# RDBENCH: ML BENCHMARK FOR RELATIONAL DATABASES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Benefiting from high-quality datasets and standardized evaluation metrics, machine learning (ML) has achieved sustained progress and widespread applications. However, while applying machine learning to relational databases, the absence of a well-established benchmark remains a significant obstacle to the development of ML. To address this issue, we introduce *ML Benchmark For Relational Databases* (RDBench), a benchmark that aims to promote hierarchical, robust, and reproducible ML research on relational databases. RDBench offers hierarchical datasets of varying scales, domains, and relations. It provides three types of data: tabular data, homogeneous graphs, and heterogeneous graphs. Importantly, all data formats share the same task definition, allowing for meaningful comparisons between methods across different data formats. Reported results are averaged over the same datasets and tasks (classification or regression), further enhancing the robustness of the experimental findings. In addition to dataset construction, we conduct extensive experiments to uncover performance differences between models. To better present our proposed RDBench, we offer a user-friendly API that provides standardized formats for three types of data.

## 1 INTRODUCTION

Providing authoritative datasets and standardized evaluation metrics, benchmarks in machine learning like ImageNet (Deng et al., 2009), GLUE (Wang et al., 2019), and OGB (Hu et al., 2020a) are playing a pivotal role in the development of the realm. These benchmarks have established clear objectives and facilitated collaboration and advancement, resulting in widespread successes across computer vision (Krizhevsky et al., 2012; He et al., 2016), natural language processing (Radford et al., 2018; OpenAI, 2023), and graph representation learning (Ying et al., 2021). However, the lack of well-established benchmarks in some domains has partially hindered the furtherance of machine learning.

As a ubiquitous and fundamental tool for storing real-world data, relational databases have gradually captivated the machine learning community (Cvitkovic, 2020). Benchmarks on relational databases (Difallah et al., 2013; Cheng et al., 2019; Zhou et al., 2023) focus on database management system or real-time feature extraction, ignoring the expanding demand from machine learning. Early works on applying machine learning to relational databases (Schlichtkrull et al., 2018; Cvitkovic, 2020) were constrained by the absence of benchmarks. It is worth noticing that building a benchmark can not only improve the performance of machine learning methods on relational databases (Gorishniy et al., 2021; Grinsztajn et al., 2022), but also facilitate potential interdisciplinary collaborations. Therefore, the establishment of a well-established machine learning benchmark for relational databases has become a top priority.

In order to address this challenge, we present *Machine Learning Benchmark For Relational Databases* (RDBench), a benchmark aiming to facilitate hierarchical, robust, and reproducible ML research on relational databases. An overview of our proposed RDBench is in Figure 1. Catering to the needs of various users, RDBench provides 3 different kinds of data formats: tabular data, homogeneous graphs, and heterogeneous graphs. And it is noteworthy that for the 3 kinds of formats, the task definition remains the same, ensuring that the comparison between results is meaningful. RDBench exhibits 11 datasets and 69 tasks; in order to provide more robust results, the presented results are the average results for identical task types performed on the same dataset. For hierarchical purposes,

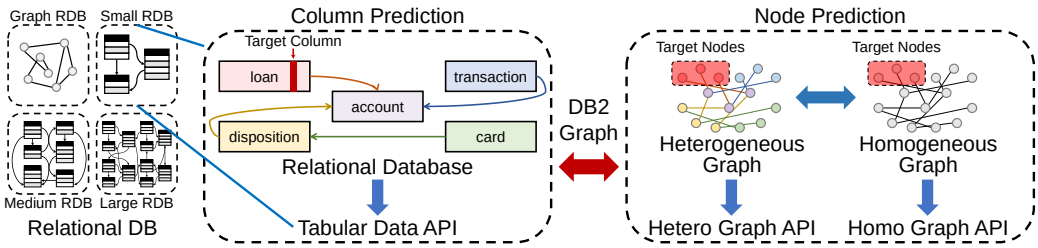



Figure 1: An overview of our proposed benchmark RDBench.

RDBench organizes datasets into 3 different levels based on their complexity, specifically the number of tables they contain. Extensive experiments are carried out on these datasets in order to present the results of methods for 3 kinds of data.

To sum up, RDBench has the following characteristics:

- **Unified Task Definition for Various Data Formats.** We provide RDBench with 3 kinds of data: tabular data, homogeneous graphs and heterogeneous graphs. For all these data formats, we propose a unified task definition, enabling results comparison between models for different formats of data.
- **Hierarchical Datasets with Comprehensive Experiments.** We present RDBench with 11 datasets with various scales, domains and relations. We categorized these datasets into three groups based on the number of tables in the relational databases, a signal for relationship complexity. Extensive experiments with 10 baselines are carried out on these datasets.
- **Easy-to-use Interfaces with Robust Results.** RDBench has a user-friendly API, utilizing popular machine learning frameworks to present the 3 kinds of data. Besides, the evaluation of RDBench is more robust because the results reported are averaged over the same dataset and same task type (classification or regression).

## 2 Related Works

**ML Benchmark.** A benchmark is more than just a simple combination of datasets and evaluation criteria today. Instead, a robust benchmark often leads the way in a machine learning field, reflecting the trends in technological progress, including Scientific Machine Learning (Takamoto et al., 2022; Thiyagalingam et al., 2022) and large language models (Valmeekam et al., 2022), and human-centric evaluations (Zhong et al., 2023). Meanwhile, the absence of benchmarks for relational databases has rendered this a crucial area that the machine learning community has long neglected (Cvitkovic, 2020).

**ML on graph data.** Graph Neural Networks (GNNs) are specialized machine learning models tailored for graph-structured data. They have gained widespread usage in domains like recommendation systems (Min et al., 2022) and social networks (Wu et al., 2020). GNNs can be categorized into Message Passing Neural Networks (MPNNs), such as GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2017), and non-MPNNs (Wu et al., 2022; Ying et al., 2021). Furthermore, there's an extended version known as Heterogeneous Graph Neural Networks (HeteroGNN) suitable for diverse graph data, such as HGCN (Peng et al., 2019) and HGT (Hu et al., 2020b). It's worth noting that the current focus of GNNs is primarily on explicit graph data and thus the lack of graph data also constrains the development of the GNNs. In contrast, the structure of databases presents new challenges for the application of GNNs.

**ML on tabular data.** Currently, most studies focus on single-tabular data. As conventional machine learning techniques, Gradient Boosting Decision Trees(GBDT) algorithms, including LightGBM (Ke et al., 2017), CatBoost (Prokhorenkova et al., 2018), XGBoost (Chen & Guestrin, 2016), sequentially train an ensemble of decision trees to predict the desired output. While the second involves more contemporary deep learning methods. They are often grouped into 4 classes: Architecture-based

methods (Chen et al., 2022; Guo et al., 2017; Popov et al., 2020), transformer-based architectures (Arik & Pfister, 2021; Gorishniy et al., 2021; Somepalli et al., 2021; Huang et al., 2020), Regularization-based methods (Kadra et al., 2021; Shavitt & Segal, 2018), LLM-based models (Wang et al., 2023b). For multi-table data, studies (Cvitkovic, 2020; Bai et al., 2021; Liu et al., 2023) propose solutions for modeling as heterogeneous graphs, while the others (Ye et al., 2023; Liu et al., 2022; Wang & Sun, 2022; Zha et al., 2023) are to perform pretraining on multi-table data to obtain meaningful representations. These early research endeavors have provided valuable applications for modeling. However, the absence of a robust benchmark severely constrains the further development of these studies.

## 3 PRELIMINARIES

**Relational Database.** A relational database, denoted as $\mathbf{D}$, comprises $N$ tables represented as $\mathbf{D} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, ..., \boldsymbol{T}_N\}$. For table $\boldsymbol{T}_n$, each column $\boldsymbol{T}_n^{:c}$ represents specific features or foreign keys pointing to another table, while each row $\boldsymbol{T}_n^{r:}$ is an instance. Each table $\boldsymbol{T}_n$ consists of multiple features $F_n = \{f_{n_1}, f_{n_2}, ..., f_{n_t}\}$ belonging to itself and foreign keys $K_n = \{k_{n_1}, k_{n_2}, ..., k_{n_s}\}$ pointing to other tables, where $s$ and $t$ are natural numbers and we have $\boldsymbol{T}_n = \{F_n, K_n\}$. For an instance $\boldsymbol{T}_n^{r:}$ at $r$-th row, its foreign key at a certain column $k_{n_w}^r$ is pointing to a row in another table $\boldsymbol{T}_m^{t:}$, denoted as $k_{n_w}^r \rightarrow \boldsymbol{T}_m^{t:}$. All foreign keys in the same column point to rows in the same table, which can be denoted as $k_{n_w} \rightarrow \boldsymbol{T}_m$. Besides, within a table, different foreign keys $k_{n_w}, k_{n_x}$ could point to the same table: $k_{n_w} \rightarrow \boldsymbol{T}_m$, $k_{n_x} \rightarrow \boldsymbol{T}_m$. For instance, in the Mutagenesis database (Debnath et al., 1991), which stores molecular information, a bond corresponds to two atom's id references pointing to the same atom table.

**Graphs.** A graph, denoted as $\mathcal{G}$, consists of two essential components: vertices and edges, represented by sets $\mathcal{V}$ and $\mathcal{E}$, respectively. Some nodes $v_i \in \mathcal{V}$ and edges $e_{(v_s, v_t)} \in \mathcal{E}$ possess attributes. These attributes can be categorized into two primary types: types (e.g., node types such as paper or author) and features (e.g., node features like age or gender), as exemplified in the context of citation networks (Wang et al., 2020). In homogeneous graphs, denoted as $\mathcal{G}_{\text{homo}}$, all nodes and edges belong to the same type or category, while heterogeneous graphs, denoted as $\mathcal{G}_{\text{hetero}}$, encompass nodes and edges of different types.

All of the notions above are in Appendix A.

## 4 MACHINE LEARNING TASKS ON RELATIONAL DATABASE

Given a multi-table relational database $\mathbf{D} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, ..., \boldsymbol{T}_N\}$, we define the machine learning tasks on it as feature prediction. More specifically, we define a task by identifying a specific feature column $f_{n_k}$ in features $F_n$ of table $\boldsymbol{T}_n$ as the prediction objective, then make use of the information from the rest part of table $\boldsymbol{T}_n$ and all of the others to predict the target column $f_{n_k}$.

To further formalize our task definition, we classify the relational database usages into 3 categories: single table, one-hop neighborhood and whole database. For all the categories, the objection is $Y = g(X)$, where $Y$ is $f_{n_k}$, and $g$ is the model. For single table, we have

$$X = \bigcup_{j \neq k} f_{n_j} \tag{1}$$

For one-hop neighborhood, the foreign keys of the current table are used to join other tables' information, i.e. features, and we have

$$X = \bigcup_{j \neq k} f_{n_j} \bigoplus \bigcup_{k_{n_i} \rightarrow T_l} F_l \tag{2}$$

Where $\bigoplus$ means the aggregation of information. For whole database, all the information of the database except for object column can be used, thus we have

$$X = \bigcup_{j \neq k} f_{n_j} \bigoplus K_n \bigoplus \bigcup_{m \neq n} T_m \tag{3}$$

The data format of a single table and one-hop neighborhood can be effectively represented as tabular data, making it easy to process through machine learning methods. However, to make optimal use of relational databases, we need to introduce graph data, including homogeneous graph data and heterogeneous graph data.

