Automatic Data Repair: Are We Ready to Deploy?

Wei Ni[*‡1], Xiaoye Miao[*★2], Xiangyu Zhao[‡3], Yangyang Wu[#4], Shuwei Liang[*#5], Jianwei Yin[*#†6]
[*]Center for Data Science, Zhejiang University, Hangzhou, China
[★]The State Key Lab of Brain-Machine Intelligence, 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
{[1]wei.ni, [2]miaoxy, [4]zjuwuyy, [5]lsw5221}@zju.edu.cn [3]xianzhao@cityu.edu.hk [6]zjuyjw@cs.zju.edu.cn

ABSTRACT

Data quality is paramount in today's data-driven world, especially in the era of generative AI. Dirty data with errors and inconsistencies 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 importance. Existing data repair algorithms differ in information utilization, problem settings, and are tested in limited scenarios. In this paper, we compare and summarize these algorithms with a driven information-based taxonomy. We systematically conduct a comprehensive evaluation of 12 mainstream data repair algorithms on 12 datasets under the settings of various data error rates, error types, and 4 downstream analysis tasks, assessing their error reduction performance with a *novel but practical* metric. We develop an effective and unified repair optimization strategy that substantially benefits the state of the arts. We conclude that, it is always worthy of data repair. The clean data does not determine the upper bound of data analysis performance. We provide valuable guidelines, challenges, and promising directions in the data repair domain. We anticipate this paper enabling researchers and users to well understand and deploy data repair algorithms in practice.

PVLDB Reference Format:
Wei Ni, Xiaoye Miao, Xiangyu Zhao, Yangyang Wu, Shuwei Liang, Jianwei Yin. Automatic Data Repair: Are We Ready to Deploy? . PVLDB, 17(10): 2617 - 2630, 2024. doi:10.14778/3675034.3675051

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/WelkinNi/Automatic-Data-Repair.

1 INTRODUCTION

The global application landscape relies heavily on data, while ubiquitously erroneous information involved in data often compromises the data analysis performance, leading to a huge economic cost [17, 43, 49, 84]. Dirty data may trigger the spread of fraud information, inaccurate decision-making, and even legal repercussions [42, 60, 62, 79, 86]. Among the strategies for efficiently mitigating erroneous information and ensuring data quality, *automatic data cleaning* [39, 86] 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 ChatGPT [67] and Midjourney [11] currently often resort to *resource-intensive manual* data cleaning [87].

The promising automatic data cleaning task is generally conducted 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, 56, 57]. Existing error detection studies [38, 57, 65, 71] have achieved obvious advancements, with an average F1 score exceeding 0.85 on real datasets [57, 71]. In contrast, the error repair is more complex and challenging. Various techniques for error repair have been employed, such as machine learning [56, 72, 85], transfer learning [36, 56], boosting algorithm [47] and few-shot learning [56]. However, within the advanced detection results, the *final* data repair performance of current methods still falls short of the expected ideal, with an average F1 score below 0.7 in real scenarios [32, 56, 72]. Thus, it is crucial and urgent to analyze and explore how to deploy *end-to-end* data cleaning approaches with effective *data repair* algorithms in real-life scenarios.

Table 1 compares and summarizes existing surveys of end-to-end data cleaning study. They either analyze the data repair process without experiments [17, 43, 44, 48], or verify the impact of data repair on downstream tasks while overlooking the inclusion of certain important algorithms [1, 51]. The limitations and analysis perspectives come from the following *three* aspects.

L1: Metric shortfalls in evaluation. Fewer data errors typically lead to higher data quality and reduced biases in subsequent analyses, which constitute the *main goals* of the data cleaning task. Prior data cleaning surveys [1, 43, 44, 48, 51] lack an evaluation of *error reduction* in final data repair results, as indicated in Table 1. They often evaluate the data repair results with metrics such as *precision* and *recall*, which are based on the proportion of correctly repaired cells, disregarding the absolute quantity changes. Thus, such metrics result in a *biased* evaluation. For example, suppose there is a dataset that initially encompasses 10 erroneous cells. One data cleaning approach repairs 100 cells, of which 80 are correct (including the initial 10 error cells). Despite 70 of 80 values being identical to their originals, they are still flagged as erroneous values, and replaced by a value within calculation, thereby should be considered as a repaired cell. It means that, the dataset currently contains

Proceedings of the VLDB Endowment, Vol. 17, No. 10 ISSN 2150-8097.
doi:10.14778/3675034.3675051

Table 1: Comparison of previous surveys and ours.

Survey study	Exp. evaluation	Error reduc. evaluation	Error rate		Error type	
			DRE*	DME*	DRE*	DME*
Ilyas et al. [43, 44]	✗	✗	✗	✗	✗	✗
Krishnan et al. [48]	✗	✗	✗	✗	✗	✗
Chu et al. [17]	✗	✗	✗	✗	✗	✗
Li et al. [51]	✓	✗	✗	✗	✗	✓
Abdelaal et al. [1]	✓	✗	✓	✗	✗	✗
Ours	✓	✓	✓	✓	✓	✓

* DRE and DME refer to data repair and downstream model evaluation.

20 erroneous cells, indicating a *doubles* amount of erroneous cells after the repair process. While the precision and recall of this repair both reach as high as 0.8 and 1.0, respectively. The correct repair of more initially right data increases the precision, according to the mediant inequality, but it cannot benefit the data quality. As a result, a higher precision/recall value does not represent a larger degree of error reduction. Furthermore, for distance-based metrics like Jaccard distance or mean squared error (MSE), they only measure the distance between the repaired and clean data, while *ignoring* the distance between the original dirty data and clean data. In a situation where the initial distance between dirty and clean data is 0.1, which then increases to 0.15 after repair. Simply stating the distance as 0.15 fails to directly indicate an increase in data errors. It cannot convey the actual *improvement degree* in data quality, either. Incorporating error reduction degree into evaluations will align repair algorithms more closely with improving data quality, and offer a direct perspective on data quality changes.

L2: Insufficient scenarios exploration in evaluation. The effectiveness of data repair algorithms differs in different scenarios. As depicted in Table 1, existing surveys lack the study of data repair algorithms' capacity under various *error types*, in either the data repair performance evaluation or the assessments of its impact on downstream tasks [1, 17, 44, 48, 51]. Meanwhile, the *error rate* analysis is absent in downstream impact evaluation, even though it is included in repair performance assessments. It results in an inadequate understanding of the proper deployment specific to distinct error types and error rates. It is vital and practical to comprehensively evaluate data repair algorithms with a diverse range of error types and rates in both repair and downstream tasks.

L3: Lacking algorithm deployment guidelines. Blindly deploying data repair algorithms may potentially hurt data quality and task performance. Although existing surveys discuss data repair algorithm performance [17, 43, 44, 48], *three* critical questions regarding algorithm deployment remain unanswered: (i) Is data repair consistently beneficial irrespective of the error rate? (ii) Based on the observation that, the suboptimal performance of data repair primarily results from incorrect repairs, how to develop a practical strategy to mitigate incorrect repairs based on error reduction evaluation? (iii) How to select the proper repair method in different scenarios based on task objectives? The data repair process plays a crucial role in the cleaning task, and addressing these issues will provide practical guidance to deploy data cleaning algorithms.

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 systematically evaluate *twelve* mainstream and state-of-the-art data repair algorithms based on a driven information-based taxonomy of data repair algorithms, including *cstr-driven* data repair algorithms, *data-driven* ones, and *hybrid-driven* ones. We also state the properly applied scenarios and graphically represent the workflow within each category.
- We define a novel and effective *metric* named *Error Drop Rate* to evaluate the error reduction condition of final data repair results. Surprisingly, in the majority of cases, most data repair algorithms tend to introduce more errors rather than eliminate erroneous information. We propose a *universal* optimization strategy that leverages error detection techniques to prevent the alteration of original right values into wrong ones.
- Extensive experiments on *twelve* real datasets under various error rates/types and data scales demonstrate both the performance of *twelve* data repair algorithms and the impact on *four* common downstream tasks. Particularly, except for five algorithms with publicly available codes, we implement *seven* additional data repair algorithms for comprehensive comparisons. We investigate the effect of *semantic* and *syntactic* data errors (that simulate complex practical scenarios) on the algorithm performance.
- We conclude several interesting findings: *i)* Repair algorithms often *increase* errors. Contrary to expectations, most existing data repair algorithms tend to raise the error rate in the data rather than eliminate it; *ii)* Using the clean data is *not* always the best. Downstream analysis models trained on purely clean data may perform worse than models trained using dirty data. *iii)* Data repair could provide universal benefits for downstream tasks. With proper algorithm selection, data repair boosts downstream tasks in the vast majority of cases; *iv) Semantic* errors pose a more stubborn challenge. Semantic errors within the data are harder to eliminate and have a more negative impact on downstream data analysis than *syntactic* ones, highlighting the need for greater consideration in real-world applications.

In the rest of this paper, Section 2 presents the preliminaries, while the new taxonomy and algorithm analysis are covered in Section 3. Comprehensive evaluations are offered in Section 4. Section 5 discusses current challenges and future directions, followed by related work in Section 6. Finally, Section 7 concludes the study.

2 PRELIMINARIES

In this section, we first introduce the problem definition. We then describe the constraints widely used in automatic repair algorithms.

Data cleaning mainly serves two primary goals: eliminating errors and ensuring data consistency, with the latter being a subgoal of the former. In this survey, we aim to evaluate *error reduction* performance over the final data repair results, as stated below.

Definition 1. **Problem Statement.** Given an instance I of relation R characterized by a set of attributes $Attrs = \{A_1, A_2, \cdots, A_n\}$. Each $t \in I$ comprises a set of cells represented by $Cells[t] = A_i[t]$ corresponding to a distinct attribute in $Attrs$. For each cell c, its unknown true value is denoted by v_c^*, its initial observed value is v_c, and its estimated true value is $\hat{v}_c$. A data repair algorithm aims to make $\hat{v}_c$ equal to v_c^* for all cells c.

This error reduction evaluation defined in *Definition* 1 encompasses correcting all erroneous cells within a dataset. It shares the same definition of *holistic repair*, as studied in most recently proposed data repair algorithms [56, 57, 71, 72].

Expert experience and field knowledge of the data serve as a critical guideline in the data repair process. These guidelines manifest as rules and constraints (cstrs). Rules provide explicit instructions for modifying erroneous values, making data repair a deterministic process [26, 34]. 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 rules in the data repair process [34]. In contrast, constraints are designed to capture the relationships between specific attributes or values. Constraints have a *wider* range of applications, since professional external information may not always be accessible.

