GFS: Graph-based Feature Synthesis for Prediction over Relational Database

Han Zhang
Shanghai Jiao Tong University
han.harry.zhang@gmail.com

Quan Gan
AWS Shanghai AI Laboratory
quagan@amazon.com

David Wipf
AWS Shanghai AI Laboratory
daviwipf@amazon.com

Weinan Zhang
Shanghai Jiao Tong University
wnzhang@sjtu.edu.cn

ABSTRACT

Relational databases are widely used in modern information systems, but traditional machine learning models are often tailored for single table settings, requiring extensive manual feature engineering to merge data from multiple tables. This process is not only labor-intensive but also destroys the inherent relational structure. We introduce Graph-based Feature Synthesis (GFS), a framework that formulates relational databases as heterogeneous graphs, preserving their relational structure and eliminating the need for manual feature engineering. GFS leverages single-table model biases to capture complex relationships within the data. Extensive experiments on four real-world datasets demonstrate that GFS consistently outperforms existing methods, achieving top rankings and superior average performance.

VLDB Workshop Reference Format:
Han Zhang, Quan Gan, David Wipf, and Weinan Zhang. GFS: Graph-based Feature Synthesis for Prediction over Relational Database. VLDB 2024 Workshop: Tabular Data Analysis Workshop (TaDA).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/zhhhhahahaha/GFS.

1 INTRODUCTION

Data mining involves extracting useful patterns from databases. *Column prediction*, where a model predicts values in a *target column* of a *target table*, is essential in many applications, such as click-through-rate prediction [4, 12, 13, 31], anomaly detection [16, 21, 27, 30], and frequent pattern mining [1, 14, 15].

Most previous works focus on single-table settings, necessitating the merging of multiple tables into one for feature engineering[24–26]. This process is labor-intensive, requires substantial domain expertise, and often destroys the inherent relational structure within the data, leading to significant information loss. Data scientists often spend 80% of their time on data integration and curation [3], highlighting the need for automated methods that preserve relational structure, reduce manual effort, and minimize information loss.

We propose Graph-based Feature Synthesis (GFS), a novel framework that formulates relational databases as heterogeneous graphs, preserving their structure and eliminating the need for manual feature engineering. GFS leverages the inductive biases from differentiable single-table models to capture intricate relationships within each table, while the graph learning process effectively learns the structural information present in the database.

Automated methods, such as DFS (Deep Feature Synthesis) [18], consolidate multiple tables into a single table using predefined rules. DFS employs depth-first search for feature aggregation but suffers from low expressiveness and sensitivity to traversal order. OneBM [20] improves upon DFS by enumerating traversal paths, reducing variance. R2N extends rule-based aggregation to LSTM [17], while ARDA [5] automates data augmentation and uses feature selection. AutoFeature [22] augments features from candidate tables using a reinforcement learning framework. However, these methods often lack open-source implementations.

Recent approaches like RDB2Graph [8] convert databases into graphs and apply Graph Neural Networks (GNNs), but they focus more on structural information and less on column interactions. GFS enhances these methods by incorporating residual connections, allowing the use of any differentiable single-table model as a node embedding function and prediction model. This design enables GFS to better capture column interaction patterns while still leveraging graph learning to capture structural information. Cvetkov et al. [7] propose a knowledge graph method that enhances target table features using related tables, but results show that DFS often performs best. ATJ-Net [2] constructs hypergraphs to fuse related tables but struggles with complex schemas.

The main contributions of this paper are:

- We propose GFS, a framework that integrates any differentiable single-table model as an embedding function and/or prediction head, leveraging existing model biases and benefit from future advances in this field.
- Relative to alternative representative paradigms in the mold of DFS and RDB2Graph, GFS offers targeted improvements, such as invariance to traversal order, greater expressiveness, and over-smoothing mitigation.
- Experiments on four real-world datasets demonstrate that GFS consistently ranks first or second with a basic single-table model,

Proceedings of the VLDB Endowment. ISSN 2150-8097.

highlighting its superiority and potential for future enhancements with more robust models.