To sum up, in our proposed baselines, we provide 3 different formats of relational databases: tabular data, homogeneous graph data as well as heterogeneous graph data. It is worth noticing that there are two kinds of tabular data: single table and one-hop neighborhood, respectively. And they can be expressed as regular and joint table for short.

# 5 BRIDGING BETWEEN RELATIONAL DATABASES AND GRAPHS

## 5.1 MOTIVATION

Though possessing strong relationship representing ability, relational databases have received relatively less attention in the field of machine learning, especially graph representation learning. On the other hand, the application of graph representation learning is considered restricted by the complicated procedure (Hu et al., 2020a) to provide suitable graph data. However, the fundamental similarity of representing relationships makes it possible to solve both problems at one time. Several attempts (Cvitkovic, 2020; Bai et al., 2021; Liu et al., 2023) explore to model relational databases as graphs. In a molecular prediction task (Debnath et al., 1991), graphs are also transformed into relational databases successfully.

In order to bridge the gap between relational databases and graphs and fully leverage the information within relational databases, we provide the graph format of relational databases. In the following section, we will introduce the procedure on both data level and task level. Besides, in order to validate the correctness and utility of the graph format, we also propose to transform graphs into relational databases and present the algorithm in Appendix C.

## 5.2 RELATIONAL DATABASES TO GRAPHS

The conversion from relational databases to graphs can be achieved through a 2-step procedure, as expressed below. The detailed procedure refer to Figure 2.

$$\mathbf{D} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, .., \boldsymbol{T}_n\} \xrightarrow{f_1} \mathcal{G}_{\text{hetero}} \xrightarrow{f_2} \mathcal{G}_{\text{homo}} \tag{4}$$

Firstly, we transform the relational database into a heterogeneous graph. The heterogeneous graph here consists of nodes and edges, where a node has node features and node type, while an edge has an edge type. In this procedure, we treat the $r$-th row $\boldsymbol{T}_n^{r:}$ of table $\boldsymbol{T}_n$ as a node $v_n^r$, assigning the row's feature $F_n^r$ to the node's feature and the index $n$ to the node's type. On the other hand, we treat the foreign keys of a row as edges, for each foreign key $k_{n_w}^r \rightarrow \boldsymbol{T}_m^{t:}$, connecting the node it from $(v_n^r)$ to the node it points to $(v_m^t)$. And the edge type is decided by both the first table and the second table.

Then, following prior works (Wang et al., 2023a) carrying out homogeneous graph learning methods on heterogeneous graph data, we opt to concatenate node features and embed the type information in them. After removing node types and edge types, heterogeneous graphs are transformed into homogeneous graphs.

Furthermore, as mentioned in Section 4, the defined tasks on relational databases are feature prediction tasks. And in heterogeneous graph, all the features in relational databases are transformed into node features. Thus the tasks on heterogeneous corresponding to relational databases are node property prediction tasks.

# 6 RDBENCH: ML BENCHMARK FOR RELATIONAL DATABASES

## 6.1 RDBENCH FEATURES

There exist diverse types of features within the tables, and we classify them into two categories: continuous and discrete. Based on the distinct characteristics of the data, we employ different approaches to process the corresponding features.

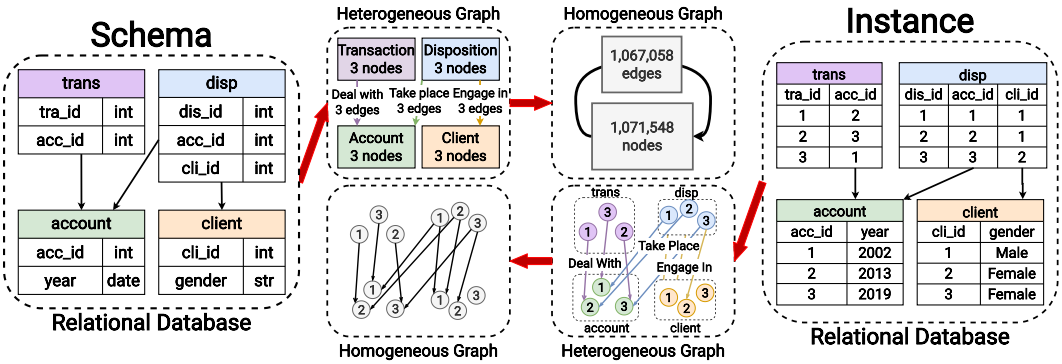



Figure 2: The procedure transforming relational databases to graphs.

**Discrete Features.** In our table, columns containing non-date string values and integer values are identified as discrete features. Strings primarily represent categorical features such as gender or geographical regions. Integers might also signify categorical features like different product kinds, though they could potentially exhibit explicit order relationships such as frequency occurrences. Categorical features lack inherent ordinal relationships and represent distinct values, thus we map them into a vector space with the same dimension.

**Continuous Features.** Columns with non-integer values and time values are regarded as continuous features. We normalize these continuous features such as prices, times or scores, and utilize them as processed features directly. It's worth noticing that we transform the integer numbers with too many unique values into floating point numbers, following the procedure we transform date into floating point numbers.

**Missing Features.** While facing missing features, we adopt 3 different kinds of ways to deal with them. Firstly, while dealing with missing target features, we just drop the rows from objective column, i.e. we do not predict these missing features. Besides, when the missing features are in other columns, we assign the average value of that column to the missing value. And when missing values occur in foreign key columns, we simply do not construct that edge.

After dealing with features, we merge discrete features and continuous features to create the features used in our training and testing procedure.

### 6.2 RDBENCH DATA

**Data Source.** We collect our datasets from a relational dataset repository (Motl & Schulte, 2015) [1]. The selection of our dataset is based on the criterion of ensuring representativity. More specifically, we choose datasets that exhibit variations in terms of number of tables, number of rows, number of columns, and source origin, aiming to capture a diverse range of relational databases. Besides, in order to validate the effectiveness of our graph transformation, we select several datasets from OGB (Hu et al., 2020a).

**Data Format.** We provide 3 kinds of data formats and 4 kinds of data in total. The 3 kinds of data formats are tabular data, heterogeneous graph data and homogeneous graph data. And tabular data has two kinds: tabular data without joint table (single table) and tabular data with joint table (one-hop neighborhood). Both kinds of tabular data are represented as tensors. Both kinds of graph data are represented with the PyTorch Geometric (PyG) library (Fey & Lenssen, 2019). Besides, all of above data are provided with their training, validation and testing masks to facilitate model evaluation.

**Data Spliting.** As for data spliting, we provide the procedure to produce the masks in our experiments part instead of providing merely fixed masks like some other benchmarks. Furthermore, while dealing with data with strong dependencies such as stock price prediction (Berka, 1999), we will partition the data based on time-series order to preserve temporal relationships.

[1]Website: `https://relational.fit.cvut.cz/`

### 6.3 RDBENCH TASKS

**Task Definition.** As previously introduced in Section 4, machine learning tasks on relational databases are defined as missing value prediction, or more specifically target column prediction. As for graphs (whether homo or hetero), the target column prediction tasks are transformed into node prediction tasks. In order to enhance robustness, for the same dataset and task, we select multiple objection columns to define a task, performing an averaging operation to get the final result.

**Task Selection.** As for task selection, we tend to choose the column with moderate difficulty and predictive value. For classification tasks, difficulty could be reflected by accuracy, where most results should be neither too high nor too low. For regression tasks, difficulty could be reflected by mean squared error, results should be better than predicting average. Predictive value has two parts. Firstly, the target column should be valuable at semantic level in the relational databases. Besides, the length of the target column should be adequate to evaluate models' performance.

**Target Leakage.** During experiments, several results imply that target leakages may have taken place. More specifically, the target column might be expressed by several other columns, resulting in extremely high accuracy or low mean squared error. We have eliminated most of the tasks with target leakage, but we intentionally retain a few sporadic instances to assess the model's capabilities, as not all models can achieve such high performance.

## 7 DATASETS

In order to bridge the existing gap between relational databases and machine learning, we present RDBench datasets. Our proposed datasets are sourced from real-world relational databases and tailored for machine learning, across various domains such as medical, business, sports, and government, ensuring their representative nature. As mentioned in Section 5, our proposed datasets provide 3 different kinds of APIs: tabular data, heterogeneous graph data and homogeneous graph data, respectively. In the following passages, we will present and analyze the performance of popular methods for different data formats on the same task.

The subsections of this section will be arranged as follows: for relational databases, we categorize them into three levels based on the complexity of their relationships, specifically the number of tables in the database. These levels consist of datasets with 3-5 tables, 6-10 tables, and more than 10 tables, respectively. Additionally, to validate the effectiveness of our graph transformation, we select two datasets from OGB (Hu et al., 2020a) and carry out experiments on both original graphs and reconstructed graphs. Each level will be presented in a separate subsection.

**Baselines.** As for experiments, we provide three kinds of representative baselines, including (1) Tabular Data: **MLP**, **XGBoost** (Chen & Guestrin, 2016), (2) Homogeneous Graph: **GCN** (Kipf & Welling, 2017), **GIN** (Xu et al., 2018), **GAT** (Veličković et al., 2017), **GraphSage** (Hamilton et al., 2017), (3) Heterogeneous Graph: **HGCN** (Peng et al., 2019), **HGT** (Hu et al., 2020b).

### 7.1 RDBENCH-L0: GRAPH DATASETS AS RELATIONAL DATABASES

In the initial stage, we focus on graph data in machine learning. Leveraging graph datasets, our intention was to not only validate the effectiveness of our graph transformation but also indicate the wide applicability nature of relational databases. We transform graphs into relational databases first and then transform them back into graphs using our API. Since only Graphsage and GCN have mature results in both tasks in the GNN dataset leaderboard [2], we primarily conduct experiments on these two models. For other models, we attempt to design the architectures by ourselves. It's worth noting that on both initial and reconstructed graphs, we preserve all network architectures and conduct parameter tuning solely for network depth and learning rate. All details have been listed in Appendix D.

**Overview.** For RDBench-L0, i.e. graph datasets, we present 2 datasets: rdb0-arxiv and rdb0-mohiv, which respectively stem from OGB (Hu et al., 2020a) datasets ogbn-arxiv and ogbg-molhiv. The

[2]Graph leaderboard: `https://ogb.stanford.edu/docs/leader_overview/`

ogbn-arxiv dataset represents the paper citation network and the task is to classify papers into forty subject areas. The ogbg-molhiv is a molecular property prediction dataset and the objective is a binary graph classification task. The statistical information is shown in Table 1.

Table 1: Statistics of Original OGB Datasets

| Dataset | Task | #Graph | #Node | #Edge | Evaluation |
|---|---|---|---|---|---|
| ogbn-arxiv | Node Clas. | 1 | 169,343 | 1,166,243 | Accuracy |
| ogbg-molhiv | Graph Clas. | 41,127 | 25.5 | 27.5 | AUC-ROC |

Table 2: Experimental results on graph datasets rdb0-arxiv and rdb0-molhiv. The symbol Recon indicates results on the reconstructed graph, while those without it are from the original graph. Results are expressed in percentages. The best results are in bold and the second-best results are underlined.

| Model<br>Evaluation | rdb0-arxiv<br>Accuracy | rdb0-molhiv<br>AUC-ROC |
|---|---|---|
| MLP | 55.46 | 70.98 |
| XGBoost | 52.38 | 54.17 |
| GCN | <u>72.08</u> | <u>76.98</u> |
| GraphSage | **72.28** | 73.88 |
| Recon GCN | 70.18 | **77.21** |
| Recon GraphSage | 69.73 | 73.43 |

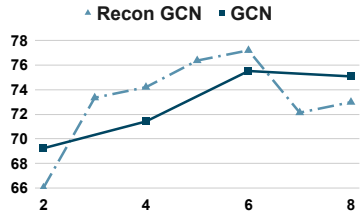



Figure 3: GCN Performance on rdb0-molhiv. X-axis is the number of GCN layers on the reconstructed graph and we double the actual number of layers in the original graph GNN to ensure correspondence. Results are expressed in percentages.

