**Automatic Data Repair: Are We Ready to Deploy?**

# ABSTRACT

Wei Ni∗‡1, Xiaoye Miao∗2, Xiangyu Zhao‡3, Yangyang Wu#4, Jianwei Yin∗#†5

∗Center for Data Science, Zhejiang University, Hangzhou, China

‡School of Data Science, City University of Hong Kong, Hong Kong, China

#School of Software Technology, Zhejiang University, Ningbo, China

†College of Computer Science, Zhejiang University, Hangzhou, China

{1wei.ni, 2miaoxy, 4zjuwuyy}@zju.edu.cn [3xy.zhao@cityu.edu.hk](mailto:3xy.zhao@cityu.edu.hk) [5zjuyjw@cs.zju.edu.cn](mailto:5zjuyjw@cs.zju.edu.cn)

cost [[13](#_bookmark30), [37](#_bookmark58), [43](#_bookmark65)]. Dirty data may trigger the spread of fraud in-

Data quality is paramount in today’s data-driven world, especially in the era of generative AI. Dirty data with errors and inconsisten- cies usually leads to flawed insights, unreliable decision-making, and biased or low-quality outputs from generative models. The study of repairing erroneous data has gained significant impor- tance. Existing data repair algorithms differ in information utiliza- tion, problem settings, and are tested in limited scenarios. In this paper, we initially compare and summarize these algorithms using a new guided information-based taxonomy. We then systematically conduct a comprehensive evaluation of 12 mainstream data repair algorithms under the settings of various data error rates, error types, and downstream analysis tasks, assessing their error reduction per- formance with a *novel* metric. Also, we develop an effective and unified repair optimization strategy that substantially benefits the state of the arts, as empirically confirmed. We demonstrate that, the *pure clean data* may not necessarily yield the best performance in data analysis tasks and data is always worth repairing regard- less of error rate. Based on the found observations and insights, we provide some practical guidelines for 5 scenarios and 2 main data analysis tasks. We anticipate this paper enabling researchers and users to well understand and deploy data repair algorithms in practice. Finally, we outline research challenges and promising future directions in the data repair field.

[arXiv:2310.00711v1](#_bookmark30) [cs.DB] 1 Oct 2023

**PVLDB Reference Format:**

Wei Ni, Xiaoye Miao, Xiangyu Zhao, Yangyang Wu, Jianwei Yin. Automatic Data Repair: Are We Ready to Deploy?. PVLDB, 17(X): XXX-XXX, 2024. [doi:XX.XX/XXX.XX](https://doi.org/XX.XX/XXX.XX)

**PVLDB Artifact Availability:**

The source code, data, and/or other artifacts have been made available at [https://github.com/WelkinNi/Automatic-Data-Repair.](https://github.com/WelkinNi/Automatic-Data-Repair)

# INTRODUCTION

The global application landscape relies heavily on data, while ubiq- uitously erroneous information involved in data often compro- mises the data analysis performance, leading to a huge economic

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org.](mailto:info@vldb.org) Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 17, No. X ISSN 2150-8097. [doi:XX.XX/XXX.XX](https://doi.org/XX.XX/XXX.XX)

formation, inaccurate decision-making, and even legal repercus- sions [[36](#_bookmark59), [68](#_bookmark90)]. Among the strategies for efficiently mitigating erro- neous information and ensuring data quality, *automatic data clean- ing* [[33](#_bookmark55), [68](#_bookmark90)] has become a *pivotal* solution, especially in the age of generative artificial intelligence (GAI), where the large volume of high-quality training data for powerful GAI models like Chat- GPT [[52](#_bookmark74)] and Midjourney [[11](#_bookmark28)] currently often resort to *resource- intensive manual* data cleaning [[39].](#_bookmark61)

The promising automatic data cleaning task is generally con- ducted in two consecutive stages: *error detection* and *error repair*. The error detection stage is to identify all wrong data values or rule violation sets, while the error repair stage is to correct these wrong values into latent right ones [[2](#_bookmark42), [46](#_bookmark68), [47](#_bookmark69)]. Existing error detection studies [[32](#_bookmark54), [47](#_bookmark69), [51](#_bookmark73), [56](#_bookmark78)] have achieved obvious advancements, with an average F1 score exceeding 0.85 on real datasets [[47](#_bookmark69), [56](#_bookmark78)]. In contrast, the error repair stage is more complex and challenging. Various techniques for error repair have been employed, such as machine learning [[46](#_bookmark68), [58](#_bookmark80), [67](#_bookmark89)], transfer learning [[30](#_bookmark52), [46](#_bookmark68)], boosting algorithm [[42](#_bookmark64)], and few-show learning [[46](#_bookmark68)]. However, the perfor- mance of error repair still falls short of the expected ideal, with about below 0.7 on F1 score in real scenarios [[26](#_bookmark48), [46](#_bookmark68), [58](#_bookmark80)]. Conse- quently, it is essential and urgent to explore how to deploy effective data repair algorithms in real-life scenarios.

Table [1](#_bookmark1) compares and summarizes existing surveys of data re- pair study [[2](#_bookmark42), [13](#_bookmark30), [37](#_bookmark58)]. They either analyze data repair algorithms without experiments [[13](#_bookmark30), [37](#_bookmark58), [38](#_bookmark60)], or verify the impact of data re- pair on downstream models while missing some *vital* data repair algorithms [[1](#_bookmark41), [44](#_bookmark66)]. The limitations and analysis perspectives are described in the following *three* aspects.

**L1: Insufficient evaluation of error elimination perfor- mance.** Prior experimental comparisons of different algorithms mainly focus on evaluating the repair performance of the algo- rithms [[1](#_bookmark41), [14](#_bookmark31), [14](#_bookmark31), [17](#_bookmark34), [40](#_bookmark62), [46](#_bookmark68), [58](#_bookmark80)]. Evaluations in these experiments typically revolve around metrics like *precision*, *recall*, and *F1 score*, which primarily show the data repair capabilities of these algo- rithms. However, these metrics only provide a relative performance of data repair algorithms, inadequately displaying actual error changes. This inadequacy arises from two factors. First, metrics like *F1 score* inherently fail to present the error elimination information directly. Second, the calculations of these metrics may potentially include correctly repaired cells from the *initially right* data, leading to higher performance on these metrics, while these repaired cells have no impact on the data. As demonstrated in Table [1,](#_bookmark1) prior data

### Table 1: Comparing perspective coverage: our research vs. previous surveys.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Experiment Evaluation | Downstream Impact Analysis | Error Reduction Analysis | Multiple Rules | Optimization Strategy |
| Chu et al. [[13]](#_bookmark30) |  |  |  |  |  |
| Ilyas et al. [[37,](#_bookmark58) [38]](#_bookmark60) |  |  |  |  |  |
| Li et al. [[44]](#_bookmark66) | **✓** | **✓** |  |  |  |
| Abdelaal et al. [[1]](#_bookmark41) | **✓** | **✓** |  |  |  |
| Ours | **✓** | **✓** | **✓** | **✓** | **✓** |

repair surveys lack the evaluation of data error reduction perfor- mance. While error reduction performance is critical in choosing the proper algorithms for real-world deployment scenarios.

**L2: Limited exploration of complex scenarios.** As shown in Table [1,](#_bookmark1) existing data cleaning studies and surveys usually overlook various error types and multiple rules. When modeling complex er- ror scenarios, they often either verify the performance of data repair algorithms solely by changing error rates [[14](#_bookmark31), [17](#_bookmark34), [24](#_bookmark46), [40](#_bookmark62), [46](#_bookmark68), [58](#_bookmark80), [67](#_bookmark89)], or only examine random string variations or original errors in the datasets [[24](#_bookmark46), [44](#_bookmark66), [46](#_bookmark68)]. Although some studies introduce various error types and rates [[1](#_bookmark41), [59](#_bookmark81)], they focus on leveraging functional depen- dencies to repair data, which solely provide dependency relation information between attributes and may ignore the broader context of the data. For practical guidelines on algorithm deployment in real applications, it is necessary and essential to consider various and complicated factors contributing to erroneous information, a di- verse range of error types, and multiple rules during the evaluation process of data repair algorithms.

**L3: Absence of algorithm deployment guidelines.** Although current surveys provide practical analyses of data repair algorithm performance across diverse scenarios and parameters [[1](#_bookmark41), [13](#_bookmark30), [37](#_bookmark58), [38](#_bookmark60)], two critical questions regarding algorithm deployment remain unanswered. (i) Is data consistently worthy of repair regardless of the error rate? (ii) Considering the suboptimal performance of current methods, how can we leverage existing tools to ensure repair quality and select proper algorithms for different scenarios? Given the crucial role of data quality in downstream tasks and the significance of data repair in ensuring data quality, addressing these questions can provide vital guidelines in real-world deployment.

Therefore, in this survey, we conduct an exhaustive comparative analysis and experimental evaluation of 12 state-of-the-art data repair algorithms.

Our main contributions are described below.

* We summarize and compare *twelve* mainstream data repair algo- rithms based on a novel algorithm taxonomy, i.e., *rule-driven data repair algorithms*, *data-driven ones*, *data&rule-driven ones* and *model-driven ones* from five aspects. In addition, we implement *six* of these algorithms to present a comprehensive comparison. Moreover, we introduce a new and effective *metric* named *Error Drop Rate* to evaluate the error reduction performance of data repair algorithms on dirty data. It is surprising to find that, in the majority of cases, most data repair algorithms tend to introduce more errors rather than eliminate them from the data.
* Extensive experiments on five datasets are conducted to verify the performance of data repair algorithms under various scenar- ios characterized by different error rates, error types, data scales and downstream analysis tasks. Specifically, to model various errors in real-world scenarios, we consider both *inner* and *outer*

data errors, originating from a multitude of potential sources and encompassing varying error rates within multiple rules. Fur- thermore, leveraging existing error detection technologies, we explore optimization strategies to prevent the alteration of origi- nal right values into wrong ones, achieving results outperforming *state-of-the-art* algorithms.

* We also evaluate the influence of these data repair algorithms on downstream tasks with various error rates and types. Four interesting findings are concluded, including i) the performance of downstream tasks is not necessarily negatively correlated with the error rate of the data; ii) completely clean data does not always guarantee optimal performance for downstream tasks within specified models; iii) the data repair process consistently proves beneficial regardless of the error rate within the dataset; and iv) *inner* errors typically have a more negative impact.
* Based on insights from our experiments, we offer rule-of-thumb algorithm recommendations in *five* scenarios and guidelines for data repair to assist deployment and support downstream tasks. We also discuss current data repair challenges, and promising future directions considering the large language model (LLM).

The rest of this paper is organized as follows: We provide neces- sary preliminaries and problem definitions in Section [2.](#_bookmark2) Section [3](#_bookmark4) analyzes current data repair algorithms from 5 aspects based on a new taxonomy. In Section [4,](#_bookmark9) we offer comprehensive experimen- tal evaluation of error reduction analysis under various complex scenarios. We discuss algorithm recommendations, guidelines chal- lenges and future directions in Section [5,](#_bookmark20) followed by related works in Section [6](#_bookmark21) and conclusions in Section [7.](#_bookmark22)

# PRELIMINARIES

In this section, we initially introduce the essential rules generally employed by a large proportion of algorithms to conduct data repair. Subsequently, we present problem definitions for data repair.

