NEURAL INFORMATION PROCESSING SYSTEMS

GADBench: Revisiting and Benchmarking Supervised Graph Anomaly Detection

Jianheng Tang[1,2], Fengrui Hua[1], Ziqi Gao[1,2], Peilin Zhao[3], Jia Li[1,2]

[1]Hong Kong University of Science and Technology (Guangzhou),

[2]Hong Kong University of Science and Technology, [3]Tencent AI Lab

香港科技大学(广州) THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU) | 数据科学与分析 DATA SCIENCE AND ANALYTICS 信息枢纽 INFORMATION HUB

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Tencent AI Lab

Self-Introduction

- Slides: https://squareroot3.github.io/

社区介绍
Introduction

由公众号图学习研讨会以及深度学习与图网络发起，不定期邀请相关领域的专家，一线科研人员和顶会论文作者进行分享，希望能够给大家提供一个相互交流，研讨，和学习的平台.

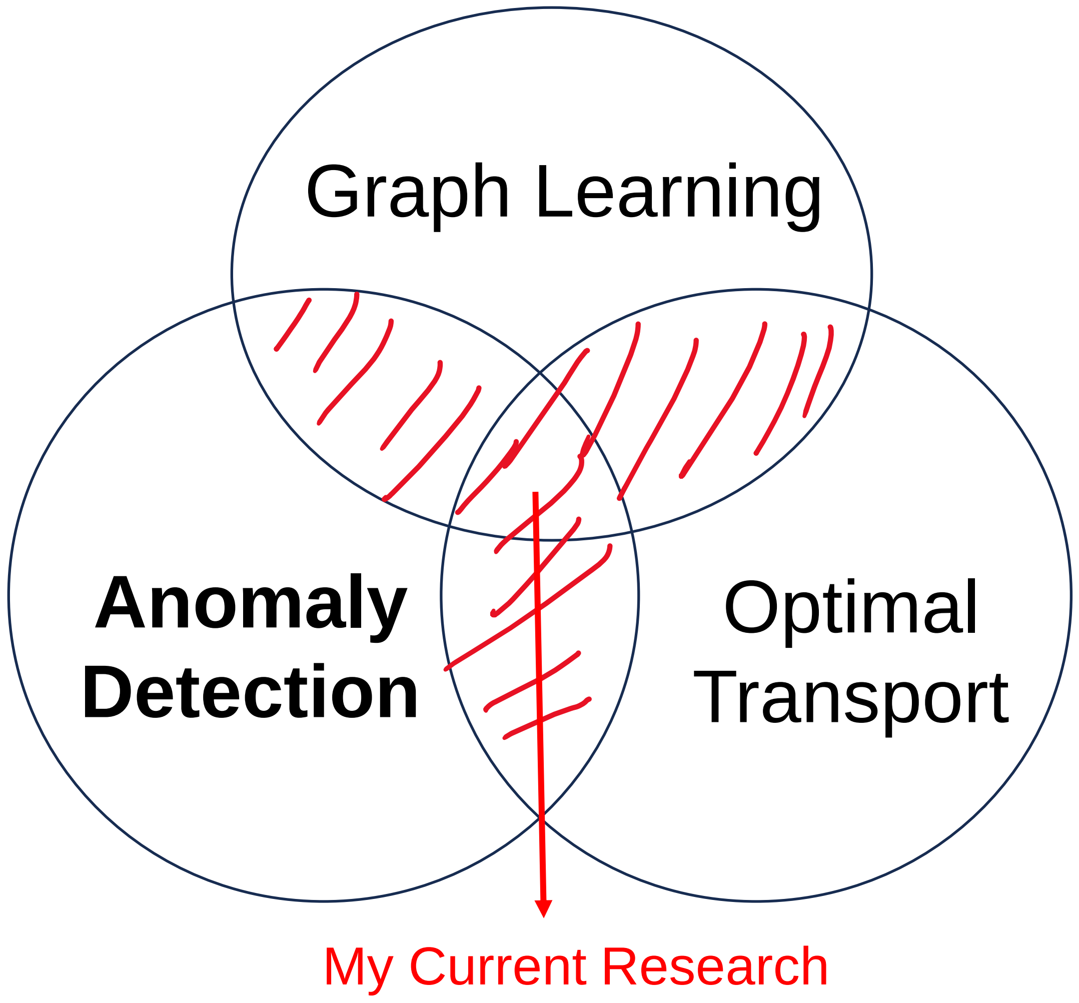
LOGS

请扫一扫

2924 0 01:06:04

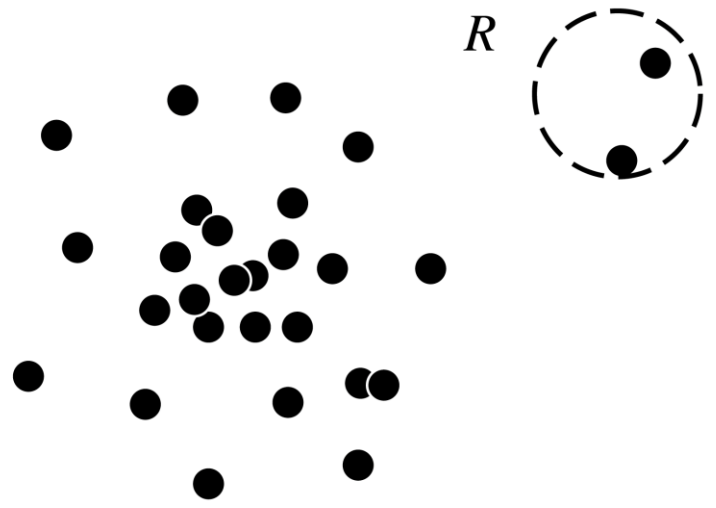
LOGS 第2022/07/16期 || 香港科技大学李佳：图异常检测再思考—我们究竟需要...

UP LOGS图学习研讨会 · 2022-7-17



Background: Anomaly Detection

- What are anomalies/outliers?
  - An anomaly or an outlier is a data object that deviates significantly from the majority of the objects, as if it was **generated by a different mechanism**. [1]
  - These anomalies can represent **errors**, but they can also indicate **critical**, **novel**, or **interesting** findings. They may contain valuable information about **abnormal behavior** or **new trends**. [2]



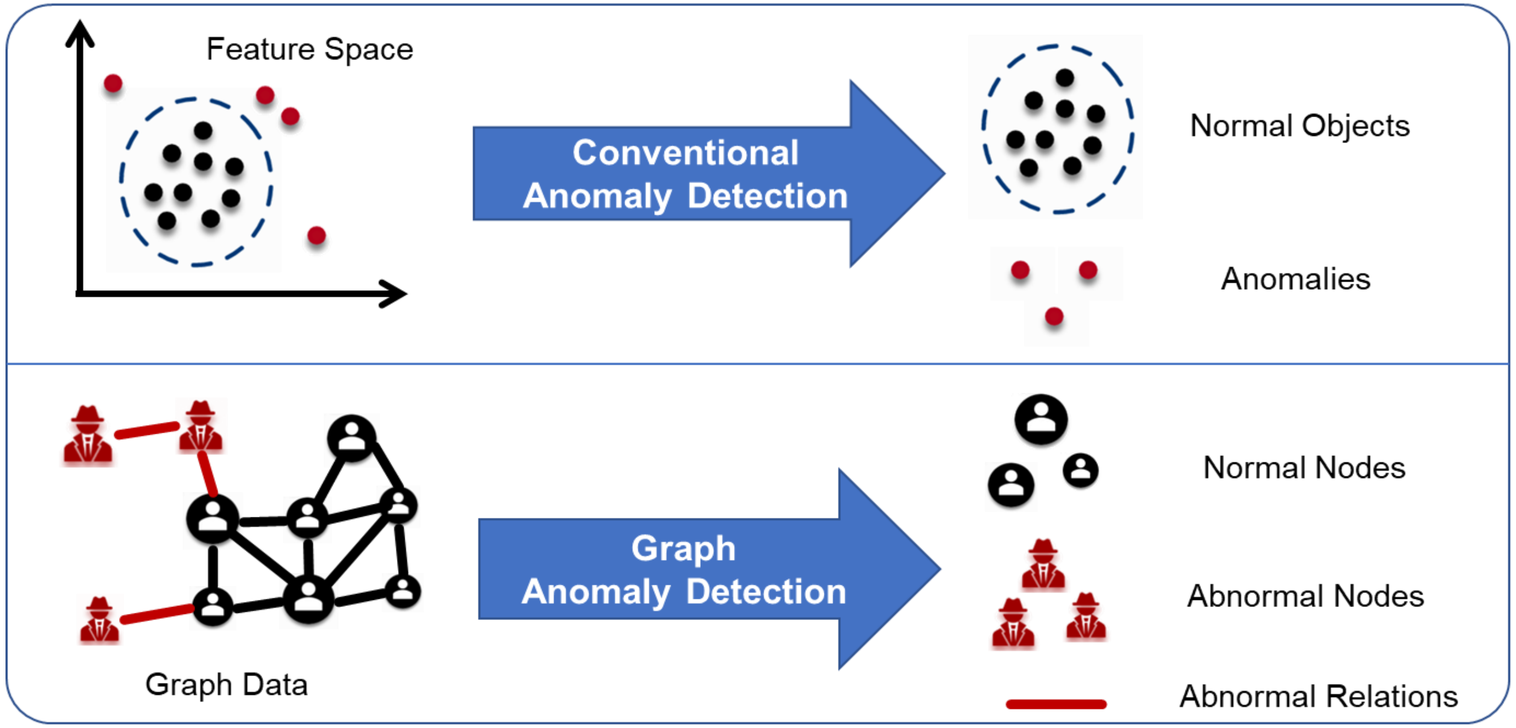
[1] Han, J., Kamber, M., and Pei, J. Data Mining: Concepts and Techniques, 3rd edition.
[2] GPT4

Background: Anomaly Detection

- Compare with other concepts
  - **Novel Detection**: The training data is **not** polluted by outliers and we are interested in detecting whether a **new** observation is an outlier.
  - **Out-Of-Distribution Detection**: It is about identifying data that is different from the data the model was trained on, **regardless** of how **unusual** or **rare** it is within its own context.
  - **Imbalance Classification**: It is a specific challenge in anomaly detection.
  - **Fraud Detection**: It is a specific application of anomaly detection that focuses on identifying fraudulent activities.

Graph-based Anomaly Detection

Graph-based Anomaly Detection (GAD) is the process of identifying **uncommon** graph objects, such as **nodes, edges, or substructures**, that significantly deviate from **the majority of** reference objects within a graph database.



Why Graph?

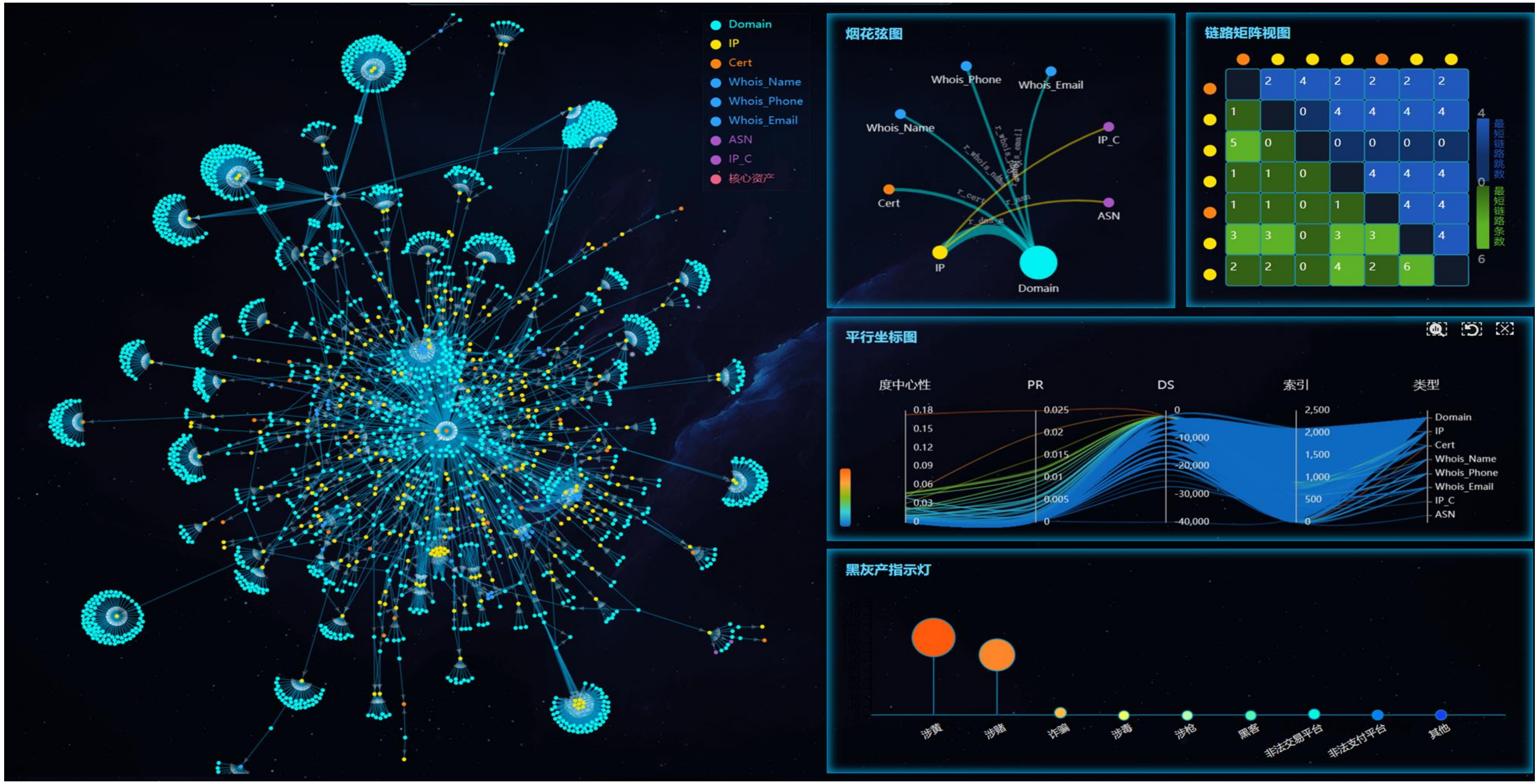
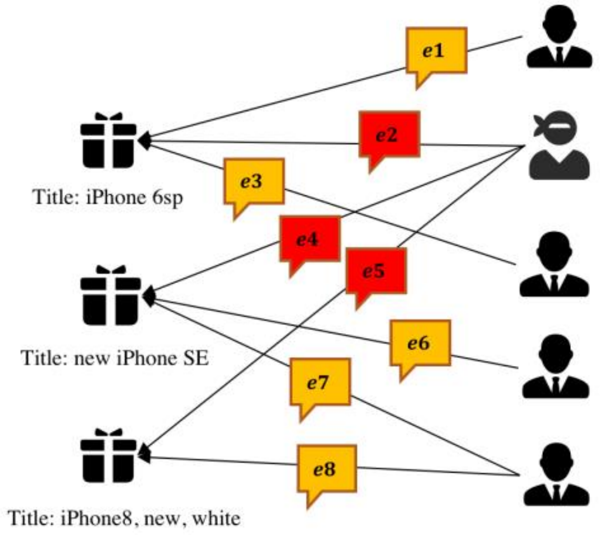


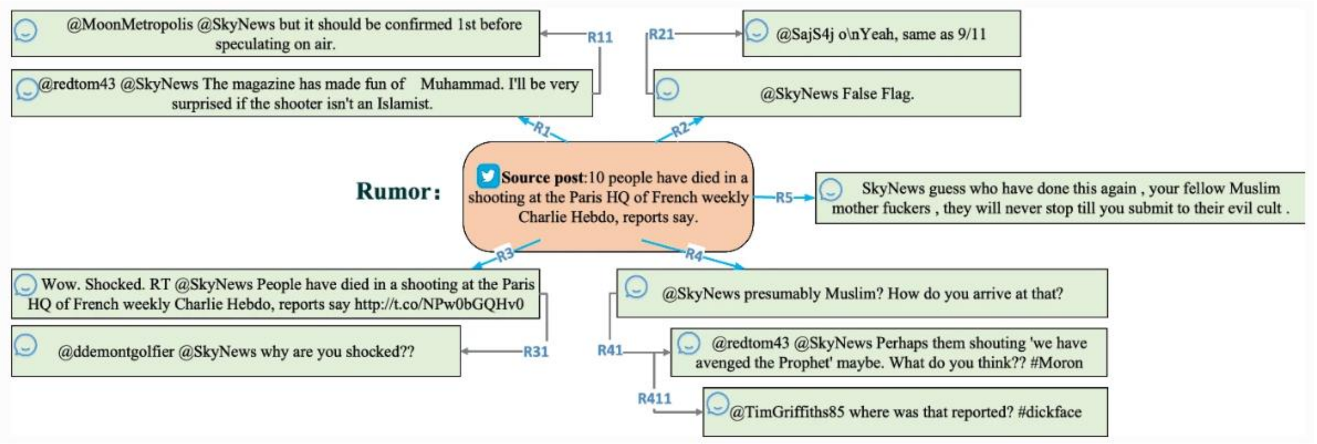
Figure credit to https://chinavis.org/2022/challenge.html

Application: GAD meets Language

- Spam-Review Filtering [1]



- Fake News/Rumor Detection [2]



Source post:10 people have died in a shooting at the Paris HQ of French weekly Charlie Hebdo, reports say.