2 PROBLEM DEFINITION

Relational Databases: A Relational Database $(\mathcal{D}, \mathcal{L})$ is comprised of a collection of tables $\mathcal{D} = \{X^1, X^2, \ldots, X^N\}$, where each X^n represents the n-th constituent table, and $\mathcal{L}$ stands for the set of relationships between tables. $X^n_{i,j}$ represents the entry value of row i and column j of table X^n. Each table has at most four types of columns: Primary Key, Foreign Key, Target Column and Attribute Column. The target column contains the values to be predicted. Primary and foreign keys define table relationships. All other columns with informational content are attribute columns. We use C_n and R_n to denote the number of attribute columns and rows in X^n, respectively. Proceeding further, there are three types of relationships between two tables, *forward*, *backward*, *no direct refer* as similar to [18].
Forward: A *forward* relationship exists when a foreign key in one table references the primary key in another table, creating a directional link from one row to another.
Backward: A *backward* relationship refers to the connection from a row in one table to all rows in another table that have a forward relationship to it.

For composite keys and many-to-many relationships, we can simplify the situation by creating new primary keys and junction tables. With the necessary terminology and notations defined, we now introduce the specific task addressed in this paper. Many data mining problems in a relational database D can be formulated as *column prediction*, which is our focus. The problem is defined as follows:

Definition 2.1 (Column Prediction Task). Given a relational database $(\mathcal{D}, \mathcal{L})$, predict the values in a target column $X^T_{:,\text{target}}$ of a *target table* $X^T \in D$ of interest using information available in the database.

3 THE GFS FRAMEWORK

We present the technical details of the GFS framework for the column prediction problem specified by Definition 2.1. We first describe the process of converting a relational database to a graph with learnable node embeddings. Then, we discuss the core steps of training and inference: message passing and label prediction. The overview of GFS is shown in Fig. 1,

3.1 Interpreting Relational Database as Graph

A relational database can be interpreted as a heterogeneous directed graph. Each row of a table is a node, and all rows within a table are nodes of the same type. A foreign key reference from $X^A_{u,i}$ to $X^B_{v,j}$ implies a directed edge from node u of type A to node v of type B, foreign key references within the same foreign key column are considered as edges of the same type. Reverse edges are added to make the graph a *heterogeneous undirected graph*, suitable for aggregating information from other tables to the target table.
Node (Row)'s Raw Features. The values of a row in attribute columns are regarded as node(row) raw features. Raw node features of row $X^n_{i,:}$ are denoted as

$$x = [X^n_{i,1}, X^n_{i,2}, \ldots, X^n_{i,C_n}] \ . \tag{1}$$

Node (Row) Embedding. The low-dimensional vector encoding node-wise information is defined as a node embedding. The node embedding of $X^n_{i,:}$ is $h(i,n) \in \mathbb{R}^d$, where n references the table X^n and i represents the corresponding intra-table row.

3.2 Attribute Column Encoder

Attribute columns are encoded into vectors for prediction. We support categorical, continuous, and date columns. Categorical values are mapped to vectors via a learnable look-up table. Continuous values are normalized and transformed using a linear layer. Dates are split into *year, month, day of month, day of week*, and treated as categorical.

Across data types, we encode each row's raw features into a sequence of real-valued vectors. We denote the whole transformation as E_θ. The raw features x of row $X^n_{i,:}$ are transformed as:

$$E_\theta(x) \in \mathbb{R}^{C_n \times d} \ , \tag{2}$$

where θ represents the learnable parameters.

3.3 Message Passing

We introduce message passing using the node corresponding to row i in table X^n as an example, with x denoting the row's raw feature. We define $F = \{X^{f_1}, \ldots, X^{f_{|F|}}\}$ as the set of tables X^n has forward relationship to, and $B = \{X^{b_1}, \ldots, X^{b_{|B|}}\}$ for backward relationships.
Node (Row) Embedding Function. The node embedding function transforms a sequence of real-valued vectors into a dense node embedding:

$$N_\phi : \mathbb{R}^{L \times d} \to \mathbb{R}^d, \tag{3}$$

L is different for different node types and ϕ represents trainable parameters. The function can be an MLP, FM [26], or FT-Transformer [11]. Nodes of the same type share the same embedding function.
Aggregation Function. For tables X^n has backward relationships to, we need to aggregate the set of node embeddings which relate into single vector. The aggregation function is defined as:

