Human-in-the-Loop Feature Discovery for Tabular Data

Ionescu, Andra; Mouw, Zeger; Aivaloglou, Efthimia; Hai, Rihan; Katsifodimos, Asterios

DOI
10.1145/3627673.3679211

Publication date
2024

Document Version
Final published version

Published in
CIKM '24

Human-in-the-Loop Feature Discovery for Tabular Data

Andra Ionescu
Delft University of Technology
Delft, The Netherlands
a.ionescu-3@tudelft.nl

Zeger Mouw
Delft University of Technology
Delft, The Netherlands
z.f.mouw@student.tudelft.nl

Efthimia Aivaloglou
Delft University of Technology
Delft, The Netherlands
e.aivaloglou@tudelft.nl

Rihan Hai
Delft University of Technology
Delft, The Netherlands
r.hai@tudelft.nl

Asterios Katsifodimos
Delft University of Technology
Delft, The Netherlands
a.katsifodimos@tudelft.nl

Abstract

In recent years, researchers have developed several methods to automate discovering datasets and augmenting features for training Machine Learning (ML) models. Together with feature selection, these efforts have paved the way towards what is termed the *feature discovery* process. Data scientists and engineers use automated feature discovery over tabular datasets to add new features from different sources and enrich training data. By surveying data practitioners, we have observed that automated feature discovery approaches do not allow data scientists to use their domain knowledge during the feature discovery process. In addition, automated feature discovery methods can leak private features or introduce biased ones.

In this paper, we introduce the first user-driven human-in-the-loop feature discovery method called `HILAutoFeat`. We demonstrate the capabilities of `HILAutoFeat`, which effectively combines automated feature discovery with user-driven insights. Our demonstration is centred around two scenarios: (*i*) an automated feature discovery scenario – `HILAutoFeat` acts as a steward in a large data lake where the user is unaware of the quality and relevance of the data, and (*ii*) a scenario where `HILAutoFeat` and the user work together – the user drives the feature discovery process by adding his domain and business knowledge, while `HILAutoFeat` performs the intensive computations.

CCS Concepts

• **Information systems** → *Data analytics.*

Keywords

Human-in-the-Loop; Feature Discovery; Tabular Data; Data Science; AutoML

ACM Reference Format:
Andra Ionescu, Zeger Mouw, Efthimia Aivaloglou, Rihan Hai, and Asterios Katsifodimos. 2024. Human-in-the-Loop Feature Discovery for Tabular Data. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3627673.3679211

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679211

1 Introduction

The long-standing presumption that the training data for a Machine Learning (ML) model is a single table does not hold true. In practice, essential predictive features often reside across multiple database tables or files, which could be part of an extensive open data repository or a data lake [7, 13, 14]. Currently, there is significant ongoing research dedicated to developing methods that automate the discovery and augmentation of tabular features for ML model training [3, 5, 12, 15]. This process, named *feature discovery*, builds upon the exploration and integration steps from dataset discovery [1, 2, 5, 11] and relies on feature selection strategies to select only the most relevant features for a given ML task.

In data lakes lacking primary-key/foreign-key constraints, employing dataset discovery [6, 10] is an essential first step for feature discovery, which reveals table relationships [3, 9] such as joinability [1, 11] or unionability [6, 14]. However, dataset discovery approaches often produce false positives [10], leading to joins that yield irrelevant tables filled with unrelated data. The issue is exacerbated when two datasets are joinable through multiple columns, a scenario where state-of-the-art feature discovery approaches typically fall short [3, 12].

The automated systems for feature discovery for ML either focus on strategies for improving the correlation metrics [5], maximizing relevance while minimizing redundancy [9], or integrating the ML model directly into the augmentation process to ensure feature compatibility [3, 12]. However, our recent user study [8] has revealed two critical issues with fully automated approaches. While data scientists find the automated feature discovery methods to be very useful for finding relevant features to train their ML models, they also lose control over which features are included in the training data for a given model. This issue is two-fold. First, fully automated feature discovery methods do not leverage the user's domain expertise, which can be pivotal in discovering important predictive features. Second, the fully automated methods can incorporate features that should not be part of the training data due to government regulations and company policies (e.g., privacy, bias).