## Rules for Data Repair

In the data repair process, rules serve as critical guidelines and can be categorized as specific rules and general rules. Specific rules pro- vide explicit instructions for modifying erroneous values, making data repair a deterministic process. However, real-world scenarios often lack external sources of authoritative information, such as master data or expert insights. This absence poses a significant challenge in employing specific rule-based approaches [[28].](#_bookmark50)

In contrast, general rules are designed to capture the relation- ships between attributes or values within specific attributes. Due to inherently simpler nature, general rules have a wider range of ap- plications. Most automatic data repair algorithms are built based on general rules like functional dependency (FD) and denial constraint (DC). Given an instance 𝐼 of relation 𝑅, which is characterized by

### Table 2: Relation of inhabitants.

its estimated true value by 𝑣ˆ . The goal of data repair is to make 𝑣ˆ

equal to

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FirstName | LastName | Gender | City | State |
| Paul | Smith | Male | New York | New York |
| Mark | White | Male | New York | New York |
| Anne | Nash | Female | Los Angeles | California |
| Anne | Nash | Male | Detroit | Michigan |
| Andy | Black | Female | Austin | Texas |

𝑐 𝑐

∗

a set of attributes 𝐴𝑡𝑡𝑟𝑠 = {𝐴1, 𝐴2, . . . , 𝐴𝑛 } that describe 𝐼 as a collection of tuples. FD [[54](#_bookmark76)] and DC [[55](#_bookmark77)] could be stated as follows.

*Definition 2.1.* **Functional Dependency.** A functional depen- dency is a statement of the form 𝑋 → 𝐴 where 𝑋 ⊆ 𝐴𝑡𝑡𝑟𝑠 and

𝐴 ∈ 𝐴𝑡𝑡𝑟𝑠, denoting that the values of attribute set 𝑋 uniquely determine the values of attribute 𝐴 across all tuples in 𝐼 .

*Definition 2.2.* **Denial Constraint.** A denial constraint 𝜑 is a statement of the form ∀𝑡𝛼 , 𝑡𝛽 ∈ 𝐼 : ¬ (𝑝1 ∧ . . . ∧ 𝑝𝑚) where 𝑝𝑖 is in the following form: 𝑡𝛼 .𝐴𝑥 𝜙 𝑡𝛽 .𝐴𝑦 or 𝑡𝛼 .𝐴𝑥 𝜙 𝑐, 𝐴𝑥 , 𝐴𝑦 ∈ 𝐴𝑡𝑡𝑟𝑠, 𝑐 is a constant, and 𝜙 is a built-in operator, i.e., ≠, =, ≤, *<*, ≥, *>*.

DC has greater expressive ability than FD. Since an FD of the form 𝑋 → 𝐴 can be equally expressed in the following DC format:

∀𝑡𝛼 , 𝑡𝛽 ∈ 𝐼, ¬ (𝑡𝛼 .𝑋0 = 𝑡𝛽 .𝑋0 ∧ . . . ∧ 𝑡𝛼 .𝑋𝑘 = 𝑡𝛽 .𝑋𝑘 ∧ 𝑡𝛼 .𝐴 ≠ 𝑡𝛽 .𝐴) ,

𝑋0 . . . 𝑋𝑘 ∈ 𝑋 . However, complex DCs cannot be equivalently ex- pressed in FD format. Due to the expressive ability and broad range of applications, our main focus for analysis and experimentation of rule-based data repair algorithms lies in DC-based algorithms.

Here is an example to further explain these two rules.

*Example 2.3.* Suppose there is a schema on information of in- habitants in Table [2,](#_bookmark3) and there are two FDs: 𝑓1 : 𝐿𝑎𝑠𝑡 𝑁 𝑎𝑚𝑒 →

𝐺𝑒𝑛𝑑𝑒𝑟 , 𝑓2 : 𝐶𝑖𝑡𝑦 → 𝑆𝑡𝑎𝑡𝑒 as well as one DC: 𝑑𝑐1 : ∀𝑡1, 𝑡2 ∈ 𝐼 :

¬ (𝑡1 .𝐶𝑖𝑡𝑦 = 𝑡2 .𝐶𝑖𝑡𝑦 ∧ 𝑡1 .𝑆𝑡𝑎𝑡𝑒 ≠ 𝑡2 .𝑆𝑡𝑎𝑡𝑒). The first FD 𝑓1 means if the values of 𝐿𝑎𝑠𝑡 𝑁 𝑎𝑚𝑒 are the same, then the values in 𝐺𝑒𝑛𝑑𝑒𝑟 should also be the same. The second FD 𝑓2 means if the values of

𝐶𝑖𝑡𝑦 are the same, then the values in 𝑆𝑡𝑎𝑡𝑒 should also be the same. The meaning of the DC is for all tuple pairs in 𝑅, the condition that the values of 𝐶𝑖𝑡𝑦 are the same while the values in 𝑆𝑡𝑎𝑡𝑒 are different should not exist. In this case, 𝑓2 and 𝑑𝑐1 are equivalent, expressing the same meaning.

## Problem Definition

Data repair mainly serves two primary goals: ensuring data consis- tency and eliminating errors. Considering that error elimination aligns more closely with attaining high-quality data, our primary focus is on evaluating error reduction performance. Recently pro- posed data repair algorithms also pursue this goal in accordance with the definition of *holistic repair* [[46](#_bookmark68), [47](#_bookmark69), [56](#_bookmark78), [58](#_bookmark80)], involving detect- ing and correcting all erroneous cells within a dataset. The formal definition of *holistic repair* [[58],](#_bookmark80) is provided below.

*Definition 2.4.* **Holistic Repair.** Given an instance 𝐼 of relation 𝑅, which is characterized by a set of attributes 𝐴𝑡𝑡𝑟𝑠 = {𝐴1, 𝐴2, . . . , 𝐴𝑛 } that describe 𝐼 as a collection of tuples, each tuple 𝑡 ∈ 𝐼 comprises a set of cells represented by 𝐶𝑒𝑙𝑙𝑠 [𝑡 ] = {𝐴𝑖 [𝑡 ]}, where each cell 𝑐 corresponds to a distinct attribute in 𝐴𝑡𝑡𝑟𝑠. For each cell, we denote its unknown true value by 𝑣∗, its initial observed value by 𝑣𝑐 , and

𝑐

𝑣𝑐 for each cell 𝑐 in 𝐷 for which 𝑣𝑐 ≠ 𝑣𝑐 .

However, earlier concepts of data repair primarily focus on en- suring consistency in databases, which is previously regarded as an important task. It is necessary to provide the *consistency repair* definition to facilitate the understanding of these algorithms, which is defined as follows.

*Definition 2.5.* **Consistency Repair.** Given a set of rules Σ over a relation 𝑅, and two instances 𝐼 and 𝐼 ′ of 𝑅, 𝐼 ′ is a repair of 𝐼 with respect to Σ if 𝐼 ′ adheres to the rule set Σ, symbolized as 𝐼 ⊨ Σ.

According to this definition, the applied algorithms usually per- form data repair with the goal of minimizing the *cost* or *cardinality* differences between 𝐼 and 𝐼 ′ based on specific scenarios [[7].](#_bookmark23)

Meanwhile, as errors can arise not only in the data but also in the given rules, several algorithms ensure database consistency by repairing both the data and given rules based on the concept of *tolerant repair* [[8,](#_bookmark25) [12,](#_bookmark29) [62],](#_bookmark84) which is defined as below.

*Definition 2.6.* **Tolerant Repair.** Given a set of rules Σ over a relation 𝑅, a set of changed rules Σ′, and two instances 𝐼 and 𝐼 ′ of

𝑅, (Σ′, 𝐼 ′) is a repair of 𝑅 such that 𝐼 ′ ⊨ Σ′.

Similar to *consistency repair*, the applied algorithms usually per- form the repair process with consideration of changing thresh- olds on differences between Σ and Σ′ based on specific scenar- ios [[8,](#_bookmark25) [12,](#_bookmark29) [62].](#_bookmark84)

# OVERVIEW OF DATA REPAIR ALGORITHMS

In this section, we firstly present a new taxonomy and overall analysis of 12 current mainstream algorithms based on the guided information. Then we introduce each of them in detail.

## Taxonomy

Current data repair algorithms can be categorized into four groups, i.e., *rule-driven* ones, *data-driven* ones, *rule&data-driven* ones, and *model-driven* ones, as presented in Table [3.](#_bookmark5) Among them, *Rule- driven* algorithms employ predefined rules to detect and repair data based equivalent classes specified in the rules, which is a set of cells that should have the same values.

Conversely, *data-driven* algorithms rely on whole data distribu- tion information rather than rules for data repair *Rule&data-driven* algorithms integrate both rule-based modeling and data distribution information, enabling possible modifications to both the original rules and data. *model-driven* algorithms repair data aiming to en- hance the downstream model performance. We believe these cate- gories adequately represent the various approaches, and we discuss their workflows and main techniques in the following sections.

## Rule-Driven Algorithms

The workflow of *rule-driven* algorithms is depicted in Figure [1](#_bookmark6) in black lines. With the support of provided rules, the data is parti- tioned based on equivalent classes, facilitating the detection of cells that violate these rules, as well as generating possible repairing candidates for the violation cells. Ultimately, the most suitable can- didates are determined based on the cost or distance computation.

### Table 3: Summary of existing data repair algorithms. There are six types of input: rules (R), original data (OD), all detection results (ADR), partial detection results (PDR), labeled data (LD), and downstream models(DM). We assess time complexity based on the cardinality of the rule set |Σ|, dataset size |D|, and attribute set size |R|.

Algorithms

Category

Input

Detection Goal

Repair Goal

Candidate Source

Time Complexity

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Holistic | Rule Driven | R+OD | Rule Violations | Consistency Repair | Equiv Class | 𝑂 ( |Σ| · | D|2 ) |
| BigDansing | Rule Driven | R+OD | Rule Violations | Consistency Repair | Equiv Class | 𝑂 ( |Σ| · | D|2 ) |
| Horizon | Rule Driven | R+OD | Rule Violations | Holistic Repair | Equiv Class | 𝑂 ( | D| · | R|2 ) |
| Nadeef | Rule Driven | R+OD | Rule Violations | Consistency Repair | Equiv Class | 𝑂 ( |Σ| · | D|2 ) |
| MLNClean | Rule Driven | R+OD | Rule Violations | Holistic Repair | Equiv Class | 𝑂 ( |Σ| · | D|2 ) |
| Daisy | Rule Driven | R+OD | Outliers | Holistic Repair | Equiv Class | 𝑂 ( |Σ| · | D|2 · | R|) |
| Scare | Data Driven | OD+PDR | Outliers | Holistic Repair | Equiv+Domain | 𝑂 ( | D| · 𝑙𝑜𝑔 | D| · | R|2 ) ) |
| Baran | Data Driven | LD+OD+ADR | - | Holistic Repair | Equiv+Domain+Str Variation | 𝑂 ( | D| · | R|2 ) ) |
| Unified | Rule&Data Driven | R+OD | Outliers | Tolerant Repair | Equiv Class | 𝑂 ( |Σ| · | D|2 · | R|) |
| Relative | Rule&Data Driven | R+OD | Outliers | Tolerant Repair | Equiv Class | 𝑂 ( | D| · | R||Σ||D| ) |
| HoloClean | Rule&Data Driven | R+OD | Wrong Values | Holistic Repair | Equiv+Domain | 𝑂 ( |Σ| · | D|2 ) |
| BoostClean | Model Driven | OD+DM | Outliers | Model Performance | Mean,Mode,Median | 𝑂 ( | D| · 𝑙𝑜𝑔 ( | D|) · | R|) |

**Blue :** Extra steps to repair data **Green:** Extra steps to repair rules

**Black:** Steps to repair data

**Ori Rules /**

**Rule ED Result**

**Error Detection**

**Data ED Result**

**Data/Rule Candidates Generation**

**Ori Rules Equiv Classes**

**Rule Candidates Data Candidates**

**Equivalent Class Group**

**Repaired Rules&Data**

**Repair Module**

**Weakly Labeled Data**

**Rule 1**

**Rule 2**

**Rule 3**

**Dirty Data**

**Rule 1'**

**Rule 2'**

**Rule 3'**

**Clean Data**

### Figure 1: The workflow of *rule-driven* and *rule&data-driven* algorithms. The black line depicts the data repair process of *rule-driven* algorithms; green and blue lines represent extra steps employed by certain *rule&data-driven* methods to re- pair rules and data respectively; ED refers to error detection.