$$\text{Agg}_\psi : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}^d, \tag{4}$$

where ψ is the learnable parameter, and $\mathcal{P}(\mathbb{R}^d)$ is the power set of $\mathbb{R}^d$. We use an aggregation method similar to PNA [6], calculating the *mean, max, min* and scaling with different factors, then using an MLP to combine them into a single vector.
Message Passing Function. Node embeddings are initialized as $\vec{0} \in \mathbb{R}^d$. For each node, the embeddings are updated iteratively, the notation here is for the example node corresponding to row i in table X^n:

$$\begin{aligned} \text{Mes}(i,n,l) :=& N_\phi\big(\text{concat}[E_\theta(x), \tilde{h}^l(i,f_1), \ldots, \tilde{h}^l(i,f_{|F|}), \quad (5) \\ & \text{Agg}_{\psi_1}(M^l(i,b_1)), \ldots, \text{Agg}_{\psi_{|B|}}(M^l(i,b_{|B|}))]\big) \\ h^{l+1}(i,n) =& \text{Mes}(i,n,l) \ . \quad (6) \end{aligned}$$

Repeating this k times captures $(k-1)$-hop information, with search depth K being the number of iterations.

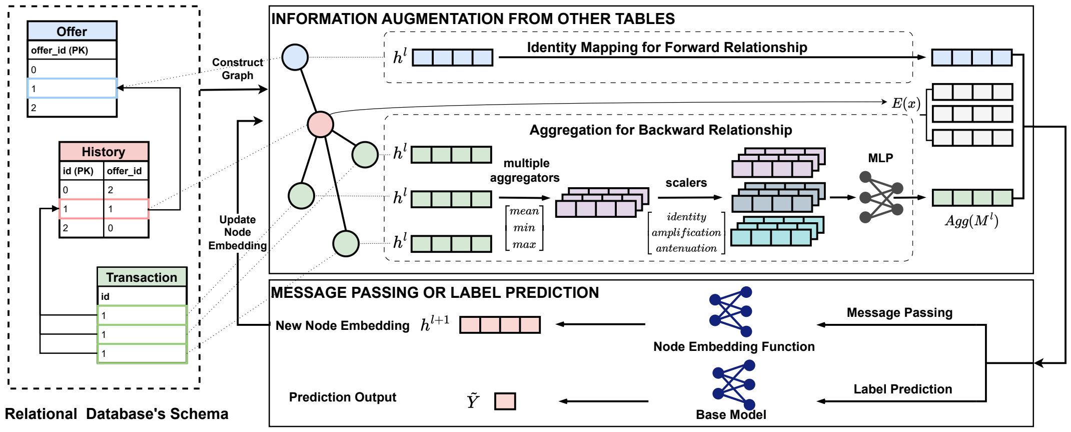
Offer
offer_id (PK)
History
id (PK)
offer_id
Transaction
id
Relational Database's Schema
Construct Graph
Update Node Embedding
INFORMATION AUGMENTATION FROM OTHER TABLES
Identity Mapping for Forward Relationship
Aggregation for Backward Relationship
multiple aggregators
mean min max
scalers
identity amplification antenuation
MLP
$E(x)$
$Agg(M^l)$
MESSAGE PASSING OR LABEL PREDICTION
New Node Embedding h^{l+1}
Node Embedding Function
Message Passing
Label Prediction
Prediction Output $\tilde{Y}$
Base Model

Figure 1: Overview of the GFS framework using the red-highlighted node as an example. It demonstrates node embedding updates and predictions for the target node. Some notations are abbreviated, and attribute columns are omitted for simplicity.

- $\tilde{h}^l(i,t) = h^l(i_t,t)$, where i_t is the row number in table X^t that X_i^n has a forward relationship to. $\tilde{h}^l(i,f_1),\ldots,\tilde{h}^l(i,f_{|F|})$ are l-step node embeddings of rows in $X^{f_1},\ldots,X^{f_{|F|}}$.
- $M^l(i,b_1),\ldots,M^l(i,b_{|B|})$ are l-step node embedding sets of rows in $X^{b_1},\ldots,X^{b_{|B|}}$ that X_i^n has backward relationships to.
- N_ϕ is the node embedding function for each node. The dimension of $E_\theta(x)$ is $C_n \times d$, $[\tilde{h}^l(i,f_1),\ldots,\tilde{h}^l(i,f_{|F|})]$ is $|F|\times d$, and $[\mathrm{Agg}_{\psi_1}(M^l(i,b_1)),\ldots,\mathrm{Agg}_{\psi_{|B|}}(M^l(i,b_{|B|}))]$ is $|B|\times d$. After concatenation, the input dimension is $(C_n+|F|+|B|)\times d$ and the output is a vector of dimension d.

