From Data Imputation to Data Cleaning – Automated Cleaning of Tabular Data Improves Downstream Predictive Performance

Sebastian Jäger[*,1] **Felix Bießmann**[1,2]

[1]Berlin University of Applied Sciences and Technology (BHT), Germany
[2]Einstein Center Digital Future, Berlin, Germany
*Correspondence to: sebastian.jaeger@bht-berlin.de

Abstract

The translation of Machine Learning (ML) research innovations to real-world applications and the maintenance of ML components are hindered by reoccurring challenges, such as reaching high predictive performance, robustness, complying with regulatory constraints, or meeting ethical standards. Many of these challenges are related to data quality and, in particular, to the lack of automation in data pipelines upstream of ML components. Automated data cleaning remains challenging since many approaches neglect the dependency structure of the data errors and require task-specific heuristics or human input for calibration. In this study, we develop and evaluate an application-agnostic ML-based data cleaning approach using well-established imputation techniques for automated detection and cleaning of erroneous values. To improve the degree of automation, we combine imputation techniques with conformal prediction (CP), a model-agnostic and distribution-free method to quantify and calibrate the uncertainty of ML models. Extensive empirical evaluations demonstrate that *Conformal Data Cleaning* (CDC) improves predictive performance in downstream ML tasks in the majority of cases. Our code is available on GitHub: `https://github.com/se-jaeger/conformal-data-cleaning`.

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

1 Introduction

Machine Learning (ML) components have become ubiquitous in modern software applications. While researchers have made significant progress in developing better models, much of this innovation is only slowly translated into real-world applications. Research at the intersection of ML and database management systems has identified several potential causes, including the difficulty of maintaining an ML system (Sculley et al., 2015) and challenges related to data quality (Breck et al., 2019; Schelter et al., 2018a). While many aspects of modern ML workflows have become simpler thanks to standardized APIs and libraries, data quality control remains one of the most impactful and difficult-to-automate parts of ML applications (Biessmann et al., 2021).

Here, we focus on one of the most common and relevant use cases of ML applications: we assume that an ML model was trained on clean data, and at inference time, the data quality deteriorates, impacting the predictive performance. When implementing ML systems, it is common to measure the data quality and remove erroneous examples using, e.g., outlier detection algorithms (Zhao et al., 2019). However, in many scenarios, low-quality data points cannot be discarded, and a prediction would be desirable. For this reason, one line of research focuses on training robust ML algorithms that are agnostic to data quality issues. For tabular data, these techniques, e.g., regularization (Kadra et al., 2021) and data augmentation (Machado et al., 2022), have been reported to be not as useful yet as for other data modalities (Borisov et al., 2022). On the other hand, there is a substantial body of literature in ML and database management communities spotlighting data quality and trying to detect which attributes of the examples are erroneous to enable data cleaning.

Existing cleaning approaches lack automation Database cleaning systems typically rely on user input, e.g., constraints, rules (Rekatsinas et al., 2017), or cleaning suggestions (Mahdavi and Abedjan, 2020). Mining these is challenging, and manually specifying them is, in practice, not scalable. Other cleaning methods use column-wise summary statistics to learn which values are erroneous or require tuning data-dependent thresholds (Breck et al., 2019; Schelter et al., 2018b). This approach is difficult to automate and ignores input data's dependencies between columns for downstream ML models.

Research question and hypothesis For missing values, it is common and efficient to use ML-based approaches to capture statistical dependencies between columns in a table (Rubin, 1987; Stekhoven and Buhlmann, 2012; Biessmann et al., 2019; Jäger et al., 2021) rather than considering only marginal statistics of single attributes (Breck et al., 2019), to impute them. Another advantage of modern imputation approaches is that they require little to no manual intervention. This raises the question of whether this approach generalizes to data cleaning problems. While typical ML algorithms do not account for prediction uncertainties or are poorly calibrated (Guo et al., 2017), it is difficult to detect erroneous attributes. Therefore, we hypothesize that using conformal inference (Vovk et al., 2005) to calibrate models' uncertainties allows us to test whether an attribute, i.e., cell, is erroneous and to clean if necessary. Conformal prediction (CP) is a model-agnostic and distribution-free method to calibrate ML models that give statistical guarantees on their performance.