Example: *Take as an example a dataset for predicting if an employee is suitable for promotion. In this dataset, the ML model has access to features such as education, years of experience, technical skills, and soft skills. Augmenting this dataset with personal information about the employees, such as age, gender, and nationality, and given the high amount of male employees the company already has, an algorithm trained with the gender feature can be biased to generate a*

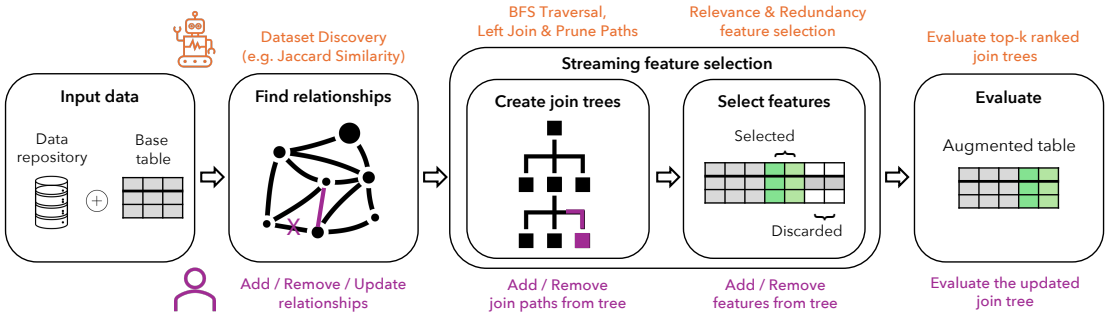

Figure 1: `HILAutoFeat` pipeline: automatic workflow and user-driven workflow.

favourable decision for male employees. Here, human input is crucial in determining the correct set of relevant and related features for augmentation.

Departing from the black-box automated approaches, we have extended our automated tool `AutoFeat` [9] to incorporate user feedback and involvement during the feature discovery process. With this human-in-the-loop approach, named `HILAutoFeat`, we address the reported issues from our user study. `HILAutoFeat` leverages the strengths of automated feature discovery methods while providing a platform for data scientists to use their domain expertise and business knowledge. `HILAutoFeat` allows users to control essential steps: users can filter the discovered relationships and the join paths and adjust the selected features while observing the effects of these updates over the augmented dataset in real-time. To the best of our knowledge, `HILAutoFeat` is the first user-driven, semi-automated feature discovery tool that dynamically adjusts to user feedback. Specifically designed for data scientists and analysts across various fields, users reported that this tool provides a more efficient augmentation process and yields effective results through the added benefit of modifying the data (i.e., relationships, join trees, features) at any given point.

Demonstration. The demonstration attendants will be able to: (*i*) navigate through a data repository while `HILAutoFeat` filters and suggests suitable tables and features for augmentation based on relevance and potential to improve the ML model performance, (*ii*) evaluate and select features that best align with their specific modelling objectives, and (*iii*) iteratively improve the feature discovery process and mitigate possible privacy and bias issues by leveraging their domain knowledge.

System Availability. The automatic feature discovery approach `AutoFeat` can be found at https://github.com/delftdata/autofeat, while the human-in-the-loop approach `HILAutoFeat` is available at https://github.com/delftdata/hci-auto-feat. Additionally, a video demonstration is accessible at https://youtu.be/tjXxCb2C3hU.

2 System Overview

We have developed a user-driven human-in-the-loop feature discovery approach, `HILAutoFeat`, which extends our automated feature discovery system, `AutoFeat` [9]. The primary objective of feature discovery is to enhance a base table by adding new features that significantly increase the predictive accuracy of a target ML model. `HILAutoFeat` streamlines the process of selecting and integrating relevant tables from a dataset collection into the base table, based on the user's input, whose domain expertise can potentially change the outcome of the augmentation process. Additionally, `HILAutoFeat` employs heuristic-based feature selection strategies to eliminate redundant or irrelevant features from this augmented table. By doing so, `HILAutoFeat` notably enhances the efficiency and accuracy of subsequent ML operations.

Figure 1 illustrates the automated feature discovery process and the user interactions which are available at every stage in the pipeline. Our human-in-the-loop feature discovery approach provides the following functionalities to the user: (1) refining dataset relationship (Section 2.1), (2) manipulating join trees (Section 2.2), and (3) refining feature sets (Section 2.3). We also discuss how `HILAutoFeat` maintains the high efficiency of `AutoFeat` through various scalability enhancements in Section 2.4.

2.1 Refine Dataset Relationships

In a data lake with hundreds or thousands of tables, the number of relationships between these tables for a fully connected graph is $n * (n-1)/2$, where n is the number of vertices. For a multi-graph, the number can be much higher. `HILAutoFeat` maps the relationships between tables using similarity scores generated by a dataset discovery algorithm. Currently, we support our schema matching tool suite Valentine [10]. With the automated feature discovery process, spurious relations are eliminated. However, the remaining relationships are not guaranteed to be relevant to augment the base table. `HILAutoFeat` enables the users to adjust the relationships discovered by the automatic process. The users possess domain knowledge [8] and can immediately recognise which tables are beneficial for the augmentation.