3.4 Label Prediction

Base Model. The base model π_ω predicts labels using the final node embeddings. The model can be any single-table model, such as MLP, FM, or FT-Transformer. The prediction function for a node corresponding to row i in table X^n is:

$$\begin{aligned}\mathrm{Pred}(i,n,l) :=& \pi_\omega\big(\mathrm{concat}[E_\theta(x),\tilde{h}^l(i,f_1),\ldots,\tilde{h}^l(i,f_{|F|}), \\ & \mathrm{Agg}_{\psi_1}(M^l(i,b_1)),\ldots,\mathrm{Agg}_{\psi_{|B|}}(M^l(i,b_{|B|}))]\big) \quad (7)\\ \tilde{Y}_i^n =& \mathrm{Pred}(i,n,l)\ . \quad (8)\end{aligned}$$

where $\tilde{Y}_i^n$ represents the output prediction vector of GFS by the example node.

4 COMPARATIVE ANALYSIS

DFS's Sensitivity to Traversal Order. DFS-based methods are strong candidates for column prediction tasks on relational databases. However, DFS's output is not invariant to traversal order, leading to different results and potential instability. This traversal order is the sequence in which DFS traverses different tables using depth-first search. We detail this issue in Section 5.1 of [29], providing the pseudo-code of DFS, formal proof, and a counterexample. We also demonstrate how this sensitivity can cause significant performance degradation in certain cases using synthetic datasets, and highlight the robustness of GFS. Additionally, we present a real-world example with the same schema as the synthetic datasets, showing that this issue can occur in real-world scenarios.

GFS generalizes DFS. We show that GFS generalizes DFS, meaning that output of a certain parametrization of GFS will be a superset of DFS's output, details are in Section 5.2 of [29].

5 EXPERIMENTS

5.1 Experimental Setup for Real-World Datasets

5.1.1 Datasets Description and Evaluation Metrics. We evaluate our model on four real-world datasets: *Acquire-valued-shoppers* [10], *KDD2015*[19], *Outbrain* [23], and *Diginetica*[9], covering domains like customer retention, click-through rate prediction, recommendation, and fraud detection. We use the area under ROC curve (*AUC*) for binary classification tasks.

5.1.2 Baselines and GFS Setting. We compare our method against two categories of approaches:

Offline Method. These methods first consolidate data into a single table, then apply single-table models. Besides DFS, we compare:

- *Target-table Only (TT)*: Only the target table is used for prediction.
- *Simple join (SJ)*: Tables are recursively joined by appending columns without aggregation.

For offline methods, we evaluate DeepFM[12] and FT-Transformer (FT-T)[11] as base models.

GNN Methods. We compare with RDB2Graph, which embeds rows into node vectors and applies GNNs, and ATJ-Net, which constructs hypergraphs for GNNs. We use RGCN, Relational GAT, and HGT as backbones for RDB2Graph.

Other similar methods like OneBM, R2N, ARDA, and AutoFeature are either proprietary or lack released source code, so we do not compare with them.

GFS Setting. GFS can use any differentiable model for single table settings as node embedding functions and base models. We found that DeepFM performs well for our datasets, making more complex models unnecessary. FT-T can be helpful for other datasets but has licensing restrictions that disallow publishing with them.

5.1.3 Parameter Settings. We use Weight & Bias [28] for hyperparameter optimization, conducting 50 trials to find the best combination of learning rate, weight decay, and dropout probability based on validation set performance. Models were rerun five times to reduce randomness.

Table 1: AUC results for real-world datasets. TT results are not applicable on datasets where the target table lacks attribute columns. * indicates significant improvements over baselines with p<0.05, ** indicates p<0.005.