@MoonMetropolis @SkyNews but it should be confirmed 1st before speculating on air.

@redtom43 @SkyNews The magazine has made fun of Muhammad. I'll be very surprised if the shooter isn't an Islamist.

@SajS4j o\nYeah, same as 9/11

@SkyNews False Flag.

SkyNews guess who have done this again , your fellow Muslim mother fuckers , they will never stop till you submit to their evil cult .

Wow. Shocked. RT @SkyNews People have died in a shooting at the Paris HQ of French weekly Charlie Hebdo, reports say http://t.co/NPw0bGQHv0

@ddemontgolfier @SkyNews why are you shocked??

@SkyNews presumably Muslim? How do you arrive at that?

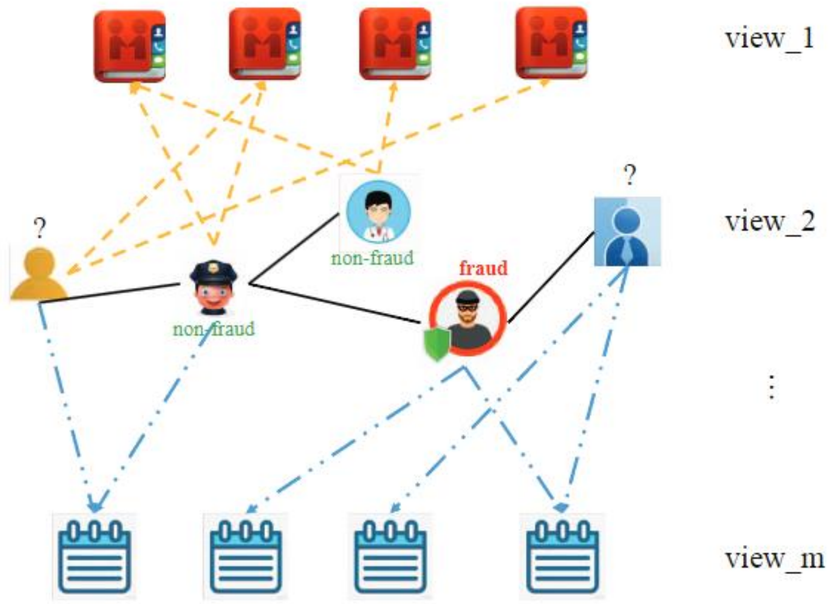
@redtom43 @SkyNews Perhaps them shouting 'we have avenged the Prophet' maybe. What do you think?? #Moron

@TimGriffiths85 where was that reported? #dickface

[1] (CIKM'19) Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks
[2] (AAAI'20) A Semi-Supervised Graph Attentive Network for Financial Fraud Detection

Application: GAD meets Fintech

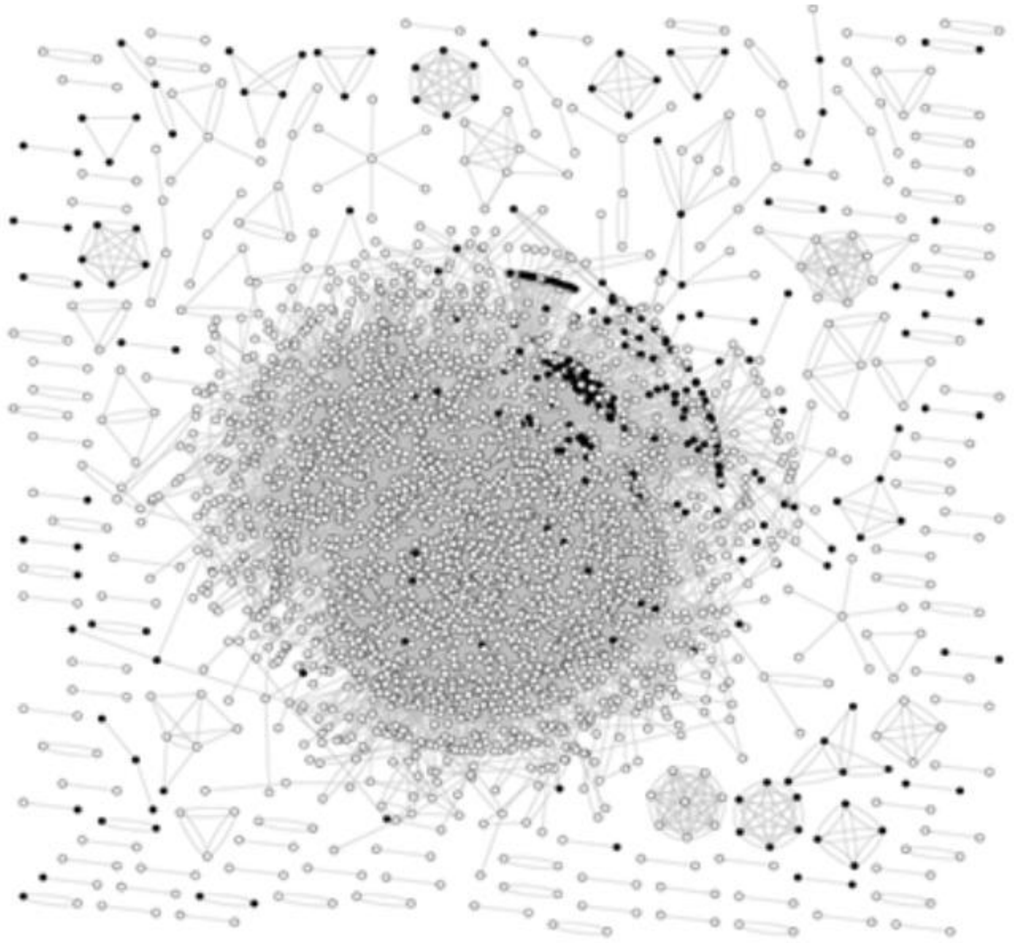
- Anti-Money Laundering [1]
- Financial Fraud Prevention [2]



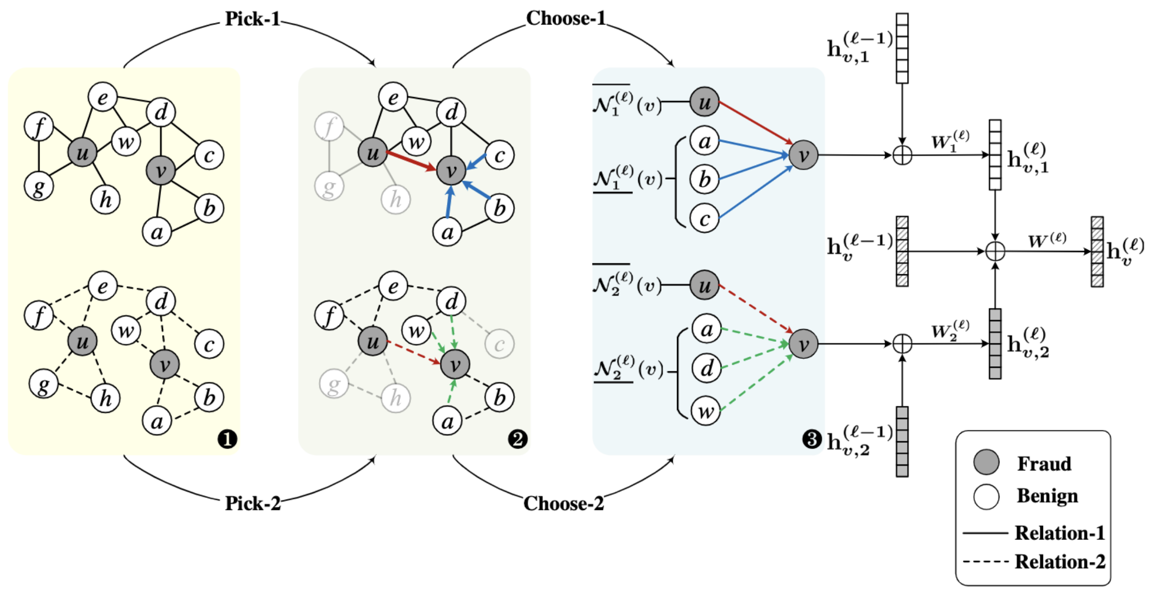
[1] (ADF@KDD'19) Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics.
[2] (ICDM'19) A Semi-Supervised Graph Attentive Network for Financial Fraud Detection

Challenges of GAD: Imbalance

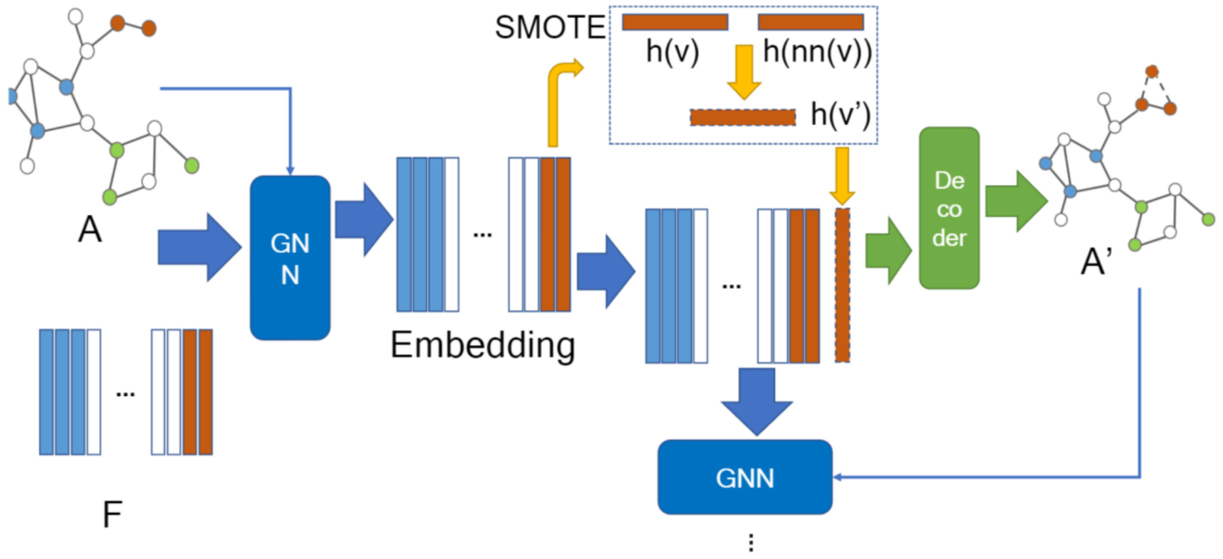
- Anomalous nodes typically constitute a small part of the total nodes, resulting in a significant label **imbalance**.



Related Work: Imbalance-aware GNN



Pick-and-Choose GNN [1]



GraphSMOTE [2]

[1] (WWW’21) Pick and choose: a gnn-based imbalanced learning approach for fraud detection.
[2] (WSDM’21) GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks

Challenges of GAD: Camouflage

- Anomalous nodes can effectively **camouflage** their relations and features to be similar to normal nodes.
  - This demands attention to intentionally manipulated edges and node features.

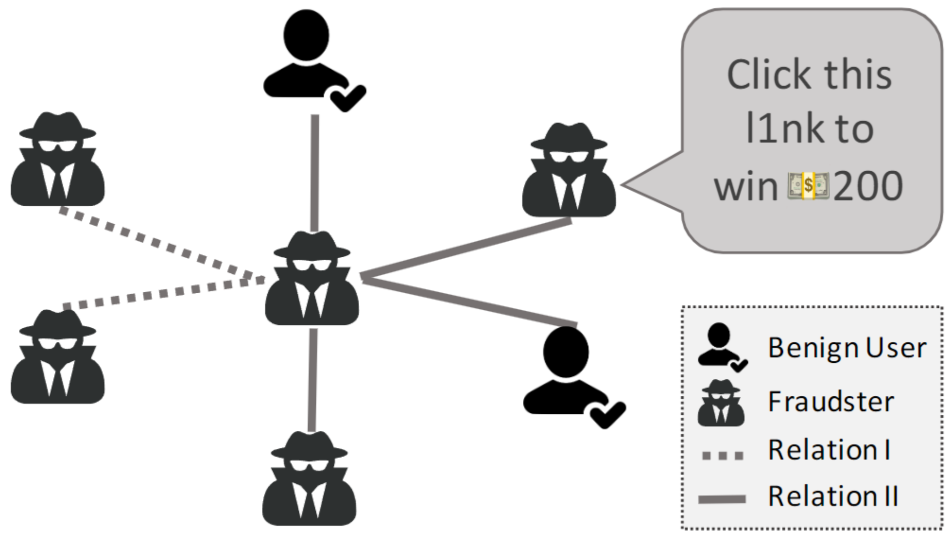
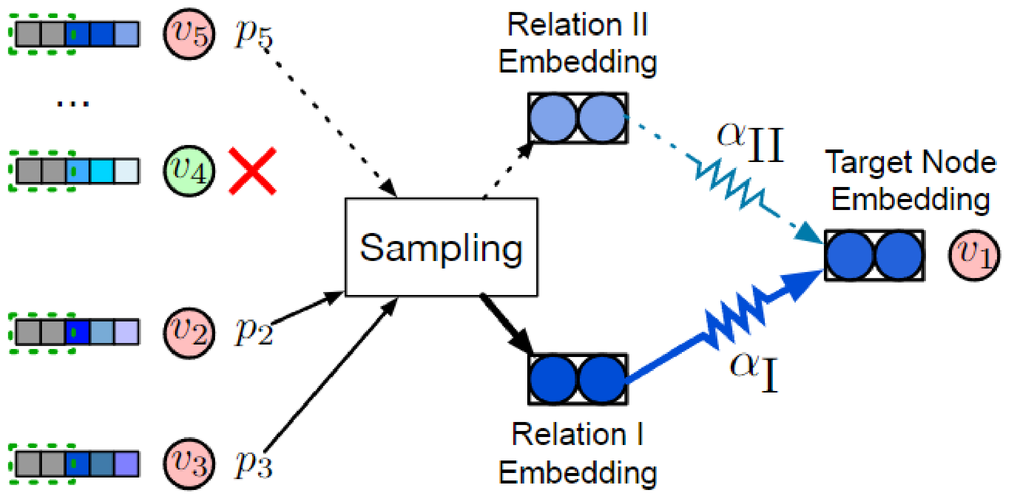
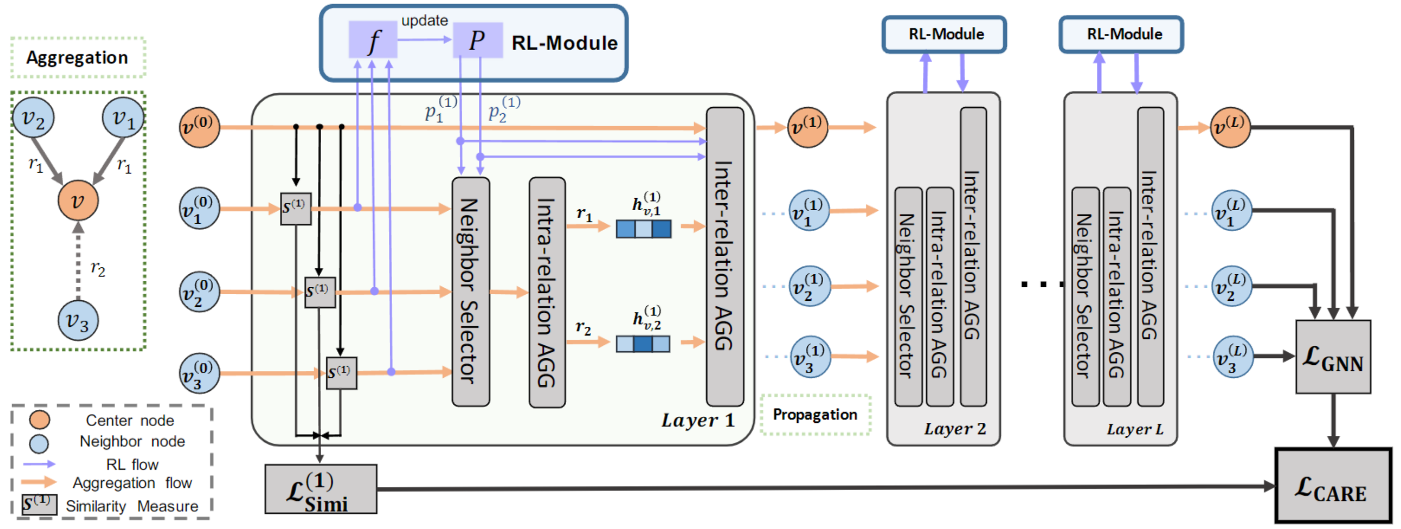


Figure 1: Two types of fraudster camouflage. (1) Feature camouflage: fraudsters add special characters to the text and make it delusive for feature-based spam detectors. (2) Relation camouflage: center fraudster connects to many benign entities under Relation II to attenuate its suspiciousness.