By default, `HILAutoFeat` displays a graph with the strongest relationships between tables, as illustrated in Figure 2. Then, it enables the user to adjust the similarity threshold, visualise all relationships, and refine them (i.e., delete, update, or add an additional one). These functionalities enhance user control over the process and allow for a more tailored and precise dataset construction, accommodating specific analytical needs and objectives.

2.2 Manipulate Join Trees

After establishing the relationships, the next step in the pipeline is streaming feature selection. In this feature selection approach, the features follow a streaming process: a new batch of features arrives with every new join. The automatic feature discovery approach computes two steps in the same streaming feature selection iteration: creates join trees and selects the features. For `HILAutoFeat`,

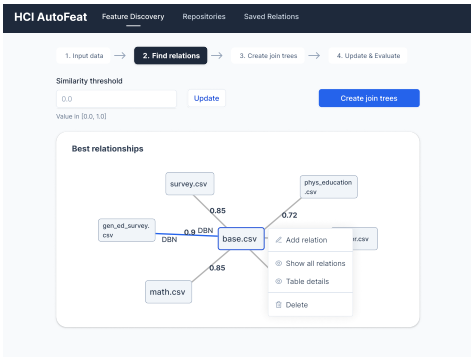

Figure 2: A snippet from the HILAutoFeat GUI showing the best relationships between tables and the user action of clicking on a table, which shows the menu to edit or view it.

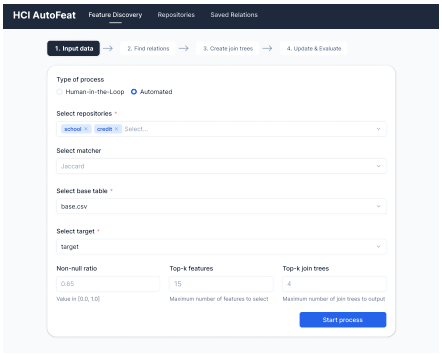

Figure 3: A snippet from the HILAutoFeat GUI showing the input required to start the automated process.

we deconstruct this process such that the user can update the join trees and the subsequent feature sets and actively observe the impact of each join and feature on the performance of the ML model.

The join trees are an early representation of the augmented table, as each node in the tree represents a table with the associated join column. We use Breadth First Search (BFS) traversal to navigate through the graph of table relationships. With BFS, we first join directly connected datasets and then proceed to join datasets that are farther away. This order of joining is crucial because it allows us to prioritise the most relevant datasets in the early stages of the traversal. Through the BFS traversal, we form join paths of varying lengths by sequentially *left*-joining the tables. The choice of a *left* join is strategic, aiming to preserve the original number of tuples and, more importantly, to maintain the number and distribution of classes in the target variable. We discuss other traversal and join strategy options in AutoFeat [9].

We refine the join trees by pruning any spurious paths. We employ similarity-based pruning – where the join column with the highest similarity score is selected, and data quality-based pruning – which involves discarding join paths that surpass a pre-defined threshold for a non-null value ratio. Each join tree is then ranked by a linear function derived from two distinct feature selection methods measuring relevance and redundancy. AutoFeat [9] provides an in-depth analysis of our feature selection methods.

With HILAutoFeat, we open the black box automatic approach and empower users by giving them control over and insights into the process. Users can actively influence the augmentation process by adding or removing paths from a join tree, acting as an external knowledge source for the algorithm. For example, in Figure 4, the user applies his domain knowledge and removes a path (i.e., table) from the join tree, which had the potential to bias the algorithm. Additionally, HILAutoFeat is enhanced with an explainability function, also illustrated in Figure 4. At each step of the algorithm, HILAutoFeat provides users with a comprehensive understanding of the process. This transparent approach fosters a deeper understanding and trust, empowering users to make informed decisions while fine-tuning their data.

2.3 Refining Feature Sets

At the heart of HILAutoFeat lies the balance between relevance and redundancy, which is crucial to its effectiveness. In the context of ML, relevance is divided into two categories: strong relevance, where removing a key feature negatively affects the optimal set of features, and weak relevance, where less important features impact the output upon removal. Redundant features, on the other hand, are those that offer no new information and can be interpreted as a duplication of relevant features. For HILAutoFeat, we apply the Spearman correlation to assess feature importance, while the MRMR metric is used to identify and manage redundancy, ensuring the model operates efficiently and effectively [9].