**Discussion.** The experimental results in Table 2 show that the best results of GCN and Graphsage closely resemble the well-established baseline on the original graph, which underscores the reliability of our modeling approach from the model perspective. GNN methods perform best in both tasks, which indicates the importance of structural information. In node classification tasks of rdb0-arxiv, XGBoost performs slightly worse than MLP, but in graph classification tasks rdb0-molhiv, XGBoost falls significantly behind MLP. Unlike node classification, where it's possible to aggregate first-order information for each node, providing truly effective graph features is quite hard for XGBoost.

Moreover, as discussed in Appendix D, regarding the properties of the reconstructed graph, on the right side of Figure 3, we illustrate the results of GCN on rdb0-molhiv dataset from the perspective of graph information at different depths. We observed the following: (1) At corresponding layers, the performance on both sides is close, indicating that the model exhibits similar information capture abilities on both the original and reconstructed graphs. (2) With an increase in the number of layers, the performance of the original GCN improves, and the performance of the transformed GCN also improves, with the optimal performance occurring at corresponding points. (3) With further layer increasing, over-smoothing occurs, leading to a decrease in model performance. This phenomenon becomes more pronounced due to the deeper layers in the reconstructed graph.

### 7.2 RDBench-L1: Relational Databases with 3-5 Tables

In the next three subsections, we turn to relational databases, which are relatively less explored in the machine learning community. We categorize our relational databases into 3 levels: databases with 3-5 tables, 6-10 tables, and more than 10 tables. At each level, we provide 3-4 representative relational databases with different sizes from different domains for hierarchical purpose.

**Overview.** For RDBench-L1, i.e. relational databases with 3-5 tables, we present 4 datasets: rdb1-ather, rdb1-rscore, rdb1-accdt, rdb1-seznam. The rdb1-ather dataset collects patients' medical examination information and their disease status. The rdb1-rscore dataset provides eateries inspection results. The rdb1-accdt dataset presents traffic accidents. The rdb1-seznam dataset collects the transaction data of Seznam's wallet. The statistical information is shown in Table 3.

**Discussion.** The experimental results from Table 4 show that, the XGBoost model demonstrated significantly superior performances compared to other models on the majority of selected tasks on datasets. Additionally, among the results of graph neural networks, GraphSage and HGCN achieve relatively better results. It is worth noticing that, HGCN emerged as the only graph model that outperformed XGBoost on a single task. Comparing the results of XGBoost and Join XGBoost, we

Table 3: Statistics of Relational Databases with 3-5 Tables (RDBench-L1)

| Dataset | Domain | # Tables | Diameter | # Cols | # Rows | # Clas. | # Reg. |
|---|---|---|---|---|---|---|---|
| rdb1-ather | Science | 4 | 2 | 198 | 12,781 | 4 | 2 |
| rdb1-rscore | Economic | 3 | 2 | 27 | 66,172 | 2 | 3 |
| rdb1-accdt | Society | 3 | 1 | 44 | 1,463,093 | 4 | 4 |
| rdb1-seznam | Economic | 4 | 2 | 17 | 2,689,257 | 4 | 2 |

Table 4: Experimental results on relational databases with 3-5 tables (RDBench-L1). Results are expressed in percentages for classification tasks, while as numerical values for regression tasks. The best results are in bold and the second-best results are underlined.

| Model | rdb1-ather | | rdb1-rscore | | rdb1-accdt | | rdb1-seznam | |
|---|---|---|---|---|---|---|---|---|
| Task | Clas. | Reg. | Clas. | Reg. | Clas. | Reg. | Clas. | Reg. |
| MLP | 27.05 | **0.156** | 75.56 | 0.610 | 53.02 | 1.353 | 31.66 | 0.014 |
| XGBoost | <u>35.00</u> | <u>0.159</u> | <u>86.98</u> | <u>0.077</u> | <u>66.36</u> | <u>0.019</u> | 37.97 | <u>0.005</u> |
| Join MLP | 21.88 | 0.190 | 53.41 | 0.812 | 50.49 | 3.764 | 31.33 | 0.401 |
| Join XGBoost | **36.09** | 0.162 | **91.40** | **0.065** | **67.76** | **0.019** | <u>44.69</u> | **0.003** |
| GCN | 29.88 | 0.186 | 60.67 | 0.186 | 58.96 | 0.261 | 39.08 | 0.040 |
| GIN | 28.30 | 0.402 | 52.10 | 0.781 | 35.88 | 1.105 | 41.49 | 0.381 |
| GraphSage | 32.80 | 0.176 | 70.57 | 0.139 | 63.62 | 0.214 | 37.57 | 0.031 |
| GAT | 28.66 | 0.173 | 53.61 | 0.149 | 53.13 | 0.280 | 32.24 | 0.029 |
| HGCN | 32.61 | 0.194 | 72.05 | 0.155 | 63.69 | 0.233 | **46.11** | 0.024 |
| HGT | 28.52 | 0.182 | 54.59 | 0.274 | 61.48 | 0.247 | 36.27 | 0.024 |

can discover that Join XGBoost obtains significant performance improvement from aggregating the information from 1-hop neighbors, which indicates the vital importance of neighborhood information. While considering the results for regression tasks, we find that the tree-based model: XGBoost can significantly surpasses other neural network based model in regression tasks, which presents the same trends as prior work on tabular data (Grinsztajn et al., 2022). An interesting result is that after gaining the information from one-hop neighborhood, XGBoost gains significant performance enhancement, while MLP gains a sharp performance drop. This phenomenon indicates that certain fixed size neural network may face difficulties capturing the real valuable information.

### 7.3 RDBench-L2: Relational Databases with 6-10 Tables

**Overview.** For RDBench-L2, i.e. relational databases with 6-10 tables, we present 4 datasets: rdb2-hbv, rdb2-ncaa, rdb2-ncaa, rdb2-imdb. The rdb2-hbv dataset presents the comparison of patients with different diseases. The rdb2-ncaa dataset collects the results of the basketball tournament. The rdb2-ncaa dataset contains successful and not successful loan data. The rdb2-imdb dataset provides the ratings of movies. The statistical information is shown in Table 5.

Table 5: Statistics of Relational Databases with 6-10 Tables (RDBench-L2)

| Dataset | Domain | # Tables | Diameter | # Cols | # Rows | # Clas. | # Reg. |
|---|---|---|---|---|---|---|---|
| rdb2-hbv | Science | 7 | 4 | 29 | 12,927 | 4 | 0 |
| rdb2-ncaa | Society | 9 | 4 | 112 | 202,305 | 2 | 4 |
| rdb2-bank | Economic | 8 | 3 | 55 | 1,079,680 | 5 | 5 |
| rdb2-imdb | Society | 7 | 4 | 27 | 1,249,411 | 5 | 0 |

**Discussion.** The experimental results from Table 6 show that, XGBoost is still the best method among baselines because of its dominating experimental performances. Besides, methods on heterogeneous graphs outperform methods on homogeneous graphs in most tasks, demonstrating the heterogeneous nature of our transformed data from relational databases. It is worth noticing that results on regression tasks show the same trend as Table 4, that tree-based method consistently outperforms neural network based models. However, in classification tasks, graph neural network methods are showing compatible performance compared with methods for tabular data, which indicates that relational information more than one-hop is also very important.

### 7.4 RDBench-L3: Relational Databases with 10+ Tables

**Overview.** For RDBench-L3, i.e. relational databases with 10+ tables, we present 3 datasets: rdb3-toxic, rdb3-lahman, rdb3-govern. The rdb3-toxic dataset presents the characteristics and carcinogenicity of compounds. The rdb3-lahman dataset is a baseball dataset collecting complete

Table 6: Experimental results on relational databases with 6-10 tables (RDBench-L2). Results are expressed in percentages for classification tasks, while as numerical values for regression tasks. The best results are in bold and the second-best results are underlined. And N/A means feature without join table is unavailable.

| **Model** | **rdb2-hbv** | **rdb2-ncaa** | | **rdb2-bank** | | **rdb2-imdb** |
|---|---|---|---|---|---|---|
| **Task** | Clas. | Clas. | Reg. | Clas. | Reg. | Clas. |
| MLP | 56.69 | 62.95 | 0.092 | 78.18 | 0.150 | N/A |
| XGBoost | 60.46 | <u>64.73</u> | <u>0.062</u> | 79.14 | <u>0.132</u> | N/A |
| Join MLP | 56.69 | 62.30 | 0.132 | 52.14 | 0.318 | 41.23 |
| Join XGBoost | 60.46 | **65.51** | **0.059** | **87.14** | **0.036** | 46.36 |
| GCN | 61.56 | 59.79 | 0.176 | 62.87 | 0.271 | 51.33 |
| GIN | 60.59 | 33.14 | 0.892 | 35.93 | 0.760 | 44.46 |
| GraphSage | <u>65.13</u> | 59.79 | 0.158 | 75.47 | 0.212 | 53.89 |
| GAT | 62.45 | 59.79 | 0.177 | 58.59 | 0.289 | 47.18 |
| HGCN | 64.46 | 59.79 | 0.197 | <u>83.53</u> | 0.243 | **60.84** |
| HGT | **72.74** | 59.79 | 0.173 | 66.89 | 0.205 | <u>60.39</u> |

batting and pitching statistics. The rdb3-govern dataset presents results of government elections. The statistical information is shown in Table 7.

Table 7: Statistics of Relational Databases with 10+ Tables (RDBench-L3)

| **Dataset** | **Domain** | **# Tables** | **Diameter** | **# Cols** | **# Rows** | **# Clas.** | **# Reg.** |
|---|---|---|---|---|---|---|---|
| rdb3-toxic | Science | 38 | 4 | 111 | 29,850 | 4 | 3 |
| rdb3-lahman | Society | 25 | 6 | 368 | 472,489 | 4 | 4 |
| rdb3-govern | Society | 19 | 7 | 138 | 803,901 | 4 | 0 |

Table 8: Experimental results on relational databases with 10+ tables (RDBench-L3). Results are expressed in percentages for classification tasks, while as numerical values for regression tasks. The best results are in bold and the second-best results are underlined. And N/A means features without joint table are unavailable.

| **Model** | **rdb3-toxic** | | **rdb3-lahman** | | **rdb3-govern** |
|---|---|---|---|---|---|
| **Task** | Clas. | Reg. | Clas. | Reg. | Clas. |
| MLP | N/A | N/A | 44.36 | 0.966 | 56.45 |
| XGBoost | N/A | N/A | <u>57.26</u> | **0.076** | 51.58 |
| Join MLP | 61.79 | 0.371 | 39.64 | 0.599 | 58.48 |
| Join XGBoost | 68.30 | 0.354 | **67.74** | <u>0.076</u> | <u>61.23</u> |
| GCN | 74.76 | 0.325 | 47.56 | 0.272 | 55.05 |
| GIN | 66.31 | 0.712 | 22.16 | 1.068 | 40.50 |
| GraphSage | 81.89 | <u>0.270</u> | 51.57 | 0.236 | 58.34 |
| GAT | 73.14 | 0.291 | 45.14 | 0.250 | 55.40 |
| HGCN | **88.67** | 0.393 | 50.23 | 0.669 | 57.86 |
| HGT | <u>88.43</u> | **0.235** | 48.42 | 0.261 | **61.97** |

**Discussion.** The experimental results from Table 8 show that, with the growing number of tables, the tree-based method XGBoost is beat by graph methods on more than half tasks and no longer holds a dominant position. This shift is related to the increased complexity of relationships and serves as significant evidence of the prominent role that relationships hold in relational databases. It is worth noticing that even in regression tasks, graph methods can outperform tabular methods, which is a substantial transformation, indicating that the relationship between tables is more and more important with the growth of number of tables, and should be taken into significant consideration.