**Holistic** [[14](#_bookmark31)]. Holistic is a classical data repair algorithm that addresses DC violations, as stated in Table [3.](#_bookmark5) Holistic encodes all DC violation cells in a *conflict hypergraph*. Then, Holistic generates diverse repair semantics based on equivalent classes, employing repair contexts (RC) considering interactions among conflicting values. However, due to the necessity of traversing data across all running processes, the time complexity escalates to 𝑂 (D2). This poses a significant challenge when dealing with large datasets.

**BigDansing** [[40](#_bookmark62)]. To alleviate the high time cost of the previous algorithms like Holistic stated in Table [3,](#_bookmark5) BigDansing optimizes the detection and repair process through the rule specification ab- straction featuring five operators (*Scope, Block, Iterate, Detect, and GenFix*), automatically designing a logical data processing plan. Though the time complexity keeps the same, these techniques ef- fectively reduce the actual time cost. Furthermore, BigDansing empowers users to focus on the logic of their rules, sparing them from the intricacies of specific implementation details.

**Horizon** [[59](#_bookmark81)]. From another view, Horizon reduces the time cost by leveraging the inherent order feature of FD. Initially, it

establishes a cost model to retain frequent patterns and generates a directed *FD pattern graph*, where edges connect the corresponding values from left to right-hand side attributes in FDs and nodes represent values. Then, the pattern graph is traversed in *linear* time to identify patterns that are most strongly supported by the data to repair FD violation cells, which significantly reduce time cost compared with other methods, as shown in Table [3.](#_bookmark5)

**Nadeef** [[17](#_bookmark34)]. In contrast to previous methods focusing on en- hancing repair performance based on specific types of rules, Nadeef emphasizes on its capacity to harness a broader spectrum of rules for various applications. This enables users to specify not only tra- ditional constraints like CFDs (FDs), MDs, and ETL rules, but also other specific rules. Based on various predefined rules, NADEEF offers two data repair algorithms tailored to different scenarios.

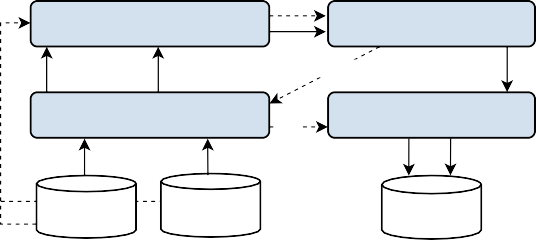
**MLNClean** [[24](#_bookmark46)]. Previous algorithms often instantiated rules by values to detect and repair data without assurance of correctness. To mitigate this issue, MLNClean leverages Markov Logic Networks (MLNs) to infer *instantiated rules* of the DCs. Subsequently, based on equivalence classes, MLNClean generates and integrates multiple data versions within the designed reliability and fusion score.

**Daisy** [[26](#_bookmark48)]. Daisy aims to optimize query execution in the data- base instead of solely repairing data. Daisy firstly checks the data and query conditions to decide on where to place the cleaning step in the logical plan. Subsequently, Daisy transforms the query result into a probabilistic outcome by replacing erroneous values with a set of candidate fixes with probabilities.

## Rule&Data-Driven Algorithms

The workflow of *rule&data-driven* methods is based on *rule-driven* ones with additional steps as shown in Figure [1.](#_bookmark6) There exist two pri- mary workflows for *rule&data-driven* methods. The first workflow aims to rectify both error data and rules. Based on the data repair process, it generates rule candidates for rule repairs. The rectified rules are subsequently applied to further facilitate data repair until reaching the optimal state or cost-effectiveness, as indicated by the green lines. The second workflow, illustrated by the blue lines, focuses on rectifying data within both rule and data contexts, thus involving weakly labeled data in the repair process.

**Repair Strategies**



**Featurization**

**Model Training**

**Labeled Data Candidates Trained Model Trained Model**

**Candidates Generation**

**Repair Module**

**Candidates**

**Dirty Data**

**+Labels**

**Detection Results**

**Clean Data**

**Data ED Results**

**Repaired Data**

**Model Training**

**Trained Model**

**Error Detection**

**No Optimal?**

**Yes**

### Figure 2: The workflow of *data-driven* algorithms. Solid lines

**Dirty Data Clean Data**

**Downstream Model**

### and dotted lines represent different data repair workflows.

**Unified** [[12](#_bookmark29)] introduces a comprehensive cost model to mini- mize the overall description length (DL) that encompasses both data and rule repairs. Unified initiates by generating core and tuple pat- terns, representing frequently occurring tuples and deviating tuples respectively. Subsequently, to determine final repairs, it employs the cost model based on the overall DL to evaluate the expenses of changing tuple patterns and modifying rules.

**Relative** [[8](#_bookmark25)]. Similar to Unified, Relative generates solutions for modifying both data and rules according to the specified threshold of relative trust. Relative firstly explores the space of rule modifi- cations with minimal changes Then, it employs an approximate algorithm for computing minimal data changes within the updated rules, ultimately deriving the final repair. But the backtracking process leads to exponential time complexity, as shown in Table [3.](#_bookmark5) **HoloClean** [[58](#_bookmark80)]. Although all guided by both data and rules, HoloClean distinguishes itself from the previously mentioned meth- ods by focusing solely on data repair. Initially, HoloClean compiles all rules into a program to capture their quantitative statistics and identify trustworthy data. Subsequently, within the *DeepDive* frame- work [[60](#_bookmark82)], HoloClean employs statistical learning and probabilistic

inference from the data to perform error detection and repair.

## Data-Driven Algorithms

As depicted in Figure [2,](#_bookmark7) there are two main workflows of *data-driven* algorithms. They usually take dirty data and (partial) detection re- sults as inputs and employ machine learning (ML) models to capture data distribution information. In the workflow represented by solid lines, dirty data is firstly processed to generate the candidates for error cells. Then, utilizing features generated by the featurization module, an ML model is trained to conduct repair process. While the workflow depicted by dotted lines employs the training of ML models to generate repair candidates. Subsequently, based on these candidates, the repair process is executed.

**Scare** [[67](#_bookmark89)] Scare is the first method that employs machine learn- ing (ML) techniques to facilitate data repair. Scare firstly partitions the data into blocks based on partition functions. For each block, a set of *classifiers* is acquired through learning from clean attributes determined by the error detection results, and utilized to forecast the candidates of dirty attributes one by one. Then Scare deter- mines the ultimate repairs by optimizing the connections among candidate values across partitions.

**Baran** [[46](#_bookmark68)]. Unlike capturing data information in the candidate generation phase in Scare, Baran firstly generates a comprehensive array of candidates within multiple models based on various con- texts, including factors such as value, vicinity, and domain shown in

### Figure 3: The workflow of *model-driven* algorithms, where ED refers to error detection.

Table [3,](#_bookmark5) thereby significantly enhancing the likelihood of including the correct value. Then, with few labeled data and the generated candidates, Baran trains a *classifier* to determine the final repair for other error cells.

## Model-Driven Algorithms

As depicted in Figure [3,](#_bookmark8) *model-driven* methods usually take the applied model and dirty data as input. Then, following the error de- tection process, various sequences of repair strategies are employed to repair the data. The applied model is trained using the repaired data and evaluated on validation data. This process continues until an optimal repair sequence that best enhances downstream perfor- mance is then employed to clean the data.

**BoostClean** [[42](#_bookmark64)]. In contrast to previous methods, the core objec- tive of BoostClean is to enhance the performance of downstream models. To achieve this goal, BoostClean employs a sequential approach: it progressively selects subsequent conditional repairs, integrates them into the existing repair sequence, and employs this updated sequence to train a new downstream classifier. The perfor- mance of this classifier is subsequently assessed using validation data. This iterative process continues until an optimal conditional repair sequence is established. The derived conditional repair se- quence is applied to repair the data.

# EXPERIMENTAL EVALUATION

This section presents an extensive experimental study involving 12 algorithms, focusing on both the actual data repair performance and the subsequent effects on downstream tasks. The evaluation aims to address the following key questions:

**Q1:** To what degree can current automatic data repair algorithms mitigate errors in the real-world datasets?

**Q2:** How effective are current algorithms at handling complex high error rates and various error types data?

**Q3:** Can we develop an improved data repair strategy based on experimental observations?

**Q4:** How does data repair impact downstream models? Can the optimization strategy still work on the downstream task?

## Experiment Settings

**Dataset&Rules.** We conduct experiments on 4 real-world datasets and 1 synthetic dataset outlined in Table [5.](#_bookmark11) Corresponding clean ver- sions exist for each dataset. *Hospital* and *Flights* are two real-world datasets obtained from previous research [[58](#_bookmark80)]. *Beers* is a real-world

### Table 4: Error detection and repair performance comparison on real-world datasets.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Metric | DataSet | Rule-Driven | | | | | | Data-Driven | | Rule&Data-Driven | | | Model-Driven |
| Bigdansing | Holistic | Nadeef | Daisy | MLNClean | Horizon | Baran | Scare | HoloClean | Unified | Relative | Boostclean |
| EDR | Hospital | -0.0819 | -0.0039 | -1.7996 | 0.0000 | 0.4322 | 0.0530 | 0.4519 | 0.0000 | 0.4872 | **0.6012** | n/a | -5.7132 |
| Flights | -0.0026 | -0.0021 | 0.0001 | 0.0000 | 0.0030 | 0.0004 | **0.0083** | 0.0000 | -0.0004 | 0.0000 | n/a | -0.0028 |
| Beers | -0.0109 | -0.0110 | -0.4783 | 0.0000 | 0.0482 | -0.0679 | **0.0708** | 0.0000 | -4.2478 | -0.1221 | n/a | -0.7174 |
| Rayyan | -0.4535 | -0.9614 | -2.5367 | 0.0000 | -0.6042 | -0.3028 | **0.0875** | 0.0000 | -1.2249 | -0.1862 | n/a | -0.6220 |
| ER\_F1 | Hospital | 0.6245 | 0.6403 | 0.0713 | 0.0000 | 0.7240 | 0.5661 | 0.6618 | 0.0469 | 0.6552 | **0.7826** | n/a | 0.3310 |
| Flights | 0.0014 | 0.0024 | 0.0007 | 0.0000 | 0.0059 | 0.0028 | **0.0219** | 0.0001 | 0.0029 | 0.0000 | n/a | 0.0000 |
| Beers | 0.0731 | 0.0688 | 0.0094 | 0.0000 | 0.1191 | 0.0818 | **0.8420** | 0.0009 | 0.0498 | 0.0106 | n/a | 0.0000 |
| Rayyan | 0.0128 | 0.0047 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.4308 | 0.0000 | **0.4819** | 0.0000 | n/a | 0.0000 |
| ED\_F1 | Hospital | 0.5903 | 0.5834 | 0.0745 | 0.0000 | **0.7297** | 0.4039 | 0.6316 | 0.2998 | 0.0753 | 0.5502 | n/a | 0.3062 |
| Flights | 0.6341 | 0.6440 | 0.8885 | 0.0000 | 0.0236 | 0.9069 | **0.9903** | 0.0000 | 0.9057 | 0.0000 | n/a | 0.0000 |
| Beers | 0.0730 | 0.0687 | 0.0385 | 0.0000 | 0.1191 | 0.0330 | **0.9947** | 0.0000 | 0.0665 | 0.0095 | n/a | 0.0000 |
| Rayyan | 0.4582 | 0.4679 | 0.2502 | 0.0000 | 0.1272 | 0.3867 | **0.7823** | 0.0000 | 0.4040 | 0.0059 | n/a | 0.0040 |