Related Work: Camouflage-resist GNN



GraphConsis [1]

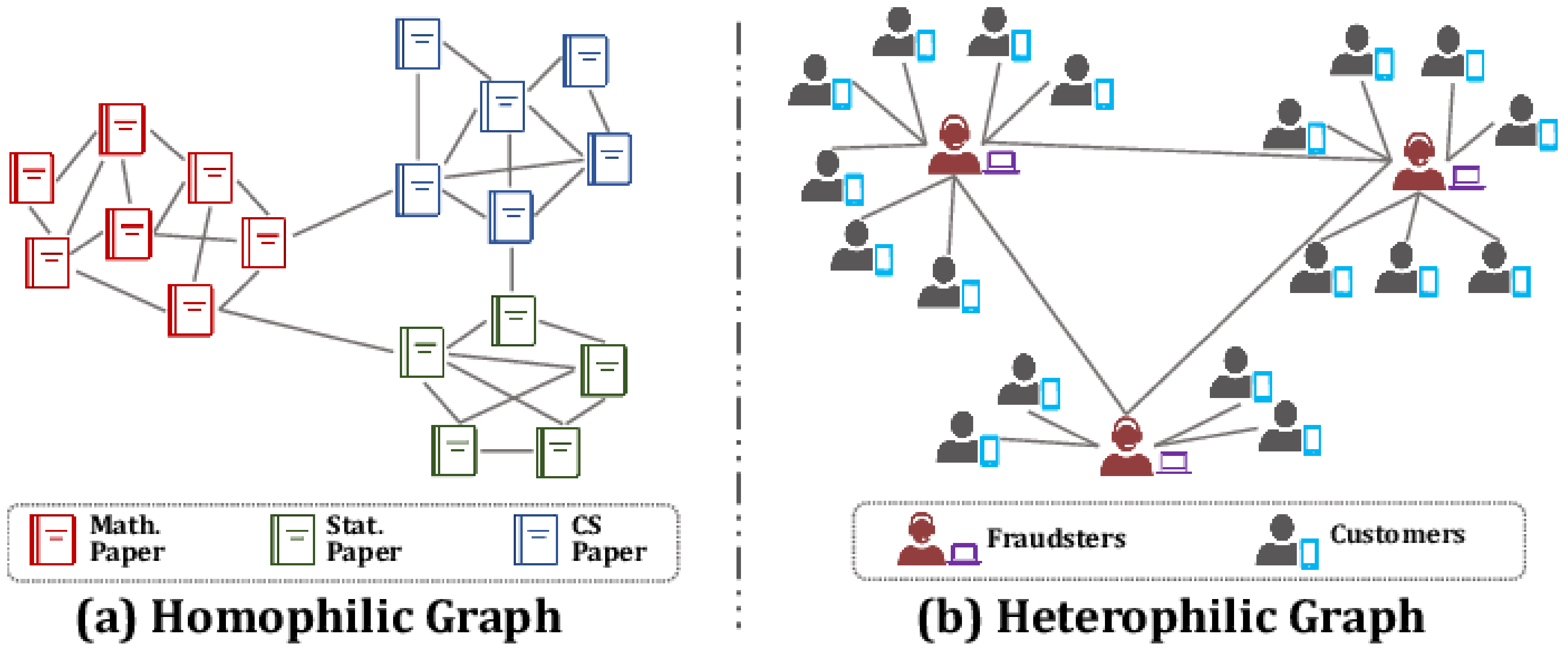


Care-GNN [2]

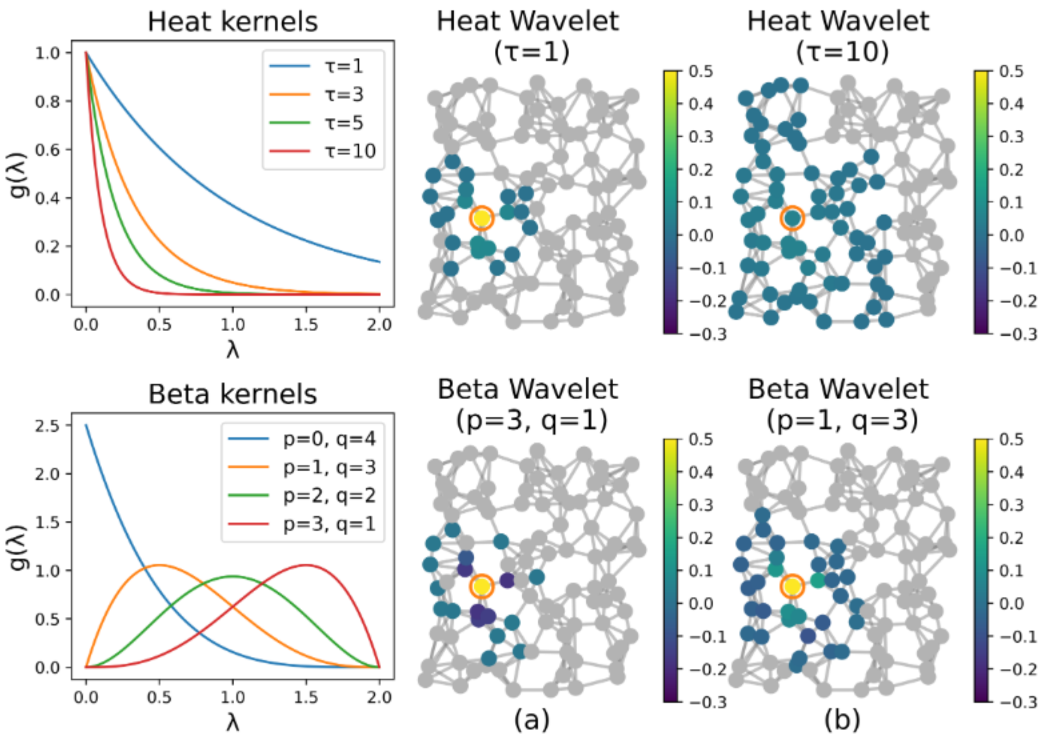
[1] (SIGIR'20) Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection
[2] (CIKM'20) Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters

Challenges of GAD: Heterophily

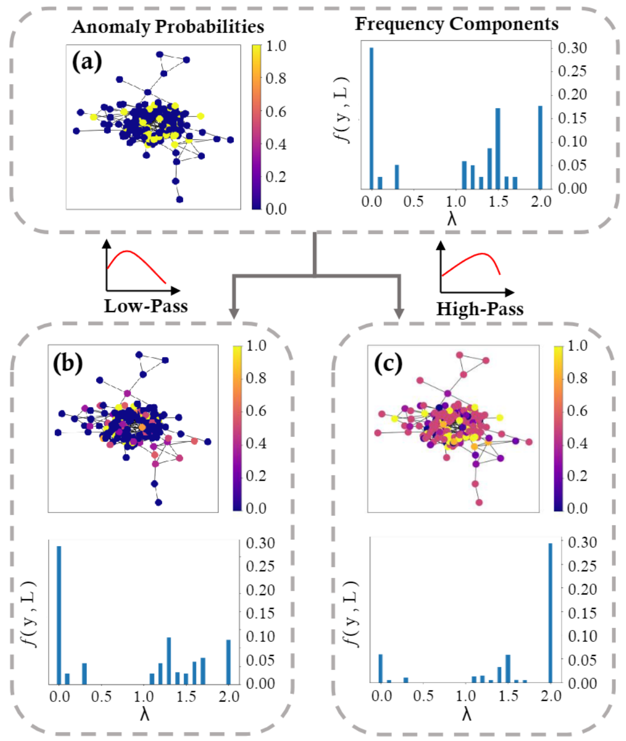
- The graph containing anomalies is often highly **heterophily**, with connected nodes having distinct attributes and labels.
  - This necessitates strategies to handle neighborhood feature disparities during message passing.



Related Work: Heterophily-aware GNN



BWGNN [1]



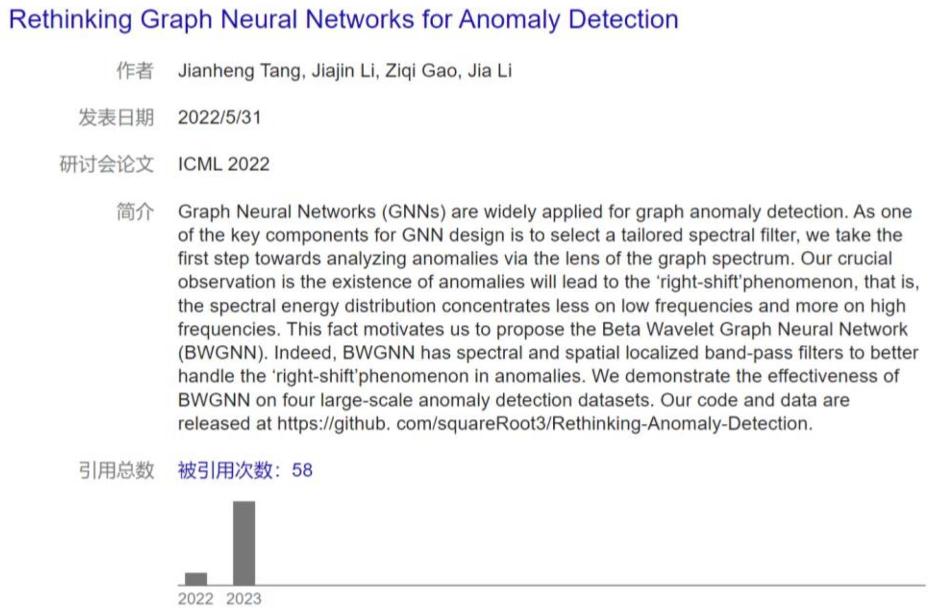
GHRN [2]

[1] (ICML’22) Rethinking Graph Neural Networks for Anomaly Detection
[2] (WWW’23) Addressing Heterophily in Graph Anomaly Detection: A Perspective of Graph Spectrum

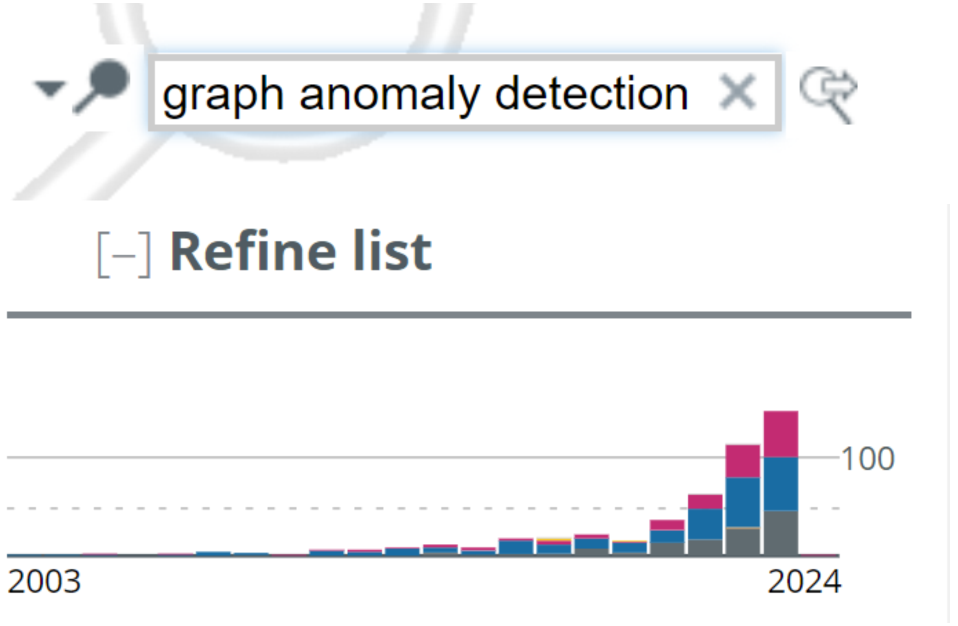
Motivation I: Are we really making progress?

Hundreds of GAD methods, ranging from traditional approaches to modern Graph Neural Networks (GNNs), have been developed.

- Which of them are truly effective?
- Most models were tested on a limited number of datasets, leaving their performance in a **standard**, **comprehensive** setting largely unexplored.


Rethinking Graph Neural Networks for Anomaly Detection

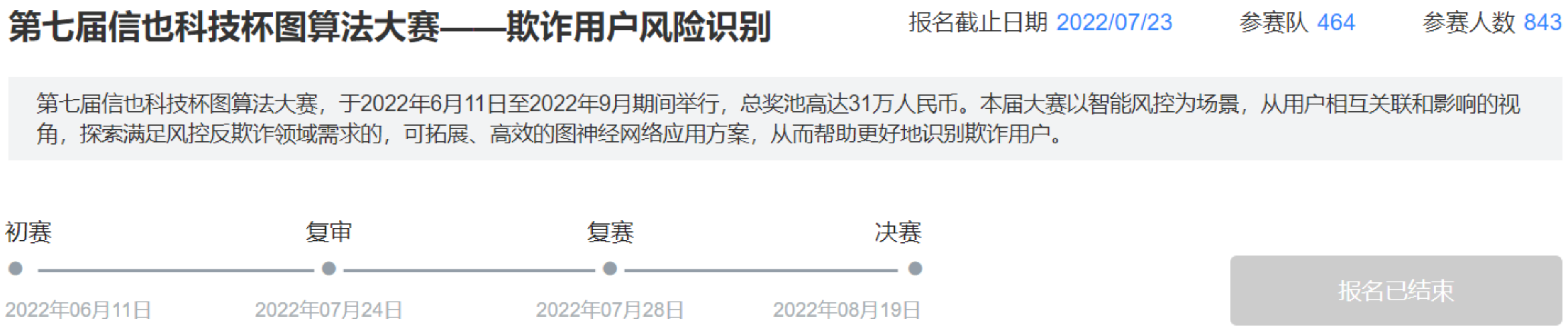
作者	Jianheng Tang, Jiajin Li, Ziqi Gao, Jia Li
发表日期	2022/5/31
研讨会论文	ICML 2022
简介	Graph Neural Networks (GNNs) are widely applied for graph anomaly detection. As one of the key components for GNN design is to select a tailored spectral filter, we take the first step towards analyzing anomalies via the lens of the graph spectrum. Our crucial observation is the existence of anomalies will lead to the 'right-shift'phenomenon, that is, the spectral energy distribution concentrates less on low frequencies and more on high frequencies. This fact motivates us to propose the Beta Wavelet Graph Neural Network (BWGNN). Indeed, BWGNN has spectral and spatial localized band-pass filters to better handle the 'right-shift'phenomenon in anomalies. We demonstrate the effectiveness of BWGNN on four large-scale anomaly detection datasets. Our code and data are released at https://github. com/squareRoot3/Rethinking-Anomaly-Detection.
引用总数	被引用次数：58



Motivation II: Academia-Industry Gap

Despite the current academic dominance of Graph Neural Network (GNN)-based methods, industry professionals appear to place greater trust in non-deep-learning techniques.

- In the 7th Finvolution Data Science Competition, the task is to detect fraudulent users in an industry financial dataset.
- **7** out of top **10** teams, including the **champion**, used tree ensembles in their solution.



第七届信也科技杯图算法大赛——欺诈用户风险识别

报名截止日期 2022/07/23　参赛队 464　参赛人数 843

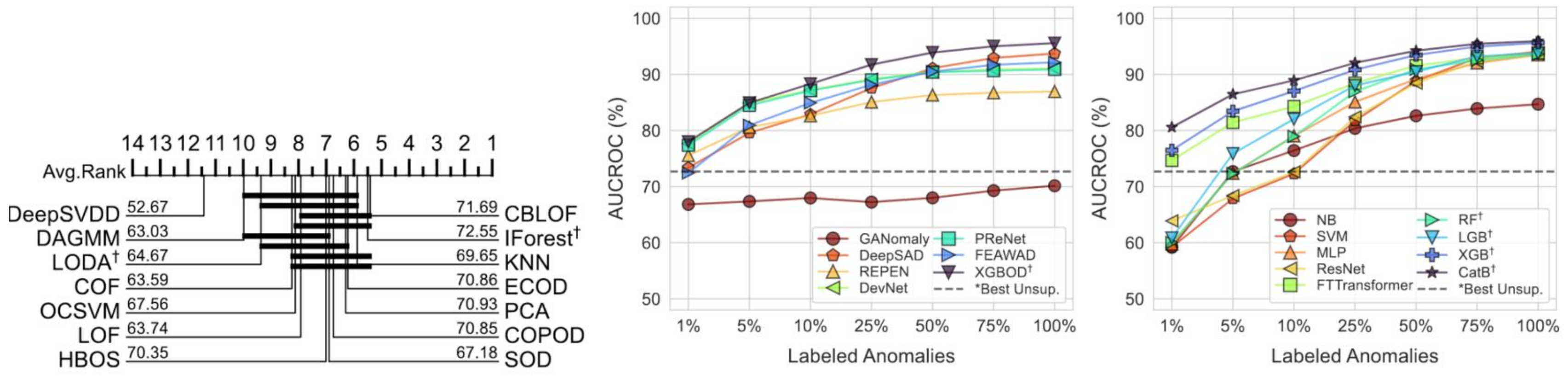
第七届信也科技杯图算法大赛，于2022年6月11日至2022年9月期间举行，总奖池高达31万人民币。本届大赛以智能风控为场景，从用户相互关联和影响的视角，探索满足风控反欺诈领域需求的，可拓展、高效的图神经网络应用方案，从而帮助更好地识别欺诈用户。

初赛	复审	复赛	决赛
2022年06月11日	2022年07月24日	2022年07月28日	2022年08月19日

https://ai.ppdai.com/mirror/goToMirrorDetailSix?mirrorId=28

Motivation III: Current Benchmark Results

- ADBench shows that non-deep-learning methods, especially ensemble trees, perform better in unsupervised, semi-supervised, and supervised settings.