The most granular operations the user can make are at the feature level. They can view the collection of selected and discarded features associated with a join tree and make updates by either adding or removing features. At this stage, users can leverage their domain expertise or business knowledge to prioritize more critical features. By modifying the feature set, the users not only alter the augmented dataset but also significantly impact the accuracy of the ML model. The refinement of the feature set introduces a higher degree of customization and precision to the feature discovery process. It allows for a dynamic interplay between automated feature selection and human judgement, ensuring that the final dataset is rich in relevant features and aligned with specific analytical goals and business objectives.

2.4 Scalability

In our approach, we rely on external dataset discovery techniques for the initial computation of table relationships, a phase that constitutes the most extensive duration within the process. Nevertheless, it is noteworthy that dataset discovery methods can be efficiently scaled to accommodate thousands of tables, enhancing both accuracy and computational speed, as shown in JOSIE [16].

When running HILAutoFeat with hundreds or thousands of tables, our methodology incorporates pruning strategies to eliminate irrelevant tables, such as similarity-based pruning and data quality-based pruning. Consequently, the number of tables in a join tree will not approach the thousand mark, given that most tables will be irrelevant to the base table targeted for augmentation. Furthermore, we have implemented a relationship-caching method, eliminating the need to recompute these connections for future usage.

In addition, our experimental evaluations with the automated approach, AutoFeat [9], reveal that our strategy offers a superior performance speed compared to existing state-of-the-art automated dataset augmentation and feature discovery techniques. This efficiency is maintained despite the integration of human interaction

within the process, as user involvement does not affect the computation time for constructing the join trees.

HILAutoFeat uses hyper-parameters to ensure that the curated set of features remains manageable for the user. Accordingly, a maximum of κ features is chosen from each table. In scenarios with large data repositories, HILAutoFeat relies on evaluating the relevance and redundancy metrics for features, thus ensuring the construction of an optimal feature set.

3 Demonstration Scenario

We will demonstrate our human-in-the-loop approach for feature discovery within a web application. In our demonstration scenario, a user aims to augment a dataset by adding features to enhance the accuracy of a tree-based ML model. The default ML model is LightGBM, which is a part of the AutoGluon AutoML framework [4]. Users, however, have the flexibility to select their desired model from the range of models supported by AutoGluon. The user starts with a base table that includes a target variable for binary classification and promising features for the ML model. Additionally, the user has access to a data repository containing multiple tables, either relevant or irrelevant, for augmentation.

Datasets. For our demonstration, we assemble a collection of datasets commonly used in evaluating state-of-the-art data augmentation techniques [3, 9, 12]. The dataset repository includes eight datasets, each comprising between five and 40 joinable tables and featuring a total of 20 to 420 attributes. All the datasets are used for binary classification problems.

3.1 Scenario #1: Distil Information

In the first demonstration scenario, we showcase the capability of HILAutoFeat to distil information. Given a large collection of tables, manually performing feature discovery implies inspecting the tables and selecting the most relevant features for the augmentation. This manual process takes a tremendous amount of effort and time to complete. In this scenario, we assume that the user is unaware of the quality and relevance of the information in the data repository. Thus, HILAutoFeat helps users by automatically filtering out irrelevant tables. The feature discovery process is entirely automatic, requiring no user intervention.

In Figure 3, we present a snippet of the user interface. The user initiates the process by specifying the input: the desired data repositories, the base table to augment and its corresponding target variable and the hyper-parameters to adjust the method. From here on, HILAutoFeat autonomously performs the steps illustrated in Figure 1: (i) finds relationships between the tables from the dataset repository, (ii) identifies suitable features for the base table and creates join trees, and (iii) returns top-κ join trees. HILAutoFeat helps the user to find relevant information and returns top-k options to the user. Now, the users have an overview of the relevant tables and features and can choose the most suitable join tree for their subsequent processes.

3.2 Scenario #2: User-Driven Feature Discovery

In this scenario, the dataset repository is smaller, and we assume the user has knowledge about the datasets. In this semi-automatic process, HILAutoFeat and the users work together. While the users

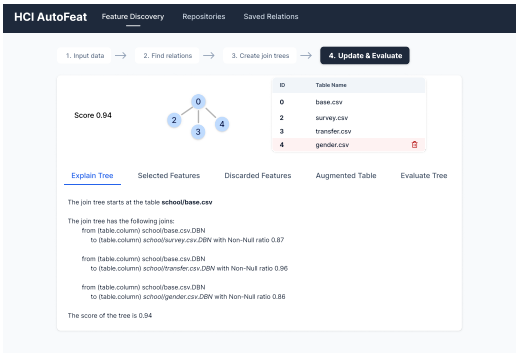

Figure 4: A snippet from the HILAutoFeat GUI showing a join tree, an explanation, and the user action of deleting a path.

rely on HILAutoFeat to perform automatic time-consuming computations, they can apply domain and business knowledge to enhance the feature discovery process and create the best-performing augmented dataset. The users start by selecting a dataset repository and the candidate tables for augmentation, similar to the automatic process from Figure 3.