**Table 5: Dataset characteristics. There are four kinds of error types: missing value (MV), typo (T), violated attribute depen- dency (VAD), and formatting issue (FI).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | #Tuples | #Attrs | Error Rate | Error Types |
| Hospital | 1000 | 20 | 3% | T, VAD |
| Flights | 2376 | 7 | 30% | MV, FI, VAD |
| Beers | 2410 | 11 | 16% | MV, FI, VAD |
| Rayyan | 1000 | 11 | 9% | MV, T, FI, VAD |
| Tax | 200000 | 15 | 4% | T, FI, VAD |

dataset sourced through web scraping and manually cleaned [[46](#_bookmark68)]. *Rayyan* is another real-world dataset cleansed by its owners [[47](#_bookmark69)], while *Tax* is a large synthetic dataset from the BART repository [[4](#_bookmark44)] with various data error types. To discover DCs, we initially employ two widely-used DC discovery methods: DCFinder [[55](#_bookmark77)] and Hy- dra [[10](#_bookmark27)]. We then manually check all discovered rules against the clean data, deciding whether to accept, repair, or deny each rule. **Error Generation.** In real-world scenarios, data errors arise from incorrect value allocation and out-of-domain values. Thus, we focus on two main error categories derived from possible error scenarios: *inner errors* and *outer errors*. *Inner errors* involve cases of improper value assignment, where the correct value is replaced with a ran- domly selected alternative from within the domain. *Outer errors* originate from diverse sources including typos, explicit and implicit missing values, and Gaussian noise. And we generate *outer errors* based on the public code of BigDaMa [[9](#_bookmark26)]. We believe this compre- hensive set of sources captures most real-world error conditions. **Evaluation Metrics.** To comprehensively assess algorithm perfor- mance, we utilize multiple data repairing metrics. For evaluating error detection effectiveness, we rely on the widely recognized met- rics of precision, recall, and F1 score. Due to space limitations, our presentation in the results focuses exclusively on the F1 score for error detection (*ED\_F1*). Considering that the error detection goals of these data repair algorithms may encompass rule violations or outliers for which explicit detection results are not guaranteed, we maintain a consistent evaluation approach based on the disparities between repaired and original cells.

For the subsequent error repair process, traditional metrics such as precision, recall, and the F1 score are commonly used [[8](#_bookmark25), [13](#_bookmark30), [14](#_bookmark31)]. However, these evaluation metrics primarily focus on the perfor- mance of method itself, often failing to adequately reflect the extent of error elimination. To address this limitation, we introduce the

concepts of *Decreased Error Count (DEC)*, *Introduced Error Count (IEC)*, and *Original Error Count (OEC)* to calculate the *Error Drop Rate (EDR)*. This approach parallels established precision, recall, and F1 score calculations, with *DEC* corresponding to *False Posi- tives* and *IEC* representing *True Negatives*. The formula for *EDR* is given by 𝐸𝐷𝑅 = 𝐷𝐸𝐶−𝐼 𝐸𝐶 . We believe this metric better reflects the actual data repair impact compared to assessing algorithm per- formance alone. Additionally, for evaluating efficiency, we include the computation time in seconds.

**Hyper-parameter Settings.** The default hyper-parameter set-

𝑂𝐸𝐶

tings of all algorithms are adopted as specified in their respective source codes or papers. In the case of algorithms that leverage la- beled data, such as MLNClean and Raha-Baran, a uniform minimum default number of 20 labeled tuples is applied across all methods. As for the *data-driven* algorithms, considering the necessary detection result, the results of the state-of-the-art error detection methods Raha [[47]](#_bookmark69) are adopted as inputs.

**Implementation Setup.** Our experiment evaluates 12 represen- tative data repairing algorithms mentioned in Section [3.](#_bookmark4)

In the case of Holistic, BigDansing, Horizon, Unified, Relative and SCARE, for which publicly available source code is unavailable, we re-implement them in Python. For the other methods, we con- duct experiments using their original publicly accessible codebases. Each experiment is conducted three times, with the reported results being derived from the mean value. All experiments are conducted on a Linux server with an Intel (R) Xeon (R) Gold 6326 CPU @ 2.90GHz and 512GB RAM, running Ubuntu 20.04.6 LTS.

## Performance on Real Scenarios

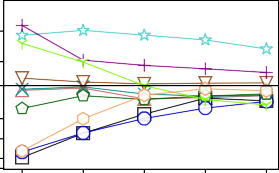
We firstly run all cleaning algorithms on each real-world dataset based on the default hyperparameters settings above. And we ex- plore their performance of *EDR*, *ER\_F1* and runtime. To show the effectiveness of *EDR*, we also present f1 score of the algorithms *ER\_F1*. The datasets utilized for this evaluation encompass *Hospital*, *Flights*, *Beers*, and *Rayyan*. "n/a" in the table means the methods can not terminated and return results in 24 hours. As none of them could finish the data repair process on *Tax* in *one* day, we segment the dataset into subsets to explore the scalability of these methods. The larger datasets encompass the entirety of the smaller ones.

**Error Repair Performance.** Regarding the *EDR* metric, as shown in Table [4,](#_bookmark10) most error repair methods lead to an increase in the

### Table 6: Runtime performance comparison across varying sizes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DataSet | Rule-Driven | | | | | | Data-Driven | | Rule&Data-Driven | | | Model-Driven |
| Bigdansing | Holistic | Nadeef | Daisy | MLNClean | Horizon | Baran | Scare | HoloClean | Unified | Relative | Boostclean |
| Tax-10k | 2691s | 45261s | 821s | n/a | **66s** | 1675s | 49346s | 41871s | 935s | 2040s | n/a | 502s |
| Tax-20k | n/a | n/a | 4421s | n/a | **122s** | 5757s | n/a | 17616s | n/a\* | 7411s | n/a | 1233s |
| Tax-30k | n/a | n/a | 7497s | n/a | **226s** | 11378s | n/a | n/a | n/a\* | 11885s | n/a | 1729s |
| Tax-40k | n/a | n/a | 15492s | n/a | **329s** | 18613s | n/a | n/a | n/a\* | 17223s | n/a | 2708s |
| Tax-50k | n/a | n/a | 19571s | n/a | **395s** | 27124s | n/a | n/a | n/a\* | 23671s | n/a | 3939s |



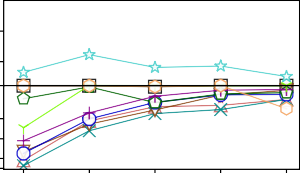
0.5

Error Drop Rate

0.25

0

0.5

0.25

0

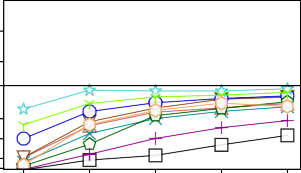
0.5

0.25



0

0.5

0.25

0

-0.5

-1

-2

-4

10% 30% 50% 70% 90%

*Error Rate*

* + 1. Hospital

-0.5

-1

-2

-4

10% 30% 50% 70% 90%

*Error Rate*

* + 1. Flights

-0.5

-1

-2

-4

10% 30% 50% 70% 90%

*Error Rate*

* + 1. Beers

-0.5

-1

-2

-4

10% 30% 50% 70% 90%

*Error Rate*

* + 1. Rayyan

### Figure 4: Data repair performance comparison under different error rates (Exp Scale).

error rate rather than a reduction with negative *EDR* values. This phenomenon can be attributed to the substantial incorrect alter- ation of correct cell values. Notably, we can see that *ER\_F1* score cannot comprehensively reflect the actual error elimination perfor- mance. In the test on *Hospital*, most algorithms achieve better *ER\_F1* performance, since it has more redundancies and simpler contexts than the others while they show significant *EDR* distinctions. This discrepancy arises because substantial yet low-proportion impre- cise data repairs can inflate precision and recall, thereby increasing the *ER\_F1* despite introducing substantial errors. Furthermore, ac- curately repair on initially correct data boosts precision and F1 but does not improve error reduction performance. Moreover, *EDR* provides an easier way to evaluate repair performance, since higher values indicate better performance, while negative values signify an increase in errors and positive values a reduction. This is more straightforward than drawing conclusions from the various *ER\_F1* scores. Among algorithms, Raha-Baran also shows superior per- formance except on the *Hospital* dataset. This is primarily due to rich features for a broad range of candidates and the strong repre- sentation ability of ML models. Conversely, most of the *rule-driven* algorithms display inferior performance except MLNClean. This demonstrates that the current candidate confidence calculation fails to capture the accurate values.

As for the *rule&data-driven* methods, HoloClean and Unified, demonstrate superior performance compared to most *rule-driven* methods. This superiority stems from their incorporation of data distribution information.

**Error Detection Performance.** The error detection performance is mainly affected by the *detection goal* and the specific dataset. In Ta- ble [4,](#_bookmark10) we could observe that the actual *EDR* is not highly correlated with the *ED\_F1* score, indicating that superior error detection does not necessarily lead to accurate identification of repair candidates. Among them, the error detection performance of Raha-Baran based on Raha is generally better than other methods. This can be attrib- uted to orientation of Raha towards detecting wrong values and it considers various types with rich features. Although HoloClean shares a similar objective of value error detection, it falls short of

achieving comparable performance. This discrepancy arises from the fact that its internal error detection process primarily focuses on rule violations, as well as valid attribute independence assumptions under real scenarios of *Naive Bayesian* model employed to generate data for conducting weakly supervised learning process. Regarding *rule-driven* algorithms oriented toward rule violation detection, they perform relatively worse than the previous algorithms, since they lack consideration of non-rule violation errors. They show varying performance owing to disparate methods of calculating candidate confidence in rule violation cell sets. Moreover, algorithms target- ing outliers like Daisy, Scare, Unified and BoostClean show even poorer performance. This could be explained by the inadequacy of relying solely on data distribution for outlier identification, which is insufficient for effectively detecting true errors. Unified performs better due to its incorporation of rule considerations, particularly evident in the *Hospital* dataset, even outperforming Raha-Baran. This is likely because *Hospital* features the highest number of rules. **Scalable Performance.** When testing scalablity performance on *Tax* dataset shown in Table [6,](#_bookmark12) we could observe that MLNClean is the most efficient data repair method. While Baran, BigDansing and Holistic consume more time than the other methods. This is due to the high cost of generating various candidates in Baran, as well as the 𝑂 (|D|2) operation to process data as analyzed previously on BigDansing and Holistic. HoloClean is terminated by the system due to out-of-memory problem when the data size is over 20k, labeling the results as n/a\*. As the data size grows, we can see that the time complexity does not totally fit the precious time complexity analysis, this is because the running time is determined by various conditions, like data distribution, error distribution, etc. But none of them finish cleaning *Tax* in 24 hours except MLNClean, showing that scalability is still a major challenge.

**Summary.** Based on the results provided in this section, we sum-

marize our observations as follows.

* Raha-Baran consistently produces nearly the best results across all test cases, making it a recommended choice for dataset repair when sufficient time is available.