(a) Avg. rank (lower the better) and avg. AUCROC (on each line) of unsupervised methods; groups of algorithms not statistically different are connected horizontally.

(b) Avg. AUCROC (on 57 datasets) vs. % of labeled anomalies (x-axis); semi-supervised (left) and fully-supervised (right). Most label-informed algorithms outperform the best *unsupervised* algorithm CBLOF (denoted as the dashed line) with 10% labeled anomalies.

[1] (NeurIPS'22) ADBench: Anomaly Detection Benchmark

Motivation III: Current Benchmark Results

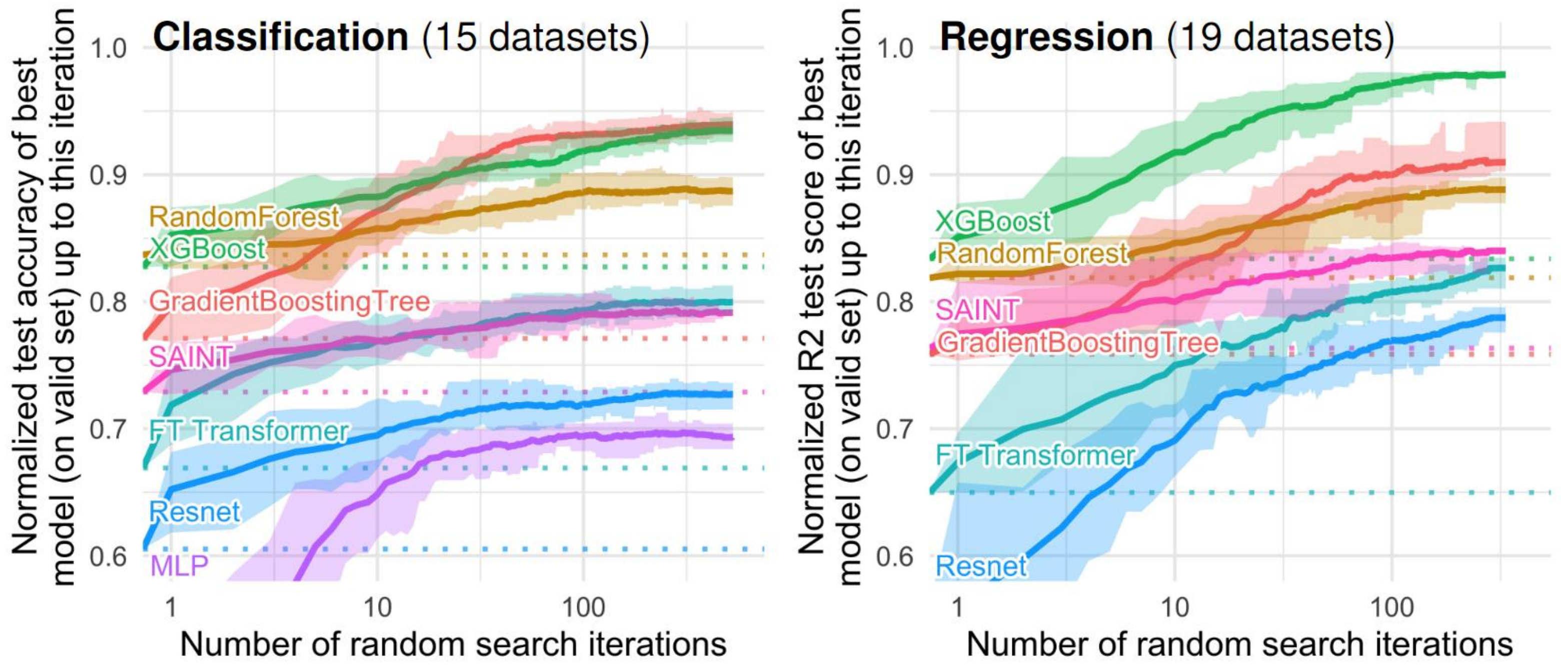
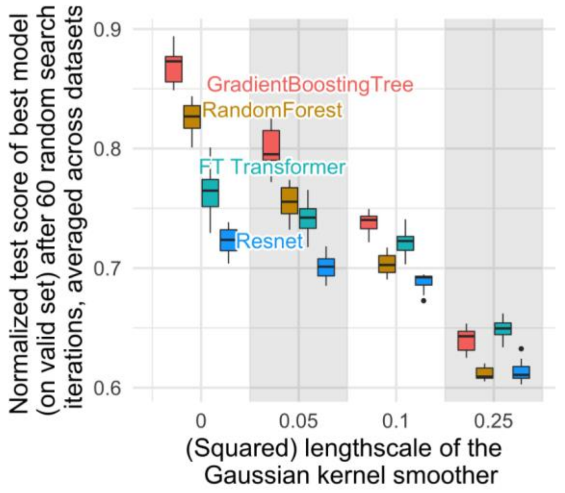


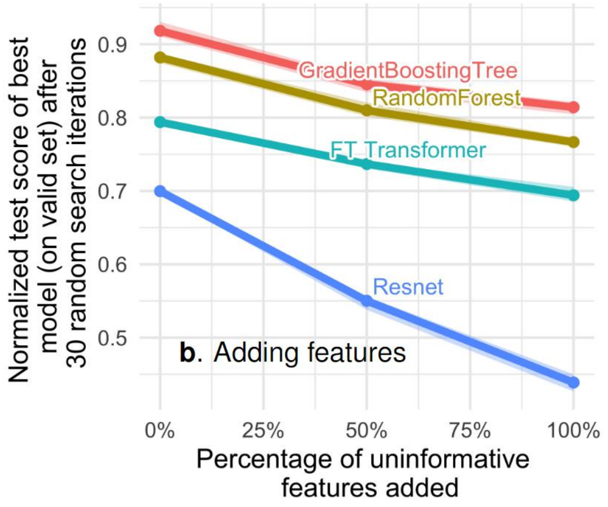
Figure 1: **Benchmark on medium-sized datasets, with only numerical features**. Dotted lines correspond to the score of the default hyperparameters, which is also the first random search iteration. Each value corresponds to the test score of the best model (on the validation set) after a specific number of random search iterations, averaged on 15 shuffles of the random search order. The ribbon corresponds to the minimum and maximum scores on these 15 shuffles.

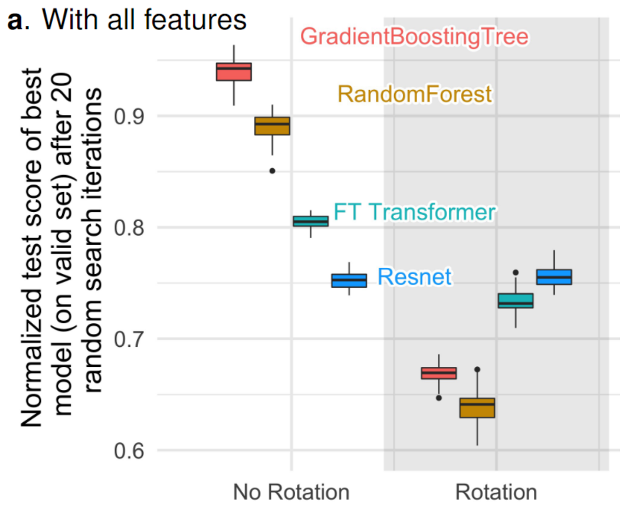
[1] (NeurIPS'22) Why do tree-based models still outperform deep learning on tabular data?

Neural Network v.s. Tree Ensembles on tabular data

- [1] explain why tree models still outperform deep learning on tabular data from the following perspectives:
  - NNs are biased to overly smooth solutions
  - Uninformative features affect more MLP-like NNs
  - Data are non invariant by rotation, so should be learning procedures







[1] (NeurIPS'22) Why do tree-based models still outperform deep learning on tabular data?

Problems in Existing GAD Benchmarks

With a long history of traditional Graph Anomaly Detection (GAD) algorithms and recently popular Graph Neural Networks (GNNs), it is still not clear:

(1) how they perform under a standard comprehensive setting,
(2) whether GNNs outperform traditional algorithms such as **tree ensembles**,
(3) their efficiency on large-scale graphs.

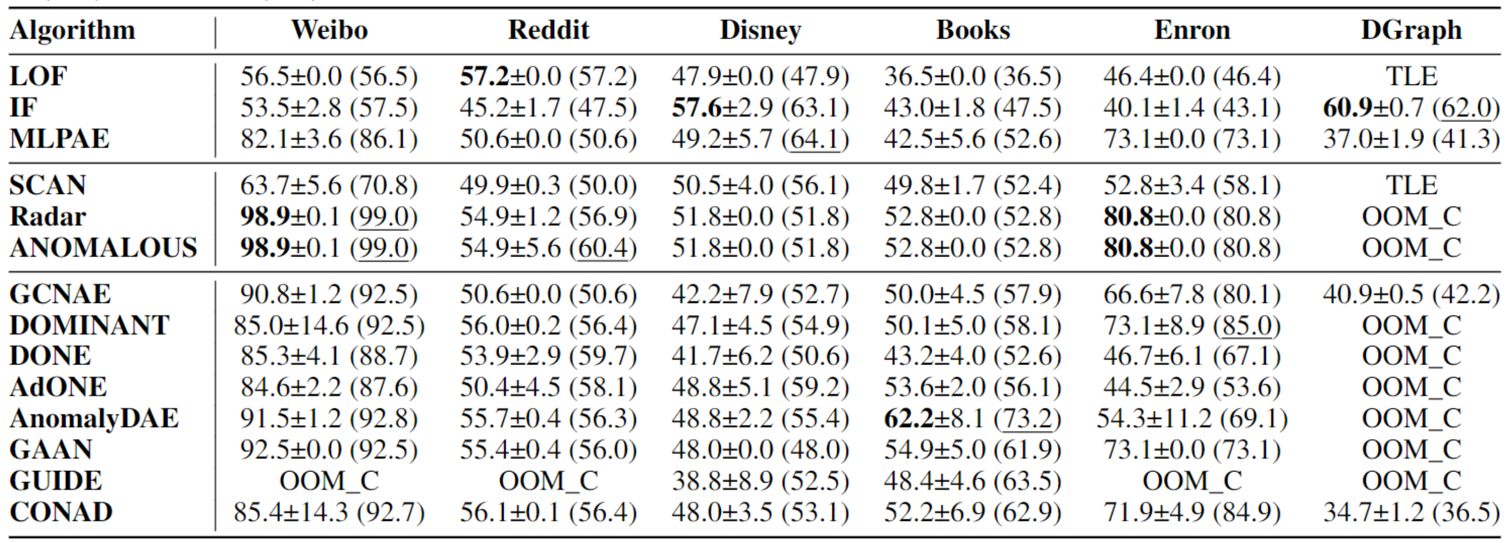
Problems in Existing GAD Benchmarks

With a long history of traditional Graph Anomaly Detection (GAD) algorithms and recently popular Graph Neural Networks (GNNs), it is still not clear:

(1) how they perform under a standard comprehensive setting,
(2) whether GNNs outperform traditional algorithms such as **tree ensembles**,
(3) their efficiency on large-scale graphs.

The latest benchmark available for GAD, **BOND** [1], only evaluates **unsupervised** methods.

However, these methods have **inferior** performance on large-scale real-world datasets (e.g., DGraph).



Algorithm	Weibo	Reddit	Disney	Books	Enron	DGraph
LOF	56.5±0.0 (56.5)	**57.2**±0.0 (57.2)	47.9±0.0 (47.9)	36.5±0.0 (36.5)	46.4±0.0 (46.4)	TLE
IF	53.5±2.8 (57.5)	45.2±1.7 (47.5)	**57.6**±2.9 (63.1)	43.0±1.8 (47.5)	40.1±1.4 (43.1)	**60.9**±0.7 (62.0)
MLPAE	82.1±3.6 (86.1)	50.6±0.0 (50.6)	49.2±5.7 (64.1)	42.5±5.6 (52.6)	73.1±0.0 (73.1)	37.0±1.9 (41.3)
SCAN	63.7±5.6 (70.8)	49.9±0.3 (50.0)	50.5±4.0 (56.1)	49.8±1.7 (52.4)	52.8±3.4 (58.1)	TLE
Radar	**98.9**±0.1 (99.0)	54.9±1.2 (56.9)	51.8±0.0 (51.8)	52.8±0.0 (52.8)	**80.8**±0.0 (80.8)	OOM_C
ANOMALOUS	**98.9**±0.1 (99.0)	54.9±5.6 (60.4)	51.8±0.0 (51.8)	52.8±0.0 (52.8)	**80.8**±0.0 (80.8)	OOM_C
GCNAE	90.8±1.2 (92.5)	50.6±0.0 (50.6)	42.2±7.9 (52.7)	50.0±4.5 (57.9)	66.6±7.8 (80.1)	40.9±0.5 (42.2)
DOMINANT	85.0±14.6 (92.5)	56.0±0.2 (56.4)	47.1±4.5 (54.9)	50.1±5.0 (58.1)	73.1±8.9 (85.0)	OOM_C
DONE	85.3±4.1 (88.7)	53.9±2.9 (59.7)	41.7±6.2 (50.6)	43.2±4.0 (52.6)	46.7±6.1 (67.1)	OOM_C
AdONE	84.6±2.2 (87.6)	50.4±4.5 (58.1)	48.8±5.1 (59.2)	53.6±2.0 (56.1)	44.5±2.9 (53.6)	OOM_C
AnomalyDAE	91.5±1.2 (92.8)	55.7±0.4 (56.3)	48.8±2.2 (55.4)	**62.2**±8.1 (73.2)	54.3±11.2 (69.1)	OOM_C
GAAN	92.5±0.0 (92.5)	55.4±0.4 (56.0)	48.0±0.0 (48.0)	54.9±5.0 (61.9)	73.1±0.0 (73.1)	OOM_C
GUIDE	OOM_C	OOM_C	38.8±8.9 (52.5)	48.4±4.6 (63.5)	OOM_C	OOM_C
CONAD	85.4±14.3 (92.7)	56.1±0.1 (56.4)	48.0±3.5 (53.1)	52.2±6.9 (62.9)	71.9±4.9 (84.9)	34.7±1.2 (36.5)

[1] (NeurIPS'22) BOND: Benchmarking Unsupervised Outlier Node Detection on Static Attributed Graphs.

Supervised GAD vs. Unsupervised GAD

Why Supervised?

- **Performance**: Many GAD models rely on labeled data to boost their performance.
- **Model Selection**: Hyper-parameter search and model selection usually require labels.

Why Unsupervised?

- **Label Budget & Quality**: the labels are expected to be noisy and of varying quality depending on the annotator.
- **Generalization Capability**: Supervised anomaly detection may be limited in finding novel anomalies.

Semi-supervised setting may strike a balance between label annotation budgets and model performance.

Our semi-supervised setting: The training/validation set has 20 anomalies and 80 normal nodes.

Introduction of GADBench

- We introduce GADBench, the first comprehensive benchmark for **fully- and semi-supervised** anomalous node detection on static attributed graphs.

- To ensure a rigorous and **fair comparison**, we implement enhancements from dataset selection, metric utilization, model training, and hyperparameter tuning.

- We integrate all models, datasets, and evaluation protocols mentioned into an **open-source repository: https://github.com/squareRoot3/GADBench**

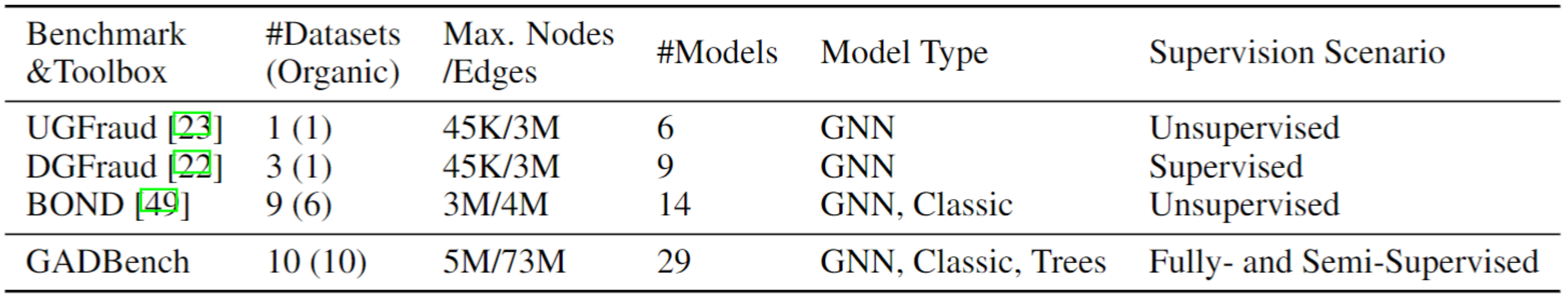
Introduction of GADBench

- We introduce GADBench, the first comprehensive benchmark for **fully- and semi-supervised** anomalous node detection on static attributed graphs.