Model	Simple Schema		Complex Schema		Rank
	AVS	KDD15	Outbrain	Diginetica	
Max Search Depth (hops) K	2	2	4	3	-
TT + DeepFM	0.6737	-	-	-	10.0
TT + FT-T	0.6720	-	-	-	11.0
SJ + DeepFM	0.6902	0.6297	0.7223	0.6278	8.0
SJ + FT-T	0.6894	0.6061	0.7188	0.6100	9.0
RDB2Graph + RGCN	0.6956	0.8557	0.7420	0.7420	6.0
RDB2Graph + GAT	0.6978	0.8629	0.7440	0.7565	4.0
RDB2Graph + HGT	0.6957	0.8719	0.7549	0.8070	3.0
DFS + DeepFM	0.6974	0.8717	0.7337	0.7963	4.3
DFS + FT-T	0.6916	0.8626	0.7303	0.8024	5.5
ATJ-Net	0.6968	**0.8812***	0.7302	0.7999	4.0
GFS (Ours)	**0.7001****	0.8781	**0.7558***	**0.8106***	**1.3**

Table 2: Training time (hours) for different models on g4dn.metal. Results are reported below the maximum search depth due to OOM issues at larger depths for some methods.

Model	AVS	KDD15	Outbrain	Diginetica
Search Depth(hops) K	2	2	3	2
RDB2Graph + RGCN	0.25	0.56	3.50	2.07
RDB2Graph + GAT	0.33	0.51	5.00	1.75
RDB2Graph + HGT	1.17	2.45	8.31	4.97
ATJ-Net	0.34	0.09	12.79	0.60
GFS (Ours)	0.60	0.39	4.05	1.84

5.2 Performance & Training Time Comparison

The results for four real-world datasets are shown in Table 1. We report optimal results for each method, ensuring search depth does not exceed the maximum allowed. The maximum search depth is set to fully traverse all relationships while maintaining reasonable training times and GPU memory usage. Online sampling models like GFS are compared with other GNN models, while offline sampling baselines like DFS and SJ cannot be fairly compared as they aggregate information prior to model training. Training time results are in Table 2. Key observations are as follow:

- GFS outperforms baselines on AVS, Outbrain, and Diginetica, and ranks second on KDD15, achieving the best average performance. Other models struggle on certain datasets; HGT performs well on Outbrain but poorly on AVS. DFS, RGCN, and GAT underperform across all datasets.
- When a lower search depth is sufficient to reach all tables and cover all PK-FK relationships (e.g., AVS and KDD15 with $K = 2$), ATJ-Net is a strong baseline with performance similar to GFS. However, for more complex schemas requiring higher search depths (e.g., Outbrain and Diginetica with $K = 4$ and $K = 3$), GFS significantly outperforms ATJ-Net. This aligns with the ATJ-Net paper's findings, which note overfitting issues as search depth K increases beyond 2. While ATJ-Net suggests random architecture search to mitigate this, our findings hold even after thorough architecture and hyperparameter tuning.
- Despite HGT's strong performance, GFS combined with DeepFM is more efficient, requiring only at most half training time compared to HGT.

Table 3: AUC results of GFS + DeepFM with different node embedding functions (N_ϕ).

N_ϕ	AVS	KDD15	Outbrain	Diginetica
MLP	0.6949	0.8660	0.7540	0.8095
DeepFM	**0.7001**	**0.8781**	**0.7558**	**0.8106**

5.3 Ablation Study

We analyze the impact of node embedding function in GFS. While choosing an advanced single table model as base model π_ω for final prediction is straightforward, using a differentiable single table model like DeepFM for row embedding function N_ϕ may seem redundant at first glance. However, this is a key distinction of GFS compared to prior GNNs which only use MLP. This approach incorporates inductive bias to extract column interaction patterns into the node embedding. To demonstrate the importance of this design, we use DeepFM as the base model but change N_ϕ from DeepFM to MLP. The results in Table 3 highlight the significance of the node embedding function. Key observations are as follows:

- In AVS and KDD15 datasets, replacing DeepFM with MLP for N_ϕ significantly drops performance, indicating the importance of feature interactions from DeepFM for non-target tables.
- In Outbrain and Diginetica datasets, the row embedding function has less impact on performance, likely because target column prediction relies more on relationships between rows in different tables rather than row feature interactions. This is supported by similar performance between GFS and HGT, where MLP is used for node embedding vectors.