Numerous automatic data repair algorithms are built based on constraints like *functional dependency* (FD) [69] and *denial constraint* (DC) [70], as stated below.

Definition 2. **Functional Dependency.** A functional dependency is a statement of the form $X \rightarrow A$ where $X \subseteq Attrs$ and $A \in Attrs$, denoting that the values of attribute set X uniquely determine the values of attribute A across all tuples in I. X and A are the left-hand and right-hand attributes, respectively.

Definition 3. **Denial Constraint.** A denial constraint φ is a statement of the form $\forall t_\alpha, t_\beta \in I : \neg (p_1 \wedge \cdots \wedge p_m)$ where p_i is in the following form: $t_\alpha.A_x \ \phi \ t_\beta.A_y$ or $t_\alpha.A_x \ \phi \ c$, $A_x, A_y \in Attrs$, c is a constant, and ϕ is a built-in operator, i.e., $\neq$, $=$, $\leq$, $<$, $\geq$, $>$.

Example 1. A sample instance from real-world tax data [4] is presented in Table 2, along with two real FDs and two DCs:

$F_1 : City \rightarrow State$,

$DC_1 : \forall t_1, t_2 \in I : \neg(t_1.City = t_2.City \wedge t_1.State \neq t_2.State)$,

$DC_2 : \forall t_1, t_2 \in I : \neg(t_1.City = t_2.City \wedge t_1.Salary \geq t_2.Salary \wedge t_1.Rate \leq t_2.Rate)$.

The FD F_1 means if the values of $City$ are the same, then the values of $State$ should also be the same. Meanwhile, DC_1 asserts that no two tuples in R can have identical $City$ values paired with differing $State$ values. DC_2 indicates that within a given $City$, a higher $Salary$ should not correspond to a lower $Rate$ of tax. In this case, F_1 and DC_1 are equivalent, expressing the same meaning. The erroneous cells violating these constraints are colored in gray.

An FD in the above form $X \rightarrow A$ can be equally expressed in the following DC format: $\forall t_\alpha, t_\beta \in I, \neg(t_\alpha.X_0 = t_\beta.X_0 \wedge \cdots \wedge t_\alpha.X_k = t_\beta.X_k \wedge t_\alpha.A \neq t_\beta.A), X_0, \cdots, X_k \in X$. While DC has greater expressive ability than FD. For instance, the monotonic relationship between values in DC_2 cannot be captured by FD, underscoring the expressivity gap between them. Considering the expressive ability and broader application spectrum, we mainly focus on DC-based data repair algorithms when evaluating the repair solutions adopting constraints.

Based on constraints, to achieve data consistency, previous work [8, 18] also offers other repair definitions, which are significant in algorithm understanding and deployment process:

Definition 4. **Tolerant Repair.** Given a set of constraints Σ over a relation schema R, along with a set of modified constraints Σ', and two instances I and I' of R. (Σ', I') is a tolerant repair of I if I' adheres to the modified constraint set Σ'.

Definition 5. **Consistency Repair.** Given a set of constraints Σ over a relation R, and two instances I and I' of R. I' is a consistency repair of I if I' adheres to the constraint set Σ.

Table 2: Relation of tax.

Tuple	FirstName	LastName	Gender	City	State	Salary	Rate
t_1	Weiming	Posthoff	Female	Vera	Okla.	45,000	6.25
t_2	Weiming	Zongtian	Male	Okemos	Mich.	10,000	3.9
t_3	Shivraj	Alpin	Female	Eastham	Mass.	90,000	5.3
t_4	Yurdaer	Thackray	Female	Eastham	LA.	30,000	5.2

3 DATA REPAIR ALGORITHMS

In this section, we elaborate 12 mainstream data repair algorithms based on a new *information-driven* taxonomy. We present a general and effective strategy to optimize these data repair algorithms.

3.1 Algorithm Taxonomy

Automatic data repair, which typically relies on data and/or constraint information, aims to enhance data quality or improve the performance of downstream tasks.

To offer insights into how they utilize constraints and data according to the driven information, we categorize existing data repair algorithms into three groups, i.e., *constraint(cstr)-driven*, *data-driven*, and *hybrid-driven* methods. The *cstr-driven* methods repair data based on constraints, typically employing *equivalent classes* (which are sets of cells that should satisfy the same constraint conditions). The *data-driven* algorithms only utilize data distribution information (without constraints) to repair data. For *hybrid-driven* algorithms, except for constraints, they also leverage data information not covered in constraints to facilitate data repair.

This taxonomy is useful for providing clear guidelines on using constraints and data, and offering targeted methods for different error types (cstr-driven for semantic, data-driven for syntactic, and hybrid for general errors). It also aligns with varying needs in deployment. Cstr-driven methods for regulated scenarios like healthcare and finance, and adaptable data-driven or hybrid methods for fields like machine learning tasks.

Moreover, there are *three* primary repair objectives within these algorithms, namely *data consistency*, *holistic repair*, and *model performance boost*. Data consistency encompasses *consistency repair*, which is applied to data with strict constraints like, geographic data where city names rely on states, and *tolerant repair*, used when constraints are either outdated or incorrect. *Holistic repair* focuses on maximizing error removal, crucial in fields like cybersecurity or healthcare. Lastly, the aim of *model performance boost* is to refine training data to enhance outcomes of data-driven tasks, such as image classification. Regarding these objectives, cstr-driven algorithms universally aim to achieve consistency repair. Data-driven approaches are for holistic repair and enhancing model performance. While the hybrid-driven methods are oriented toward tolerant repair or holistic repair.

Guided by the taxonomy, we select methods based on the utilization of constraints and data. In *cstr-driven* methods, we examine widely cited Holistic [18] and Nadeef [22] for mainstream graph-based and statistic-based data modeling with DCs. For *data-driven* ones, methods predominantly encompass statistical boosting and machine learning (ML)-based techniques, with BoostClean [47] as the only example for the former, Baran [56] and Scare [85] pioneered for ML applications in data repair. *Hybrid-driven* tools like Relative [8] and Unified [16] are representative in heuristic-based methods, while HoloClean [72] merges constraints with data

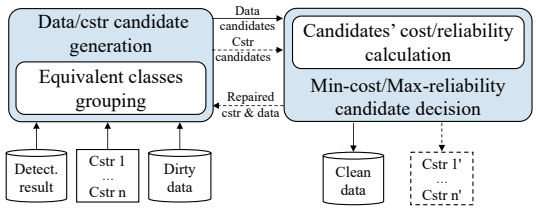


Figure 1: Workflow of *cstr/hybrid-driven* repair algorithms.

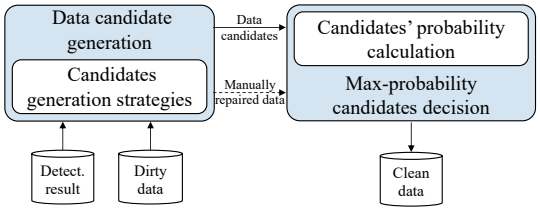


Figure 2: Workflow of *data-driven* repair algorithms.

using ML models. Then from a scalability viewpoint, *cstr-driven* techniques face time cost challenges; BigDaning [45] alleviate this with methods to reduce redundancy and group data. Horizon [73] proposes a fresh strategy to reduce time costs using FDs; meanwhile, Daisy [32] showcases data repair's practicality for query tasks. Lastly, MLNClean [30] introduces a novel weighted constraint application using Markov logic networks, showcasing enhanced performance and novel constraint application methods.

3.2 Algorithm Description

Each data repair algorithm falls into one of the three categories. We summarize the workflow of each group of data repair algorithms, and describe the representative algorithms belonging to the group.

3.2.1 Cstr-driven Data Repair Algorithms. Figure 1 depicts the workflow of *cstr-driven* algorithms (not including dashed arrows and modified constraints in the figure). It takes a dirty dataset with constraints and the *constraints violation* detection results as inputs. It generally includes two phases to get the clean data: getting the possible correct candidates for *constraint violation* cells and repairing the data based on cost minimization or reliability maximization.

Specifically, data candidate generation of constraint violation cells is based on *equivalent classes*, which shows that a set of attribute values should satisfy the same condition. For example, in Table 2, the $State$ value of t_3 and t_4 should be identical due to the same value of $City$ based on F_2. Thus, $(t_3.City, t_3.State)$ and $(t_4.City, t_4.State)$ form an equivalent class, indicating that their values *should* be identical. All values in the equivalent classes could be the data repair candidates.

In addition, the choice of candidates for repair is determined by candidates within minimal calculated costs like edit distance [34] and cardinality [18, 22], or maximal reliability metrics such as frequency [32] and overall data support degree [73].

Holistic [18]. Holistic is a classical repair algorithm addressing DC violations. When generating data candidates, it encodes all equivalent classes in a *conflict hypergraph*, where nodes and hyperedges represent violation cells and associated constraints, respectively. By analyzing nodes' interactions, Holistic generates a variety of repair candidates. To determine candidates, it first calculates the applied costs of all candidates, and then heuristics are employed to achieve cost minimum and data consistency. Due to the necessity of comparing tuple pairs to find all equivalent classes, the time complexity of Holistic escalates to $O(|I|^2)$. This poses a significant challenge when dealing with large datasets.

BigDansing [45]. To alleviate the high time cost of repair algorithms, BigDansing proposes two key acceleration strategies. The first one is removing data irrelevant to the given constraints from the repair process. The second one is grouping equivalent classes with the *same keys* to narrow down the candidates' number. Thus, they accelerate the tuple comparison process and reduce the actual time cost of data repair.

Horizon [73]. To avoid exhaustively tuple comparisons in grouping equivalent classes, Horizon constructs a directed *FD pattern graph* with multiple hierarchies, each corresponding to the attributes in FDs. Directed edges connect nodes across hierarchies, linking values from the left-hand attributes to right-hand ones indicated by FDs. To group equivalent classes and generate data candidates, the graph is traversed with a time complexity *linear* to the number of edges. Horizon finally determines candidates for constraint violation cells with the highest support score.

Nadeef [22]. Unlike prior methods leveraging a certain constraint type, Nadeef focuses on utilizing various constraints by compiling and managing them in a unified format. It enables users to specify not only constraints like FDs and DCs but also other constraints. Based on the candidates generated with equivalent classes, Nadeef minimizes the cardinality cost with two data repair algorithms tailored to emphasize efficiency or effectiveness.