- To ensure a rigorous and **fair comparison**, we implement enhancements from dataset selection, metric utilization, model training, and hyperparameter tuning.

- We integrate all models, datasets, and evaluation protocols mentioned into an **open-source repository:** **https://github.com/squareRoot3/GADBench**

Table 1: Comparison of existing GAD benchmarks in terms of datasets, models, and scenarios.



Benchmark &Toolbox	#Datasets (Organic)	Max. Nodes /Edges	#Models	Model Type	Supervision Scenario
UGFraud [23]	1 (1)	45K/3M	6	GNN	Unsupervised
DGFraud [22]	3 (1)	45K/3M	9	GNN	Supervised
BOND [49]	9 (6)	3M/4M	14	GNN, Classic	Unsupervised
GADBench	10 (10)	5M/73M	29	GNN, Classic, Trees	Fully- and Semi-Supervised

Selected Models in GADBench

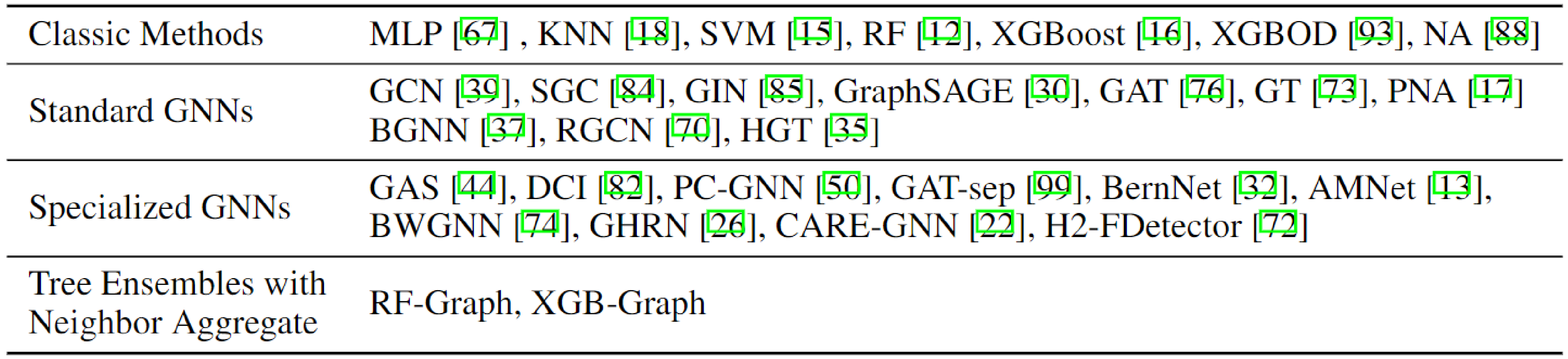


Table 2: Categorization of all models used in our evaluation.

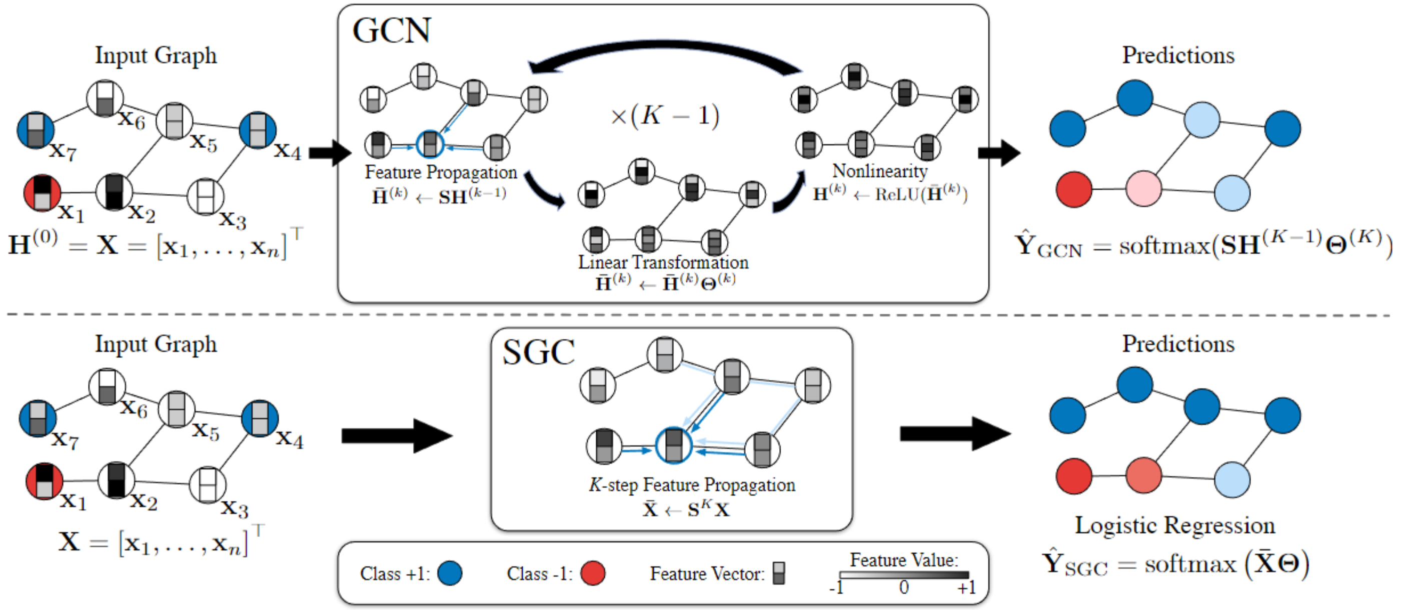
Classic Methods	MLP [67] , KNN [18], SVM [15], RF [12], XGBoost [16], XGBOD [93], NA [88]
Standard GNNs	GCN [39], SGC [84], GIN [85], GraphSAGE [30], GAT [76], GT [73], PNA [17] BGNN [37], RGCN [70], HGT [35]
Specialized GNNs	GAS [44], DCI [82], PC-GNN [50], GAT-sep [99], BernNet [32], AMNet [13], BWGNN [74], GHRN [26], CARE-GNN [22], H2-FDetector [72]
Tree Ensembles with Neighbor Aggregate	RF-Graph, XGB-Graph

Our evaluation encompasses a total of **29** models:

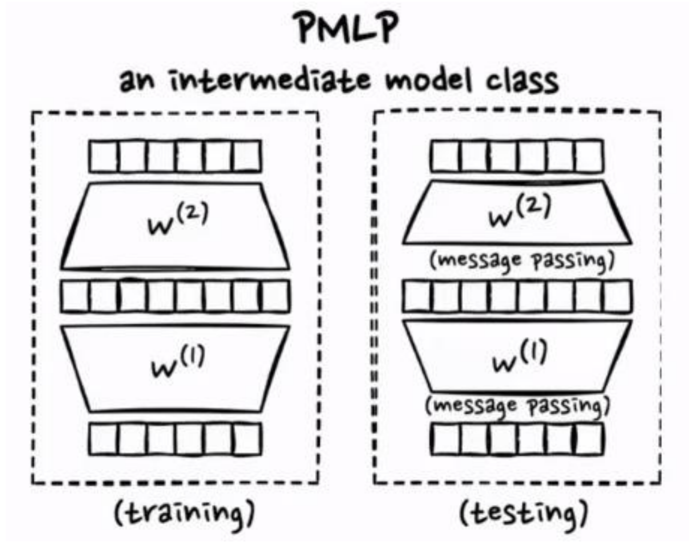
- 7 classic non-graph models
- 10 standard GNNs
- 10 state-of-the-art GNNs specifically designed for graph anomaly detection
- 2 tree ensembles with neighbor aggregation

Tree Ensembles With Neighbor Aggregation

- Inspired by a subclass of simplified GNNs with parameter-free massage passing, e.g., Simple Graph Convolution [1] and Propagational MLP [2]



Input Graph
GCN
$\times(K-1)$
Feature Propagation
$\bar{\mathbf{H}}^{(k)} \leftarrow \mathbf{S}\mathbf{H}^{(k-1)}$
Linear Transformation
$\bar{\mathbf{H}}^{(k)} \leftarrow \bar{\mathbf{H}}^{(k)}\Theta^{(k)}$
Nonlinearity
$\mathbf{H}^{(k)} \leftarrow \mathrm{ReLU}(\bar{\mathbf{H}}^{(k)})$
Predictions
$\mathbf{H}^{(0)} = \mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$
$\hat{\mathbf{Y}}_{\mathrm{GCN}} = \mathrm{softmax}(\mathbf{S}\mathbf{H}^{(K-1)}\Theta^{(K)})$
SGC
K-step Feature Propagation
$\bar{\mathbf{X}} \leftarrow \mathbf{S}^K\mathbf{X}$
$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$
Logistic Regression
$\hat{\mathbf{Y}}_{\mathrm{SGC}} = \mathrm{softmax}(\bar{\mathbf{X}}\Theta)$
Class +1:
Class -1:
Feature Vector:
Feature Value:
-1 0 +1



[1] Simplifying Graph Convolutional Networks
[2] LOGS第2023/06/03期|| 上海交通大学杨晨晓：连接MLP与GNN：探讨图神经网络天生的强大泛化性

Tree Ensembles With Neighbor Aggregation

- Our tree ensembles with neighbor aggregation adopt the following computational paradigm:

$$\boldsymbol{h}_{v_i}^{(l)} = \text{Aggregate}\{\boldsymbol{h}_{v_j}^{(l-1)} | v_i \in \text{Neighbor}(v_j)\}$$
$$\text{Score}(v_i) = \text{TreeEnsemble}([\boldsymbol{h}_{v_i}^{0} || \boldsymbol{h}_{v_i}^{1} || \cdots || \boldsymbol{h}_{v_i}^{L}]).$$

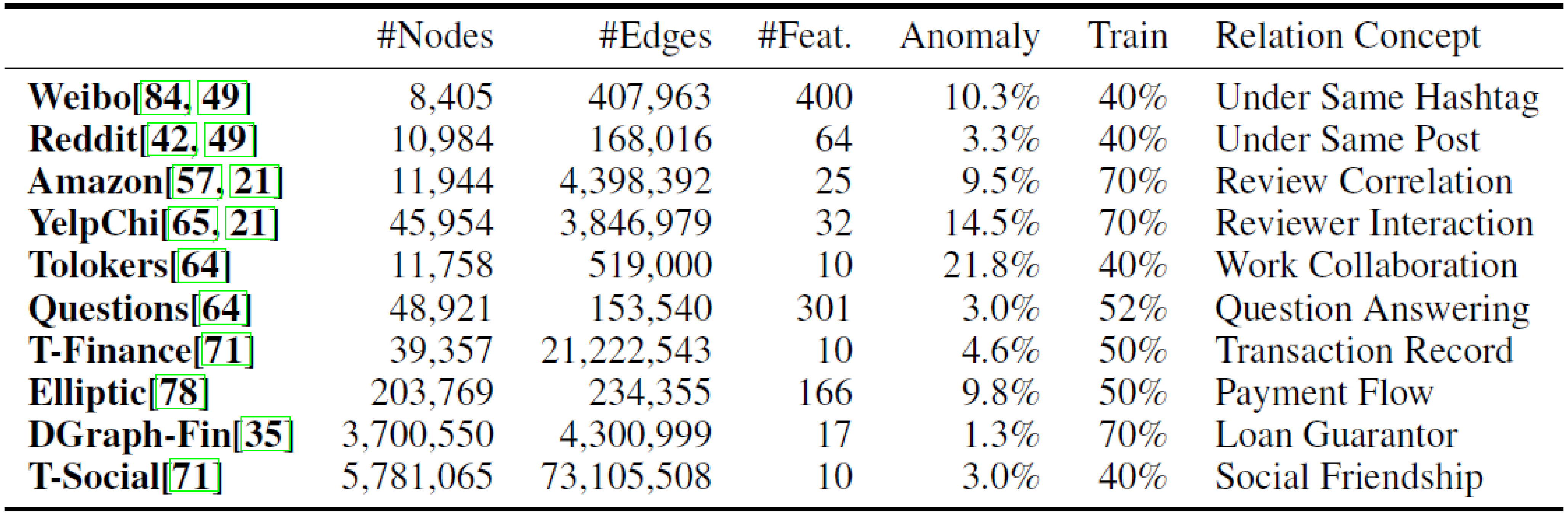
- $\boldsymbol{h}_{v_i}^{(0)} = \mathbf{x}_i$ denotes the initial node attributes.
- $\boldsymbol{h}_{v_i}^{(l)}$ represents the node feature after l-layers of neighbor aggregation.
- $\text{Aggregate}(\cdot)$ can take on any aggregation function such as mean, max, or sum pooling.
- $\text{TreeEnsemble}(\cdot)$ can be any tree ensembles that takes the aggregated features as input to predict the anomaly score of each node, e.g., **Random Forest** and **XGBoost**.

Selected Datasets in GADBench

In GADBench, we have collected **10** diverse and representative datasets, which are chosen based on the following criteria:

- **Organic anomalies.** Datasets in GADBench exclusively contain anomalies that naturally emerge in real-world scenarios, a distinction from previous studies that employ synthetic anomalies for GAD evaluations.
- **Various domains.** Datasets in GADBench span multiple domains, including social media, e-commerce, e-finance, crowd-sourcing, etc.
- **Diverse scale and Imbalance ratio.** GADBench datasets cover a wide scale, from tens of thousands of nodes to millions, with different anomaly ratios.

Dataset Information

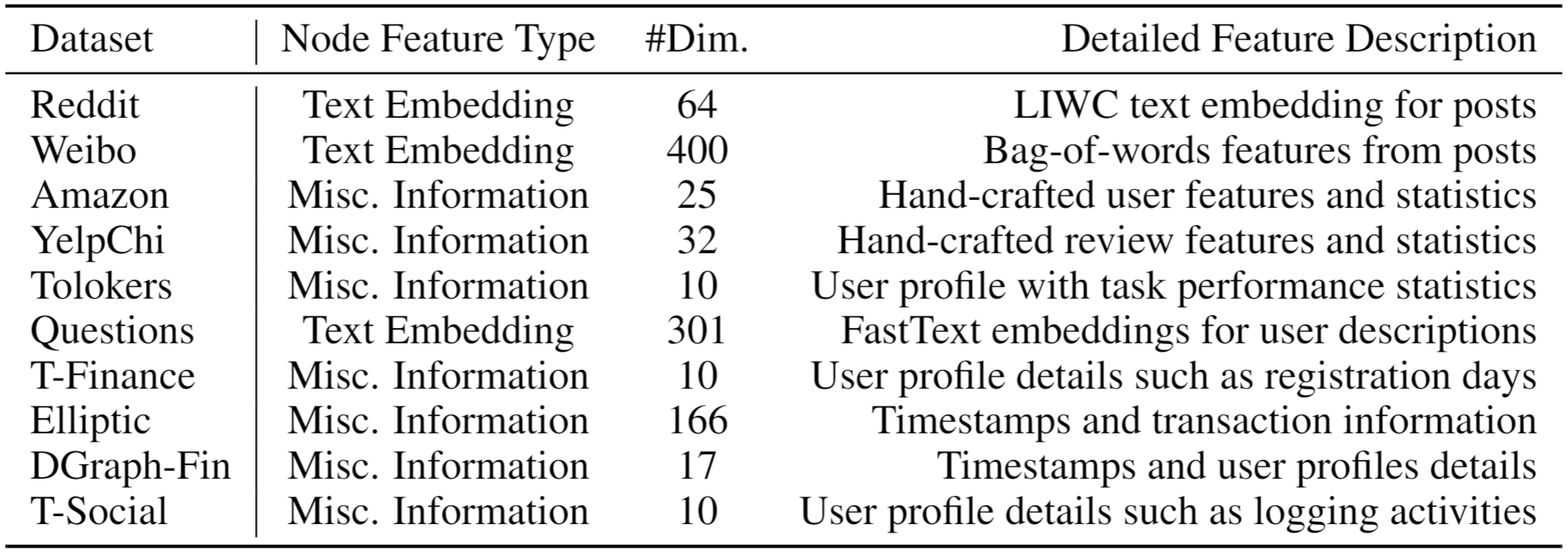


	#Nodes	#Edges	#Feat.	Anomaly	Train	Relation Concept
Weibo[84, 49]	8,405	407,963	400	10.3%	40%	Under Same Hashtag
Reddit[42, 49]	10,984	168,016	64	3.3%	40%	Under Same Post
Amazon[57, 21]	11,944	4,398,392	25	9.5%	70%	Review Correlation
YelpChi[65, 21]	45,954	3,846,979	32	14.5%	70%	Reviewer Interaction
Tolokers[64]	11,758	519,000	10	21.8%	40%	Work Collaboration
Questions[64]	48,921	153,540	301	3.0%	52%	Question Answering
T-Finance[71]	39,357	21,222,543	10	4.6%	50%	Transaction Record
Elliptic[78]	203,769	234,355	166	9.8%	50%	Payment Flow
DGraph-Fin[35]	3,700,550	4,300,999	17	1.3%	70%	Loan Guarantor
T-Social[71]	5,781,065	73,105,508	10	3.0%	40%	Social Friendship

- **Weibo, Reddit, Questions, and T-Social** are designed to identify anomalous accounts on social media platforms.
- **T-Finance, Elliptic, and DGraph-Fin** concentrate on identifying fraudulent users, illicit entities and overdue loans in financial networks,
- **Tolokers, Amazon and YelpChi** aim to detect fraudulent workers, reviews and reviewers on crowd-sourcing or e-commerce platforms.