6 CONCLUSION

In this paper, we introduce GFS, a novel framework for general *column prediction* tasks on relational databases. GFS is an embedding update and prediction framework that integrates any differentiable model designed for single table settings. It improves upon previous methods like DFS, RDB2Graph, and ATJ-Net, addressing their inherent issues. Comprehensive experiments on real-world datasets demonstrate that GFS outperforms baselines and exhibits superior efficiency compared to the most powerful GNN baseline, HGT. These results highlight GFS's potential as an effective and efficient solution for machine learning tasks on relational databases.

REFERENCES

[1] Rakesh Agrawal, Ramakrishnan Srikant, et al. 1994. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215. Santiago, Chile, 487–499.

[2] Jinze Bai, Jialin Wang, Zhao Li, Donghui Ding, Ji Zhang, and Jun Gao. 2021. ATJ-Net: Auto-Table-Join Network for Automatic Learning on Relational Databases. In *Proceedings of the Web Conference 2021*. 1540–1551.

[3] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 1335–1349. https://doi.org/10.1145/3318464.3389742

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[5] Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. 2020. ARDA: Automatic Relational Data Augmentation for Machine Learning. *Proc. VLDB Endow.* 13, 9 (may 2020), 1373–1387. https://doi.org/10.14778/3397230.3397235

[6] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. 2020. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems* 33 (2020), 13260–13271.

[7] Alexis Cvetkov-Iliev, Alexandre Allauzen, and Gaël Varoquaux. 2023. Relational data embeddings for feature enrichment with background information. *Machine Learning* 112, 2 (2023), 687–720.

[8] Milan Cvitkovic. 2020. Supervised learning on relational databases with graph neural networks. *arXiv preprint arXiv:2002.02046* (2020).

[9] diginetica 2016. https://competitions.codalab.org/competitions/11161

[10] DMDave, Todd B, and Will Cukierski. 2014. Acquire Valued Shoppers Challenge. https://kaggle.com/competitions/acquire-valued-shoppers-challenge

[11] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems* 34 (2021), 18932–18943.

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[13] Wei Guo, Can Zhang, Zhicheng He, Jiarui Qin, Huifeng Guo, Bo Chen, Ruiming Tang, Xiuqiang He, and Rui Zhang. 2022. Miss: Multi-interest self-supervised learning framework for click-through rate prediction. In *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 727–740.

[14] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. 2007. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery* 15, 1 (2007), 55–86.

[15] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. *ACM sigmod record* 29, 2 (2000), 1–12.

[16] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. 2022. Adbench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems* 35 (2022), 32142–32159.

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[18] James Max Kanter and Kalyan Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 1–10.

[19] kddcup 2015. https://www.biendata.xyz/competition/kddcup2015/

[20] Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. 2017. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327* (2017).

[21] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[22] Jiabin Liu, Chengliang Chai, Yuyu Luo, Yin Lou, Jianhua Feng, and Nan Tang. 2022. Feature augmentation with reinforcement learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 3360–3372.

[23] mjkistler, Ran Locar, Ronny Lempel, RoySassonOB, Rwagner, and Will Cukierski. 2016. Outbrain Click Prediction. https://kaggle.com/competitions/outbrain-click-prediction

[24] Kwanghyun Park, Karla Saur, Dalitso Banda, Rathijit Sen, Matteo Interlandi, and Konstantinos Karanasos. 2022. End-to-end optimization of machine learning prediction queries. In *Proceedings of the 2022 International Conference on Management of Data*. 587–601.

[25] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-based neural networks for user response prediction over multi-field categorical data. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 1–35.

[26] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[27] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. 2022. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*. PMLR, 21076–21089.

[28] Weight & Bias 2023. https://wandb.ai/site

[29] Han Zhang, Quan Gan, David Wipf, and Weinan Zhang. 2023. GFS: Graph-based Feature Synthesis for Prediction over Relational Databases. arXiv:2312.02037 [cs.LG] https://arxiv.org/abs/2312.02037

[30] Yue Zhao and Maciej K Hryniewicki. 2018. Xgbod: improving supervised outlier detection with unsupervised representation learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[31] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.