MLNClean [30]. Previous algorithms often instantiate the given constraints by corresponding values to repair data, which may not be trustworthy, thus lacking robustness. To mitigate this issue, MLNClean leverages Markov logic networks to learn the trustworthy degree of each *instantiated constraint*, thereby enhancing the robustness. Based on the learned instantiated constraints, MLNClean generates multiple data candidates. Subsequently, a *reliability* score of each data candidate is calculated with the trustworthy degree. To decide the proper candidates, it first selects candidates with maximal reliability scores, thus generating multiple repaired data versions. Then, MLNClean designs a *fusion* score to resolve conflicts across different data versions and determine the final repairs.

Daisy [32]. Unlike previous methods concerning offline data repair, Daisy aims to repair results from online queries. It first generates candidates from the query data based on equivalent classes indicated by the given DCs. To evaluate candidates' reliability, Daisy computes their conditional probability using the frequency appearing with other attribute values. Finally, Daisy merges distinct conditional probabilities to derive the repair candidates.

3.2.2 Data-driven Data Repair Algorithms. The workflow of *data-driven* data repair algorithms is plotted in Figure 2. It takes dirty data and the *error* detection results as inputs, and extracts data distribution information. Data candidates are generated for error cells based on configured strategies. Then, the probability of a candidate is calculated using ML models [56] or the likelihood benefit formulas [85]. Note that, when computing the probability of each candidate, some algorithms may involve few manually repaired data, as indicated by the dashed line. Finally, the candidates with the highest probabilities are decided as the repair decisions.

Scare [85]. To achieve holistic repair, Scare incorporates ML models into the data repair process. To generate data candidates,

Table 3: Summary and analysis of existing data repair algorithms.

Cats.	Algorithms	Base model	Human config.	Repair goal	Candidate source	Candidate evaluation	Time complexity	Scala. Strategy
Cstr-driven	Holistic [18]	Graph-based	C+Cost	Consistency repair	Equivalent class	Cardinality/Distance cost	$O(\|\Sigma\| \cdot \|I\|^2 \cdot \|A\|)$	None
	BigDansing [45]	Graph-based	C+Cost	Consistency repair	Equivalent class	Cardinality/Distance cost	$O(\|\Sigma\| \cdot \|I\|^2 \cdot \|A\|)$	RR+Group.
	Horizon [73]	Graph-based	C	Consistency repair	Equivalent class	Support score	$O(\|I\|^2 \cdot \|A\|^2)$	VA
	Nadeef [22]	Stats-based	C+Cost	Consistency repair	Equivalent class	Cardinality cost	$O(\|\Sigma\| \cdot \|I\|^2 \cdot \|A\|)$	None
	MLNClean [30]	Stats-based	C+LD	Consistency repair	Equivalent class	Reliability score	$O(\|\Sigma\| \cdot \|I\|^2 \cdot \|A\|)$	VA
	Daisy [32]	Stats-based	C	Consistency repair	Equivalent class	Probability	$O(\|\Sigma\| \cdot \|I\|^2 \cdot \|A\|)$	None
Data-driven	Scare [85]	ML-based	LD+ML model	Holistic repair	Domain	Probability	$O(\|I\| \cdot log\|I\| \cdot \|A\|^2))$	None
	Baran [56]	ML-based	LD+ML model	Holistic repair	Domain+Str variation	Probability	$O(\|I\|^2 \cdot \|A\|^2))$	PS
	BoostClean [47]	Stats. boosting	Rep libs.	Model perf. boost	Mean+Mode+Median	Model performance	$O(\|I\| \cdot \|A\|)$	Rep libs.
Hybrid-driven	Unified [16]	Heuristic-based	C+4 Paras	Tolerant repair	Equivalent class	Description length cost	$O(\|\Sigma\| \cdot \|I\|^2 \cdot \|A\|)$	None
	Relative [8]	Heuristic-based	C+1 Para	Tolerant repair	Equivalent class	Cardinality cost	$O(\|I\|^2 \cdot \|A\|^{\|\Sigma\|\|I\|})$	None
	HoloClean [72]	ML-based	C+13 Paras+ML	Holistic repair	Domain	Probability	$O(\|\Sigma\| \cdot \|I\|^2 \cdot \|A\|)$	DP+TP+PS

a set of *classifiers* is obtained by learning from attribute values detected as clean. These classifiers are utilized to predict potential candidates for values of dirty attributes. Then, the joint probability of candidates is computed based on their co-occurring frequency. Finally, Scare determines the final tuple repair by selecting the candidate values of dirty attributes with the highest joint probability.

Baran [56]. Same to Scare, Baran also pursues holistic repair. When generating data candidates, instead of solely relying on dirty data information, Baran provides *more comprehensive* candidate generation strategies based on various contexts like vicinity, domain, and string variation. It thus significantly enhances the likelihood of including the correct values. Within manually cleaned data and generated candidates of few wrong data, Baran trains a *classifier* to determine the repair candidates for other error cells with the highest probabilities.

BoostClean [47]. BoostClean aims to boost the performance of *downstream analysis* models. Unlike the previous two methods, BoostClean follows an *iterative* process. It first generates data candidates using strategies like mean, mode, and median. Repaired data stemming from these candidates then trains the downstream analysis model. BoostClean computes The candidates' probability is computed by assessing the trained model's performance on validation data. This iterative process continues until maximum probability is reached, which translates into optimized model performance and enhanced quality of data repairs.

3.2.3 Hybrid-driven Data Repair Algorithms. Figure 1 also depicts the workflow of *hybrid-driven* methods (including the part with dashed arrows and modified constraints). After the equivalent classes grouping process, *hybrid-driven* algorithms generate *both* constraint *and* data candidates. The repair algorithm computes the costs of data and *constraint* candidates to determine the minimum-cost candidates. These rectified constraints and data are then applied iteratively to further enhance repair, until the optimal state or cost minimum is reached, as indicated by the dashed arrows.

Unified [16]. Unified aims to achieve *tolerant repair* with minimal description length (DL) [74] for the entire data with FDs. It first calculates the overall DL of each FD and its equivalent classes. The constraints are then processed one by one in the decreasing order of DL. For each FD, Unified generates both constraint and data candidates based on the equivalent classes. To determine the final repair, the candidates are evaluated by DL-based cost. Candidates with minimal cost are decided for the repair. The repaired constraints and data are then taken into consideration when processing the next constraint, continuing until reaching the overall minimal DL.

Relative [8]. Similar to Unified, Relative aims to achieve *tolerant repair* with FDs. To generate both constraint and data candidates, Relative first explores FD modification space to generate constraint candidates, following grouping equivalent classes. For each constraint candidate, Relative endeavors to repair data with the specified threshold. Finally, from the candidates that meet threshold conditions, the ones with the least cardinality costs are determined as the final repairs.

HoloClean [72]. Though utilizing both constraint and data information, HoloClean focuses on *holistic repair*, thus lacking the steps with dashed arrows and modified constraints in Figure 1. After the detection process, HoloClean skips the equivalent classes grouping, treating all values in the dirty data as data candidates. For candidate determination, HoloClean extracts quantitative statistics of constraint and identifies probable correct values with *Naive Bayesian* model. Utilizing these statistics and data, HoloClean learns a statistical model and calculates candidate probabilities via *DeepDive* framework [75]. Ultimately, data candidates with highest probability are selected as final repairs.

3.2.4 Discussion. Table 3 summarizes these data repair algorithms from several aspects. The scalable strategy involves reducing redundancy (RR), parallel strategy (PS), domain pruning (DP), tuple partition (TP), and value-based approach (VA). The human configuration includes cost function (Cost), matching learning (ML) model, constraints (C), repair libraries (Rep libs.), parameter (Para), and labeled data (LD) The time complexity is estimated based on the constraint set size $|\Sigma|$, instance size $|I|$, and attribute set size $|A|$.

It is shown in Table 3 that *cstr-driven* and some *hybrid-driven* algorithms mainly use equivalent classes as candidate sources, minimizing differences between candidates and original dirty data. However, these methods mainly consider constraint equivalence, potentially overlooking other logical or domain-specific relationships. This leads to possible omission of latent correct values and inadequate data error elimination. In contrast, unconstrained methods like HoloClean and *data-driven* algorithms make more liberal use of information, including domain and string variations. This increases the likelihood of including latent clean data in the candidate pool. However, BoostClean's reliance on mean, mode, and median values falls short in identifying latent clean data, resulting in inferior performance in experiments.

The evaluation of data repair candidates hinges on the *minimization* of costs and *maximization* of certain scores. Among them, probability is heavily influenced by the applied models, forming the basis of data-driven and some hybrid-driven methods. Other evaluation metrics depend on heuristic approaches and are constrained

Table 4: Case study of EDR.

Case	$\#d_w$	$\#d_{w2r}$	$\#d_{w2w}$	$\#d_{r2w}$	EDR	Precision	Recall
case 1	100	10	0	10	0	0.50	0.10
case 2	100	10	90	0	0.10	0.10	0.10
case 3	100	10	10	10	0	0.33	0.10

by available information within equivalent classes, as employed by *cstr-driven* and some *hybrid-driven* methods. These metrics can be customized to prioritize specific elements, making the derived repair decisions easy to explain. However, these evaluates presume the majority of the data is accurate, posing challenges when dealing with significantly erroneous datasets.

The time complexity analysis is primarily based on the instance size $|I|$, the constraint set size $|\Sigma|$, and the number of attributes $|A|$. Most data repair methods maintain an $O(|I|^2)$ complexity due to necessary tuple comparisons. The real execution time is subject to various factors such as error and data distribution. Besides, methods like MLNClean and Horizon focus on operating on the values, rather than the tuples, often exhibiting shorter running time in the experiments. For Scare, its time complexity is at the $O(|I| \cdot \log |I|)$ level, but the classifier training process involved in candidate generation can be time-consuming. BoostClean operates with a time complexity at the $O(|I|)$ level, owing to the limited candidate sources. Relative stands out as an exception. Although it exhibits a time complexity at $O(|I|^2)$ level, its exploration of the FD modification space results in exponential time complexity with $O(|A|)$, which is also confirmed in our experimental study.

For human configurations, they can be classified into two categories of human cost: for elements such as constraints, Rep libs, and labeled data, in-depth knowledge is imperative; for ML models, parameters, and cost functions, the emphasis is on practical deployment experience. Algorithms such as Horizon, BoostClean, and Daisy necessitate minimal knowledge and deployment expenses. While HoloClean demands extensive knowledge and thorough deployment analysis, suggesting a higher cost. Moreover, BigDansing, Holistic, Nadeef, Relative, and Unified require additional deployment experience beyond the standard constraints. MLNClean also calls for additional specialized knowledge. Scare and Baran require labeled data and their implementation of ML models may incur additional training time costs.

3.3 Optimization Strategy

In this subsection, we propose an effective strategy to optimize the data repair algorithms. We also present a new metric to fairly evaluate the final data repair results.