0.5

Error Drop Rate

0

-0.5

-4

0.5

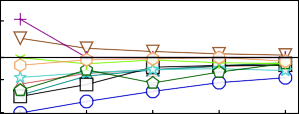
Error Drop Rate

0

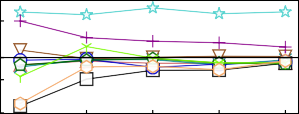
-0.5

-4

10% 30% 50% 70% 90%

*Error Rate*

1. *Inner error*, Hospital



10% 30% 50% 70% 90%

*Error Rate*

* + 1. *Outer error*, Hospital

0.5

0

-0.5

-4

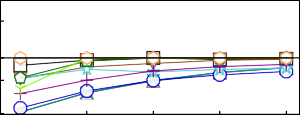
0.5

0

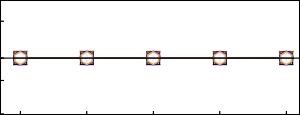
-0.5

-4

10% 30% 50% 70% 90%

*Error Rate*

1. *Inner error*, Flights



10% 30% 50% 70% 90%

*Error Rate*

* + 1. *Outer error*, Flights

0.5

0

-0.5

-4

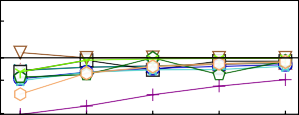
0.5

0

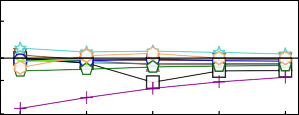
-0.5

-4

10% 30% 50% 70% 90%

*Error Rate*

1. *Inner error*, Beers



10% 30% 50% 70% 90%

*Error Rate*

* + 1. *Outer error*, Beers

0.5

0

-0.5

-4

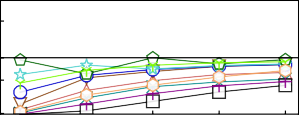
0.5

0

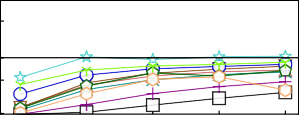
-0.5

-4

10% 30% 50% 70% 90%

*Error Rate*

1. *Inner error*, Rayyan



10% 30% 50% 70% 90%

*Error Rate*

* + 1. *Outer error*, Rayyan

### Figure 5: Data repair performance comparison under different error types (Exp Scale).



0.5

EDR

0

-0.5

Opt Ratio

4

2

0

0.5

EDR

0

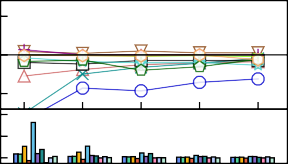
-0.5

Opt Ratio

4

2

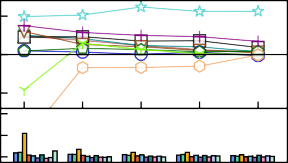
0



10% 30% 50% 70% 90%

*Error Rate*

1. *Inner error*, Hospital



10% 30% 50% 70% 90%

*Error Rate*

1. *Outer error*, Hospital

0.5

0

-0.5

4

2

0

0.5

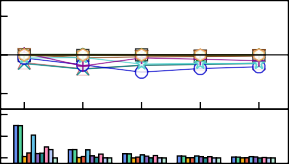
0

-0.5

4

2

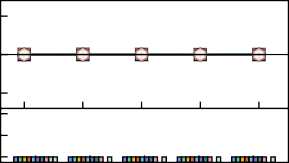
0



10% 30% 50% 70% 90%

*Error Rate*

1. *Inner error*, Flights



x x x x

10% 30% 50% 70% 90%

*Error Rate*

1. *Outer error*, Flights

0.5

0

-0.5

4

2

0

0.5

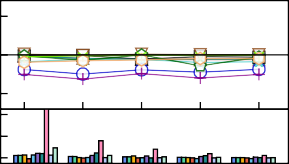
0

-0.5

4

2

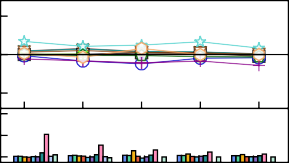
0



10% 30% 50% 70% 90%

*Error Rate*

1. *Inner error*, Beers



x x x

10% 30% 50% 70% 90%

*Error Rate*

1. *Outer error*, Beers

0.5

0

-0.5

4

2

0

0.5

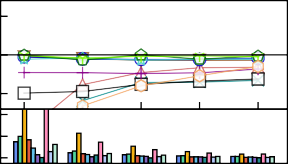
0

-0.5

4

2

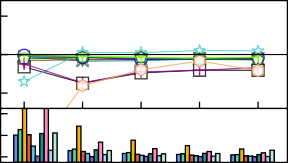
0



10% 30% 50% 70% 90%

*Error Rate*

1. *Inner error*, Rayyan



10% 30% 50% 70% 90%

*Error Rate*

1. *Outer error*, Rayyan

### Figure 6: Optimization using error detection tools

* Most of the other methods do not repair the error in the original data ideally. They tend to increase the error rate in the original data instead of eliminating the errors after the repair process. This phenomenon stems from a substantial number of cells tran- sitioning from accurate values to erroneous ones.
* Current methods can not process data within complex errors. In *Rayyan* and *Beers*, characterized by complex and lengthy data, almost all repair methods struggle to attain optimal performance. Conversely, the *Hospital* dataset, abundant in redundancy, proves easier to repair, consequently yielding improved results.
* Almost all the methods show various results across these four datasets, which means the repair performance is highly related to inherent features and distribution of the specific data. Addi- tional metrics for evaluating data and error distribution prove indispensable in guiding the data repair process.

## Robustness to Complex Data Errors

In this section, we aim to simulate the real and complex scenarios by introducing both *inner* and *outer* errors into the datasets at varying levels of error rates. We assume that each cause of error occurs randomly and independently of others, thus the proportion

of *inner* errors to *outer* errors is 1:4. Then we evaluate the impact of inner and outer error on the data repair algorithms separately. Moreover, considering the low relation between *ED\_F1* and *EDR*, we only report *EDR* in the following sections. Since the Daisy and Relative cannot get the results in 24 hours or have no influence on the datasets, we exclude them in this part of the experiments.

**Performance under different error rate.** As depicted in figure [4,](#_bookmark13) consistent with the previous experiment, the majority of results across the datasets fail to reduce errors in the data. Furthermore, as the error rate increases, most methods exhibit less effects on the repaired data. Raha-Baran, Unified, and MLNClean consistently demonstrate relatively stable performance, surpassing the others in most conditions. Besides, on the *Hospital* dataset, methods involv- ing data information, such as HoloClean, Unified, and Raha-Baran, generally perform better due to substantial data redundancies and rules. *Rule-driven* methods appear to struggle in significantly reduc- ing errors across varying error rates. Conversely, on the *Beers* and *Rayyan* datasets, most algorithms yield unfavorable results, possibly due to complex data structures and patterns. Notably, Raha-Baran effectively reduces errors even at a 90% error rate, demonstrating the value of repairing data with high error levels.

1

0.75

F1 Score

0.5

0.25

0

1

0.75

F1 Score

0.5

0.25

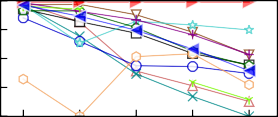
0

EDR

0.5

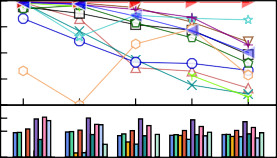
0

-1



10% 30% 50% 70% 90%

* + 1. *XGBoost*, Hospital



x

10% 30% 50% 70% 90%

*Error Rate*

1. *MLP*, Hospital

1

0.75

0.5

0.25

0

1

0.75

0.5

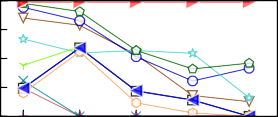
0.25

0

0.5

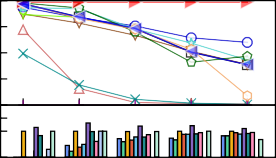
0

-1



10% 30% 50% 70% 90%

* + 1. *XGBoost*, Flights



x x x

10% 30% 50% 70% 90%

*Error Rate*

1. *MLP*, Flights

1

0.75

0.5

0.25

0

1

0.75

0.5

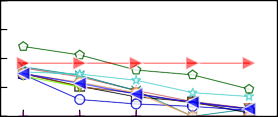
0.25

0

0.5

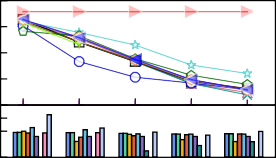
0

-1



10% 30% 50% 70% 90%

* + 1. *XGBoost*, Beers



x x x

10% 30% 50% 70% 90%

*Error Rate*

1. *MLP*, Beers

1

0.75

0.5

0.25

0

1

0.75

0.5

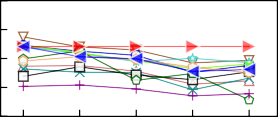
0.25

0

0.5

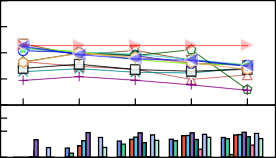
0

-1



10% 30% 50% 70% 90%

* + 1. *XGBoost*, Rayyan



10% 30% 50% 70% 90%

*Error Rate*

1. *MLP*, Rayyan

### Figure 7: Classification task performance within *XGBoost* and *MLP*, where *EDR* is presented with exp scale.

**Performance under different error types.** As depicted in Fig- ure [5,](#_bookmark14) *inner* errors pose greater challenges for methods leveraging data information, like HoloClean, Unified, and Raha-Baran, while *rule-driven* methods exhibit greater robustness against various error types. Aligning with previous experiments, Raha-Baran effectively reduces errors even at 90% rates on *Hospital*, demonstrating the value of repairing highly erroneous data, especially in the presence of outer errors. Boostclean exhibits mixed results, indicating sen- sitivity to error complexity and distribution. Notably, almost all methods show minimal impact on the *Flights* dataset, likely due to its original high cardinality, which outer errors further improve. **Summary.** We present a summary of the experimental results along with key insights in this subsection as follows.

* Increasing error rates diminish algorithm effects on original datasets and methods incorporating data information outperform solely reliance on rule ones, especially for *outer errors*.
* Most of *rule-driven* methods yield consistently negative out- comes across all tests, underscoring the importance of avoiding incorrect fixes to correct data when employing these methods.
* *Inner* errors have a greater influence than *outer* errors on repair performance, particularly for data-driven methods.
* Data with a high error rate is still worthy of conducting repair, especially within outer errors.

## Optimizations

Based on the aforementioned findings, to avoid substantial transfor- mations of cells within correct values into incorrect ones, we con- sider leveraging the current state-of-the-art error detection method. We ensure that values identified as correct by detection methods re- main unaltered by data repair algorithms. Among them, Raha [[47](#_bookmark69)] emerges as a suitable choice, having consistently demonstrated state-of-the-art performance in the conducted tests [[1,](#_bookmark41) [47,](#_bookmark69) [56].](#_bookmark78)

**Optimizations with error detection.** We evaluate the *optimiza- tion ratio* by computing the difference in *EDR* between cases with and without error detection tools. As illustrated in Figure [6,](#_bookmark15) it is evident that the implementation within error detection tools yields substantial improvements for most results. However, as the error rate increases, the benefits provided by error detection tools di- minish. *Rule-driven* methods like Horizon, BigDansing and Holistic

exhibit more improvements involving error detection tools, even surpassing Raha-Baran in certain instances. Conversely, *data-driven* methods see limited gains, primarily due to their stronger initial performance. Moreover, the optimization ratio is higher for *inner* versus *outer* errors. This stems from poorer method performance on *inner* errors, enabling more significant improvements. Moreover, most algorithms manage to decrease or only marginally increase errors handling datasets with *outer* errors.