## 8 Conclusion

In response to the gap between machine learning and relational databases, we present *ML Benchmark For Relational Databases* (RDBench). We provide hierarchical datasets and a user-friendly toolkit, offering various commonly used data format interfaces: tabular data, homogeneous graph data, and heterogeneous graph data. It's worth noticing that all these data formats share an identical task definition. We also carried out extensive experiments on these datasets. With the help of our unified task definition and easy-to-use interfaces, researchers from different ML subdomains can deploy models, get comparable results and further enhance collaboration.

## REFERENCES


Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, pp. 6679–6687. AAAI Press, 2021.

Jinze Bai, Jialin Wang, Zhao Li, Donghui Ding, Ji Zhang, and Jun Gao. Atj-net: Auto-table-join network for automatic learning on relational databases. In *WWW*, pp. 1540–1551. ACM / IW3C2, 2021.

Petr Berka. Workshop notes on Discovery Challenge PKDD'99. 1999. URL `http://lisp.vse.cz/pkdd99/`.

Jintai Chen, Kuanlun Liao, Yao Wan, Danny Z Chen, and Jian Wu. Danets: Deep abstract networks for tabular data classification and regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3930–3938, 2022.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pp. 785–794. ACM, 2016.

Yijian Cheng, Pengjie Ding, Tongtong Wang, Wei Lu, and Xiaoyong Du. Which category is better: benchmarking relational and graph database management systems. *Data Science and Engineering*, 4:309–322, 2019.

Milan Cvitkovic. Supervised learning on relational databases with graph neural networks. *CoRR*, abs/2002.02046, 2020.

A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991. ISSN 0022-2623. doi: 10.1021/jm00106a046.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. IEEE Computer Society, 2009.

Djellel Eddine Difallah, Andrew Pavlo, Carlo Curino, and Philippe Cudre-Mauroux. Oltp-bench: An extensible testbed for benchmarking relational databases. *Proceedings of the VLDB Endowment*, 7 (4):277–288, 2013.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *NeurIPS*, pp. 18932–18943, 2021.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *NeurIPS*, 2022.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. In *IJCAI*, pp. 1725–1731. ijcai.org, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778. IEEE Computer Society, 2016.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pp. 2704–2710, 2020b.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar S. Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *CoRR*, abs/2012.06678, 2020.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. In *NeurIPS*, pp. 23928–23941, 2021.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, pp. 3146–3154, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR (Poster)*. OpenReview.net, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Guang Liu, Jie Yang, and Ledell Wu. Ptab: Using the pre-trained language model for modeling tabular data. *CoRR*, abs/2209.08060, 2022.

Shengchao Liu, David Vázquez, Jian Tang, and Pierre-André Noël. Flaky performances when pretraining on relational databases (student abstract). In *AAAI*, pp. 16266–16267. AAAI Press, 2023.

Erxue Min, Yu Rong, Tingyang Xu, Yatao Bian, Peilin Zhao, Junzhou Huang, Da Luo, Kangyi Lin, and Sophia Ananiadou. Masked transformer for neighhourhood-aware click-through rate prediction. *CoRR*, abs/2201.13311, 2022.

Jan Motl and Oliver Schulte. The ctu prague relational learning repository. *arXiv preprint arXiv:1511.03086*, 2015.

OpenAI. Gpt-4 technical report, 2023.

Hao Peng, Jianxin Li, Qiran Gong, Yangqiu Song, Yuanxing Ning, Kunfeng Lai, and Philip S Yu. Fine-grained event categorization with heterogeneous graph convolutional networks. *arXiv preprint arXiv:1906.04580*, 2019.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *ICLR*. OpenReview.net, 2020.

Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *NeurIPS*, pp. 6639–6649, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pp. 593–607. Springer, 2018.

Ira Shavitt and Eran Segal. Regularization learning networks: Deep learning for tabular datasets. In *NeurIPS*, pp. 1386–1396, 2018.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. SAINT: improved neural networks for tabular data via row attention and contrastive pre-training. *CoRR*, abs/2106.01342, 2021.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. In *NeurIPS*, 2022.

Jeyan Thiyagalingam, Mallikarjun Shankar, Geoffrey Fox, and Tony Hey. Scientific machine learning benchmarks. *Nature Reviews Physics*, 4(6):413–420, 2022.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). *arXiv preprint arXiv:2206.10498*, 2022.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR (Poster)*. OpenReview.net, 2019.

Junfu Wang, Yuanfang Guo, Liang Yang, and Yunhong Wang. Enabling homogeneous gnns to handle heterogeneous graphs via relation embedding. *IEEE Transactions on Big Data*, 2023a.

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1): 396–413, 2020.

Zifeng Wang and Jimeng Sun. Transtab: Learning transferable tabular transformers across tables. In *NeurIPS*, 2022.

Zifeng Wang, Chufan Gao, Cao Xiao, and Jimeng Sun. Anypredict: Foundation model for tabular prediction. *arXiv preprint arXiv:2305.12081*, 2023b.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401, 2022.

Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. Graph convolutional networks with markov random field reasoning for social spammer detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 1054–1061, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Chao Ye, Guoshan Lu, Haobo Wang, Liyao Li, Sai Wu, Gang Chen, and Junbo Zhao. CT-BERT: learning better tabular representations through cross-table pre-training. *CoRR*, abs/2307.04308, 2023.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, Tao Zhang, Chen Zhou, Kaizhe Shou, Miao Wang, Wufang Zhu, Guoshan Lu, Chao Ye, Yali Ye, Wentao Ye, Yiming Zhang, Xinglong Deng, Jie Xu, Haobo Wang, Gang Chen, and Junbo Zhao. Tablegpt: Towards unifying tables, nature language and commands into one GPT. *CoRR*, abs/2307.08674, 2023.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. *CoRR*, abs/2304.06364, 2023.

Xuanhe Zhou, Cheng Chen, Kunyi Li, Bingsheng He, Mian Lu, Qiaosheng Liu, Wei Huang, Guoliang Li, Zhao Zheng, and Yuqiang Chen. Febench: A benchmark for real-time relational data feature extraction. *Proceedings of the VLDB Endowment*, 16(12):3597–3609, 2023.

# A NOTIONS

All necessary notations are organized below.

**Notations**

| | |
|---|---|
| $\mathbf{D}$ | The relational database |
| $\boldsymbol{T}_n$ | The $n$-th table in relational database $\mathbf{D}$ |
| $\boldsymbol{T}_n^{r:}$, $\boldsymbol{T}_n^{:c}$ | The $n$-th table's $r$-th row, $c$-th column |
| $\boldsymbol{T}_n^{r:c}$ | The value in the $c$-th column of the $r$-th row of the $n$-th table |
| $F_n$, $K_n$ | The features and foreign keys of the $n$-th table $\boldsymbol{T}_n$ |
| $k_{n_w}^r \rightarrow \boldsymbol{T}_m^{t:}$ | The $w$-th foreign key of the $r$-th row of the table $\boldsymbol{T}_n$ points to the $t$-th row of table $\boldsymbol{T}_m$ |
| $k_{n_w} \rightarrow \boldsymbol{T}_m$ | The $w$-th foreign key of table $\boldsymbol{T}_n$ points to table $\boldsymbol{T}_m$ |
| $\mathcal{G}_{\text{homo}}$, $\mathcal{G}_{\text{hetero}}$ | Homogeneous Graph, Heterogeneous Graph |
| $\mathcal{V}$, $\mathcal{E}$ | The set of nodes, edges in Graph $\mathcal{G}$ |

# B IMPLEMENTATION DETAILS OF RELATIONAL DATABASES

## B.1 HYPERPARAMETERS

**Neural network based models.** For all neural network-based models, we adopt Adam optimizer (Kingma & Ba, 2014) with learning rate set to $0.001$ and a cosine learning rate scheduler. The number of layers is set to $4$ considering the diameter of datasets, and the hidden dimension is set to $128$. More specifically, number of head is set to $4$ for GAT and HGT. For dataset rdb1-seznam, number of epoch is set to 50. For datasets rdb1-accdt, rdb2-bank, rdb2-imdb, the number of epochs is set to 100. For other datasets, number of epoch is set to 200. Early stop epochs is set to 50, i.e. training without better results on validation set for 50 epochs is terminated.

**Tree based models.** For the only tree-based model XGBoost, we adopt the default configuration. Max depth is set to 3, learning rate is set to $0.1$, number of estimators is set to $100$, booster is set to gbtree, and weight of L2 regularization is set to $1$.

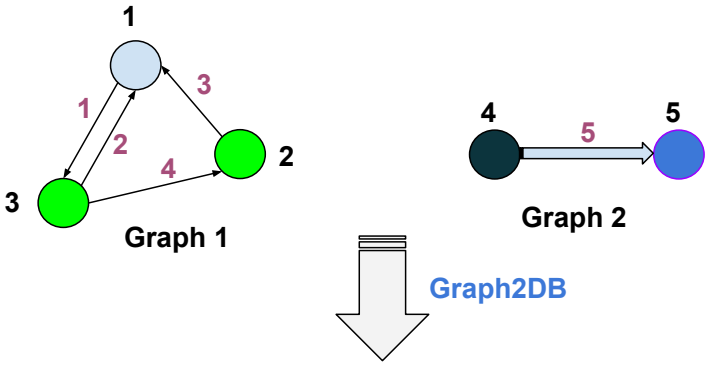
## B.2 EVALUATION

Datasets are randomly split into training set, validation set and testing set, the ratio is $0.8$, $0.1$ and $0.1$. And we conduct 3 times for each experiment, reporting the detailed average and squared error in the Appendix F.

# C FROM GRAPHS TO DATABASES

Expanding upon the approach introduced in (Debnath et al., 1991), we present a unified method for transforming graphs into up to three essential tables. This approach enables the flexible migration of node, edge, and graph classification tasks from the graph domain to a database environment.

More specifically, our database consists of three tables, denoted as $T_{edge}$, $T_{node}$, and $T_{graph}$, each dedicated to different data types: edges, nodes, and graphs. Notably, the first two columns in $T_{edge}$ contain node information, serving as foreign keys to $T_{node}$. Additionally, the second column of $T_{node}$ includes graph information for each node, acting as a foreign key to $T_{graph}$. The remaining columns in each table are features and types. We also provide an example of a graph containing two directed subgraphs, as illustrated in Figure 4 below.




| Edge_id | Node_id | Node_id2 | Edge_type | Edge_feature1 | Edge_feature2 |
|---|---|---|---|---|---|
| 1 | 1 | 3 | 0 | 0.21 | 0.69 |
| 2 | 3 | 1 | 0 | 0.82 | 0.43 |
| 3 | 2 | 1 | 0 | 0.71 | 0.49 |
| 4 | 3 | 2 | 0 | 0.79 | 0.12 |
| 5 | 4 | 5 | 1 | 0.38 | 0.75 |

**Node**

| Node_id | Graph_id | Node_type | Node_feature1 | Node_feature2 |
|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 0.52 |
| 2 | 3 | 2 | 5 | 0.93 |
| 3 | 2 | 2 | 8 | 0.63 |
| 4 | 3 | 3 | 1 | 0.65 |
| 5 | 4 | 4 | 10 | 0.56 |

**Edge**

| Graph_id | Graph_type | Graph_feature1 | Graph_feature2 |
|---|---|---|---|
| 1 | 1 | 0.99 | 0.12 |
| 2 | 2 | 0.49 | 0.63 |

**Graph**

Figure 4: An example from Graphs to Databases. In this illustration, we present two graphs, where node order is delineated in black, and the node sequence is highlighted in Dark Magenta. Different colored nodes represent different types. Nodes of different colors denote distinct types. Within Graph 1, we assume uniform node types, but the edge types vary between the two graphs. Node, edge, and graph features are represented by two-dimensional random numbers.