The primary cause of suboptimal in existing repair algorithms is *incorrect repair* on *initial right* data. For methods aiming at consistency repair and tolerant repair, this stems mainly from two factors. First, during repair candidate selection, *faulty* information is used as constraint violation cells are not classified as incorrect or correct. Second, the selection process may result in *local-optimal* repairs due to the sole use of equivalent classes. For algorithms targeting *holistic repair*, despite considering a more comprehensive data range, the error detection performance may fall short compared to a dedicated error detection model.

As a result, we propose to adopt Raha [57] as an *optimization strategy* to further improve existing data repair algorithms, for its state-of-the-art (sota) error detection performance [1, 57, 71]. Specifically, the error detection results derived by Raha can be employed to ensure that, the identified correct data remains unchanged during the data repair process, thereby effectively mitigating the risk of erroneously changing *right* data. Although inconsistencies may arise, our experiments validate its efficiency.

Table 5: The used datasets in repair evaluation.

Name	#Tuples	#Attrs	Error rate	Error types	#Cstrs
Hospital	1,000	20	3%	T, VAD	15
Flights	2,376	7	30%	MV, FI, VAD	6
Beers	2,410	11	16%	MV, FI, VAD	5
Rayyan	1,000	11	9%	MV, T, FI, VAD	10
Tax	200,000	15	4%	T, FI, VAD	8

Moreover, we attempt to more accurately measure the ultimate data repair performance. As analyzed earlier, metrics like *precision* and *recall* are *biased*, and distance-based metrics fail to indicate the *improvement degree* or conditions of data error increasing. We denote the initial right data repaired into right and wrong ones by $\#d_{r2r}$ and $\#d_{r2w}$, respectively. The number of initial wrong data repaired into right and wrong ones are denoted by $\#d_{w2r}$ and $\#d_{w2w}$, respectively. Hence, precision is calculated as: $pre = \frac{\#d_{r2r}+\#d_{w2r}}{\#d_{r2r}+\#d_{w2r}+\#d_{r2w}+\#d_{w2w}}$. Obviously, $\#d_{r2r}$ increases precision according to the mediant inequality, and $\#d_{w2w}$ distorts the precision calculation, making a high precision not guarantee error reduction, and vice versa, as cases shown in Table 4. Further, the *F1* score, as the harmonic mean of precision and recall, cannot explicitly reflect the error reduction degree of data.

To this end, we introduce a new definition termed *Error Drop Rate (EDR)* to measure the error reduction performance of data repair algorithms. It disregards irrelevant data quality factors, and directly evaluates the error reduction degree. For generality, we defined it as follows.

$$EDR = \frac{dis^{d2c} - dis^{r2c}}{dis^{d2c}}. \quad (1)$$

Here, dis^{d2c} indicates the distance between dirty and clean data, whereas dis^{r2c} is the distance from repaired data to clean data. Aligning with the problem statements, in the experiments, in the experiments, dis^{d2c} equals to d_w, and dis^{r2c} equals to $d_w - (d_{w2r} - d_{r2w})$. Though ignoring d_{r2r} and d_{w2w} somehow neglects the algorithms' repair ability, the benefits of EDR are multiple. Firstly, EDR allows effective evaluation of data quality enhancements by contrasting pre and post repaired data distance. Secondly, the normalization process enables equal comparison on repair effectiveness irrespective of initial data range and quality. Lastly, EDR provides a straightforward depiction of error reduction/raising (with positive and negative values), and its monotonic feature signifies that increased values align with larger data quality improvements.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate 12 mainstream data repair algorithms to address the following key questions: **Q1:** To what degree can existing automatic data repair algorithms mitigate errors in real-world datasets? **Q2:** Can algorithms with suboptimal repair performance enhance the performance of downstream analysis models? **Q3:** How effective is the proposed general optimization strategy? **Q4:** How effectively can these algorithms manage high error rates and different error types?

Datasets. We conduct repair experiments over five real-world datasets, as outlined in Table 5. *Hospital* and *Flights* [72] include

Table 6: Data repair performance comparison on real-world datasets.

Cat.	Algos.	Hospital			Flights			Beers			Rayyan			Tax-10k		
		EDR	F1	Hybrid	EDR	F1	Hybrid	EDR	F1	Hybrid	EDR	F1	Hybrid	EDR	F1	Hybrid
Rule-driven	BigDansing	−0.0897⑧	0.6239⑥	0.0298⑧	−0.1665⑪	0.3772⑥	0.4193⑩	−0.0111⑥	0.0802③	0.0185④	−0.4540⑤	0.0259③	0.4010⑥	−1.1938⑤	0.1463②	0.3058⑧
	Holistic	−0.0039⑦	0.6403④	0.0282⑥	−0.1335⑧	0.3912④	0.4014⑦	−0.0110④	0.0688④	0.0186⑤	−0.9614⑧	0.0047④	0.4616⑧	−1.1938⑤	0.1463②	0.0099④
	Horizon	0.0530⑤	0.5661⑦	0.0264④	0.1148②	0.3869⑤	0.3290②	−0.0110④	0.0688④	0.0219⑥	−0.9614⑧	0.0047④	0.3528⑤	−50.957⑧	0.1134④	0.0162⑤
	Nadeef	−1.7996⑩	0.0713⑩	0.1074⑩	−0.0528⑦	0⑨	0.3828⑥	−0.4783⑧	0.0094⑧	0.1101⑨	−2.5367⑪	0⑥	0.9503⑪	−55.387⑨	0.0009⑧	0.2074⑦
	MLNClean	0.4322④	0.7240②	0.0152②	−0.0126⑥	0.0051⑦	0.3576⑤	0.0482②	0.1191②	**0.0057**①	−0.6042⑥	0⑥	0.4128⑦	−0.1147④	**0.1927**①	0.0053③
	Daisy	0⑥	0⑪	0.0284⑦	0④	0⑨	0.3519④	0③	0⑨	0.0152③	0②	0⑥	0.2652③	n/a	n/a	n/a
Data-driven	Scare	−0.5350⑨	0.1511⑨	0.0566⑨	−0.1364⑨	0.0002⑧	0.4017⑧	−0.5238⑨	0.0015⑨	0.1036⑧	−0.0886③	0⑥	0.3000④	−7.4933⑦	0.0013⑦	0.0337⑥
	Baran	0.4872③	0.6299⑤	0.0156③	**0.4910**①	**0.6369**①	**0.1894**①	**0.7245**①	**0.7514**①	0.0073②	**0.7403**①	**0.8415**①	**0.1005**①	**0.0160**①	0.0634⑤	0.0050②
	BoostClean	−5.7132⑪	0.3310⑧	0.3152⑪	−0.0028⑤	0⑨	0.5088⑪	−0.7174⑩	0⑨	0.1581⑩	−0.6220⑦	0⑥	0.4869⑨	0②	0⑨	**0.0049**①
Hybrid-driven	Unified	**0.6012**①	**0.7826**①	0.0268⑤	0.0415③	0.5579②	0.3414③	−0.1221⑦	0.0106⑦	0.0538⑦	−0.1862④	0⑥	0.2612②	n/a	n/a	n/a
	Relative	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	HoloClean	0.4872②	0.6515③	**0.0137**①	−0.1390⑩	0.4711③	0.4019⑨	−3.7310⑪	0.0652⑥	0.7141⑪	−1.8897⑩	0.6467②	0.7674⑩	−0.0213③	0.0417⑥	0.7020⑨

Table 7: The case study for in-depth metric analysis.

Algos.	Hospital					Flights					Beers					Rayyan					Tax-10k				
	$\#d_w$	$\#d_{w2r}$	$\#d_{w2w}$	$\#d_{r2r}$	$\#d_{r2w}$	$\#d_w$	$\#d_{w2r}$	$\#d_{w2w}$	$\#d_{r2r}$	$\#d_{r2w}$	$\#d_w$	$\#d_{w2r}$	$\#d_{w2w}$	$\#d_{r2r}$	$\#d_{r2w}$	$\#d_w$	$\#d_{w2r}$	$\#d_{w2w}$	$\#d_{r2r}$	$\#d_{r2w}$	$\#d_w$	$\#d_{w2r}$	$\#d_{w2w}$	$\#d_{r2r}$	$\#d_{r2w}$
BigDansing	509	417	2	58	462	4,920	2,188	1,521	2	3,026	3,358	176	0	3	213	2,873	112	418	19	1,338	375	49	200	4	448
Baran	509	221	48	0	3	4,920	2,773	214	0	230	3,358	2,378	837	0	1	2,873	2,011	108	0	3	375	27	173	0	2
HoloClean	509	248	187	10,603	0	4,920	1,712	3,208	6,933	1	3,358	121	121	2,483	744	2,873	1,217	813	3,335	1,920	375	8	254	7,371	19

a high degree of duplicate tuples and correlated columns. *Beers* is a real-world dataset sourced through web scraping and manually cleaned by the dataset owner [56]. *Rayyan* is another real-world dataset cleansed by its owners [57]. It contains all the error types, thus making it hard to repair. *Tax* is a large dataset describing tax payment records [4]. It contains various data error types with 200,000 tuples and 15 attributes. Rich with insights into social resource allocation and commercial values, these datasets are valuable for academic inquiry and *data-driven* decision-making. Four kinds of error types, namely missing value (MV), typo (T), violated attribute dependency (VAD), and formatting issue (FI), exist in the datasets, and are primarily encountered in real-world scenarios [4, 56, 57, 71]. They stem from either incorrect value allocation, termed semantic errors, or the presence of out-of-domain values, known as syntactic errors [56, 71]. For each dataset, the corresponding clean version exists, applied to get evaluation metrics and further experiments. To discover denial constraints (DCs), we initially employ two widely-used DC discovery methods DCFinder [70] and Hydra [10]. We then manually check all discovered rules, deciding whether to accept, modify, or deny each rule. The final applied rules can be viewed in the code repository.

To comprehensively explore various error rates and error types scenarios *not covered* by real-world datasets, we generate dirty data by adding erroneous values randomly into clean data. For semantic errors, we generate errors by randomly selecting alternatives from the domain. Regarding syntactic errors, we introduce typos, both *explicit* and *implicit* missing values, and Gaussian noise, aligning with T, MV, and FI but with a broader range. These errors are generated using the publicly available code from BigDaMa [9].