**Summary.** Here are the summaries of the current section.

* Leveraging current error detection tools can significantly im- prove the data repairing result when the error rate is *low*. Fur- thermore, a high degree of error reduction can be achieved when addressing data with *outer* errors.
* It is highly recommended to combine *rule-driven* algorithms with error detection methods, as this approach could yield results that approach top-tier performance.

## Effects on Downstream Tasks

In this section, we conduct experiments encompassing two preva- lent ML tasks: *Regression* and *Classification*. The experiments are based on the original dirty data, ground truth clean data, and data repaired by selected algorithms. For *classification*, we randomly select one column from each of the previous four datasets as the multi-class target. For *regression*, we choose a numerical column of *Hospital* and *Beers* as the other datasets lack numerical columns. We partition the data into 80% training sets and 20% testing sets, employing *MLP* and *XGBoost* models. Our selection of these two models is driven by two factors. Firstly, both models can effectively handle two aforementioned tasks. Secondly, *MLP* model serves as a foundational component of current deep learning models, while *XGBoost* stands as a prevailing and robust model on tabular data. **Performance on classification.** In Figure [7,](#_bookmark16) we can see that in most cases, classification performance decreases almost linearly with increasing error, without the sharp drop expected. Among all algorithms, Raha-Baran consistently improves classification across most tests, likely due to its superior data repair performance. Datasets repaired by other algorithms, especially *rule-driven* meth- ods excluding Horizon and MLNClean in some cases, lead to worse

1.0

EDR F1 Score

0.75

0.5

0.25

0

0.5

0.0

-1.0

1.0

EDR F1 Score

0.75

0.5

0.25

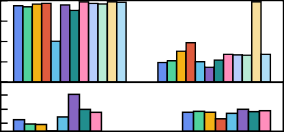
0

0.5

0.0

-1.0



10% 90%

10% 90%

*[]*

10% 90%

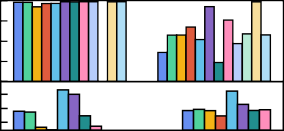
*[]*

10% 90%

*[]*

*Error Rate*

1. *Inner error*, Hospital



10% 90%

10% 90%

*[]*

*Error Rate*

1. *Outer error*, Hospital

1.0

0.75

0.5

0.25

0

0.5

0.0

-1.0

1.0

0.75

0.5

0.25

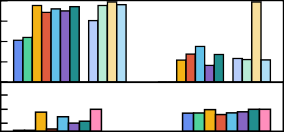
0

0.5

0.0

-1.0

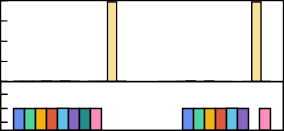
 



10% 90%

*Error Rate*

1. *Inner error*, Flights



x

x

10% 90%

10% 90%

*[]*

*Error Rate*

1. *Outer error*, Flights

1.0

0.75

0.5

0.25

0

0.5

0.0

-1.0

1.0

0.75

0.5

0.25

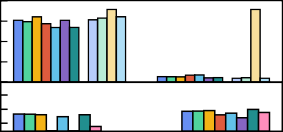
0

0.5

0.0

-1.0



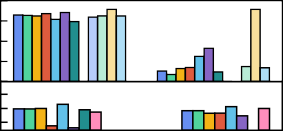
10% 90%

10% 90%

*[]*

*Error Rate*

1. *Inner error*, Beers



x

x

10% 90%

10% 90%

*[]*

*Error Rate*

1. *Outer error*, Beers

1.0

0.75

0.5

0.25

0

0.5

0.0

-1.0

1.0

0.75

0.5

0.25

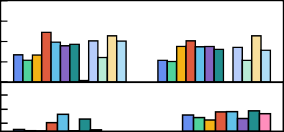
0

0.5

0.0

-1.0

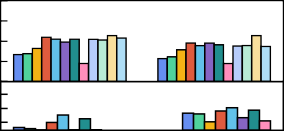
   



10% 90%

*Error Rate*

1. *Inner error*, Rayyan



10% 90%

10% 90%

*[]*

*Error Rate*

1. *Outer error*, Rayyan

### Figure 8: Classification performance within *MLP* under different error types, where *EDR* is presented with exp scale.

**Effects evaluation of error types.** Figure [8](#_bookmark17) shows experiments

1.0

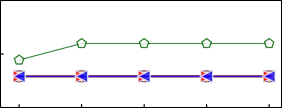
0.5

MSE

0.0

31

×10



10% 30% 50% 70% 90%

*Error Rate*

1. *XGBoost*, Hospital

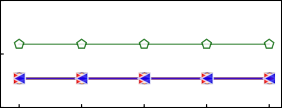
1.0

0.5

0.0

31

×10

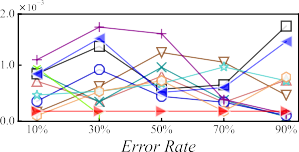
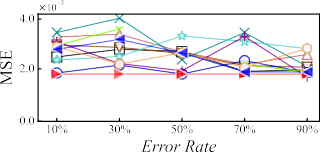


10% 30% 50% 70% 90%

*Error Rate*

1. *XGBoost*, Beers

conducted respectively for inner and outer errors, focusing on the *classification* task due to the minimal influence of error rate on *regression*. We employ *MLP* model due to its relatively better perfor- mance. Across all datasets except *Flights*, *inner* errors have a more negative impact than outer errors, becoming especially prominent at 90% rates. This may be attributed to the distortion of the intrinsic data distribution, thereby complicating the decision-making pro- cess of the model. For *Flights*, *outer* errors lead to extremely low performance. This could be attributed to the high cardinality of the original feature data, where the introduction of additional unknown values substantially increases the feature cardinality, consequently hindering the prediction of final results. Since data-driven methods

1. *MLP*, Hospital (d) *MLP*, Beers

**Figure 9: Regression performance within *XGBoost* and *MLP*** performance than the original datasets. Notably, the *EDR* perfor- mance is not necessarily negatively correlated with the actual down-

stream performance. Despite poor *EDR* results from Horizon, Scare, and MLNClean, they may still *enhance* the classification perfor- mance. In particular, for XGBoost on *Beers*, Scare repaired data outperforms clean data at 10% and 30% error rates. Also, on *Rayyan*, data repaired by MLNClean outperforms clean data at 10%. A pos- sible explanation is that the improved feature-label consistency in the repaired datasets makes it easier for models to learn the rela- tions, although more errors are introduced. Conversely, BoostClean specifically designed for the downstream model performs not as well as expected, even decreasing the performance. Besides, gener- ally *MLP* performs relatively better and more robust than *XGBoost* across almost all the tests.

**Performance on regression.** In the regression task, in Figure [9](#_bookmark18) it is evident that the error rate has little effect. In particular, during the cases employing *XGBoost*, *MSE* exhibits almost no fluctuation across error rates. In the cases of *MLP*, most algorithms also in- crease the overall performance compared with original dirty data. In Figure [9,](#_bookmark18) the performance appears to fluctuate across error rates while the actual changes are minor. Furthermore, similar to the phenomenon in the classification task, certain algorithms enhance performance despite subpar cleaning. Especially on *Beers*, data re- paired by Horizon and BoostClean demonstrate the capability to outperform the clean data.

like Raha-Baran and Unified can more easily repair outer errors, their performance is better. Conversely, the performance on re- paired data within *inner* errors usually falls short of that observed in the original dataset. A possible reason is that the repair process of these methods may further disrupt the data distribution, making it harder for the model to learn feature-label relations.

**Effects evaluation of the optimization strategy.** This part of the experiment primarily affirms the effects on downstream models within error detection optimization. As depicted in Figure [10,](#_bookmark19) most methods exhibit improvements across all tests, outperforming the original dirty data. We can also observe that the optimization of error detection is more effective on data with less than half of the error rates. Among them, *rule-driven* algorithms can approach or even surpass top-tier algorithms with precise error detection tools. This shows the importance of avoiding incorrect changes to correct values. In contrast, *data-driven* methods like Raha-Baran and Scare have fewer improvements, possibly due to their relatively good initial performance. Moreover, it is notable that by leveraging error detection optimization, more repaired data could achieve perfor- mance on par with clean data, especially at low error rates. This is evident in tests of Holistic, BigDansing, Horizon and Scare on *Rayyan* and *Hospital*. Notably, in the tests on the *Hospital* dataset, BoostClean shows increasing performance at higher error rates. This may indicate that simpler repair strategies like using the mean and mode, are more effective when the error rate is high.

**Summary.** Here is the summary of the current section.

1

Opt Rate F1 Score

0.75

0.5

0.25

0

0.5

0

-0.5

1

Opt Rate F1 Score

0.75

0.5

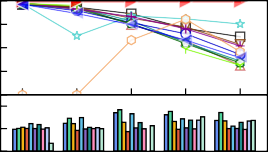
0.25

0

0.5

0

-0.5

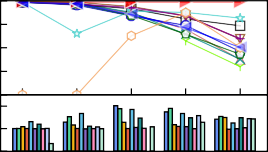


x

10% 30% 50% 70% 90%

*Error Rate*

1. *XGBoost*, Hospital



x

10% 30% 50% 70% 90%

*Error Rate*

1. *MLP*, Hospital

1

0.75

0.5

0.25

0

0.5

0

-0.5

1

0.75

0.5

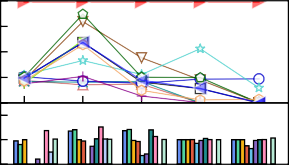
0.25

0

0.5

0

-0.5



x

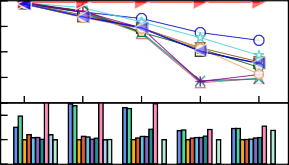
x

x

10% 30% 50% 70% 90%

*Error Rate*

1. *XGBoost*, Flights



x

x

x

10% 30% 50% 70% 90%

*Error Rate*

1. *MLP*, Flights

1

0.75

0.5

0.25

0

0.5

0

-0.5

1

0.75

0.5

0.25

0

0.5

0

-0.5

10% 30% 50% 70% 90%

*Error Rate*

1. *XGBoost*, Beers

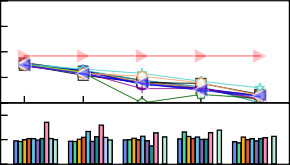
10% 30% 50% 70% 90%

*Error Rate*

1. *MLP*, Beers

1

0.75



x

x

x

0.5

0.25

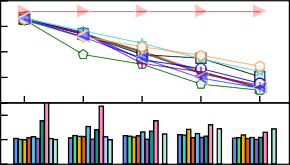
0

0.5

0

-0.5

1



x

x

x

0.75

0.5

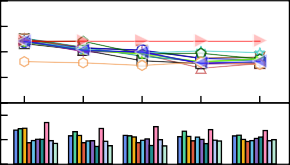
0.25

0

0.5

0

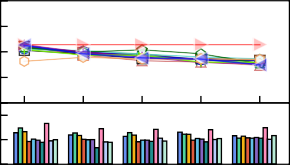
-0.5



10% 30% 50% 70% 90%

*Error Rate*

1. *XGBoost*, Rayyan



10% 30% 50% 70% 90%

*Error Rate*

1. *MLP*, Rayyan

### Figure 10: Classification performance within error detection optimization using *MLP* and *XGBoost* under different error rates.

