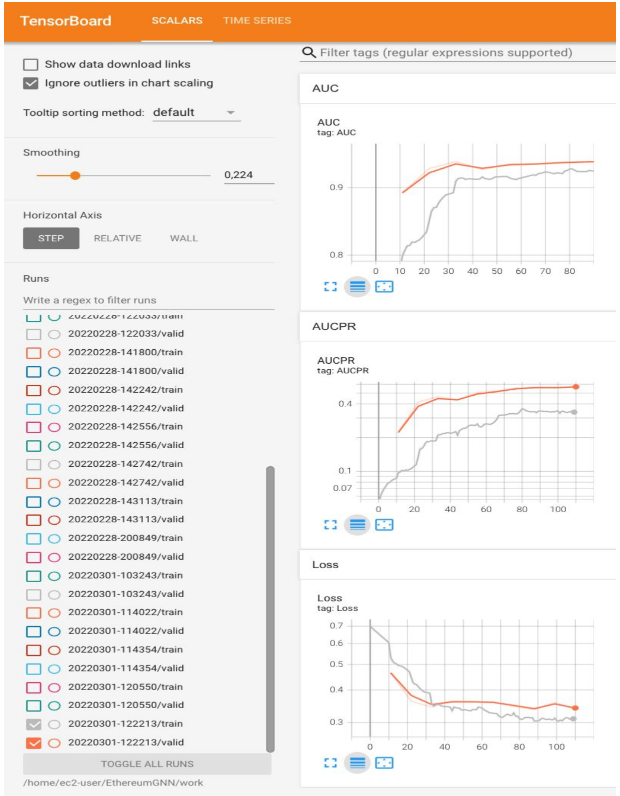
GNN-based Fraud Detection



(1) Graph Construction. (2) Training GNN on the Graph. (3) Classifying Unlabeled Nodes.

Key idea: the connected nodes are similar (homophily assumption)

Neighbor Loading and Model Training



Define Train Mini-Batch Loader

```
[7]: from tgml.dataloaders import NeighborLoader
```

```
[8]: train_loader = NeighborLoader(
         graph=tgraph,
         tmp_id="tmp_id",
         v_in_feats="in_degree:int,out_degree:int,send_amount:double,send_min:double,recv_amount:double,recv_min:double,pagerank:double",
         v_out_labels="is_fraud:bool",
         v_extra_feats="is_training:bool",
         output_format="PyG",
         batch_size=hp["batch_size"],
         num_neighbors=hp["num_neighbors"],
         num_hops=hp["num_hops"],
         filter_by = "is_training",
         shuffle=True,
         timeout=600000
     )
```

```
Installing and optimizing queries. It might take a minute if this is the first time you use this loader.
```

```
Start Training epoch: 9
Epoch 9, Train Batch 0, Loss 0.2951, AUC 0.9254, AUCPR 0.3485, Precision 0.4102, Recall 0.6225
Epoch 9, Train Batch 1, Loss 0.2882, AUC 0.9321, AUCPR 0.3547, Precision 0.4185, Recall 0.6067
Epoch 9, Train Batch 2, Loss 0.2800, AUC 0.9368, AUCPR 0.3784, Precision 0.4338, Recall 0.6129
Epoch 9, Train Batch 3, Loss 0.2853, AUC 0.9345, AUCPR 0.3747, Precision 0.4255, Recall 0.6168
Epoch 9, Train Batch 4, Loss 0.2851, AUC 0.9354, AUCPR 0.3822, Precision 0.4240, Recall 0.6179
Epoch 9, Train Batch 5, Loss 0.2907, AUC 0.9315, AUCPR 0.3688, Precision 0.4374, Recall 0.5736
Epoch 9, Train Batch 6, Loss 0.2965, AUC 0.9288, AUCPR 0.3658, Precision 0.4326, Recall 0.5486
Epoch 9, Train Batch 7, Loss 0.2994, AUC 0.9293, AUCPR 0.3618, Precision 0.4264, Recall 0.5392
Start validation epoch: 9
Epoch 9, Valid Loss 0.3013, Valid AUC 0.9516, Valid AUCPR 0.5181, Valid Precision 0.6451, Valid Recall 0.3685
```

Explainability of Phishing Patterns via Subgraphs

Account ID	in_degree	out_degree	send_amount	send_min	recv_amount	recv_min	pagerank
4303	0.246572	0.712181	0.733618	0.260401	0.254661	0.275504	0.722594
16424 (16424)	0.730702	0.268578	0.255523	0.245975	0.730027	0.294349	0.289655
16089 (16089)	0.760887	0.261722	0.273178	0.266076	0.717297	0.254782	0.256013