Evaluation metrics. We employ the proposed metric *EDR*, and running time *runtime* (in seconds), CPU (in percent), and memory (in Mb) usage to measure the effectiveness and efficiency of data repair algorithms. For comparison, we report *F1 score* in real scenarios, considering its wide application in previous studies [8, 17, 18]. We have also applied a *hybrid distance metric*, denoted by $hybrid\_dis = w_1 * MSE + w_2 * Jac\_dis$, where *MSE* represents mean squared error, and Jac_dis denotes Jaccard distance. The weights w_1 and w_2 are the proportion of values calculated using *MSE* and Jac_dis, respectively. For downstream data analysis tasks, we employ prevail *F1* score for classification, *mean squared error* (MSE) for regression, *silhouette score* for clustering, and *accuracy* for k nearest neighbor query (kNN), where k=1 in the experiments.

Implementation details. We re-implement *seven* pieces of codes for Holistic [18], BigDansing [45], Horizon [73], Daisy [32], Scare [85], Unified [16], and Relative [8], due to the unavailable source codes. We use publicly accessible codebases for other five methods. For MLNClean and Baran, which require manual cleaning, a consistent minimum of 20 tuples is used. To feed detection results into Baran and Scare, we use Raha [57]. For other algorithms, we adopt their initial detection methods as their repair processes are closely tied to them. For Holistic, BigDansing, Nadeef and Relative, we employ the original cardinality functions used in the paper. The other hyper-parameter settings are adopted as specified in the source codes or papers. Each reported result represents the average of *three* repeated experiments. 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.

4.1 Repair Performance on Real Scenarios

The first set of experiments evaluates the performance of all data repair algorithms over five real-world datasets. Table 6 lists the corresponding experimental results in terms of *EDR*, *F1* score, and *Hybrid_dis* (Hybrid). The best values for each metric within the same dataset are identified using **bold** formatting. We also label the rank in the circle of algorithms within each dataset. As none of the methods could finish the repair process in 24 hours (labeled with "n/a") except MLNClean, we partition *Tax* into subsets, where the larger ones encompass the entirety of the smaller ones. The smallest subset is used to evaluate the repair performance of algorithms, the others are for the repair cost study. For further illustration, Table 7 reports the specific values of $\#d_w$, $\#d_{w2r}$, $\#d_{r2r}$ and $\#d_{r2w}$ of each dataset over relatively good algorithm from each group. Based on the repaired data, we then conduct a set of experiments under *four* data analysis tasks: classification and regression

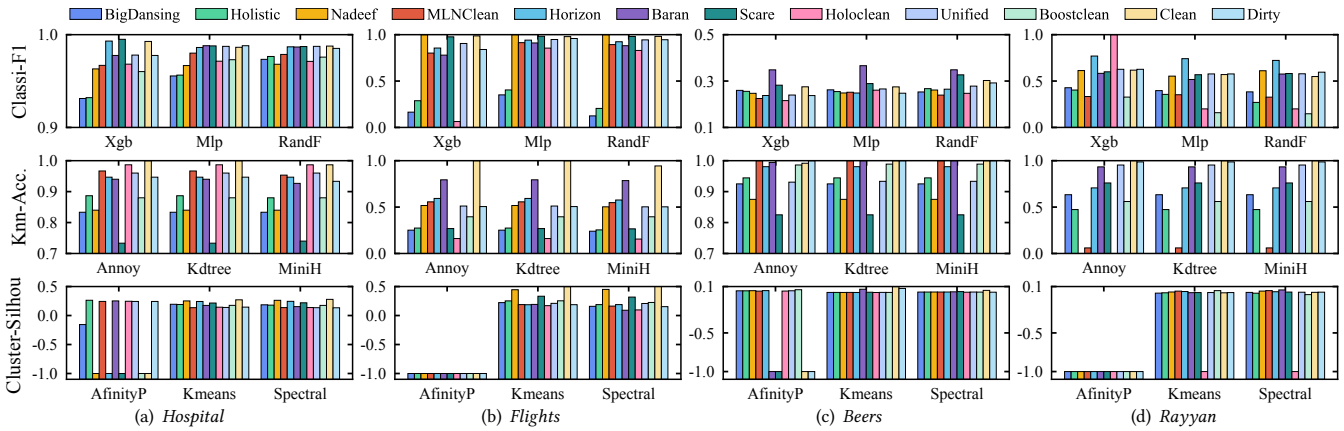


Figure 3: Performance of classification, kNN, and cluster on repaired data.

(with XGBoost [14], MLP [78], and RandomForest models [54]), as well as clustering (with Affinity Propagation [28], Spectral Clustering [66], and K-means models [55]) and kNN query (with Annoy [52], MiniHash [13], and KD-tree models [6]). For classification, kNN, and clustering, we select 'HospitalName', 'flight', 'style', and 'article_title' for the datasets of Hospital, Flights, Beers, and Rayyan as the target column, respectively. For regression, 'Score' and 'ibv' are chosen for the Hospital and Rayyan datasets, as the sole two featuring numerical attributes. These attributes exhibit a strong correlation with other values and are important features in the data.

Data repair performance. As shown in Table 6, most data repair methods exhibit *negative* values for the EDR metric. This phenomenon can be attributed to the fact that, there are more incorrect repairs on correct values compared to the correct repair of wrong ones, according to the definition of EDR in Eq. 1. Across datasets, most algorithms exhibit better EDR value on the Hospital dataset. This can be attributed to its abundance of redundancies, substantial cosntraints, and simpler contexts. While other datasets prove challenging to repair due to either high error rate (on Flights), or fewer redundancies with more complex contexts (on Beers, and Rayyan). As for Tax, the data repair algorithms tend to introduce a larger amount of errors, possibly due to its scare errors and large size.

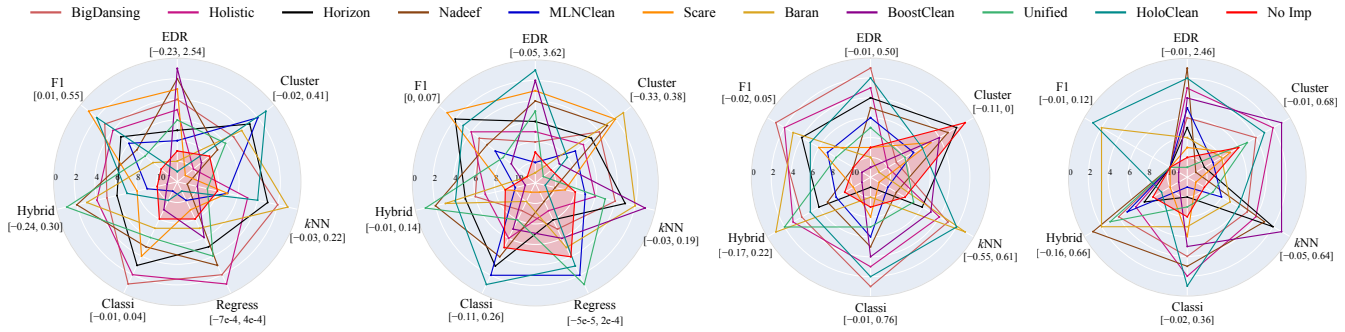
Among these methods, *Baran* shows superior *EDR* performance on most datasets except *Hospital*. It can be attributed to its utilization of more information(like manually cleaned data and precise error detection results), and comprehensive candidate generation strategies, which increase the likelihood of including latent clean data in the candidate sets to a large extent. For other *data-driven* methods, BoostClean and Scare perform much worse due to the simple strategies for generating candidates, and the dependence on fully clean attributes, respectively. Moreover, most of the *cstr-driven* algorithms display inferior performance except MLNClean. This observation highlights that, leveraging models like the Markov network to learn instantiated constraints may be promising. Besides, Daisy does not repair any of the data because the union of all the conditional probabilities potentially assigns the highest probability to the initial dirty tuples, leaving them unchanged. For *hybrid-driven* methods, HoloClean and Unified perform much better on *Hospital* than other datasets, due to substantial redundancies. HoloClean is more sensitive to data than Unified. The reason is that, HoloClean employs Naive Bayesian model to identify possible correct data, which assumes attribute independence, being challenging in practice. Relative can not finish the repair in all cases due to its exponential level of time complexity.

Note that, in the rest of experiments, we exclude Daisy and Relative, since they either fail to produce results within 24 hours or have no repair impact on the datasets. The *Tax* dataset is omitted, since only *MLNClean* can complete the repair process.

Downstream task performance. Figure 3 shows the performance of classification, kNN, and clustering tasks on repaired data. While Table 8 illustrates the performance of regression. Firstly, it can also be observed that, with proper selections of repair algorithms, regardless of the applied model, the downstream task performance can always be enhanced over the dirty data. Besides, among four tasks, performance on regression is rarely influenced by the repair data, probably due to the few numerical values in the data. Notably, we can see that applying clean data to train the model using clean data is not always the best for downstream tasks, as demonstrated by the performance of Xgboost models on Flights. Also, as evidenced by the RandomForest model on Rayyan, downstream analysis models trained on purely clean data may perform worse than models trained using dirty data.

Repair metric evaluation. As shown in Table 6, *EDR* provides a simpler and more intuitive repair performance evaluation. With higher *EDR* values denoting superior error reduction and negative (resp. positive) fluctuations indicating error increments (resp. decrements). While it is hard to interpret the actual error reduction degree from F1 score and Hybrid_dis. Notably, EDR shares a 0.64 mean overlap rate with the F1 score top 5/11 methods, showing 0.8, 0.6, 0.8, 0.4, and 0.6 scores for Hospital, Flights, Beers, Rayyan, and Tax-10k. It also registers a 0.84 overlap rate with Hybrid_dis, posting 1.0, 0.8, 0.8, 0.8, and 0.8 across the same datasets. This phenomenon suggests that while there are some discrepancies in the rankings produced by these metrics, they are highly correlated in their evaluation of relative repair performance.

In particular, when considering the F1 score for Flights, HoloClean achieves *F1 score* as high as 0.47, placing it in the top 3 performance. But the fact is that, it tends to introduce more errors than it eliminates. This is because most of its correctly repaired cells are from the initial right data, as shown in Table 7. It indicates that,



(a) *Hospital* (b) *Beers* (c) *Flights* (d) *Rayyan*

Figure 4: Optimization gain ranking.

Table 8: Performance of regression on repaired data.