* The error rate in the data does not necessarily determine the downstream model performance. This highlights the necessity for the exploration of the improved *metrics* to evaluate the data quality according to downstream models.
* It is essential to recognize that completely clean data does not always represent the performance upper bound. In fact, in both classification and regression tasks, the alignment between fea- tures and labels may potentially hold more significance.
* Data is always worth conducting repair. Even when the error rate is as high as 90%, appropriate error repair algorithms can still lead to significant performance enhancements.
* Suitable model choice is significant, especially for data with low error rates. The choice of model potentially may have more effects than the error rate in the data.
* It is also noteworthy error rate of the original data has more influence in the classification task compared to the regression one on the current datasets.

# DISCUSSION

According to the experiments of algorithms and components on real- world and synthetic datasets, we discuss our findings as follows. **Recommendations.** Our recommendations for data repair algo- rithms are based on 5 scenarios considering 3 critical factors: *time limit*, *error reduction*, and *downstream performance*. Recognizing that *time limit* is rarely the sole consideration, we analyze this factor in conjunction with the other two.

*S1: Prioritizing error reduction without time limits.* From previous experiments, we observe that if the primary goal is extensive reduc- tion of the overall data error rate, such as cleaning a database, Baran is the best choice due to its superior capability in error reduction. *S2: Balancing error reduction and time limits.* And if the require- ment for the error reduction is not strict and has limited time, MLNClean and Unified emerge as balanced choices. Their notably efficient processing time and commendable error reduction perfor-

mance make them a good option.

*S3: Prioritizing downstream performance with time limit.* If down- stream performance is emphasized, Scare and MLNClean can be

advantageous choices. In our experiments, we observed that with only inner or outer errors, Scare sometimes struggled to complete repairs within the defined time limits.

*S4: Balancing downstream performance and time limits.* If the time limit is flexible, Raha-Baran remains a conservative choice as it rarely results in worse downstream performance. Especially when the majority of errors in the dataset are of the outer error type, Raha-Baran becomes a much more favorable option. However, it is worth noting that Raha-Baran may encounter challenges when addressing inner errors within the data.

*S5: Balancing downstream performance and error reduction.* In cases where the dual priorities include enhancing both *Downstream performance* and *Error Reduction*, Baran and MLNClean rise as the ideal choices. Their consistent ability to deliver commendable results in both domains establishes it as a fitting selection.

Additionally, if the dataset comprises rules and the primary objective is to resolve data conflicts, BigDansing and Horizon stand as suitable options. However, it is important to acknowledge that their error reduction performance might not consistently meet expectations, additional error detection tools are suggested to be used together to avoid the right values changed into wrong ones. **Guidelines.** We provide guidelines based on the above experiment results from two aspects, *data repair task* and *downstream task*.

*For data repair task.* Intuitively, data repair process contains two stages, namely *error detection* and *error correction*, From the above experiment, we could observe that the performance of *error detection* does not determine the *error correction* result. However, in the practical scenarios, we suggest that data repair algorithms should be applied with error detection methods together, especially when pursuing the error reduction performance using *rule-driven* algorithms. In this way, a large number of cells with right values could avoid being changed into wrong ones, this could significantly improve the *error reduction performance*.

*For downstream tasks.* First of all, data is always worthy of con- ducting repair. Previous experiments underscore that irrespective of the error rate within the data, data repair could consistently lead to significant performance enhancements. Besides, mitigating inner

errors should be placed on special attention. The distortion intro- duced into the inner data distribution magnifies negative impacts, surpassing those incurred by outer errors. Then, model selection is equally important as the data quality, different data models the feature distribution to the label from different perspectives, thus leading to huge differences. Finally, if the data contains more cate- gory text data than the regression data in the regression task, model selection for data repair might be redundant. Given the high influ- ence of the dirty data, common repair methods, such as utilizing mode or mean values, may prove effective.

**Challenges.** At present, most of the data repair methods lies in error rate reduction within datasets, and they are designed to deal with various errors in real scenarios. Within the development of various downstream models, especially the arising of large language model (LLM) [[34](#_bookmark56)], more complex data scenarios and problems are appearing. We summarize the challenges as follows.

*Effective and Efficient Data Repair Algorithms.* Current data repair algorithms fall short of achieving ideal results when dealing with complex scenarios as shown in the experiments. This limitation stems from the fact that the cardinality of the error domain is infinite, whereas there exists only one correct value. Not only local values in the table are important, but the prior distribution of dirty values is also important, which is notoriously difficult to estimate. Efficiency is another problem. Effective algorithms like *Baran* could not deal with datasets containing 20k records within one day, which is considerably lengthy in this era of big data. Thus, there is an urgent need for effective data repair algorithms.

*Exploring Metrics Evaluating Data Quality for Models.* From the experiments, we could observe that the downstream task perfor- mance is not necessarily negatively related to the error rate in the data. Moreover, with the advent of LLMs, issues pertaining to data quality such as hallucination [[48](#_bookmark70)] and chain-of-thought problems [[65](#_bookmark87)] should also be considered. Low-quality data used to train LLMs may introduce unintended biases or false inferences which could propagate through the model’s reasoning chains [[52](#_bookmark74)]. Current data quality metrics like accuracy and completeness are not enough to evaluate the data quality for these tasks. Therefore, considering *metrics* related to data errors can not comprehensively evaluate data quality. This puts an urgent on exploring new metrics to evaluate data quality according to the applied models.

*Data Repair for Downstream Models.* Existing data repair tech- niques aimed at enhancing downstream performance, focus either on error detection [[43](#_bookmark65)] or have limited improvements [[42](#_bookmark64)]. This is due to two reasons. Firstly, current evaluation methods like in- fluence function [[41]](#_bookmark63) can only estimate the influence at *tuple level* while repair for downstream models requires estimation at *cell level*. Secondly, evaluating the influence of unseen value is hard. Since most of the repair candidates for the dirty value are unseen in the repaired tuples. Consequently, data repair for downstream models stands as a hard challenge within the domain of data repair.

**Future Directions.** At present, most of the data repair methods lie in error rate reduction within datasets, within the development of various downstream models, especially the arising of large language model (LLM) [[34](#_bookmark56)], more complex data scenarios and problem are appearing. We summarize the future directions as follows.

*Combining Rule Discovery and Data Information for Data Repair.* It is noteworthy that current state-of-the-art algorithms, such as Baran [[46](#_bookmark68)] and HoloClean [[58](#_bookmark80)], employ data information in the data repair process. These methods harness the strong represen- tation capabilities of ML models, enabling them to capture data distribution information and assess the reliability of potential candi- dates with rich features, which are beyond the scope of traditional methods. However, these algorithms are easily influenced by unre- lated values, since values in the data are usually determined by few cells, which is the strength of rules. While in the big data era with high human cost, efficiently rule discovering is also an important task [[49](#_bookmark71)]. Therefore, combining rule discovery and data information for data repair is a future trend for data repair.

*LLM for Data Repair.* Typically, current data repair algorithms can only generate candidates for error cells based on the values in the data, specific string transfer rules, or certain knowledge bases [[15](#_bookmark32), [21](#_bookmark38), [46](#_bookmark68)]. However, when confronting data within less redundancy, it is hard for current methods to generate candidates outside the limited provided information, therefore current Large Language Model (LLM) containing a large range of knowledge is a natural choice. By carefully designing the prompt [[64](#_bookmark86)] fed into LLM with comprehensive data context and rules information, LLM can generate repair candidates that are both grammatically valid and semantically meaningful. Compared to current data repair methods, LLM can generate candidates beyond the limited information in the data, while properly considering the data semantics and structures. *Fairness-Aware Data Repair.* Previous research has demonstrated that automatic data cleaning can potentially hurt fairness in ma- chine machine learning-based decision systems [[27](#_bookmark49)]. However, ex- isting data repair methods do not consider fairness issues during the repair process. Consequently, developing fairness-aware data repair techniques emerges as a critical research direction to ensure repaired data does not propagate unfair model predictions. This requires jointly optimizing data quality and fairness metrics when generating repair candidates. Incorporating techniques from fair machine learning, including adversarial debiasing and constrained optimization, into data repair frameworks presents a promising

approach to imbue repaired data with notions of fairness.

# RELATED WORK

**Error Detection**. As the prior step in data cleaning process [[13](#_bookmark30)], error detection involves two work lines, i.e., *rule-driven* and *data- driven* error detection. *Rule-based error detection algorithms* assume the data must adhere to a predefined set of rules.

Error detection algorithms identify conflicting values based on the given rules like FDs [[3](#_bookmark43), [8](#_bookmark25), [59](#_bookmark81)], DCs [[14](#_bookmark31), [24](#_bookmark46), [26](#_bookmark48), [58](#_bookmark80)], conditional

functional dependencies [[19](#_bookmark36)], user-defined functions [[5](#_bookmark45), [17](#_bookmark34)], and

manually defined parameters [[6,](#_bookmark24) [57].](#_bookmark79)

In contrast, *data-driven error detection algorithms* identify errors based on (weak) labeled data without relying on predefined rules. These algorithms can be categorized into supervised and unsu- pervised ones. Supervised algorithms [[32](#_bookmark54), [47](#_bookmark69), [51](#_bookmark73), [56](#_bookmark78)] use (a few) labeled data. For unsupervised methods, statistical hypothesis [[63](#_bookmark85)], co-occurrence dependency [[35](#_bookmark57)], and gradient of the downstream model [[43]](#_bookmark65) are usually employed to detect errors.

**Error Repair.** Error repair is the subsequent correction process in data cleaning. These algorithms typically employ internal and ex- ternal information. Internally, they utilize integrity constraints [[14](#_bookmark31), [25](#_bookmark47), [28](#_bookmark50), [58](#_bookmark80), [59](#_bookmark81)] and the distribution of data itself [[46](#_bookmark68), [58](#_bookmark80), [62](#_bookmark84), [67](#_bookmark89)]. Externally, they incorporate information from master data [[21](#_bookmark38)], knowledge base [[15]](#_bookmark32) and downstream data analysis models [[42].](#_bookmark64)

Besides, some methods perform data repair for downstream models. They generally delete noisy data at the tuple level, like cleansing based on SGD [[29](#_bookmark51)]. While we focus on repairing incorrect values to correct ones without deletion operations.