Node Feature Information

Table 7: Overview of datasets in GADBench with their corresponding node feature types, feature dimension, and detailed descriptions.



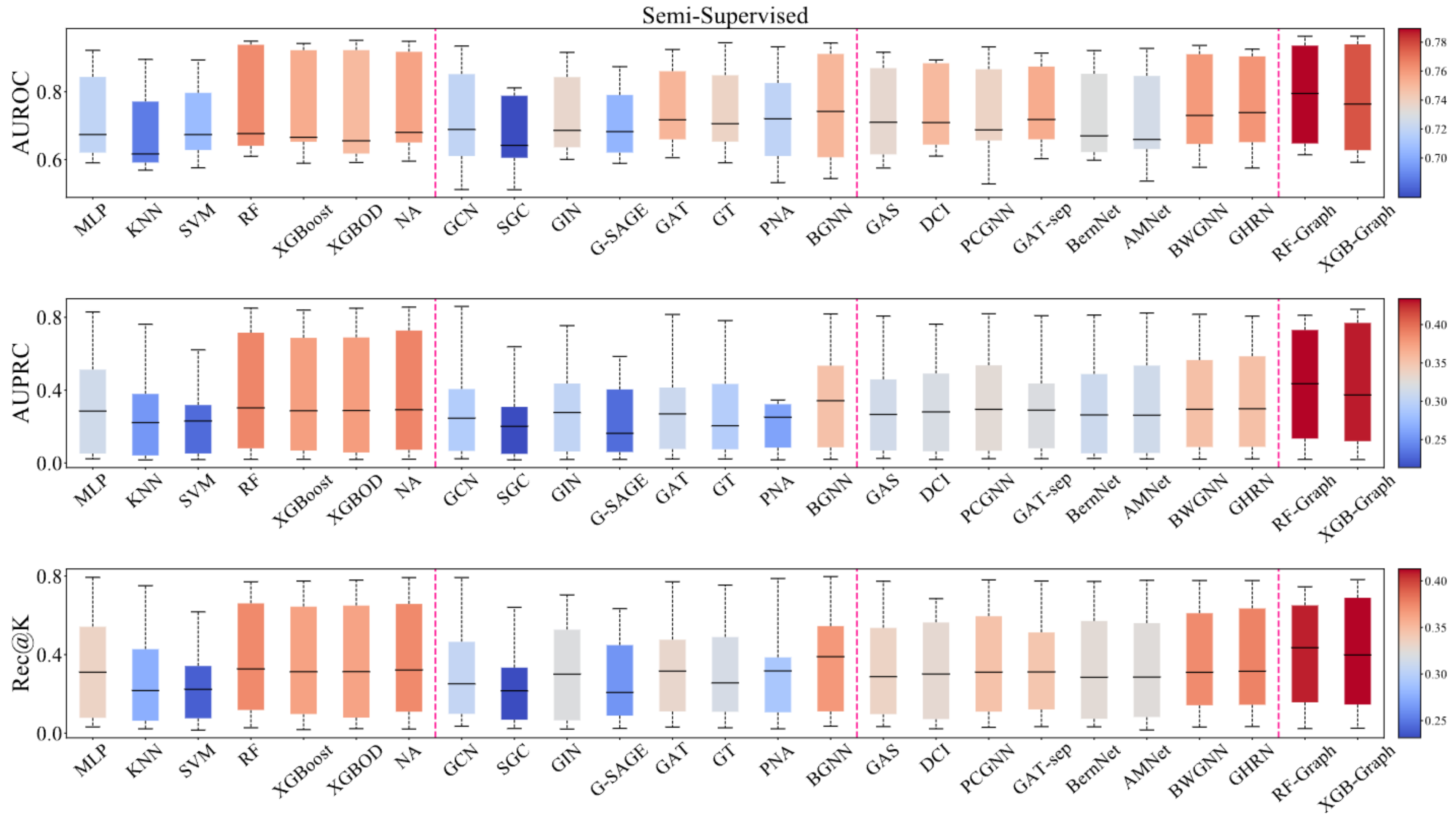
Dataset	Node Feature Type	#Dim.	Detailed Feature Description
Reddit	Text Embedding	64	LIWC text embedding for posts
Weibo	Text Embedding	400	Bag-of-words features from posts
Amazon	Misc. Information	25	Hand-crafted user features and statistics
YelpChi	Misc. Information	32	Hand-crafted review features and statistics
Tolokers	Misc. Information	10	User profile with task performance statistics
Questions	Text Embedding	301	FastText embeddings for user descriptions
T-Finance	Misc. Information	10	User profile details such as registration days
Elliptic	Misc. Information	166	Timestamps and transaction information
DGraph-Fin	Misc. Information	17	Timestamps and user profiles details
T-Social	Misc. Information	10	User profile details such as logging activities

Selected Metrics in GADBench

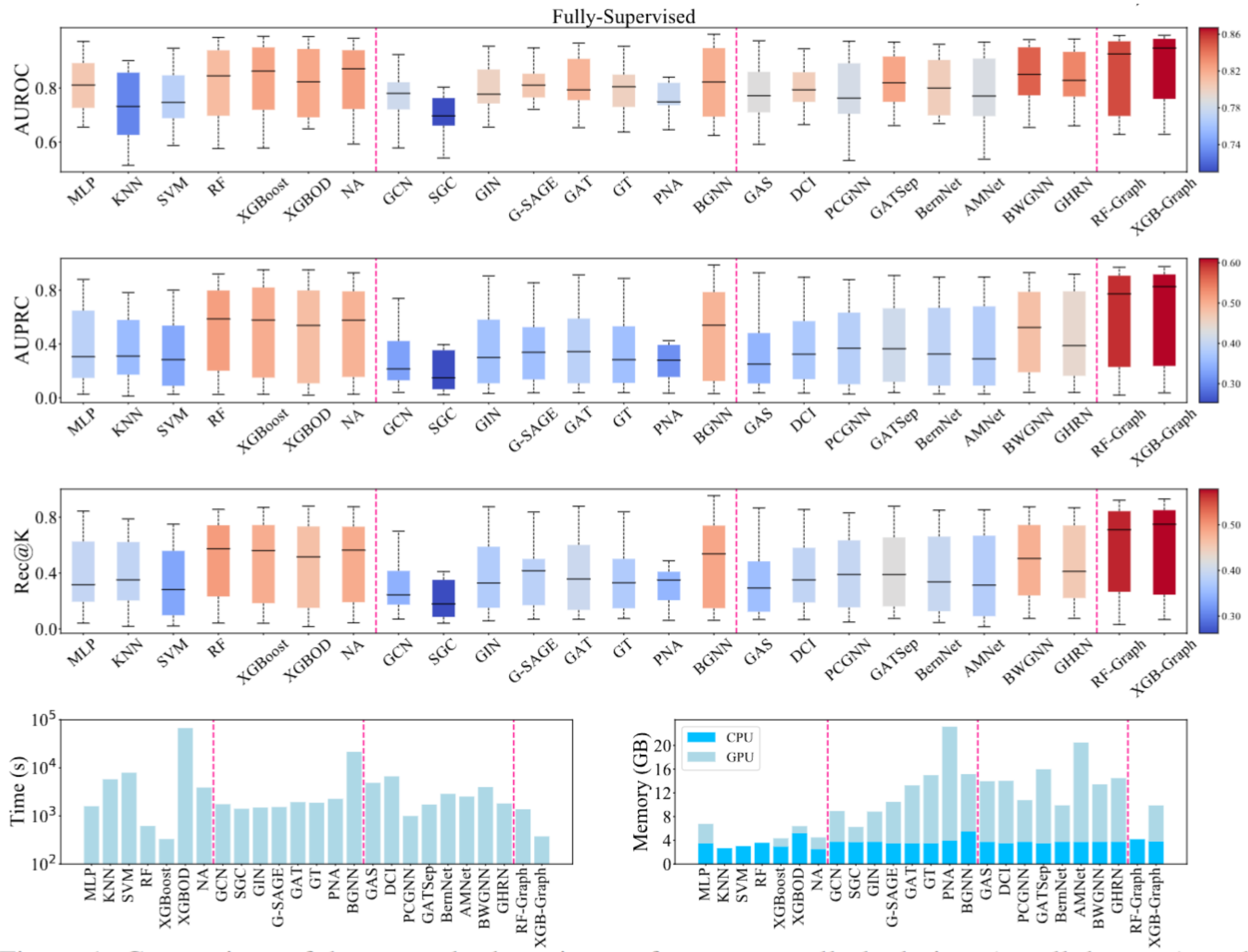
- Area Under the Receiver Operating Characteristic Curve (**AUROC**)
- Area Under the Prevision Recall Curve (**AUPRC**) estimated by average precision
- Recall score within top-k predictions (**Rec@K**)
- Running time and memory consumption

Among these metrics, **AUROC** primarily focuses on overall performance and is not sensitive to top-K predictions, **Rec@K** only cares top-K performance, and **AUPRC** strikes a balance between the two.

Semi-Supervised Results (Default Hyper-parameters)

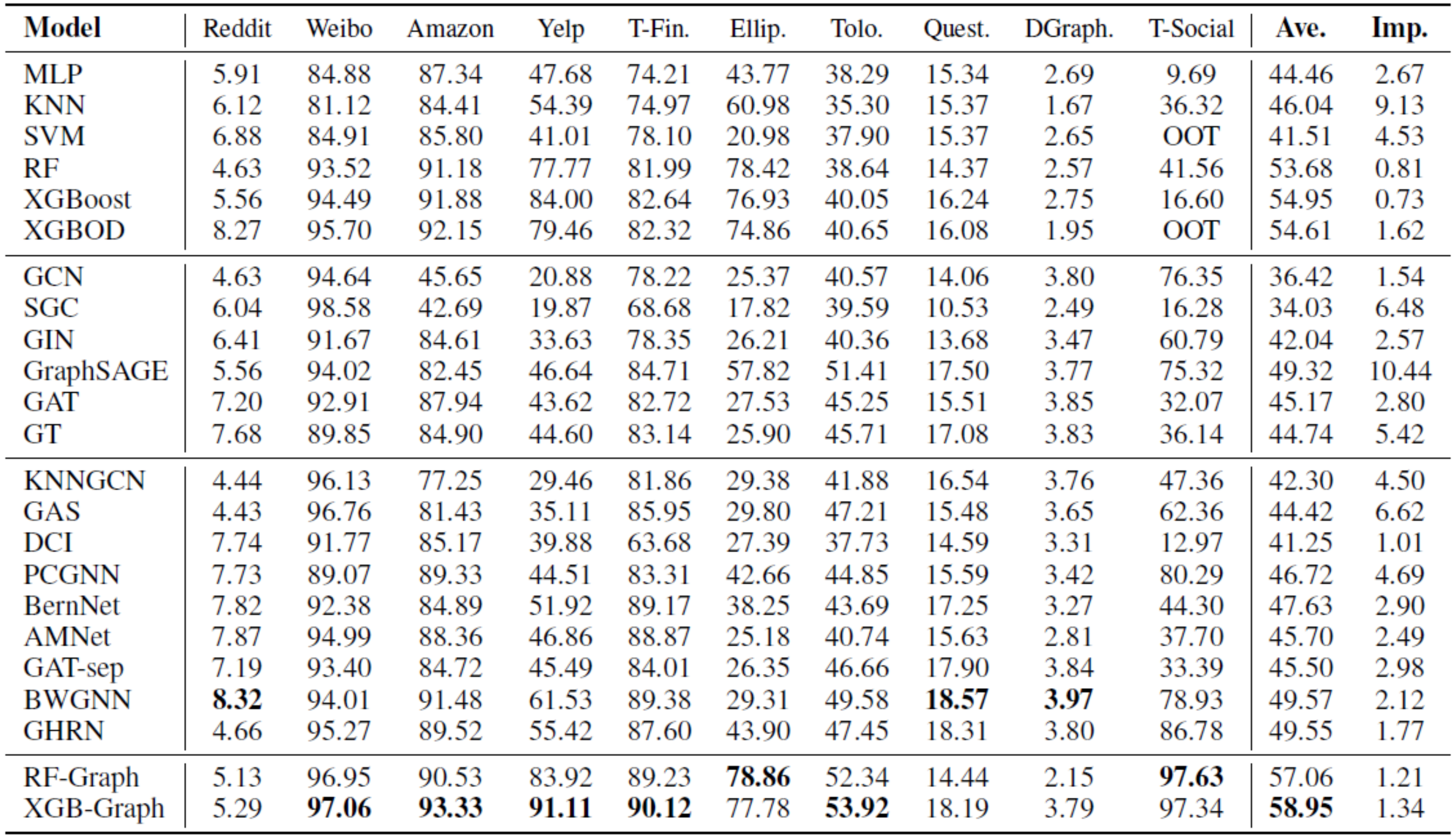


Fully-Supervised Results (Default Hyper-parameters)



Fully-Supervised Results: Random Search Hyperparameter Tuning

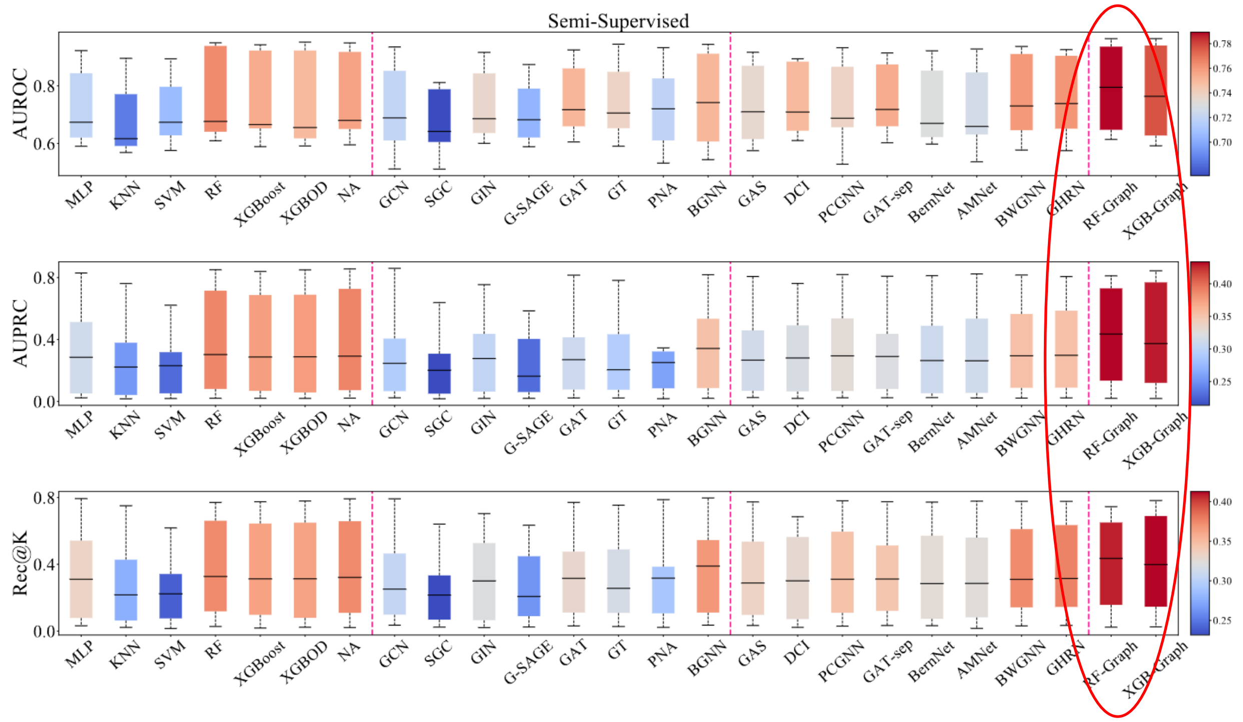
We run random search for 100 trials and save the best hyperparameter configuration on the validation set.