## D Analysis of Reconstructed Graph Properties

### D.1 Transformed Graphs

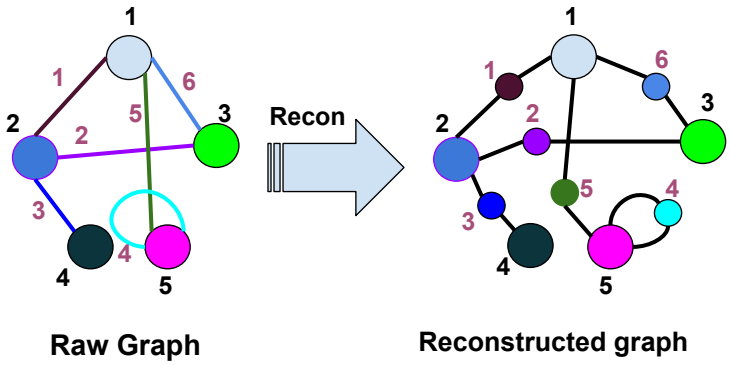



Figure 5: An Example of Graph Transformation Process. In this illustration, node order is delineated in black, and the node sequence is highlighted in Dark Magenta. The nodes and edges of the raw graph are marked with the same colors as their corresponding nodes in the reconstructed graph. The symbol (Recon) the reconstruction process

At Level 0, we conducted comparative experiments between the original graph and the reconstructed graph. Concurrently, to enhance our comprehension of the characteristics of the reconstructed graph, we provide an example in Figure 5 here. Subsequently, we take the topological properties on the homogenous graph as an example. By comparing the original graph with the transformed homogenous graph, we can discover some interesting properties and better understand the modeling process.

### D.2 THE MODELING PROCESS

We denote the original graph as $\mathcal{G}$, the transformed graph as $\tilde{\mathcal{G}}$. The sets of nodes and edges in the original graph are $\mathcal{V}$ and $\mathcal{E}$, while the sets of nodes and edges in the transformed graph are $\tilde{\mathcal{V}}$ and $\tilde{\mathcal{E}}$. The reconstruction mapping is denoted as $f$, and we describe this process as follows:

$$\mathcal{G} \xrightarrow{f} \tilde{\mathcal{G}} \tag{5}$$

$$\forall v_i \in \mathcal{V}, \quad v_i \xrightarrow{f} \tilde{v}_i \in \tilde{\mathcal{V}} \tag{6}$$

$$\forall e_{(s,t)} \in \mathcal{E}, \quad e_{(s,t)} \xrightarrow{f} \tilde{v}_{e_{(s,t)}} \in \tilde{\mathcal{V}} \tag{7}$$

Furthermore, if nodes $v_i$ and $e_{(s,t)}$ are connected in $\mathcal{G}$, then nodes $\tilde{v}_i$ and $\tilde{v}_{e_{(s,t)}}$ are connected as well.

### D.3 PROPERTIES

Shortest Path Distance(SPD) is used to describe the number of edges that two nodes or edges pass along the shortest path (SPD >= 1). In message-passing neural networks, information needs to go through "SPD" layers in order to be transmitted from one node to another.

**Proposition 1.** *(Shortest Path Distance): Let edge $e_{(i,j)}$ represent the connection between nodes $v_i$ and $v_j$. $\mathcal{V}$ and $\tilde{\mathcal{V}}$ respectively represent the node sets of the original graph and the reconstructed graph. The relationship between the Shortest Path Distance on the original graph $\mathcal{G}$ and the reconstructed graph $\tilde{\mathcal{G}}$ can be described as:*

- *For the SPD between nodes after transformation,*

$$\forall v_i, v_j \in \mathcal{V}, \quad SPD(\tilde{v}_i, \tilde{v}_j) = 2 * SPD(v_i, v_j) \tag{8}$$

- *For the SPD between edges and nodes after transformation,*

$$\forall v_i \in \mathcal{V}, \forall e_{s,t} \in \mathcal{E}, \quad SPD(\tilde{v}_i, \tilde{v}_{e_{s,t}}) = 2 * \min_{(a,b)\in\{(i,s),(i,t)\}} SPD(v_a, v_b) \tag{9}$$

- *For the SPD between edges after transformation,*

$$\forall e_{s,t}, e_{i,j} \in \mathcal{E}, \quad SPD(\tilde{v}_{e_{i,j}}, \tilde{v}_{e_{s,t}}) = 2 * \min_{(a,b)\in\{(i,s),(i,t),(j,s),(j,t)\}} SPD(v_a, v_b) + 1 \tag{10}$$

Here, we count each self-loop as an increment of one to the node's degree. For instance, in the original graph 5, the degree of node 5 is recorded as 2.

**Proposition 2.** *(Degrees of nodes) Supoose $\nVdash^{v_i}_{self}$ denote the number of self-loop edges for node $v_i$. The node degrees on the reconstructed graph $\tilde{\mathcal{G}}$ can be described as:*

- *For nodes transformed from edges,*

$$\forall e_{s,t} \in \mathcal{E}, \quad deg(\tilde{v}_{e_{s,t}}) = 2 \tag{11}$$

- *For nodes transformed from nodes,*

$$\forall v_i \in \mathcal{V}, \quad deg(\tilde{v}_i) = deg(v_i) + \nVdash^{v_i}_{self} \tag{12}$$

**Proposition 3.** *(Number of nodes and edges) Suppose the original graph $\mathcal{G}$ has $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges, the transformed graph $\tilde{\mathcal{G}}$ has $|\tilde{\mathcal{V}}| = |\mathcal{V}| + |\mathcal{E}|$ nodes. The number of edges $|\tilde{\mathcal{E}}|$ is equal to $2 * |\mathcal{E}|$.* $\mathcal{A} = \tilde{\mathcal{A}}^2[node\_index]$

This proposition can be obtained by summing the degrees of all nodes.

$$\frac{1}{2}\{\sum_{v_i \in \mathcal{V}} (deg(v_i) + \mathbb{1}_{self}^{v_i}) + \sum_{e_{(s,t)} \in \mathcal{E}} 2\} \quad = \quad 2 * |\mathcal{E}| \tag{13}$$

Furthermore, we calculate the average node degrees, denoted as $\rho_{\mathcal{G}}$ and $\rho_{\tilde{\mathcal{G}}}$. When compared to the original graph, it becomes apparent that the transformed graph tends to be sparser, with an average degree not exceeding two.

$$\rho_{\mathcal{G}} = \frac{|\mathcal{E}|}{|\mathcal{V}|}, \quad \rho_{\tilde{\mathcal{G}}} = 2 * \frac{|\mathcal{E}|}{|\mathcal{V}| + |\mathcal{E}|} = 2 * \frac{\rho_{\mathcal{G}}}{1 + \rho_{\mathcal{G}}} \leq 2, \quad \frac{\rho_{\tilde{\mathcal{G}}}}{\rho_{\mathcal{G}}} = \frac{2}{1 + \rho_{\mathcal{G}}} \tag{14}$$

# E LEVEL0 EXPERIMENT DETAILS

In regard to each result, we conducted three experiments and took the average. For each model, we perform a grid search where we adjust the learning rate in the range of $\{0.1, 0.01, 0.001, 0.0005, 0.00001\}$. For the original graph, we tune the number of layers in the range of $\{1, 2, 3, 4, 5\}$. For the reconstructed graph, we tune the number of layers in the range of $\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

Here are the specific procedures for each experiment.

## E.1 NODE PREDICTION: RDB0-ARXIV

rdb0-arxiv is a homogeneous graph dataset where each node has a 128-dimensional feature vector. The ultimate prediction task is a multiclass classification problem with 40 categories.

**Implementation details.**

1. We established baseline results by directly applying Graph Neural Networks (GNNs) to the original graph.
2. For the Multilayer Perceptron (MLP) model, we generated predictions by directly utilizing the features of each node.
3. In the case of XGBoost, we aggregated the average features of each node's neighbors and concatenated them with the original node features, enabling XGBoost to make predictions.
4. Initially, we converted the graph into a database and subsequently transformed it back into a new graph using our proposed API. We ensured that the training set, validation set, test set, and the GNN architecture precisely matched the original graph to guarantee fairness in the comparison.

## E.2 GRAPH PREDICTION: RDB0-MOLHIV

rdb0-molhiv is a homogeneous graph dataset that incorporates both node and edge features. The prediction task can be described as a binary classification problem, with model performance assessed using the ROC-AUC metric.

**Implementation details.**

1. In MLP, the original node features are used as input for the prediction task.
2. For xgboost, we calculate the mean, sum, or maximum of the features of all nodes or edges and concatenate them together as each graph's features for training. We retain the best results among these three operations.
3. The reconstructed graph and the original graph utilize consistent label indexing and splitting.

### E.3 EXPERIMENTS ON GAT AND GIN

Since the aim of our level 0 experiments is to demonstrate the validity of the graph API modeling approach rather than proposing new GNN algorithms, we fixed the number of GAT headers and the default settings for each layer of GIN. Parameter tuning was performed only by adjusting the learning rate and network depth.

| **Model** / **Evaluation** | **rdb0-arxiv** Accuracy | **rdb0-molhiv** AUC-ROC |
|---|---|---|
| MLP | 55.46 | 70.98 |
| XGBoost | 52.38 | 54.17 |
| GIN | <u>62.13</u> | **75.23** |
| GAT | **69.83** | 69.64 |
| Recon GIN | 55.00 | <u>73.65</u> |
| Recon GAT | 61.28 | 66.70 |

Table 9: Experimental results of GAT and GIN on graph datasets(Level 0). All results are expressed in percentages. The symbol (Recon) indicates results on the reconstructed graph, while those without it are from the original graph. The best results are in bold and the second-best results are underlined

**Discussion.** Similar to the results in the main text, XGBoost performs slightly worse than MLP in node classification tasks in Table 9. Due to the challenge of transforming the entire graph information into a unified feature representation for XGBoost, it significantly lags behind MLP in graph classification tasks. In the case of the rdb0-molhiv, GIN leverage graph information but exhibit mixed performance compared to MLP. Moreover, the performance gap between GIN and GAT on both the original graph and the reconstructed graph is larger than that shown in Table 3 of the main text for GCN and GraphSage. This difference can be attributed, in large part, to the fact that we have rigidly constrained several hyperparameters and model architectures. Simultaneously, it underscores the challenge of identifying appropriate learning rates and network depths within the confines of the constrained parameter search space, particularly when dealing with larger graphs.

## F DETAILED DATASETS INFORMATION AND EXPERIMENTAL RESULTS

In this section, we will list the background information of our datasets to help you better understand the tasks.