Algos.	Hospital			Beers		
	XGB($\times e^{-32}$)	MLP	RandF($\times e^{-32}$)	XGB($\times e^{-32}$)	MLP	RandF($\times e^{-32}$)
All clean	2.9108	0.0019	1.3207	2.7342	0.0002	1.2741
No repair	2.9108	0.0026	1.3207	2.7342	0.0002	1.2741
BigDansing	2.9108	0.0031	1.3207	2.7342	0.0010	1.2741
Holistic	2.9108	0.0028	1.3207	2.7342	0.0010	1.2741
Horizon	2.9108	0.0027	1.3207	2.7342	0.0010	1.2741
Nadeef	2.9108	0.0025	1.3207	2.7342	0.0002	1.2741
MLNClean	2.9108	0.0022	1.3207	2.7342	0.0005	1.2741
Unified	2.9108	0.0026	1.3207	2.7342	0.0004	1.2741
HoloClean	2.9108	0.0019	1.3207	2.7342	0.0009	1.2741
Scare	2.9108	0.0020	1.3207	2.7342	0.0008	1.2741
Baran	2.9108	0.0027	1.3207	2.7342	0.0010	1.2741
BoostClean	2.9108	0.0024	1.3207	2.7342	0.0010	1.2741

Table 9: Anal. of repair metric and downstream performance.

Task	Corr. indicator	EDR	F1 score	Hybrid_dis
Classi.	Mean *rank corr.*	**0.5311**	0.2423	0.0816
	Mean *top-5 overlap.*	**0.5733**	0.5200	0.4400
Regress.	Mean *rank corr.*	0.1909	**0.3394**	0.0244
	Mean *top-5 overlap.*	**0.6000**	**0.6000**	0.5000
Cluster	Mean *rank corr.*	**0.1158**	0.0974	−0.0893
	Mean *top-5 overlap.*	**0.4133**	**0.4133**	**0.4133**
kNN	Mean *rank corr.*	**0.1920**	0.0278	0.0456
	Mean *top-5 overlap.*	**0.5200**	0.4000	0.4400

the inclusion of $\#d_{r2r}$ may significantly improve F1 score, while it actually has no impact on the data error rate, thus making it accurate in evaluating the quality of repaired data. Similar situations also occur on Beers and Rayyan when assessing HoloClean. Meanwhile, $\#d_{w2w}$ also influences the F1 score calculation. Baran achieves a low F1 score on Tax-10k, due to incorrect repairs of 173 original wrong cells, constituting a large proportion of all repaired cells. However, even with an F1 score as low as 0.0634, Baran can still decrease errors in the data. These instances suggest that, the F1 score falls short in evaluating the actual error change condition. With a high F1 score, there may also be an increase in the data error rate due to a large $\#d_{r2r}$. A low F1 score may be attributed to large $\#d_{w2w}$, and thus the data errors can still be reduced.

Furthermore, according to the rank of the repair tools in Table 6, and the performance rank of downstream tasks on different repaired data in Figure 3 and Table 8, we evaluate the mean Pearson correlation coefficient and the mean Top-5 methods' overlap rate between repair and downstream performance (across all evaluated models on the repaired dataset, i.e., Hospital, Flights, Beers, and Rayyan). As shown in Table 9, EDR displays the strongest correlation with downstream performance, and the highest overlap rate among repair metrics in most cases. This suggests that EDR is a more reliable indicator of data quality when performing data repairs aimed at enhancing downstream task performance. Thus, it is advisable to accord greater importance to EDR when assessing repair tools in data analysis tasks.

Table 10: Overhead of the optimization strategy.

Overhead	Hospital	Flights	Beers	Rayyan
Max CPU usage	4840%	5120%	2432%	5690%
Max Memory usage	331Mb	307Mb	387Mb	252Mb
Runtime	294s	88s	208s	107s

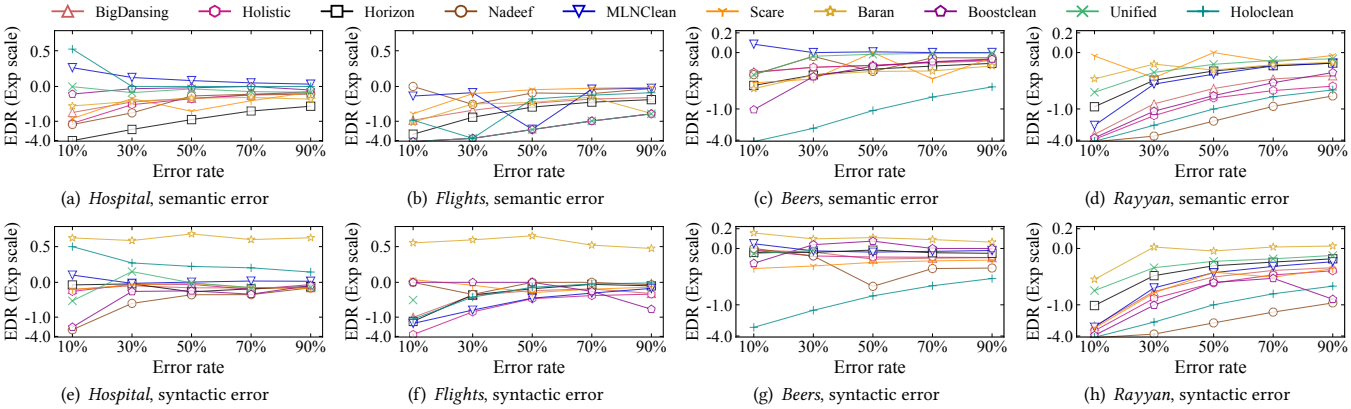
Table 11: Frequency of incorrect prevention v.s. effective opt.

Algos.	Hospital		Flights		Beers		Rayyan	
	Iden-W	Iden-R	Iden-W	Iden-R	Iden-W	Iden-R	Iden-W	Iden-R
Big.	0.29	**0.52**	0.04	**0.40**	0	**0.55**	0	**0.72**
Holi.	0.31	**0.51**	0.05	**0.40**	0	**0.55**	0	**0.81**
Nadeef	0.02	**0.91**	0	**0.54**	0	**0.81**	0	**0.71**
MLNC.	0.28	**0.30**	0.01	**0.36**	0.01	**0.20**	0	**0.81**
Hori.	0.34	**0.47**	0.06	**0.19**	0	**0.56**	0	**0.98**
Baran	0.55	0.24	0.03	0.03	0.02	**0.04**	0.02	**0.36**
Scared	0.07	**0.79**	0	0	0	**0.92**	0	**0.46**
Holo.	0	**0.44**	0.01	0	0.01	**0.57**	0	**0.96**
Uni.	0	**0.19**	0.08	**0.14**	0.01	**0.86**	0	0
Boost.	0.11	**0.65**	0	**0.50**	0	**0.78**	0	**0.12**

Effect of optimization strategy. Firstly, to fully represent the effect of the optimization strategy, we provide the ranks of advancement for both repair metrics and downstream performance. Besides, we also show the range for the absolute values of improvement, all of which is demonstrated in Figure 4. The term 'No Imp' represents the rank where no enhancements are observed. In other words, points that are positioned outside the 'No Imp' range signal improvements. Notably, in a majority of instances, the optimization strategy results in observable improvements. Secondly, we report the frequency of instances where the optimization strategy blocks the correct alteration of initially incorrect values, and successfully prevents incorrect modifications of initially correct values. We denote the former and latter frequency as Iden-W and Iden-R, respectively, as depicted in Table 11. It is noticeable that, despite the presence of incorrect preventative actions, the instances of Iden-R exceed those of Iden-W in most scenarios. This outcome suggests that, on balance, the optimization strategy serves its intended purpose effectively. Finally, to report the extra cost of the optimization strategy, we have reported its maximum CPU (in percent, e.g. 4840% indicates the use of 48 CPU cores) and memory usage (in Mb), as

Table 12: Maximum calculation resource usage of the algorithms across varying data sizes.

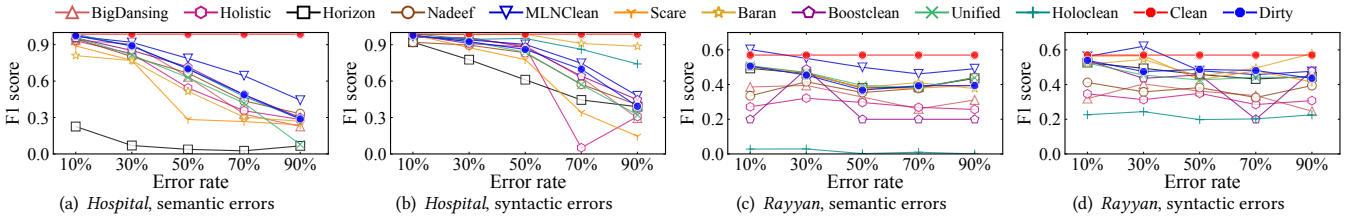
Algos.	Tax-10k			Tax-20k			Tax-30k			Tax-40k			Tax-50k		
	Runtime	CPU	Memory	Runtime	CPU	Memory	Runtime	CPU	Memory	Runtime	CPU	Memory	Runtime	CPU	Memory
BigDansing	2,691	109%	6,557Mb	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Holistic	45,261	109%	47,670Mb	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Horizon	1,675	110%	118Mb	5,757	111%	162Mb	11,378	109%	186Mb	18,613	110%	249Mb	27,124	110%	288Mb
Nadeef	821	110%	3,004Mb	4,421	111%	6,281Mb	7,497	112%	6,763Mb	15,492	110%	8,744Mb	19,571	110%	9,839Mb
MLNClean	66	110%	4,796Mb	122	110%	5,250Mb	226	110%	5,735Mb	329	112%	7,776Mb	395	113%	8,671Mb
Scare	38,096	5760%	8,693Mb	23,705	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Baran	49,346	6400%	2,070Mb	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
BoostClean	502	1180%	6919Mb	1,233	1090%	15,820Mb	1,729	1140%	25,379Mb	2,708	1020%	37,712Mb	3,939	1100%	51,345Mb
HoloClean	935	4820%	191,019Mb	n/a*	n/a*	n/a*	n/a*	n/a*	n/a*	n/a*	n/a*	n/a*	n/a*	n/a*	n/a*



(a) *Hospital*, semantic error (b) *Flights*, semantic error (c) *Beers*, semantic error (d) *Rayyan*, semantic error

(e) *Hospital*, syntactic error (f) *Flights*, syntactic error (g) *Beers*, syntactic error (h) *Rayyan*, syntactic error

Figure 5: Data repair performance vs. different error rates.



(a) *Hospital*, semantic errors (b) *Hospital*, syntactic errors (c) *Rayyan*, semantic errors (d) *Rayyan*, syntactic errors

Figure 6: Classification performance vs. different error rates.

well as the end-to-end runtime across four datasets, as outlined in Table 10. Though the max CPU usage appears high, the memory usage and runtime are within reasonable limits, ensuring that the optimization strategy remains practical applications.