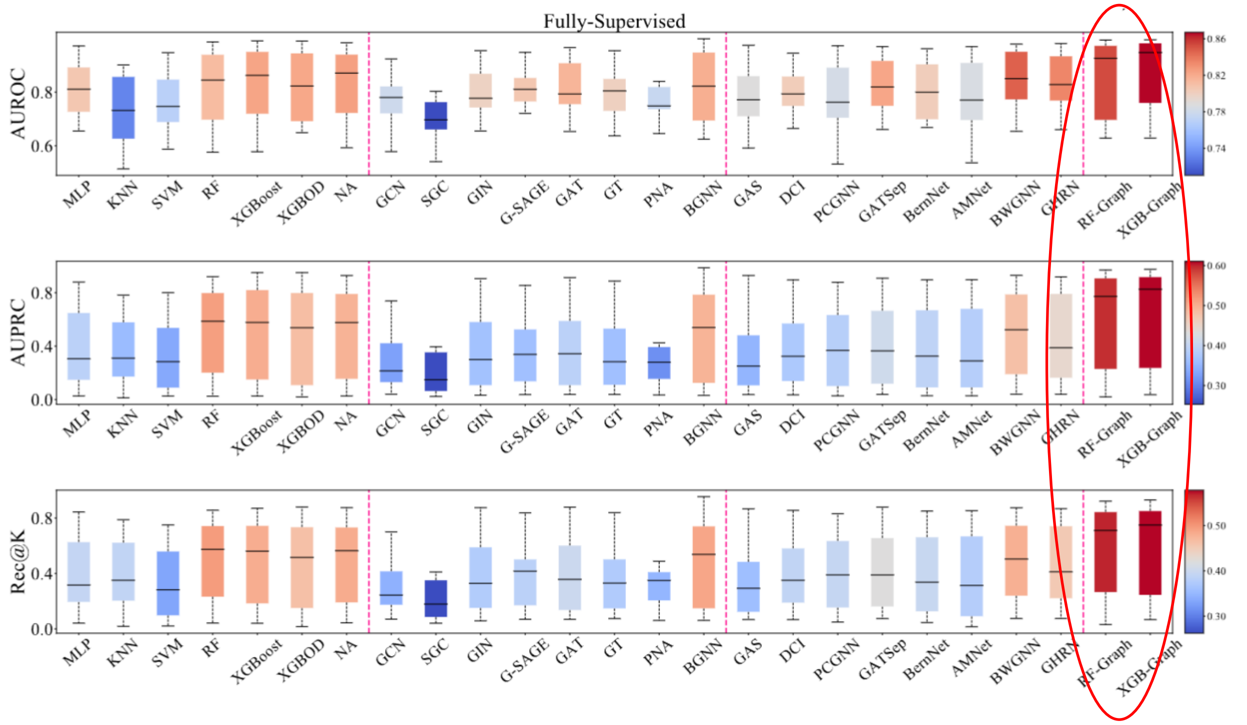


Model	Reddit	Weibo	Amazon	Yelp	T-Fin.	Ellip.	Tolo.	Quest.	DGraph.	T-Social	**Ave.**	**Imp.**
MLP	5.91	84.88	87.34	47.68	74.21	43.77	38.29	15.34	2.69	9.69	44.46	2.67
KNN	6.12	81.12	84.41	54.39	74.97	60.98	35.30	15.37	1.67	36.32	46.04	9.13
SVM	6.88	84.91	85.80	41.01	78.10	20.98	37.90	15.37	2.65	OOT	41.51	4.53
RF	4.63	93.52	91.18	77.77	81.99	78.42	38.64	14.37	2.57	41.56	53.68	0.81
XGBoost	5.56	94.49	91.88	84.00	82.64	76.93	40.05	16.24	2.75	16.60	54.95	0.73
XGBOD	8.27	95.70	92.15	79.46	82.32	74.86	40.65	16.08	1.95	OOT	54.61	1.62
GCN	4.63	94.64	45.65	20.88	78.22	25.37	40.57	14.06	3.80	76.35	36.42	1.54
SGC	6.04	98.58	42.69	19.87	68.68	17.82	39.59	10.53	2.49	16.28	34.03	6.48
GIN	6.41	91.67	84.61	33.63	78.35	26.21	40.36	13.68	3.47	60.79	42.04	2.57
GraphSAGE	5.56	94.02	82.45	46.64	84.71	57.82	51.41	17.50	3.77	75.32	49.32	10.44
GAT	7.20	92.91	87.94	43.62	82.72	27.53	45.25	15.51	3.85	32.07	45.17	2.80
GT	7.68	89.85	84.90	44.60	83.14	25.90	45.71	17.08	3.83	36.14	44.74	5.42
KNNGCN	4.44	96.13	77.25	29.46	81.86	29.38	41.88	16.54	3.76	47.36	42.30	4.50
GAS	4.43	96.76	81.43	35.11	85.95	29.80	47.21	15.48	3.65	62.36	44.42	6.62
DCI	7.74	91.77	85.17	39.88	63.68	27.39	37.73	14.59	3.31	12.97	41.25	1.01
PCGNN	7.73	89.07	89.33	44.51	83.31	42.66	44.85	15.59	3.42	80.29	46.72	4.69
BernNet	7.82	92.38	84.89	51.92	89.17	38.25	43.69	17.25	3.27	44.30	47.63	2.90
AMNet	7.87	94.99	88.36	46.86	88.87	25.18	40.74	15.63	2.81	37.70	45.70	2.49
GAT-sep	7.19	93.40	84.72	45.49	84.01	26.35	46.66	17.90	3.84	33.39	45.50	2.98
BWGNN	**8.32**	94.01	91.48	61.53	89.38	29.31	49.58	**18.57**	**3.97**	78.93	49.57	2.12
GHRN	4.66	95.27	89.52	55.42	87.60	43.90	47.45	18.31	3.80	86.78	49.55	1.77
RF-Graph	5.13	96.95	90.53	83.92	89.23	**78.86**	52.34	14.44	2.15	**97.63**	57.06	1.21
XGB-Graph	5.29	**97.06**	**93.33**	**91.11**	**90.12**	77.78	**53.92**	18.19	3.79	97.34	**58.95**	1.34

Benchmark Findings

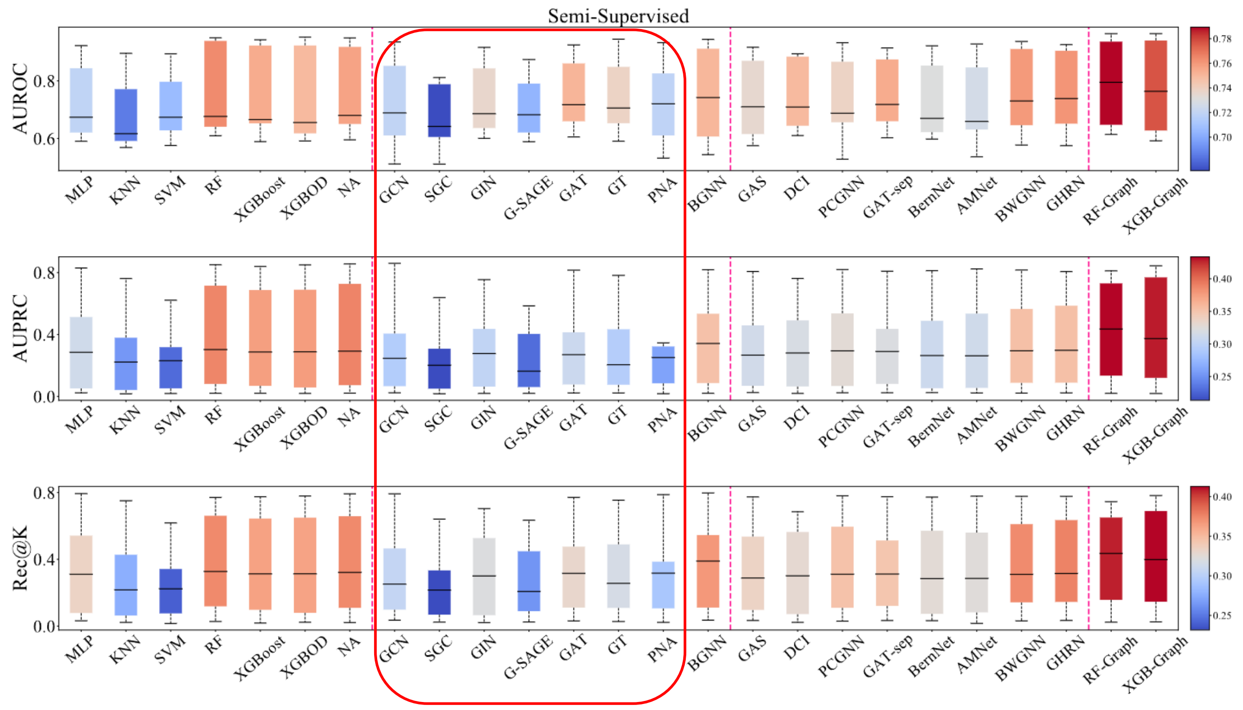
- **Findings Ⅰ**: Ensemble trees with neighbor aggregation have superior performance.

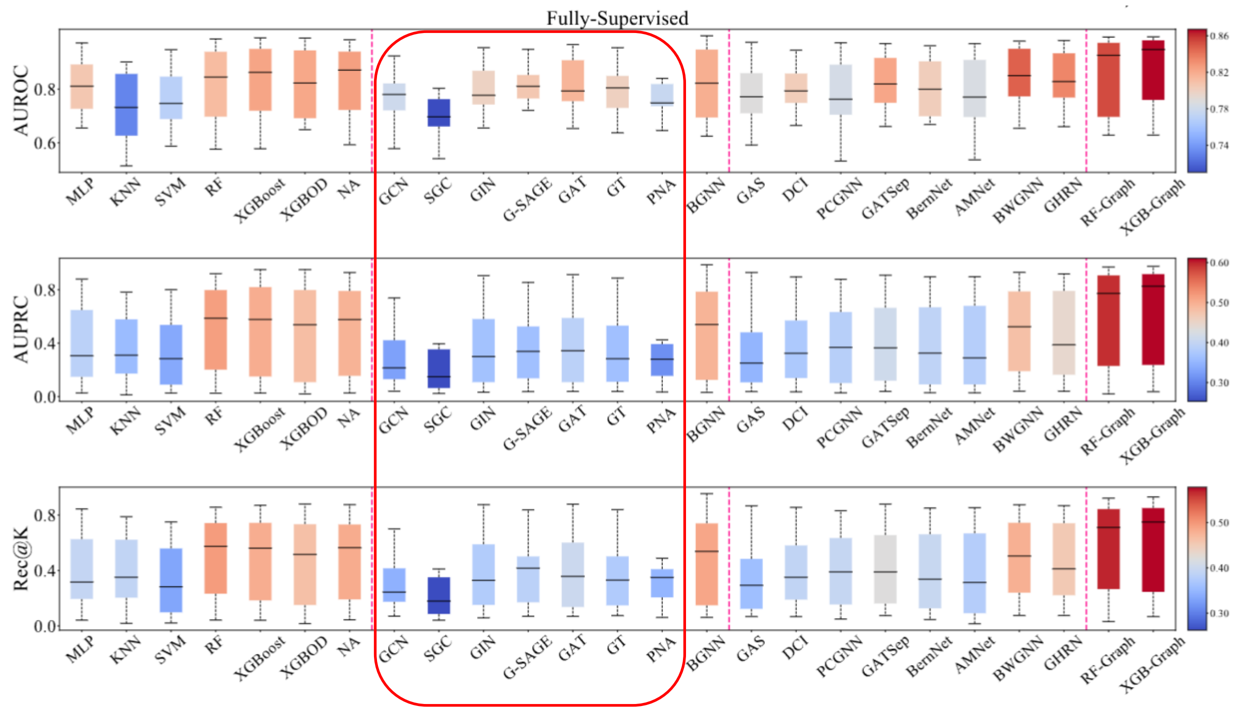




Benchmark Findings

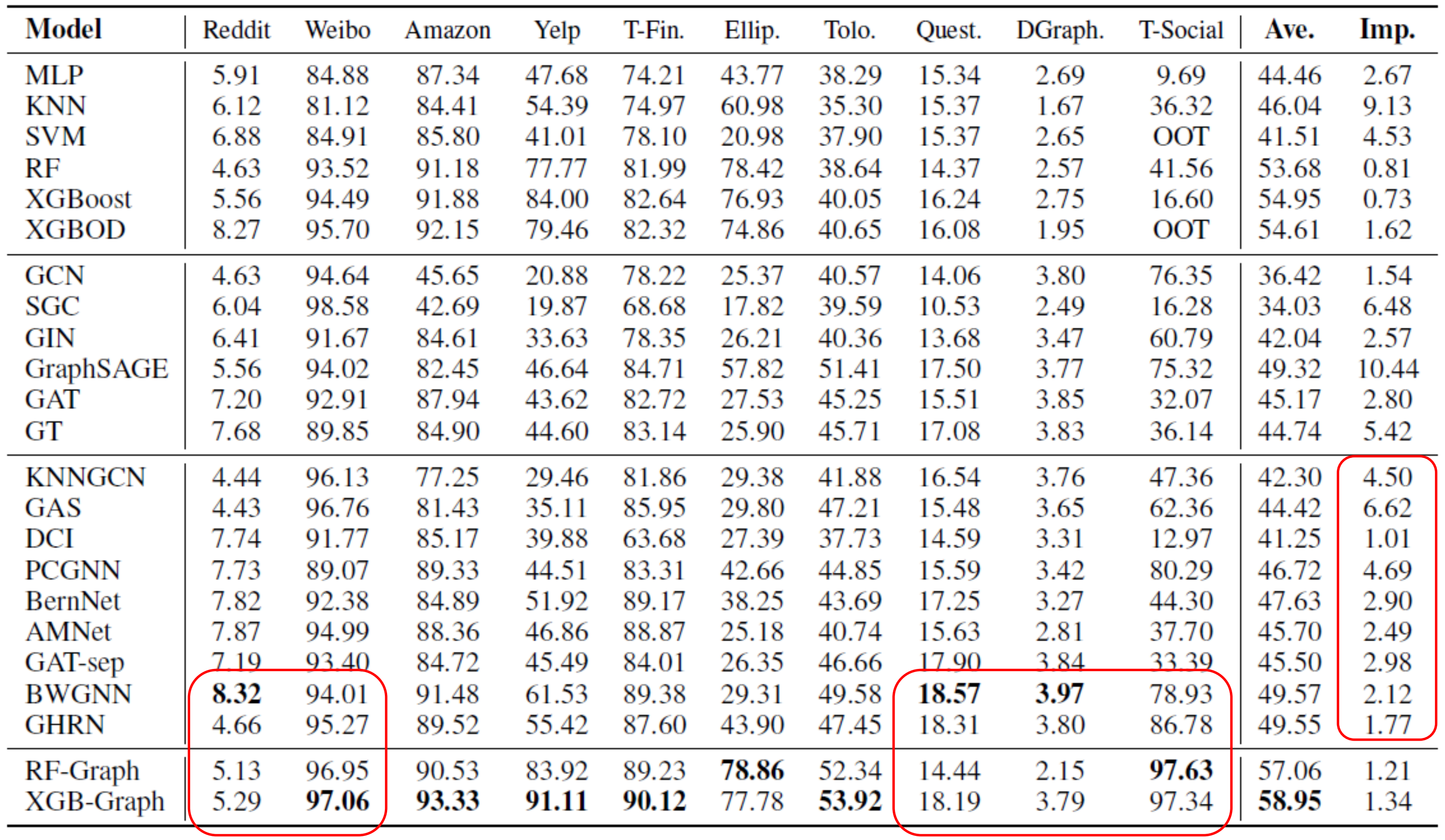
- **Findings Ⅱ**: Most standard GNNs prove unsuitable for GAD.





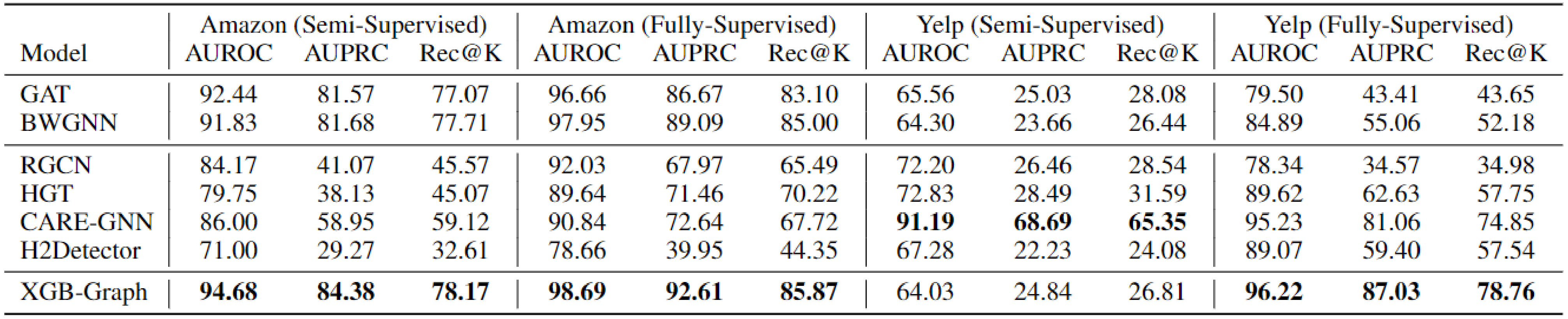
Benchmark Findings

Findings Ⅲ: Specialized GNNs require hyperparameter tuning to achieve satisfactory performance.