### F.1 RDBENCH-L1

#### F.1.1 RDB1-ATHER

This dataset is about one study on atherosclerosis. The study aims to identify prevalence of atherosclerosis risk factors in middle-aged men, who are considered to be the most influenced by the disease.

The explanation of this dataset is in Table 10. And the detail of experiments is in 11 and 12.

#### F.1.2 RDB1-RSCORE

This dataset is about food safety of restaurants in San Francisco. It contains Information about inspections on those restaurants, and violations found during these inspections.

The explanation of this dataset is in Table 13. And the detail of experiments is in 14 and 15.

#### F.1.3 RDB1-ACCDT

This is one dataset consisting of all accidents that happened in Slovenia's capital city Ljubljana between the years 1995 and 2005.

The explanation of this dataset is in Table 16. And the detail of experiments is in 17 and 18.

#### F.1.4 RDB1-SEZNAM

Seznam.cz is a web portal and search engine in the Czech Republic. The data represent online advertisement expenditures of users' accounts. Normally, users need to prepay some money into their accounts, and Seznam will charge from their accounts if users utilize some service. In some time, payments are made directly by users without utilizing their prepaid balance.

The explanation of this dataset is in Table 19. And the detail of experiments is in 20 and 21.

### F.2 RDBENCH-L2

#### F.2.1 RDB2-HBV

This dataset describes examinations of 206 patients of Hepatitis B or C.

The explanation of this dataset is in Table 22. And the detail of experiments is in 23.

#### F.2.2 RDB2-NCAA

This dataset is about NCAA men's basketball tournament. It contains information about historical games from 1985 to 2014 and detailed information for games from 2003 to 2014.

The explanation of this dataset is in Table 24. And the detail of experiments is in 25 and 26.

#### F.2.3 RDB2-BANK

Financial dataset contains 606 successful and 76 not successful loans along with their information and transactions.

The explanation of this dataset is in Table 27. And the detail of experiments is on 28 and 29.

#### F.2.4 RDB2-IMDB

MovieLens is a web-based recommender system and virtual community. This dataset contains users' rating on movies, actors and directors.

The explanation of this dataset is in Table 30. And the detail of experiments is in 31.

### F.3 RDBENCH-L3

#### F.3.1 RDB3-TOXIC

This dataset is about whether one compound is carcinogenic, or not. It contains information on the structure of compounds.

The explanation of this dataset is in Table 32. And the detail of experiments is in 33 and 34.

#### F.3.2 RDB3-LAHMAN

This dataset comes from famous reporter Sean Lahman. It contains information about baseball competitions from 1871 to 2014, including complete batting and pitching statistics, fielding statistics, standings, team statistics , managerial records, post-season data, and more.

The explanation of this dataset is in Table 35. And the detail of experiments is in 36 and 37.

#### F.3.3 RDB3-GOVERN

This dataset is about parliamentarians from the Czech Republic, and votes on bills.

The explanation of this dataset is in Table 38. And the detail of experiments is in 39.

Table 10: Basic Information of tasks on rdb1-ather, # Class means the number of classes, StdErr means standard error.

| Table | Target | Type | # Class | StdErr |
|---|---|---|---|---|
| death | DeathReason | Clas. | 8 | N/A |
| death | DeathMonth | Clas. | 12 | N/A |
| entry | Group | Clas. | 6 | N/A |
| control | ExamYear | Clas. | 24 | N/A |
| entry | Cholesterol | Reg. | N/A | 0.245 |
| entry | Triglycerides | Reg. | N/A | 0.166 |

Table 11: Experiment result details of classification tasks on dataset rdb1-ather. The best results are in bold and the second-best results are underlined.

| **Table** | death | | entry | control |
|---|---|---|---|---|
| **Column** | DeathReason | DeathMonth | Group | ExamYear |
| MLP | 27.03±2.21 | 7.21±5.55 | 44.21±0.33 | 29.77±1.65 |
| XGBoost | 25.22±1.28 | <u>10.81±0.00</u> | 58.87±2.31 | <u>45.13±2.13</u> |
| Join MLP | 18.02±4.59 | 9.01±4.59 | 43.02±0.89 | 17.50±1.64 |
| Join XGBoost | 26.13±3.37 | 9.01±3.37 | 59.58±5.05 | **49.67±1.86** |
| GCN | 25.23±2.55 | 8.11±2.20 | 76.83±5.47 | 9.37±1.70 |
| GIN | **30.63±5.10** | 6.31±2.55 | 68.79±6.10 | 7.47±0.66 |
| GraphSage | <u>27.93±10.20</u> | 5.41±2.21 | **81.56±3.01** | 16.34±0.71 |
| GAT | 22.52±3.37 | 2.70±2.21 | <u>79.20±2.61</u> | 10.22±4.41 |
| HGCN | 26.13±3.37 | **12.61±2.55** | 78.25±3.49 | 13.47±1.38 |
| HGT | 23.43±5.10 | 9.01±2.55 | 73.29±0.67 | 8.39±1.05 |

Table 12: Experiment result details of regression tasks on dataset rdb1-ather. The best results are in bold and the second-best results are underlined.

| **Table** | entry | |
|---|---|---|
| **Column** | Cholesterol | Triglycerides |
| MLP | **0.161±0.010** | **0.150±0.027** |
| XGBoost | 0.164±0.012 | <u>0.155±0.023</u> |
| Join MLP | 0.225±0.011 | 0.155±0.029 |
| Join XGBoost | <u>0.161±0.012</u> | 0.162±0.026 |
| GCN | 0.196±0.016 | 0.175±0.042 |
| GIN | 0.439±0.302 | 0.366±0.167 |
| GraphSage | 0.180±0.002 | 0.171±0.036 |
| GAT | 0.178±0.007 | 0.168±0.037 |
| HGCN | 0.208±0.013 | 0.179±0.033 |
| HGT | 0.197±0.007 | 0.167±0.047 |

Table 13: Basic Information of tasks on rdb1-rscore, # Class means the number of classes, StdErr means standard error.

| Table | Target | Type | # Class | StdErr |
|---|---|---|---|---|
| violations | RiskLevel | Clas. | 3 | N/A |
| inspections | Type | Clas. | 12 | N/A |
| inspections | Score | Reg. | N/A | 0.236 |
| restaurant | Latitude | Reg. | N/A | 0.079 |
| restaurant | Longitude | Reg. | N/A | 0.002 |

Table 14: Experiment result details of classification tasks on dataset rdb1-rscore. The best results are in bold and the second-best results are underlined.

| **Table** | violations | inspections |
|---|---|---|
| **Column** | RiskLevel | Type |
| MLP | 95.65±1.44 | 55.47±0.39 |
| XGBoost | **100.00±0.00** | <u>73.96±0.21</u> |
| Join MLP | 51.48±0.47 | 55.34±0.27 |
| Join XGBoost | <u>100.00±0.00</u> | **82.80±0.86** |
| GCN | 65.23±9.86 | 56.12±0.86 |
| GIN | 48.65±4.47 | 55.56±0.32 |
| GraphSage | 84.74±8.74 | 56.41±1.03 |
| GAT | 51.86±0.64 | 55.37±0.80 |
| HGCN | 81.48±7.34 | 62.63±1.24 |
| HGT | 53.83±0.79 | 55.36±0.79 |

Table 15: Experiment result details of regression tasks on dataset rdb1-rscore. The best results are in bold and the second-best results are underlined.

| **Table** | inspections | restaurant | |
|---|---|---|---|
| **Column** | Score | Latitude | Longitude |
| MLP | 0.228±0.008 | 0.841±0.244 | 0.760±0.208 |
| XGBoost | 0.229±0.005 | **0.000±0.000** | **0.000±0.000** |
| Join MLP | 0.833±0.096 | 0.841±0.244 | 0.760±0.208 |
| Join XGBoost | **0.196±0.003** | <u>0.000±0.000</u> | <u>0.000±0.000</u> |
| GCN | 0.207±0.008 | 0.266±0.027 | 0.084±0.001 |
| GIN | 0.700±0.693 | 1.283±0.202 | 0.362±0.036 |
| GraphSage | 0.211±0.004 | 0.129±0.015 | 0.076±0.002 |
| GAT | 0.233±0.007 | 0.156±0.014 | 0.058±0.017 |
| HGCN | 0.213±0.004 | 0.169±0.005 | 0.083±0.001 |
| HGT | <u>0.206±0.012</u> | 0.537±0.021 | 0.078±0.006 |

Table 16: Basic Information of tasks on rdb1-accdt, # Class means the number of classes, StdErr means standard error.

| Table | Target | Type | # Class | StdErr |
|---|---|---|---|---|
| person | IsCauseOfAccident | Clas. | 2 | N/A |
| accident | WeatherCondition | Clas. | 11 | N/A |
| accident | TrafficCondition | Clas. | 5 | N/A |
| accident | AccidentClass | Clas. | 6 | N/A |
| person | Age | Reg. | N/A | 0.295 |
| person | Experience | Reg. | N/A | 0.203 |
| accident | X | Reg. | N/A | 0.407 |
| accident | Y | Reg. | N/A | 0.371 |

Table 17: Experiment result details of classification tasks on dataset rdb1-accdt. The best results are in bold and the second-best results are underlined.

| **Table** | person | | accident | |
|---|---|---|---|---|
| **Column** | IsCauseOfAccident | WeatherCondition | TrafficCondition | AccidentClass |
| MLP | 64.50±0.16 | 23.78±1.31 | 55.94±0.16 | 67.87±0.26 |
| XGBoost | 65.94±0.12 | <u>58.63±0.29</u> | <u>62.51±0.16</u> | 78.39±0.15 |
| Join MLP | 54.69±0.02 | 23.48±1.17 | 55.96±0.17 | 67.87±0.26 |
| Join XGBoost | **71.23±0.08** | **58.86±0.34** | **62.58±0.09** | 78.41±0.09 |
| GCN | 59.12±0.64 | 39.88±0.45 | 56.12±0.04 | 80.74±2.91 |
| GIN | 50.32±0.44 | 20.73±2.83 | 22.46±23.60 | 50.03±23.68 |
| GraphSage | <u>66.18±0.15</u> | 42.90±0.10 | 57.43±0.12 | <u>88.00±0.43</u> |
| GAT | 59.26±0.15 | 29.73±0.50 | 55.63±0.17 | 67.93±0.08 |
| HGCN | 63.21±1.07 | 44.45±1.74 | 57.83±0.13 | **89.29±2.24** |
| HGT | 60.25±0.13 | 41.68±0.21 | 56.15±0.77 | 87.87±1.09 |

Table 18: Experiment result details of regression tasks on dataset rdb1-accdt. The best results are in bold and the second-best results are underlined.

| **Table** | person | | accident | |
|---|---|---|---|---|
| **Column** | Age | Experience | X | Y |
| MLP | 0.069±0.002 | 0.048±0.004 | 2.552±0.276 | 2.742±0.412 |
| XGBoost | **0.046±0.000** | <u>0.029±0.000</u> | **0.001±0.000** | **0.001±0.000** |
| Join MLP | 5.331±2.057 | 4.248±1.632 | 3.203±2.441 | 2.275±0.778 |
| Join XGBoost | <u>0.046±0.000</u> | **0.029±0.000** | <u>0.001±0.000</u> | <u>0.001±0.000</u> |
| GCN | 0.274±0.004 | 0.180±0.002 | 0.288±0.013 | 0.301±0.004 |
| GIN | 1.067±0.561 | 0.808±0.278 | 1.254±0.475 | 1.293±0.484 |
| GraphSage | 0.227±0.012 | 0.136±0.004 | 0.238±0.008 | 0.256±0.008 |
| GAT | 0.288±0.001 | 0.189±0.004 | 0.330±0.004 | 0.312±0.005 |
| HGCN | 0.243±0.004 | 0.166±0.005 | 0.264±0.016 | 0.260±0.012 |
| HGT | 0.260±0.003 | 0.163±0.001 | 0.284±0.032 | 0.282±0.020 |