Scalable performance. Table 12 displays the execution time (in seconds), max CPU (in percent), and Memory usage (in Mb) of data repair algorithms on *Tax*. It is observed that only half of the algorithms can finish the experiments, indicating the significant time cost challenges. Among them, Horizon consumes the least resources, showing almost a linear increase in resource usage as the data size increases. This verifies the effectiveness of its value-based modeling approaches. BoostClean also scales but has a notably higher CPU and Memory usage. MLNClean appears to be the most efficient in terms of runtime for smaller datasets, although the memory usage is still significant. HoloClean and Holistic have extremely high memory usage even for the smallest data size. It is important to note that, BigDansing consumes much less resources than Holistic, validating its scaling strategies. Unified cannot complete tasks, largely due to the extensive time consumption in assessing both data and constraint cost. For Baran, the challenge in its scalability arises from the substantial cost of generating diverse candidates. Its high CPU usage also indicates the validity of the parallel strategy.

4.2 Effects of Varying Errors

The second set of experiments is to fully verify the effect of data error *both* on the repair performance *and* downstream effects of data repair algorithms, considering the *semantic* and *syntactic* errors across various error rates. We only report *EDR* metric for its superiority. Regarding downstream tasks, we focus solely on the classification tasks, given their extensive applications.

For repair performance, Figure 5 illustrates the experimental results when the error rate changes from 10% to 90%. The vertical axis is in an exponential scale. Remarkably, an increase in the error rate usually corresponds to the reduced correction effectiveness of these algorithms. Existing data repair algorithms struggle to eliminate *semantic* errors, while *syntactic* errors can be reduced more or less. It is because that, semantic errors involve similar value patterns and inter-value relations as the clean data, which may make it hard to detect and decide the correct candidate. Concerning syntactic errors, Baran excels across all cases. Even with 90% error rates in

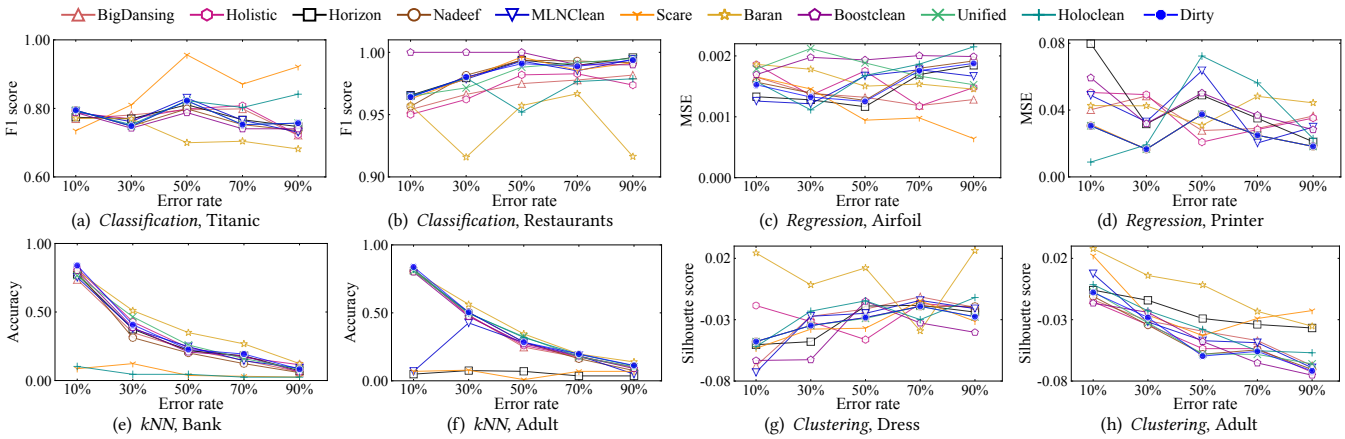


Figure 7: Real workloads v.s. error rates.

Hospital, it can rectify over half errors, demonstrating its strength in handling syntactic errors. However, it underperforms in addressing semantic errors, possibly due to its feature extraction strategies' imperfections and reliance on minimal manually cleaned data. *Cstr-driven* methods appear to encounter challenges in significantly reducing errors across varying error rates. For *hybrid-driven* methods, consistent with previous results, HoloClean performs much better on *Hospital* than other datasets, while Unified demonstrates greater robustness. Regarding semantic errors, *cstr-driven* methods exhibit greater robustness. In particular, MLNClean demonstrates superior performance compared to syntactic errors, especially on Hospital and Beers, showing the effectiveness of its strategy to learn the trustworthy degree of instantiated constraints.

For the classification task, Figure 6 illustrates the corresponding experimental results. We employ MLP on Hospital and Rayyan, as MLP generally performs better than XGBoost. We can observe that, the classification performance decreases almost linearly with an increasing error rate in most cases, without the sharp drop expected. Semantic errors generally have larger negative impacts on MLP than syntactic ones. In terms of semantic errors, except for data repaired by MLNClean, the repaired dataset scarcely helps improve the classification performance of the trained model beyond that achieved with the original dirty dataset. It is attributed to the further disruption of the data distribution by the repair process, making it harder for the model to learn feature-label relations.

4.3 Repair Effect on Downstream Tasks

To further answer the critical question of how data repair algorithms impact the downstream task, we conduct a set of experiments on datasets with real workloads.

Following previous related works [1, 20, 50, 51], for classification, we employ *Restaurants* and *Titanic* datasets; for regression, *Airfoil* and *Printer* datasets are used; clustering analysis is conducted using *Adult* and *Dress* datasets; and k-nearest neighbor (kNN) queries perform on *Adult* and *Bank* datasets. We employ Xgboost, Spectral clustering, and KD-tree models to conduct this experiment, selected based on their median performance in previous experiments, thereby proving generality. Due to the lack of clean data, we treat the initial data as correct, and introduce dirty data by adding previously mentioned *semantic* and *syntactic* errors. Since errors can be introduced by independent factors in reality, thus the proportion of each specific error (i.e., implicit missing, explicit missing, typos, gaussian noise, and misallocated values) is equal. These errors are both injected into features and labels. With a 1:4 ratio of semantic to syntactic errors, we illustrate the performance of *four* workloads on *both* the repaired data *and* the initial dirty data, given the absence of corresponding clean data. Besides, for cstr-driven and some hybrid-driven methods, we adopt the aforementioned DC discovery tools, i.e., DCFinder [70] and Hydra [10] to discover constraints, which we then manually check.

As shown in Figure 7, aligning with earlier results, data repair enhances performance in *almost every* condition across all 4 tasks and 7 datasets, with few negligible deviations at a 10% error rate. With proper algorithm selection, data repair can enhance downstream model performance compared to the original dirty data. Typically, the performance of models with repaired data fluctuates between that with original dirty data. Among them, Baran can consistently augment downstream task performance across nearly all datasets, with the exception of Restaurants. Coupled with MLNClean, it boosts model performance except in the case of the Adult dataset with the kNN task. While Nadeef provides less substantial improvements, contributing only to the performance enhancement on Restaurants and Adult within the clustering task. Other methods also enhance model performance on more than two datasets, confirming the need for data repair algorithms in practical scenarios. Interestingly, on the Restaurant and Dress datasets, as the error rate increases, performance improves. This counter intuitive phenomenon may occur due to various factors, such as the model possibly leveraging noise to avoid overfitting, thereby gaining a more generalized performance on varied datasets.

4.4 Discussion

It is observed that existing automatic data repair algorithms can effectively reduce syntactic errors given adequate redundancies but struggle to eliminate semantic errors. However, with proper methodology, data repair can *benefit* downstream tasks in the majority of situations. We summarize some takeaways for data repair algorithms regarding the scenarios based on the experiments.

Recommendations. Baran, MLNClean, Horizon, and BigDansing are better alternatives in real-life scenarios.

- Dominant syntactic errors: Baran emerges as the foremost recommendation, demonstrating its unique efficacy in dealing with syntactic errors. Previous experiments have consistently illustrated its superior performance over other algorithms.

- Dominant semantic errors: To solve semantic errors, the use of MLNClean is recommended for its beneficial application of Markov Logic Networks in learning the credibility of constraints, showing significant performance in managing semantic errors.
- Large-scale Datasets: Horizon deserves consideration. In situations characterized by large volumes of data, the value-based approach of Horizon can handle large datasets without excessive computation resource consumption.
- User and Domain Expert Involvement: Considering that HoloClean demands considerable human input in the form of configurations, it can be beneficial to incorporate this approach as a way to efficiently leverage user and expert insights.

Guidelines. Based on the experiments, we provide four guidelines regarding errors, optimization strategies, and data analysis models.

i) *The optimization strategy is suggested.* The key cause of subpar performance in current data repair algorithms is incorrect data repair. Integrating the sota error detection methods can reduce the risk of incorrectly repairing accurate data.

ii) *EDR should be paid more attention.* EDR provides an intuitive and quantifiable assessment of data quality enhancements. By focusing on it, a more profound understanding of variations in data quality can be achieved. Furthermore, it exhibits the *strongest* correlation with the performance of downstream tasks, augmenting forecasting performances with a higher degree of trustworthiness.

iii) *It is always worthy of repairing data for downstream tasks.* Data repair remains a critical operation regardless of the error rate. Via carefully selecting appropriate data repair algorithms, it is possible to eliminate up to half of the errors even when the error rate is as high as 90%. It can significantly benefit the downstream tasks.

iv) *Semantic errors should be paid more attention. Semantic* errors may detrimentally affect downstream models more than *syntactic* ones. The repair algorithms even exacerbate this disruption. Hence, resolving *semantic* errors should be prioritized.

v) *Choice of data analysis model also matters.* Different models make different assumptions on data distribution. If improving the data quality is challenging, taking the time to systematically evaluate different models may lead to better performance, like the F1 score discrepancies between *XGBoost* and *MLP* in the experiments.

5 FUTURE DIRECTIONS

In this section, we underscore research challenges and potential avenues for further exploration within the data repair domain.

Research challenges. Within the development of various downstream models, especially the arising of large language model (LLM) [40], more complex data scenarios and problems are appearing. We summarize current challenges as follows.

Effective and efficient data repair. Existing repair tools struggle with complex datasets due to the infinite error domain versus a single correct value. Efficiency issues persist as well, with solutions like *Baran* unable to process 20K records within a day, demonstrating the pressing demand for improved data repair algorithms.

Proper metrics evaluating data quality for models. Our experiments show no necessarily negative correlation between the downstream task performance and data error rate. With the rise of LLMs, considering data quality issues like hallucination [58, 82] gains significance. This necessitates the exploration of new metrics to thoroughly assess data quality *w.r.t.* applied models.