3.2.1 Refine Relationships. HILAutoFeat works with multiple dataset discovery algorithms, as described in Section 2.1. For this demonstration scenario, we use Jaccard similarity for ease of understanding the process. Following the initialisation phase, users have the opportunity to examine and modify each step of the pipeline. This process begins with determining the relationships between the tables in the data repository. Subsequently, users can visualize and refine these relationships as illustrated in Figure 2. Users can add or remove a relation or alter the similarity score of an existing relationship, ensuring that only the most relevant tables are considered for augmentation.

3.2.2 Manipulate Join Trees. Once we establish the relationships between tables, the next step in the workflow is computing join trees. The user can visualize and conduct a detailed examination of these trees. In this detailed view, they can observe the ranking of each join tree and examine all the join paths that constitute the tree, as shown in Figure 4. Users can actively influence the augmentation process by adding or removing paths. They can also verify in real time the effect of each refinement on the augmented dataset by evaluating the tree.

3.2.3 Refine Features. The most granular operations the user can make are at the feature level. They can view the selected and discarded feature sets associated with a join tree and make updates by adding or removing features. The user can remove highly redundant features and re-evaluate the dataset by retraining the ML model with the revised feature set.

With these scenarios, we demonstrate how incorporating the user's domain knowledge is instrumental in shaping a significantly more robust and tailored training dataset, thereby enhancing the overall effectiveness of our approach.

Acknowledgments

We would like to extend our gratitude to Alexandra Pituru for her invaluable support and guidance in designing our user interface.

References

[1] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 709–720. https://doi.org/10.1109/ICDE48307.2020.00067

[2] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A Data Discovery System. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. 1001–1012. https://doi.org/10.1109/ICDE.2018.00094

[3] 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

[4] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505* (2020).

[5] Mahdi Esmailoghli, Jorge-Arnulfo Quiané-Ruiz, and Ziawasch Abedjan. 2021. COCOA: COrrelation COefficient-Aware Data Augmentation. In *Proceedings of the 24th International Conference on Extending Database Technology (EDBT)*. 331–336. https://doi.org/10.5441/002/edbt.2021.30

[6] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023. Table Discovery in Data Lakes: State-of-the-art and Future Directions. In *Companion of the 2023 International Conference on Management of Data* (Seattle, WA, USA) *(SIGMOD '23)*. Association for Computing Machinery, New York, NY, USA, 69–75. https://doi.org/10.1145/3555041.3589409

[7] Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. 2023. Data Lakes: A Survey of Functions and Systems. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (Dec 2023), 12571–12590. https://doi.org/10.1109/TKDE.2023.3270101

[8] Andra Ionescu, Zeger Mouw, Efthimia Aivaloglou, and Asterios Katsifodimos. 2024. Key Insights from a Feature Discovery User Study. In *Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics* (Santiago, AA, Chile) *(HILDA 24)*. Association for Computing Machinery, New York, NY, USA, 1–5. https://doi.org/10.1145/3665939.3665961

[9] Andra Ionescu, Kiril Vasilev, Florena Buse, Rihan Hai, and Asterios Katsifodimos. 2024. AutoFeat: Transitive Feature Discovery over Join Paths. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 1861–1873. https://doi.org/10.1109/ICDE60146.2024.00150

[10] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 468–479. https://doi.org/10.1109/ICDE51399.2021.00047

[11] Arun Kumar, Jeffrey Naughton, Jignesh M. Patel, and Xiaojin Zhu. 2016. To Join or Not to Join? Thinking Twice about Joins before Feature Selection. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 19–34. https://doi.org/10.1145/2882903.2882952

[12] 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)*. 3360–3372. https://doi.org/10.1109/ICDE53745.2022.00317

[13] Fatemeh Nargesian, Abolfazl Asudeh, and H. V. Jagadish. 2022. Responsible Data Integration: Next-generation Challenges. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 2458–2464. https://doi.org/10.1145/3514221.3522567

[14] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table union search on open data. *Proc. VLDB Endow.* 11, 7 (mar 2018), 813–825. https://doi.org/10.14778/3192965.3192973

[15] Zixuan Zhao and Raul Castro Fernandez. 2022. Leva: Boosting Machine Learning Performance with Relational Embedding Data Augmentation. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 1504–1517. https://doi.org/10.1145/3514221.3517891

[16] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 847–864. https://doi.org/10.1145/3299869.3300065