Table 19: Basic Information of tasks on rdb1-seznam, # Class means the number of classes, StdErr means standard error.

| Table | Target | Type | # Class | StdErr |
|---|---|---|---|---|
| charge | PrepayService | Clas. | 8 | N/A |
| prepay | PrepayService | Clas. | 8 | N/A |
| client | Location | Clas. | 14 | N/A |
| client | Domain | Clas. | 52 | N/A |
| charge | ChargeMoney | Reg. | N/A | 0.007 |
| prepay | PrepayMoney | Reg. | N/A | 0.005 |

Table 20: Experiment result details of classification tasks on dataset rdb1-seznam. The best results are in bold and the second-best results are underlined.

| **Table** | charge | prepay | client | |
|---|---|---|---|---|
| **Column** | PrepayService | PrepayService | Location | Domain |
| MLP | 42.63±0.28 | 48.51±0.06 | 26.83±0.43 | 8.70±0.40 |
| XGBoost | 59.03±0.01 | 56.08±0.06 | 27.31±0.27 | 9.46±0.42 |
| Join MLP | 42.03±1.21 | 47.78±0.09 | 26.83±0.43 | 8.70±0.40 |
| Join XGBoost | **71.98±0.09** | 70.02±0.27 | 27.31±0.27 | 9.46±0.42 |
| GCN | 49.60±5.48 | 65.24±3.46 | <u>27.93±0.40</u> | 13.59±0.90 |
| GIN | 53.29±10.53 | <u>70.02±4.06</u> | 27.92±0.41 | **14.75±0.07** |
| GraphSage | 49.86±6.81 | 59.19±0.04 | 27.93±0.40 | 13.33±1.06 |
| GAT | 43.67±2.53 | 47.72±0.23 | 27.93±0.40 | 9.65±1.82 |
| HGCN | <u>68.86±0.32</u> | **72.88±1.43** | **28.28±0.41** | <u>14.44±0.51</u> |
| HGT | 47.20±3.70 | 57.09±3.35 | 27.93±0.40 | 12.88±1.04 |

Table 21: Experiment result details of regression tasks on dataset rdb1-seznam. The best results are in bold and the second-best results are underlined.

| **Table** | charge | prepay |
|---|---|---|
| **Column** | ChargeMoney | PrepayMoney |
| MLP | 0.010±0.004 | 0.019±0.008 |
| XGBoost | <u>0.006±0.001</u> | **0.004±0.001** |
| Join MLP | 0.161±0.068 | 0.641±0.076 |
| Join XGBoost | **0.003±0.000** | <u>0.004±0.001</u> |
| GCN | 0.008±0.002 | 0.073±0.003 |
| GIN | 0.259±0.157 | 0.503±0.346 |
| GraphSage | 0.008±0.001 | 0.054±0.007 |
| GAT | 0.006±0.001 | 0.052±0.010 |
| HGCN | 0.010±0.002 | 0.039±0.007 |
| HGT | 0.006±0.001 | 0.043±0.005 |

Table 22: Basic Information of tasks on rdb2-hbv, # Class means the number of classes.

| Table | Target | Type | # Class |
|---|---|---|---|
| patient | Type | Clas. | 2 |
| exam_result | Got | Clas. | 5 |
| exam_result | Ztt | Clas. | 6 |
| exam_result | Tp | Clas. | 4 |

Table 23: Experiment result details on dataset rdb2-hbv. The best results are in bold and the second-best results are underlined.

| **Type** | **Classification** | | | |
|---|---|---|---|---|
| **Table** | patient | exam_result | | |
| **Column** | Type | Got | Ztt | Tp |
| MLP | 54.00±9.09 | 64.26±1.70 | 56.36±1.33 | 52.14±3.01 |
| XGBoost | 56.67±2.49 | <u>68.60±1.53</u> | <u>59.81±0.36</u> | 56.77±0.90 |
| Join MLP | 54.00±9.09 | 64.26±1.70 | 56.36±1.33 | 52.14±3.01 |
| Join XGBoost | 56.67±2.49 | 68.60±1.53 | 59.81±0.36 | 56.77±0.90 |
| GCN | 75.33±8.38 | 63.09±1.45 | 56.94±0.72 | 50.91±1.83 |
| GIN | 76.00±8.64 | 64.73±2.65 | 56.53±2.12 | 45.11±5.77 |
| GraphSage | 80.00±5.89 | 66.14±1.30 | 59.29±1.79 | 55.13±0.46 |
| GAT | 67.33±3.40 | 64.85±1.37 | 58.35±1.55 | <u>59.29±0.41</u> |
| HGCN | <u>87.33±0.94</u> | 64.21±1.30 | 55.36±0.90 | 50.96±2.87 |
| HGT | **95.33±1.89** | **70.88±4.40** | **61.80±2.47** | **62.97±2.41** |

Table 24: Basic Information of tasks on rdb2-ncaa, # Class means the number of classes, StdErr means standard error.

| Table | Target | Type | # Class | StdErr |
|---|---|---|---|---|
| compact_results | WinnerLocation | Clas. | 3 | N/A |
| detailed_results | WinnerLocation | Clas. | 3 | N/A |
| compact_results | WinnerScore | Reg. | N/A | 0.160 |
| compact_results | LoserScore | Reg. | N/A | 0.174 |
| detailed_results | WinnerScore | Reg. | N/A | 0.204 |
| detailed_results | LoserScore | Reg. | N/A | 0.181 |

Table 25: Experiment result details of classification tasks on dataset rdb2-ncaa. The best results are in bold and the second-best results are underlined.

| **Table** | compact_results | detailed_results |
|---|---|---|
| **Column** | WinnerLocation | WinnerLocation |
| MLP | 59.49±0.32 | 66.42±0.66 |
| XGBoost | <u>61.70±0.26</u> | <u>67.76±0.14</u> |
| Join MLP | 59.47±0.32 | 65.14±0.45 |
| Join XGBoost | **62.45±0.22** | **68.59±0.18** |
| GCN | 59.84±0.07 | 59.75±0.49 |
| GIN | 39.54±14.32 | 26.75±23.82 |
| GraphSage | 59.84±0.07 | 59.75±0.49 |
| GAT | 59.84±0.07 | 59.75±0.49 |
| HGCN | 59.85±0.06 | 59.75±0.49 |
| HGT | 59.84±0.07 | 59.75±0.49 |

Table 26: Experiment result details of regression tasks on dataset rdb2-ncaa. The best results are in bold and the second-best results are underlined.

| **Table** | compact_results | | detailed_results | |
|---|---|---|---|---|
| **Column** | WinnerScore | LoserScore | WinnerScore | LoserScore |
| MLP | 0.131±0.004 | 0.143±0.002 | 0.047±0.005 | 0.047±0.003 |
| XGBoost | <u>0.111±0.000</u> | <u>0.121±0.001</u> | <u>0.010±0.000</u> | <u>0.007±0.000</u> |
| Join MLP | 0.177±0.005 | 0.194±0.004 | 0.083±0.003 | 0.074±0.010 |
| Join XGBoost | **0.105±0.000** | **0.115±0.001** | **0.010±0.000** | **0.007±0.000** |
| GCN | 0.157±0.001 | 0.172±0.001 | 0.200±0.001 | 0.177±0.001 |
| GIN | 0.833±0.139 | 0.958±0.139 | 0.913±0.169 | 0.866±0.169 |
| GraphSage | 0.119±0.004 | 0.156±0.005 | 0.190±0.002 | 0.168±0.001 |
| GAT | 0.157±0.001 | 0.172±0.001 | 0.201±0.000 | 0.178±0.002 |
| HGCN | 0.172±0.005 | 0.185±0.005 | 0.226±0.006 | 0.205±0.007 |
| HGT | 0.152±0.004 | 0.170±0.002 | 0.195±0.003 | 0.174±0.001 |

Table 27: Basic Information of tasks on rdb2-bank, # Class means the number of classes, StdErr means standard error.

| Table | Target | Type | # Class | StdErr |
|---|---|---|---|---|
| loan | Status | Clas. | 4 | N/A |
| loan | Duration | Clas. | 5 | N/A |
| transaction | PaymentType | Clas. | 8 | N/A |
| card | Type | Clas. | 3 | N/A |
| order | PaymentType | Clas. | 4 | N/A |
| order | Amount | Reg. | N/A | 0.343 |
| loan | Amount | Reg. | N/A | 0.337 |
| loan | Payments | Reg. | N/A | 0.428 |
| transaction | Balance | Reg. | N/A | 0.177 |
| transaction | Amount | Reg. | N/A | 0.218 |

Table 28: Experiment result details of classification tasks on dataset rdb2-bank. The best results are in bold and the second-best results are underlined.

| **Table** | loan | | transaction | card | order |
|---|---|---|---|---|---|
| **Column** | Status | Duration | PaymentType | Type | PaymentType |
| MLP | 83.33±3.02 | **97.06±1.20** | 81.92±1.98 | 74.15±1.59 | 54.46±0.64 |
| XGBoost | **86.27±3.02** | 90.69±4.22 | 97.72±0.01 | 68.16±4.34 | 52.86±1.42 |
| Join MLP | 60.29±3.60 | 19.61±3.67 | 49.90±1.86 | 74.90±2.65 | 56.00±0.89 |
| Join XGBoost | 84.80±2.50 | 89.22±3.86 | **99.20±0.04** | **100.00±0.00** | 62.49±0.57 |
| GCN | 69.61±3.67 | 39.70±6.24 | 62.26±9.72 | 72.66±2.12 | 70.14±1.25 |
| GIN | 26.96±18.62 | 16.18±2.08 | 29.06±2.21 | 72.66±2.12 | 34.84±26.22 |
| GraphSage | 79.41±4.33 | 71.08±3.86 | 84.07±1.38 | 72.66±2.12 | 70.14±1.25 |
| GAT | 58.82±7.21 | 59.31±15.99 | 32.05±0.18 | 72.66±2.12 | 70.14±1.25 |
| HGCN | 78.43±6.04 | 85.78±5.93 | 86.10±0.39 | 87.26±2.12 | **80.09±1.58** |
| HGT | 75.00±3.60 | 69.12±1.20 | 34.17±2.61 | 81.65±8.48 | 74.52±1.30 |

Table 29: Experiment result details of regression tasks on dataset rdb2-bank. The best results are in bold and the second-best results are underlined.