**Deduplication.** Deduplication, also known as entity resolution or entity matching, identifies records that refer to the same real- world entity [[16](#_bookmark33)], which is an essential operation in data prepa- ration at *tuple* level. It can be categorized into *rule-driven* and *data-driven* methods. *Rule-driven* methods leverage various meta- information [[22](#_bookmark39), [23](#_bookmark40)], attribute similarities [[31](#_bookmark53)], and entity matching rules [[20](#_bookmark37), [61](#_bookmark83)] to deduplicate the original data. As for *data-driven* algorithms, they either employ ML models trained on labeled data to perform deduplication [[18](#_bookmark35), [45](#_bookmark67), [50](#_bookmark72), [53](#_bookmark75)] or rely on general-purpose clustering algorithms like Gaussian mixture models [[66].](#_bookmark88)

# CONCLUSIONS

In this paper, we consider 12 representative data repair algorithms based on a novel taxonomy. Subsequently, we conduct a compre- hensive evaluation and discussion of the performance of all these algorithms on four real-world datasets and one synthetic dataset in a unified framework. Additionally, we assess both the analysis of data repair performance under complex scenarios and the effects on downstream models. We also provide optimization strategies

1. Laure Berti-Equille, Tamraparni Dasu, and Divesh Srivastava. 2011. Discovery of complex glitch patterns: A novel approach to quantitative data cleaning. In *ICDE*. 733–744.
2. George Beskales, Ihab F. Ilyas, and Lukasz Golab. 2010. Sampling the repairs of functional dependency violations under hard constraints. *Proceedings of the* *VLDB Endowment* 3, 1–2 (2010), 197–207.
3. George Beskales, Ihab F. Ilyas, Lukasz Golab, and Artur Galiullin. 2013. On the relative trust between inconsistent data and inaccurate constraints. In *ICDE*. 541–552.
4. BigDaMa. [n.d.]. error-generator. [https://github.com/BigDaMa/error-generator.](https://github.com/BigDaMa/error-generator) Accessed April 18, 2023.
5. Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. 2017. Efficient denial constraint discovery with hydra. *Proceedings of the VLDB Endowment* 11, 3 (2017), 311–323.
6. Ali Borji. 2023. Generated Faces in the Wild: Quantitative Comparison of Stable Diffusion, Midjourney and DALL-E 2. *ArXiv Preprint ArXiv:2210.00586* (2023).
7. Fei Chiang and Renée J. Miller. 2011. A unified model for data and constraint repair. In *ICDE*. 446–457.
8. Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data cleaning: Overview and emerging challenges. In *SIGMOD*. 2201–2206.
9. Xu Chu, I. F. Ilyas, and P. Papotti. 2013. Holistic data cleaning: Putting violations into context. In *ICDE*. 458–469.
10. Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*. 1247–1261.
11. Xin Luna Dong and Theodoros Rekatsinas. 2018. Data integration and machine learning: A natural synergy. In *SIGMOD*. 1645–1650.
12. Amr Ebaid, Ahmed Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, Jorge-Arnulfo Quiane-Ruiz, Nan Tang, and Si Yin. 2013. NADEEF: A generalized data cleaning system. *Proceedings of the VLDB Endowment* 6, 12 (2013), 1218–1221.
13. Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1454–1467.
14. Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2008. Con- ditional functional dependencies for capturing data inconsistencies. *ACM Trans-* *actions on Database Systems* 33, 2, Article 6 (2008), 48 pages.
15. Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. 2009. Reasoning about record matching rules. *Proceedings of the VLDB Endowment* 2, 1 (2009), 407–418.
16. Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Wenyuan Yu. 2012. Towards certain fixes with editing rules and master data. *The VLDB Journal* 21, 2 (2012),

based on current technologies. Besides, we provide some useful rec- ommendations about promising research directions and insightful

213–238.

1. Ivan P. Fellegi and Alan B. Sunter. 1969. A theory for record linkage.

*Statist. Assoc.* 64, 328 (1969), 1183–1210.

*J. Amer.*

principles for algorithm optimizations.

Finally, it is vital to note that, our study exclusively considers core algorithms based on main memory. As we move forward, we plan to explore various optimizations and leverage recent technologies to further enhance the overall performance of data repair.

# ACKNOWLEDGMENTS

This work was supported by the [...] Research Fund of [...] (Number [...]). Additional funding was provided by [...] and [...]. We also thank [...] for contributing [...].

# REFERENCES

1. Mohamed Abdelaal, Christian Hammacher, and Harald Schöning. 2023. REIN: A comprehensive benchmark framework for data cleaning methods in ML pipelines. In *EDBT*. 499–511.
2. Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. De- tecting data errors: Where are we and what needs to be done? *Proceedings of the* *VLDB Endowment* 9, 12 (2016), 993–1004.
3. Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases:* *The Logical Level*. Addison-Wesley Longman Publishing Co., Inc.
4. Patricia C. Arocena, Boris Glavic, Giansalvatore Mecca, Renée J. Miller, Paolo Papotti, and Donatello Santoro. 2015. Messing up with BART: Error generation for evaluating data-cleaning algorithms. *Proceedings of the VLDB Endowment* 9, 2 (2015), 36–47.
5. Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Eui- jong Whang, and Jennifer Widom. 2009. Swoosh: A generic approach to entity resolution. *The VLDB Journal* 18, 1 (2009), 255–276.
6. Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, and Cristian-

Augustin Saita. 2001. Declarative data cleaning: Language, model, and algorithms.

*Proceedings of the VLDB Endowment*, 371–380.

1. Congcong Ge, Yunjun Gao, Xiaoye Miao, Bin Yao, and Haobo Wang. 2022. A hybrid data cleaning framework using markov logic networks. *IEEE Transactions* *on Knowledge and Data Engineering* 34, 5 (2022), 2048–2062.
2. Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC data-cleaning framework. *Proceedings of the VLDB Endowment* 6, 9 (2013), 625–636.
3. Stella Giannakopoulou, Manos Karpathiotakis, and Anastasia Ailamaki. 2020. Cleaning denial constraint violations through relaxation. In *SIGMOD*. 805–815.
4. Shubha Guha, Falaah Arif Khan, Julia Stoyanovich, and Sebastian Schelter. 2023. Automated data cleaning can hurt fairness in machine learning-based decision making. In *ICDE*. 3747–3754.
5. Shuang Hao, Nan Tang, Guoliang Li, Jian He, Na Ta, and Jianhua Feng. 2017. A novel cost-based model for data repairing. *IEEE Transactions on Knowledge and* *Data Engineering* 29, 4 (2017), 727–742.
6. Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. 2019. *Data cleansing for* *models trained with SGD*.
7. Md Kamrul Hasan and Mohammad Mahdavi. 2021. Automatic Error Correction Using the Wikipedia Page Revision History. In *CIKM*. 3073–3077.
8. Oktie Hassanzadeh, Fei Chiang, Hyun Chul Lee, and Renée J. Miller. 2009. Frame- work for evaluating clustering algorithms in duplicate detection. *Proceedings of* *the VLDB Endowment* 2, 1 (2009), 1282–1293.
9. Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. 2019. HoloDetect: Few-Shot Learning for Error Detection. In *SIGMOD*. 829–846.
10. Yue Hu, Yanbing Wang, Canwen Jiao, Rajesh Sankaran, Charles E Catlett, and Daniel B Work. 2019. Automatic data cleaning via tensor factorization for large urban environmental sensor networks. In *Proceedings of the NeurIPS Workshop* *on Tackling Climate Change with Machine Learning*.
11. Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. *ArXiv Preprint ArXiv:2212.10403* (2023).
12. Zhipeng Huang and Yeye He. 2018. Auto-Detect: Data-Driven Error Detection in Tables. In *SIGMOD*. 1377–1392.
13. Kadir Ider and Andreas Schmietendorf. 2018. Data privacy for AI fraud detection models. (2018), 102–107.
14. Ihab F. Ilyas and Xu Chu. 2015. Trends in cleaning relational data: consistency and deduplication. *Foundations and Trends in Databases* 5, 4 (2015), 281–393.
15. Ihab F. Ilyas and Xu Chu. 2019. *Data cleaning*. Association for Computing Machinery.
16. Deepa Seetharaman Karen Hao. 2023. Cleaning Up ChatGPT Takes Heavy Toll on Human Workers. [https://www.wsj.com/articles/chatgpt-openai-content-](https://www.wsj.com/articles/chatgpt-openai-content-abusive-sexually-explicit-harassment-kenya-workers-on-human-workers-cf191483) [abusive-sexually-explicit-harassment-kenya-workers-on-human-workers-](https://www.wsj.com/articles/chatgpt-openai-content-abusive-sexually-explicit-harassment-kenya-workers-on-human-workers-cf191483) [cf191483.](https://www.wsj.com/articles/chatgpt-openai-content-abusive-sexually-explicit-harassment-kenya-workers-on-human-workers-cf191483)
17. Zuhair Khayyat, Ihab F. Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. 2015. BigDans- ing: A system for big data cleansing. In *SIGMOD*. 1215–1230.
18. Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*. 1885–1894.
19. Sanjay Krishnan, Michael J. Franklin, Ken Goldberg, and Eugene Wu. 2017. BoostClean: Automated error detection and repair for machine learning. *ArXiv* *Preprint ArXiv:1711.01299* (2017).
20. Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Gold- berg. 2016. ActiveClean: Interactive data cleaning for statistical modeling. *Pro-* *ceedings of the VLDB Endowment* 9, 12 (2016), 948–959.
21. Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. 2021. CleanML: A study for evaluating the impact of data cleaning on ML classification tasks. In *ICDE*. 13–24.
22. Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proceedings of* *the VLDB Endowment* 14, 1 (2020), 50–60.
23. Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error cor- rection via a unified context representation and tansfer learning. *Proceedings of* *the VLDB Endowment* 13, 12 (aug 2020), 1948–1961.
24. Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Mad- den, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A configuration-free error detection system. In *SIGMOD*. 865–882.
25. Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero- resource black-box hallucination detection for generative large language models. *ArXiv Preprint arXiv:2303.08896* (2023).
26. Yinan Mei, Shaoxu Song, Chenguang Fang, Ziheng Wei, Jingyun Fang, and Jiang Long. 2023. Discovering editing rules by deep reinforcement learning. In *ICDE*. 355–367.
27. Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *SIGMOD*. 19–34.
28. Felix Neutatz, Mohammad Mahdavi, and Ziawasch Abedjan. 2019. ED2: A case for active learning in error detection. In *CIKM (CIKM ’19)*. 2249–2252.
29. Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv Preprint* *ArXiv:2203.02155* (2022).
30. Matteo Paganelli, Francesco Del Buono, Andrea Baraldi, and Francesco Guerra. 2022. Analyzing how BERT performs entity matching. *Proceedings of the VLDB* *Endowment* 15, 8 (2022), 1726–1738.
31. Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Func- tional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment* 8, 10 (2015), 1082–1093.
32. Eduardo H. M. Pena, Eduardo C. de Almeida, and Felix Naumann. 2019. Dis- covery of approximate (and exact) denial constraints. *Proceedings of the VLDB* *Endowment* 13, 3 (2019), 266–278.
33. Minh Pham, Craig A. Knoblock, Muhao Chen, Binh Vu, and Jay Pujara. 2021. SPADE: A semi-supervised probabilistic approach for detecting errors in tables. In *IJCAI*. 3543–3551.
34. Clément Pit-Claudel, Zelda E. Mariet, Rachael Harding, and Samuel Madden. 2016. Outlier detection in heterogeneous datasets using automatic tuple expansion. (2016).
35. Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. Holo- Clean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB* *Endowment* 10, 11 (2017), 1190–1201.
36. El Kindi Rezig, Mourad Ouzzani, Walid G. Aref, Ahmed K. Elmagarmid, Ahmed R. Mahmood, and Michael Stonebraker. 2021. Horizon: Scalable dependency-driven data cleaning. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2546–2554.
37. Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. 2015. Incremental knowledge base construction using deepdive. *Proceedings* *of the VLDB Endowment* 8, 11 (2015), 1310–1321.
38. Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama,

and Nan Tang. 2017. Generating concise entity matching rules. In *SIGMOD*. 1635–1638.

1. Shaoxu Song, Han Zhu, and Jianmin Wang. 2016. Constraint-Variance Tolerant Data Repairing. In *SIGMOD*. 877–892.
2. Pei Wang and Yeye He. 2019. Uni-detect: A unified approach to automated error detection in tables. In *SIGMOD*. 811–828.
3. Zijie J Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. 2022. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models. *ArXiv Preprint arXiv:2210.14896* (2022).
4. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reason- ing in large language models. *NIPS* (2022), 24824–24837.
5. Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumu- ruganathan. 2020. ZeroER: Entity resolution using zero labeled examples. In *SIGMOD*. 1149–1164.
6. Mohamed Yakout, Laure Berti-Équille, and Ahmed K. Elmagarmid. 2013. Don’t be scared: Use scalable automatic repairing with maximal likelihood and bounded changes. In *SIGMOD*. 553–564.
7. Haojun Zhang, Chengliang Chai, AnHai Doan, Paris Koutris, and Esteban Ar- caute. 2020. Manually detecting errors for data cleaning using adaptive crowd- sourcing strategies. In *EDBT*. 311–322.