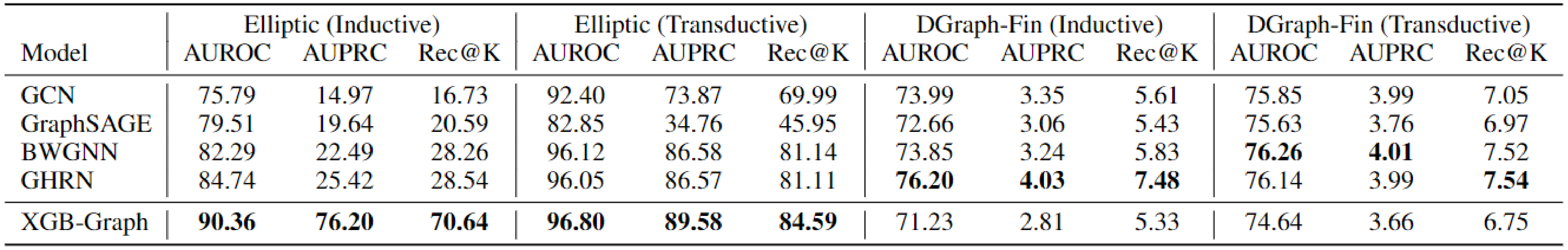


Model	Reddit	Weibo	Amazon	Yelp	T-Fin.	Ellip.	Tolo.	Quest.	DGraph.	T-Social	**Ave.**	**Imp.**
MLP	5.91	84.88	87.34	47.68	74.21	43.77	38.29	15.34	2.69	9.69	44.46	2.67
KNN	6.12	81.12	84.41	54.39	74.97	60.98	35.30	15.37	1.67	36.32	46.04	9.13
SVM	6.88	84.91	85.80	41.01	78.10	20.98	37.90	15.37	2.65	OOT	41.51	4.53
RF	4.63	93.52	91.18	77.77	81.99	78.42	38.64	14.37	2.57	41.56	53.68	0.81
XGBoost	5.56	94.49	91.88	84.00	82.64	76.93	40.05	16.24	2.75	16.60	54.95	0.73
XGBOD	8.27	95.70	92.15	79.46	82.32	74.86	40.65	16.08	1.95	OOT	54.61	1.62
GCN	4.63	94.64	45.65	20.88	78.22	25.37	40.57	14.06	3.80	76.35	36.42	1.54
SGC	6.04	98.58	42.69	19.87	68.68	17.82	39.59	10.53	2.49	16.28	34.03	6.48
GIN	6.41	91.67	84.61	33.63	78.35	26.21	40.36	13.68	3.47	60.79	42.04	2.57
GraphSAGE	5.56	94.02	82.45	46.64	84.71	57.82	51.41	17.50	3.77	75.32	49.32	10.44
GAT	7.20	92.91	87.94	43.62	82.72	27.53	45.25	15.51	3.85	32.07	45.17	2.80
GT	7.68	89.85	84.90	44.60	83.14	25.90	45.71	17.08	3.83	36.14	44.74	5.42
KNNGCN	4.44	96.13	77.25	29.46	81.86	29.38	41.88	16.54	3.76	47.36	42.30	4.50
GAS	4.43	96.76	81.43	35.11	85.95	29.80	47.21	15.48	3.65	62.36	44.42	6.62
DCI	7.74	91.77	85.17	39.88	63.68	27.39	37.73	14.59	3.31	12.97	41.25	1.01
PCGNN	7.73	89.07	89.33	44.51	83.31	42.66	44.85	15.59	3.42	80.29	46.72	4.69
BernNet	7.82	92.38	84.89	51.92	89.17	38.25	43.69	17.25	3.27	44.30	47.63	2.90
AMNet	7.87	94.99	88.36	46.86	88.87	25.18	40.74	15.63	2.81	37.70	45.70	2.49
GAT-sep	7.19	93.40	84.72	45.49	84.01	26.35	46.66	17.90	3.84	33.39	45.50	2.98
BWGNN	**8.32**	94.01	91.48	61.53	89.38	29.31	49.58	**18.57**	**3.97**	78.93	49.57	2.12
GHRN	4.66	95.27	89.52	55.42	87.60	43.90	47.45	18.31	3.80	86.78	49.55	1.77
RF-Graph	5.13	96.95	90.53	83.92	89.23	**78.86**	52.34	14.44	2.15	**97.63**	57.06	1.21
XGB-Graph	5.29	**97.06**	**93.33**	**91.11**	**90.12**	77.78	**53.92**	18.19	3.79	97.34	**58.95**	1.34

Primary Results on Heterogeneous & Transductive Settings



Model	Amazon (Semi-Supervised) AUROC	AUPRC	Rec@K	Amazon (Fully-Supervised) AUROC	AUPRC	Rec@K	Yelp (Semi-Supervised) AUROC	AUPRC	Rec@K	Yelp (Fully-Supervised) AUROC	AUPRC	Rec@K
GAT	92.44	81.57	77.07	96.66	86.67	83.10	65.56	25.03	28.08	79.50	43.41	43.65
BWGNN	91.83	81.68	77.71	97.95	89.09	85.00	64.30	23.66	26.44	84.89	55.06	52.18
RGCN	84.17	41.07	45.57	92.03	67.97	65.49	72.20	26.46	28.54	78.34	34.57	34.98
HGT	79.75	38.13	45.07	89.64	71.46	70.22	72.83	28.49	31.59	89.62	62.63	57.75
CARE-GNN	86.00	58.95	59.12	90.84	72.64	67.72	**91.19**	**68.69**	**65.35**	95.23	81.06	74.85
H2Detector	71.00	29.27	32.61	78.66	39.95	44.35	67.28	22.23	24.08	89.07	59.40	57.54
XGB-Graph	**94.68**	**84.38**	**78.17**	**98.69**	**92.61**	**85.87**	64.03	24.84	26.81	**96.22**	**87.03**	**78.76**



Model	Elliptic (Inductive) AUROC	AUPRC	Rec@K	Elliptic (Transductive) AUROC	AUPRC	Rec@K	DGraph-Fin (Inductive) AUROC	AUPRC	Rec@K	DGraph-Fin (Transductive) AUROC	AUPRC	Rec@K
GCN	75.79	14.97	16.73	92.40	73.87	69.99	73.99	3.35	5.61	75.85	3.99	7.05
GraphSAGE	79.51	19.64	20.59	82.85	34.76	45.95	72.66	3.06	5.43	75.63	3.76	6.97
BWGNN	82.29	22.49	28.26	96.12	86.58	81.14	73.85	3.24	5.83	**76.26**	**4.01**	7.52
GHRN	84.74	25.42	28.54	96.05	86.57	81.11	**76.20**	**4.03**	**7.48**	76.14	3.99	**7.54**
XGB-Graph	**90.36**	**76.20**	**70.64**	**96.80**	**89.58**	**84.59**	71.23	2.81	5.33	74.64	3.66	6.75

Impact of different number of neighbor aggregation layers

- The performance on most datasets improves when the number of neighbor aggregation layers increases **from 0 to 2.**

- Further increments in the number of layers do not contribute to any significant improvement.

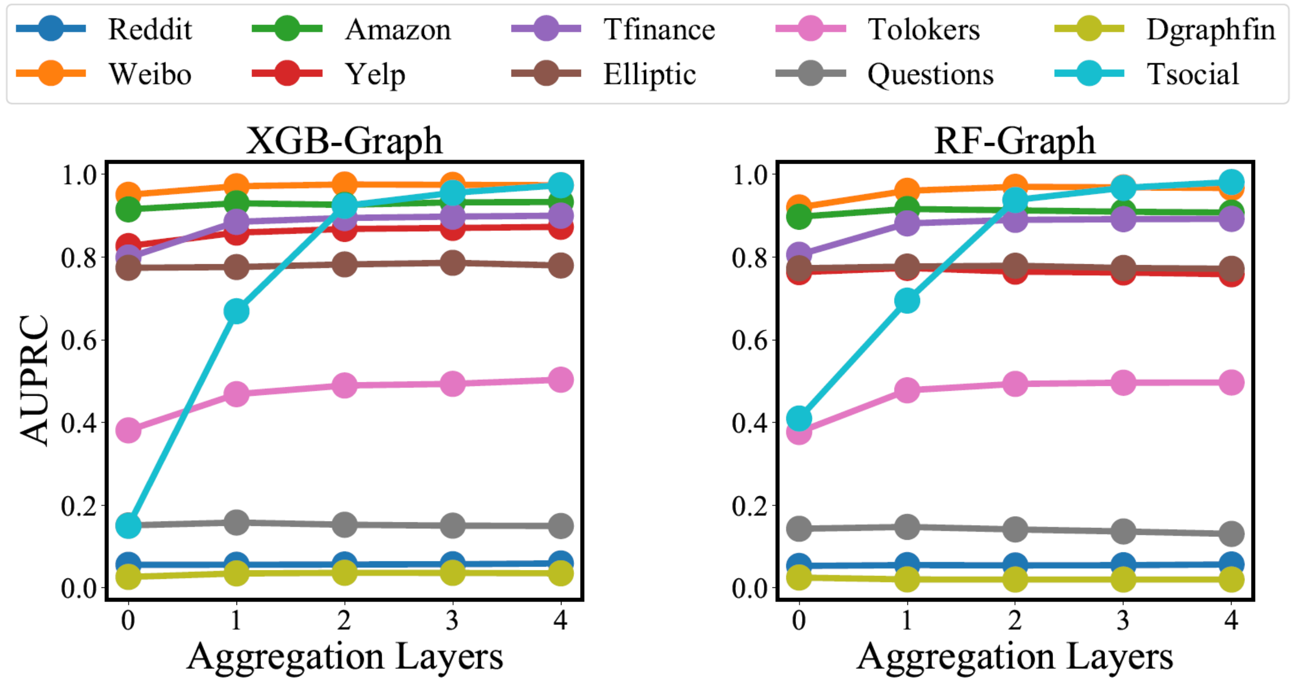


Figure 2: The impact of different number of neighbor aggregation layers on the performance of XGB-Graph and RF-Graph.

Why Do Tree Ensembles with Neighbor Aggregation Outperform GNNs?

Anomaly instances tend to **form multiple clusters** and are **coupled with normal instances**:

- It matches the inductive bias of tree ensembles that favor **complex and disjoint decision boundaries.**
- GNNs, which typically employ an MLP as the final layer, tend to generate **simple and continuous decision boundaries.**

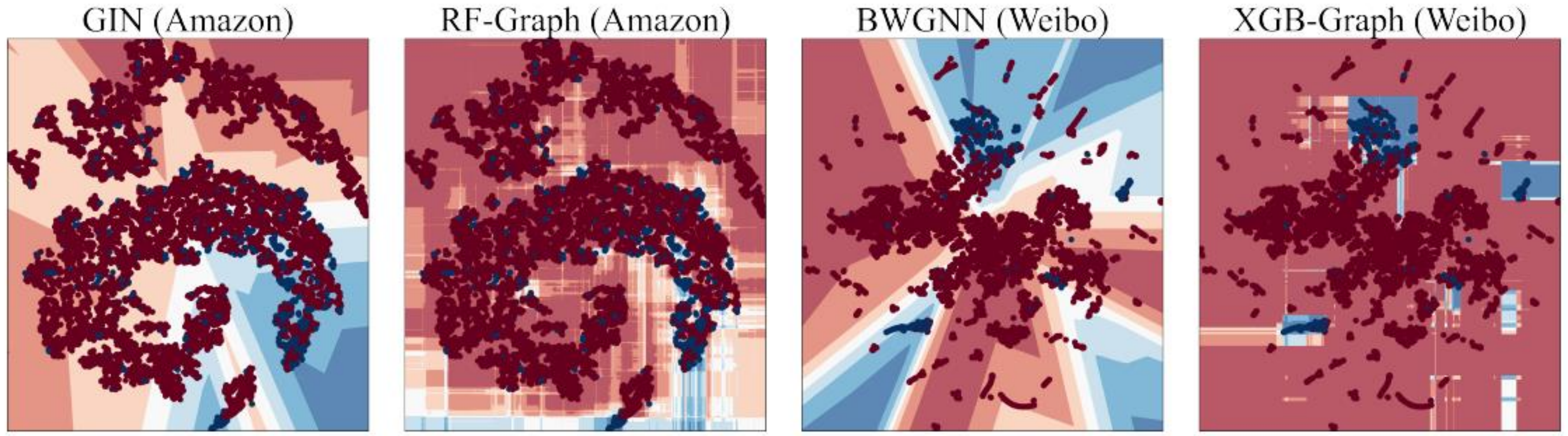


Figure 3: Decision boundaries comparison of different approaches. Blue points represent anomalies while red points are normal nodes. Similarly, the blue/red regions correspond to model predictions for anomalous/normal classes.

When Do Tree Ensembles with Neighbor Aggregation Outperform GNNs?

Out of the 10 datasets in GADBench, 3 datasets purely use text embeddings as node features, while in the remaining 7 datasets, node features contain miscellaneous information such as the combination of numerical, categorical, and temporal features.

- For datasets that rely on **text-based features**—namely Reddit, Weibo, and Questions—**GNNs** showcase competitive performance in comparison to tree ensembles.

- Conversely, in the other 7 datasets with **diverse feature types** that have low correlation (e.g., gender and age), **tree ensembles** with neighbor aggregation typically exhibit superior performance.

Conclusion

- We introduce GADBench, the first comprehensive benchmark for semi- and fully-supervised anomalous node detection on static attributed graphs.

- Our evaluation of 29 models on 10 real-world datasets shows that tree ensembles with simple neighborhood aggregation generally outperform other models, including GNNs specifically designed for the GAD task.
  - The rationale behind this finding is initially examined from the standpoints of decision boundary and node feature type.

- By making GADBench open-source, we aim to foster further research and refinement of GAD algorithms, as well as their more informed evaluations and comparisons.

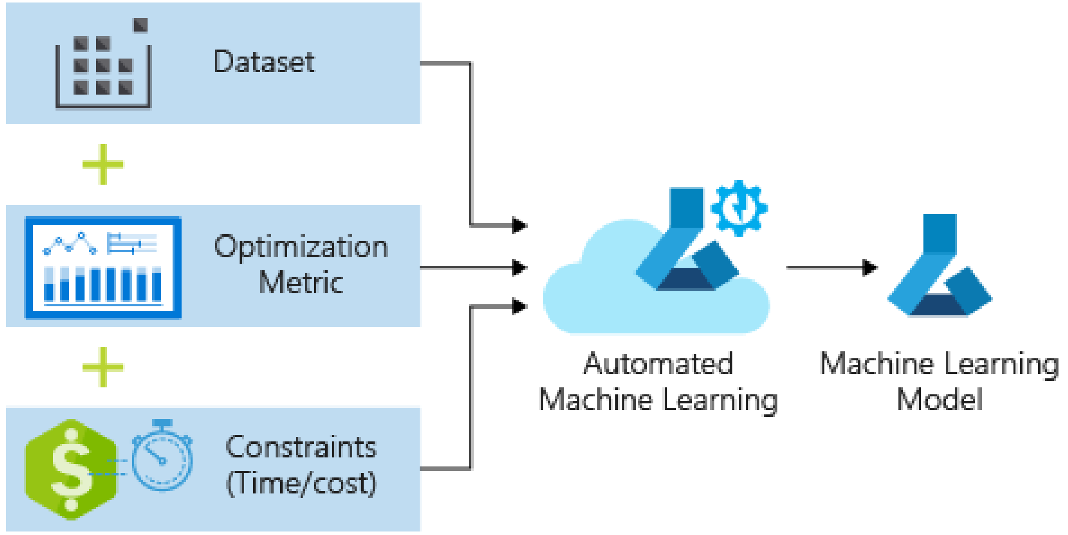
Future Direction: GAD Model Design

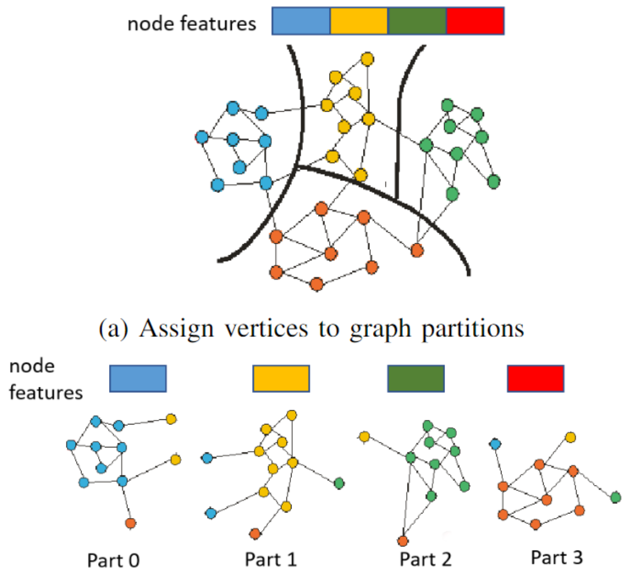
- Tree ensembles with neighbor aggregation is just an initial attempt to integrate graph information:
  - explore more sophisticated ways to combine GNNs and tree ensembles
  - e.g., through end-to-end training strategies.

- How to enhance ensemble trees in the unsupervised GAD scenario?
  - e.g., combine neighborhood aggregation with isolation forest.

Future Direction: GAD System Design

- Automated Algorithm Selection [1,2]
- System-Level Optimization [3,4]





(a) Assign vertices to graph partitions

(b) Generate graph partitions with HALO vertices (the vertices with different colors from majority of the vertices in the partition).

Fig. 4: Graph partitioning with METIS in DistDGL.

[1] AutoGluon: AutoML for Image, Text, Time Series, and Tabular Data
[2] ADGym: Design Choices for Deep Anomaly Detection
[3] DistDGL: Distributed Graph Neural Network Training for Billion-Scale Graphs
[4] GPU-accelerated Outlier Detection via Tensor Operations

Thank you for listening!

Q&A

WeChat: sqrt3tjh
Email: sqrt3tjh@gmail.com
Github: https://github.com/squareRoot3/GADBench