| **Table** | order | loan | | transaction | |
|---|---|---|---|---|---|
| **Column** | Amount | Amount | Payments | Balance | Amount |
| MLP | 0.330±0.004 | 0.027±0.006 | 0.106±0.024 | 0.151±0.001 | 0.139±0.001 |
| XGBoost | 0.351±0.009 | **0.018±0.002** | **0.034±0.012** | 0.129±0.000 | 0.126±0.000 |
| Join MLP | 0.358±0.003 | 0.299±0.013 | 0.514±0.053 | 0.188±0.003 | 0.233±0.017 |
| Join XGBoost | **0.002±0.000** | 0.019±0.005 | 0.040±0.017 | **0.116±0.000** | **0.000±0.000** |
| GCN | 0.314±0.013 | 0.288±0.050 | 0.405±0.028 | 0.166±0.003 | 0.184±0.012 |
| GIN | 0.905±0.403 | 0.889±0.395 | 1.194±0.423 | 0.516±0.267 | 0.298±0.095 |
| GraphSage | 0.307±0.014 | 0.156±0.025 | 0.293±0.011 | 0.157±0.003 | 0.147±0.004 |
| GAT | 0.315±0.008 | 0.355±0.033 | 0.431±0.009 | 0.166±0.002 | 0.176±0.002 |
| HGCN | 0.329±0.015 | 0.222±0.013 | 0.311±0.016 | 0.160±0.001 | 0.193±0.011 |
| HGT | 0.297±0.012 | 0.174±0.027 | 0.188±0.034 | 0.162±0.000 | 0.202±0.001 |

Table 30: Basic Information of tasks on rdb2-imdb, # Class means the number of classes.

| Table | Target | Type | # Class |
|---|---|---|---|
| movies2actors | CastNum | Clas. | 4 |
| movies2directors | MovieGenre | Clas. | 9 |
| actor | ActorQuality | Clas. | 6 |
| director | DirectorQuality | Clas. | 6 |
| user2movie | Rating | Clas. | 5 |

Table 31: Experiment result details on dataset rdb2-imdb. The best results are in bold and the second-best results are underlined. In some cases, the input table donot have any feature, so we mark it as N/A.

| **Type** | **Classification** | | | | |
|---|---|---|---|---|---|
| **Table** | movies2actors | movies2directors | actor | director | user2movie |
| **Column** | CastNum | MovieGenre | ActorQuality | DirectorQuality | Rating |
| MLP | N/A | N/A | 41.60±0.64 | 51.97±0.43 | N/A |
| XGBoost | N/A | N/A | 41.90±0.22 | 51.51±0.57 | N/A |
| Join MLP | 47.32±0.20 | 30.28±2.02 | 41.60±0.64 | 51.97±0.43 | 35.01±0.26 |
| Join XGBoost | **55.43±0.27** | 41.79±1.72 | 41.90±0.22 | 51.51±0.57 | **41.17±0.22** |
| GCN | 50.14±1.40 | 31.48±0.80 | 68.96±7.42 | 70.45±3.77 | 35.63±0.11 |
| GIN | 36.51±8.22 | 20.13±8.07 | 65.63±17.81 | 71.06±16.83 | 29.00±4.23 |
| GraphSage | 49.42±0.27 | 35.27±2.91 | 80.68±1.87 | 68.79±4.77 | 35.31±0.22 |
| GAT | 47.71±0.28 | 29.55±1.31 | 66.76±0.21 | 56.97±6.10 | 34.95±0.05 |
| HGCN | 50.18±0.20 | 40.50±2.38 | **84.61±2.09** | **90.61±0.77** | 38.32±0.34 |
| HGT | 50.70±0.78 | **42.19±0.99** | 84.01±1.80 | 87.12±4.39 | 37.96±0.55 |

Table 32: Basic Information of tasks on rdb3-toxic, # Class means the number of classes, StdErr means standard error.

| Table | Target | Type | # Class | StdErr |
|---|---|---|---|---|
| bond_count | Count | Clas. | 4 | N/A |
| active | Activity | Clas. | 2 | N/A |
| bond | BondArg | Clas. | 4 | N/A |
| has_property | Arg2 | Clas. | 2 | N/A |
| atom | Arg4 | Reg. | N/A | 0.192 |
| max_charge | Max | Reg. | N/A | 0.280 |
| min_charge | Min | Reg. | N/A | 0.560 |

Table 33: Experiment result details of classification tasks on dataset rdb3-toxic. The best results are in bold and the second-best results are underlined. In some cases, the input table donot have any feature, so we mark it as N/A.

| **Table** | bond_count | active | bond | has_property |
|---|---|---|---|---|
| **Column** | Count | Activity | BondArg | Arg2 |
| MLP | N/A | N/A | N/A | 59.03±1.57 |
| XGBoost | N/A | N/A | N/A | 54.45±5.08 |
| Join MLP | 55.58±0.94 | 60.92±13.30 | 72.65±0.66 | 58.02±1.65 |
| Join XGBoost | 60.47±3.09 | 83.91±7.08 | 74.40±1.68 | 54.45±5.08 |
| GCN | 84.56±1.77 | 70.12±14.45 | 78.73±8.52 | 65.65±3.74 |
| GIN | 79.13±3.40 | 50.58±8.60 | 77.55±6.86 | 58.02±7.19 |
| GraphSage | **89.22±0.99** | 81.61±15.51 | 93.38±1.41 | 63.36±6.51 |
| GAT | 58.18±1.31 | **100.00±0.00** | 72.82±0.55 | 61.58±3.14 |
| HGCN | <u>88.07±1.04</u> | <u>98.85±1.63</u> | <u>95.53±0.75</u> | **72.26±3.20** |
| HGT | 86.24±0.50 | 98.85±1.63 | **98.68±0.20** | <u>69.98±2.52</u> |

Table 34: Experiment result details of regression tasks on dataset rdb3-toxic. The best results are in bold and the second-best results are underlined. In some cases, the input table donot have any feature, so we mark it as N/A.

| **Table** | atom | max_charge | min_charge |
|---|---|---|---|
| **Column** | Arg4 | Max | Min |
| MLP | 0.124±0.012 | N/A | N/A |
| XGBoost | <u>0.080±0.009</u> | N/A | N/A |
| Join MLP | 0.215±0.019 | 0.352±0.038 | 0.547±0.045 |
| Join XGBoost | **0.077±0.008** | 0.403±0.042 | 0.583±0.058 |
| GCN | 0.225±0.019 | <u>0.255±0.024</u> | 0.495±0.049 |
| GIN | 0.434±0.216 | 0.696±0.345 | 1.006±0.429 |
| GraphSage | 0.120±0.013 | 0.272±0.037 | <u>0.418±0.069</u> |
| GAT | 0.187±0.007 | **0.214±0.096** | 0.473±0.028 |
| HGCN | 0.173±0.015 | 0.405±0.018 | 0.602±0.061 |
| HGT | 0.091±0.010 | 0.260±0.036 | **0.353±0.090** |

Table 35: Basic Information of tasks on rdb3-lahman, # Class means the number of classes, StdErr means standard error.

| Table | Target | Type | # Class | StdErr |
|---|---|---|---|---|
| pitchingpost | Game | Clas. | 8 | N/A |
| teams | Rank | Clas. | 13 | N/A |
| fielding | Position | Clas. | 11 | N/A |
| awardsplayers | League | Clas. | 4 | N/A |
| managers | RankTeam | Reg. | N/A | 0.470 |
| fielding | Position | Reg. | N/A | 0.178 |
| appearances | GameAll | Reg. | N/A | 0.573 |
| players | Height | Reg. | N/A | 0.128 |

Table 36: Experiment result details of classification tasks on dataset rdb3-lahman. The best results are in bold and the second-best results are underlined.

| **Table** | pitchingpost | teams | fielding | awardsplayers |
|---|---|---|---|---|
| **Column** | Game | Rank | Position | League |
| MLP | 67.78±0.20 | 33.82±1.52 | 32.90±0.65 | 42.95±0.43 |
| XGBoost | **74.78±0.49** | **49.69±2.52** | **72.19±0.38** | 32.41±2.62 |
| Join MLP | 56.08±1.03 | 33.82±1.52 | 26.17±1.01 | 42.49±1.87 |
| Join XGBoost | 74.30±1.58 | 49.69±2.52 | 71.78±0.28 | **75.19±1.04** |
| GCN | 45.51±6.31 | 20.42±2.30 | 52.91±0.38 | 71.44±2.30 |
| GIN | 27.53±19.12 | 12.67±0.17 | 10.92±2.31 | 37.54±8.22 |
| GraphSage | 57.44±3.30 | 22.14±2.97 | 53.97±0.37 | 72.77±0.50 |
| GAT | 40.81±0.20 | 23.74±1.55 | 44.53±1.62 | 71.50±1.90 |
| HGCN | 57.76±1.36 | 20.05±2.56 | 51.73±1.40 | 71.39±1.75 |
| HGT | 50.04±0.69 | 17.47±3.36 | 53.08±0.36 | 73.11±1.26 |

Table 37: Experiment result details of regression tasks on dataset rdb3-lahman. The best results are in bold and the second-best results are underlined.

| **Table** | managers | fielding | appearances | players |
|---|---|---|---|---|
| **Column** | RankTeam | Position | GameAll | Height |
| MLP | 0.258±0.013 | 0.113±0.001 | 0.054±0.005 | 3.439±0.483 |
| XGBoost | **0.196±0.010** | 0.020±0.000 | **0.002±0.000** | 0.087±0.000 |
| Join MLP | 0.249±0.006 | 0.182±0.041 | 0.359±0.171 | 1.606±0.221 |
| Join XGBoost | 0.197±0.007 | **0.020±0.000** | 0.002±0.000 | **0.087±0.002** |
| GCN | 0.420±0.011 | 0.119±0.007 | 0.364±0.005 | 0.187±0.004 |
| GIN | 0.853±0.524 | 0.689±0.283 | 0.945±0.321 | 1.785±0.189 |
| GraphSage | 0.418±0.025 | 0.074±0.005 | 0.321±0.005 | 0.131±0.001 |
| GAT | 0.403±0.019 | 0.111±0.021 | 0.360±0.014 | 0.127±0.001 |
| HGCN | 0.474±0.034 | 0.399±0.041 | 0.512±0.021 | 1.292±0.310 |
| HGT | 0.413±0.020 | 0.141±0.003 | 0.379±0.003 | 0.109±0.000 |

Table 38: Basic Information of tasks on rdb3-govern, # Class means the number of classes.

| Table | Target | Type | # Class |
|---|---|---|---|
| individual_vote | Result | Clas. | 5 |
| vote | Result | Clas. | 2 |
| member | Region | Clas. | 22 |
| member | Term | Clas. | 7 |

Table 39: Experiment result details on dataset rdb3-govern. The best results are in bold and the second-best results are underlined.

| **Type** | **Classification** | | | |
|---|---|---|---|---|
| **Table** | individual_vote | vote | member | |
| **Column** | Result | Result | Region | Term |
| MLP | 51.82±0.02 | 97.82±0.27 | 18.10±2.56 | 58.06±1.36 |
| XGBoost | 56.81±0.11 | 70.93±1.09 | 22.08±2.44 | 56.51±2.77 |
| Join MLP | 51.80±0.02 | **97.82±0.36** | 14.35±2.98 | 69.98±4.60 |
| Join XGBoost | **61.19±0.09** | 70.93±1.09 | **48.56±2.98** | 64.24±1.43 |
| GCN | 51.71±0.19 | 72.26±3.22 | 13.25±1.43 | 83.00±1.90 |
| GIN | 51.71±0.19 | 71.88±2.85 | 6.18±1.74 | 32.23±4.13 |
| GraphSage | 52.99±0.99 | 73.77±3.03 | 19.87±2.35 | **86.76±2.48** |
| GAT | 51.71±0.19 | 71.88±2.85 | 12.81±1.13 | 85.21±3.26 |
| HGCN | 53.27±0.31 | 73.11±2.08 | 18.99±1.36 | 86.09±3.01 |
| HGT | 52.35±1.09 | 74.34±3.03 | 38.85±2.77 | 82.34±1.13 |