Potential directions. current data repair methods mainly focus on reducing errors, but the advent of LLMs and subsequent complex data scenarios point towards a shift in future directions:

Combining rule discovery and data information for data repair. Existing sota algorithms like Baran [46, 56, 61] leverage data information. However, their susceptibility to unrelated values highlights the importance of combining rule discovery and data information, especially given the high human costs in the big data era [59].

LLM for data repair. Current automated data repair algorithms struggle to generate candidate solutions beyond the available data [19, 26, 56]. Embracing LLMs, with their reasoning ability and extensive knowledge, can overcome this by generating semantically significant and grammatically valid repair candidates [15, 33, 79, 81], making it a fascinating research direction.

6 RELATED WORK

Data cleaning mainly comprises two steps: error detection and repair [17]. The error detection involves two work lines, i.e., *cstr-driven* algorithms [3, 5, 7, 8, 12, 18, 22, 24, 72], and *data-driven* ones [38, 41, 49, 57, 62, 63, 65, 71, 80]. The error repair process typically employs intrinsic data properties like integrity constraints [17, 18, 24, 31, 34, 73] and the data distribution [56, 77, 85], or external information like master data [26], knowledge base [19], and downstream data analysis models [35, 47, 62] to correct the values. Deduplication is also an essential operation in data preparation [21], leveraging meta-information [27, 29, 37], entity matching rules [25, 76], ML models [23, 53, 64, 68], and clustering algorithms [83] to deduplicate the original data. This survey primarily concentrates on evaluating final data repair results.

7 CONCLUSIONS

In this paper, we comprehensively examine 12 mainstream data repair algorithms through a new taxonomy in a unified framework. We propose an effective optimization strategy to improve all of these algorithms. We also introduce a novel metric for fair data repair evaluation, highlighting the limitations of existing metrics in evaluating error reduction. Through thorough testing on five real-world datasets and analysis across four common downstream tasks encompassing 7 practical workloads, we assess the algorithms' efficacy in various complex scenarios and downstream tasks. We conclude a series of key insights, observations, and future research directions in the field of data repair. Moving forward, we intend to further enhance the overall data repair performance in practice.

ACKNOWLEDGMENTS

This work is supported by Leading Goose R&D Program of Zhejiang (No.2024C01109), the NSFC (No.62372404), Beijing Life Science Academy (No.2024900CB0080, No.2023000CB0020), Henan Institute of Chinese Engineering Development Strategies (No.2023HENZDB01), and the Fundamental Research Funds for the Central Universities (No.226-2024-00030). Xiangyu Zhao is partially supported by Research Impact Fund (No.R1015-23), APRC-CityU New Research Initiatives (No.9610565), CityU-HKIDS Early Career Research Grant (No.9360163), Huawei (Huawei Innovation Research Program), Tencent (CCF-Tencent Open Fund, Tencent Rhino-Bird Focused Research Program), Ant Group (CCF-Ant Research Fund, Ant Group Research Fund), CCF-BaiChuan-Ebtech Foundation Model Fund, and Kuaishou. Xiaoye Miao is the corresponding author.

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. Detecting 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 Cooperation, 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 Euijong Whang, and Jennifer Widom. 2009. Swoosh: A generic approach to entity resolution. *The VLDB Journal* 18, 1 (2009), 255–276.

[6] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (1975), 509–517.

[7] 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.

[8] 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.

[9] BigDaMa. 2023. Error generator. https://github.com/BigDaMa/error-generator. Accessed: 2024-06-14.

[10] Tobias Bleifuß, Sebastian Kruse, and Felix Naumann. 2017. Efficient denial constraint discovery with hydra. *Proceedings of the VLDB Endowment* 11, 3 (2017), 311–323.

[11] Ali Borji. 2023. Generated faces in the wild: Quantitative comparison of stable diffusion, midjourney and DALL-E 2. *ArXiv Preprint ArXiv:2210.00586* (2023).

[12] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. 2020. Outlier detection: Methods, models, and classification. *Comput. Surveys* 53, 3, Article 55 (2020), 37 pages.

[13] Andrei Z Broder. 1997. On the resemblance and containment of documents. In *SEQUENCES*. 21–29.

[14] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *KDD*. 785–794.

[15] Zui CHen, Lei Cao, Sam Madden, Ju Fan, Nan Tang, Zihui Gu, Zeyuan Shang, Chunwei Liu, Michael Cafarella, and Tim Kraska. 2023. SEED: Simple, efficient, and effective data management via large language models. *ArXiv Preprint ArXiv:2310.00749* (2023).

[16] Fei Chiang and Renée J. Miller. 2011. A unified model for data and constraint repair. In *ICDE*. 446–457.

[17] Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data cleaning: Overview and emerging challenges. In *SIGMOD*. 2201–2206.

[18] Xu Chu, I. F. Ilyas, and P. Papotti. 2013. Holistic data cleaning: Putting violations into context. In *ICDE*. 458–469.

[19] 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.

[20] Ireneusz Czarnowski. 2012. Cluster-based instance selection for machine classification. *Knowledge and Information Systems* 30, 5 (2012), 113–133.

[21] Xin Luna Dong and Theodoros Rekatsinas. 2018. Data integration and machine learning: A natural synergy. In *SIGMOD*. 1645–1650.

[22] 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.

[23] 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.

[24] Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2008. Conditional functional dependencies for capturing data inconsistencies. *ACM Transactions on Database Systems* 33, 2, Article 6 (2008), 48 pages.

[25] 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.

[26] 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), 213–238.

[27] Ivan P. Fellegi and Alan B. Sunter. 1969. A theory for record linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.

[28] Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science* 315, 5814 (2007), 972–976.

[29] 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.

[30] 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.

[31] 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.

[32] Stella Giannakopoulou, Manos Karpathiotakis, and Anastasia Ailamaki. 2020. Cleaning denial constraint violations through relaxation. In *SIGMOD*. 805–815.

[33] 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.

[34] 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.

[35] Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. 2019. Data cleansing for models trained with SGD. In *NeurIPS*. 4213–4222.

[36] Md Kamrul Hasan and Mohammad Mahdavi. 2021. Automatic Error Correction Using the Wikipedia Page Revision History. In *CIKM*. 3073–3077.

[37] Oktie Hassanzadeh, Fei Chiang, Hyun Chul Lee, and Renée J. Miller. 2009. Framework for evaluating clustering algorithms in duplicate detection. *Proceedings of the VLDB Endowment* 2, 1 (2009), 1282–1293.

[38] Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. 2019. HoloDetect: Few-shot learning for error detection. In *SIGMOD*. 829–846.

[39] 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*.

[40] Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. *ArXiv Preprint ArXiv:2212.10403* (2023).

[41] Zhipeng Huang and Yeye He. 2018. Auto-detect: Data-driven error detection in tables. In *SIGMOD*. 1377–1392.

[42] Kadir Ider and Andreas Schmietendorf. 2018. Data privacy for AI fraud detection models. In *ICDS*. 102–107.

[43] 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.

[44] Ihab F. Ilyas and Xu Chu. 2019. *Data cleaning*. Association for Computing Machinery.

[45] Zuhair Khayyat, Ihab F. Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. 2015. BigDansing: A system for big data cleansing. In *SIGMOD*. 1215–1230.

[46] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*. 1885–1894.

[47] 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).

[48] Sanjay Krishnan, Daniel Haas, Michael J. Franklin, and Eugene Wu. 2016. Towards reliable interactive data cleaning: A user survey and recommendations. In *HILDA*. Article 9, 5 pages.

[49] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Goldberg. 2016. ActiveClean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment* 9, 12 (2016), 948–959.

[50] Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. 2014. A data- and workload-aware algorithm for range queries under differential privacy. *Proceedings of the VLDB Endowment* 7, 5 (2014), 341–352.

[51] 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.

[52] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2019), 1475–1488.

[53] 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.

[54] Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.

[55] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137.

[56] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 1948–1961.

[57] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A configuration-free error detection system. In *SIGMOD*. 865–882.

[58] 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).

[59] 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.
[60] Xiaoye Miao, Yunjun Gao, Su Guo, and Wanqi Liu. 2018. Incomplete data management: A survey. *Frontiers of Computer Science* 12, 1 (2018), 4–25.
[61] Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, Jun Wang, and Jianwei Yin. 2021. Efficient and effective data imputation with influence functions. *Proceedings of the VLDB Endowment* 15, 3 (2021), 624–632.
[62] Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, and Jianwei Yin. 2023. An experimental survey of missing data imputation algorithms. *IEEE Transactions on Knowledge and Data Engineering* 35, 7 (2023), 6630–6650.
[63] Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin. 2021. Generative semi-supervised learning for multivariate time series imputation. In *AAAI*. 8983–8991.
[64] 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.
[65] Felix Neutatz, Mohammad Mahdavi, and Ziawasch Abedjan. 2019. ED2: A case for active learning in error detection. In *CIKM*. 2249–2252.
[66] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *NeurIPS*. 849–856.
[67] 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).
[68] 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.
[69] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment* 8, 10 (2015), 1082–1093.
[70] Eduardo H. M. Pena, Eduardo C. de Almeida, and Felix Naumann. 2019. Discovery of approximate (and exact) denial constraints. *Proceedings of the VLDB Endowment* 13, 3 (2019), 266–278.
[71] 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.
[72] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1190–1201.
[73] 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.
[74] Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica* 14, 5 (1978), 465–471.
[75] 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.
[76] 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.
[77] Shaoxu Song, Han Zhu, and Jianmin Wang. 2016. Constraint-variance tolerant data repairing. In *SIGMOD*. 877–892.
[78] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang. 2015. Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks and Learning Systems* 27, 4 (2015), 809–821.
[79] Nan Tang, Chenyu Yang, Ju Fan, Lei Cao, Yuyu Luo, and Alon Halevy. 2023. VerifAI: verified generative AI. *ArXiv Preprint ArXiv:2307.02796* (2023).
[80] Pei Wang and Yeye He. 2019. Uni-detect: A unified approach to automated error detection in tables. In *SIGMOD*. 811–828.
[81] 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).
[82] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS* (2022), 24824–24837.
[83] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. ZeroER: Entity resolution using zero labeled examples. In *SIGMOD*. 1149–1164.
[84] Yangyang Wu, Xiaoye Miao, Zilinghan Li, Shilan He, Xinkai Yuan, and Jianwei Yin. 2023. An efficient generative data imputation toolbox with adversarial learning. In *ICDE*. 3651–3654.
[85] 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.
[86] Haojun Zhang, Chengliang Chai, AnHai Doan, Paris Koutris, and Esteban Arcaute. 2020. Manually detecting errors for data cleaning using adaptive crowdsourcing strategies. In *EDBT*. 311–322.
[87] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *ArXiv Preprint ArXiv:2303.18223* (2023).