Contributions We propose Conformal Data Cleaning (CDC) that aims at a higher degree of automation in cleaning tasks and generally ML applications by combining established imputation techniques with a statistically well-founded calibration method, conformal inference. The calibration of imputation models allows us to control the cleaning performance using a generic hyperparameter independent of the datasets and variable types, allowing fully automated application of CDC. We experimentally evaluate CDC on realistic scenarios, where ML models and CDC are trained on clean training data, and at inference time, the test data can have data errors. The benchmark consists of 16 heterogeneous tabular datasets, which we perturb with four realistic error types and five error fractions each. In the majority of these 320 experiments, CDC outperforms the baselines and improves the downstream ML task. Our code is available on GitHub: `https://github.com/se-jaeger/conformal-data-cleaning`.

2 Related Work

The term *data cleaning* has different meanings depending on the research area. Here, we briefly describe the differences and highlight how these relate to and influence our work. We focus on tabular data and do not consider methods for other data types (e.g., texts or images).

Outlier detection Outlier detection (also known as anomaly detection) identifies data points that differ substantially from others and is well-known in the machine learning (ML) community. Unsupervised ML approaches to predict whether or not a data point is an outlier has been investigated in machine learning (Ramaswamy et al., 2000; Liu et al., 2008), in work on empirical cumulative distribution function estimation (Li et al., 2020, 2022) and neural networks (Hawkins et al., 2002; Chen et al., 2017; Wang et al., 2020). The idea of outlier detection is to remove outliers, entire data points, from a dataset. In many application scenarios, a complementary goal is to detect and potentially clean anomalous attributes, meaning single dimensions of a multivariate data point or cells in a data table. In this study, we focus on this application use case.

Cell-based error detection and correction Cell-based error detection and correction focuses on errors in individual attributes of data points. This task is less studied in the statistical learning community than outlier detection. In contrast, the data management community has a large body of literature investigating cell-based error detection and correction methods for relational databases, i.e., tabular data. However, these approaches are often specialized for detecting specific error types, e.g., violations of rules, constraints or functional dependencies (Dallachiesa et al., 2013; Rekatsinas et al., 2017), inconsistencies (Ham, 2013), or (disguised) missing values ("999999" for a phone number) (Qahtan et al., 2018), and they often rely on user input. Also, in the data management community, ML methods are increasingly being used for data cleaning, employing semi-supervision (Mahdavi et al., 2019), active learning (Neutatz et al., 2019), or self-supervision (Peng et al., 2022; Liu et al., 2022) to exploit user input more efficiently. Most correction approaches can tackle all error types but use user input (e.g., Krishnan et al., 2016; Rekatsinas et al., 2017; Mahdavi and Abedjan, 2020). Abdelaal et al. (2023) show in an extensive benchmark that many of these detection and correction methods can be combined, which is similar to our approach. Since we focus on improving the automation of cleaning tasks, we build upon methods that do not rely on user input but instead rest on the assumption of clean training data

and corrupted data at inference time.

Missing data and data imputation Missing values are common data quality issues (Kumar et al., 2017). Various strategies to deal with missing values were developed in the ML community and beyond, which range from discarding rows or columns over column-wise imputing (Rubin, 1976) and imputing depending on other columns' values (Rubin, 1987; Stekhoven and Buhlmann, 2012; Biessmann et al., 2019) to deep generative models (Yoon et al., 2018; Camino et al., 2019; Yoon and Sull, 2020; Nazábal et al., 2020). In this study, we build on this work and leverage recent work in ML-based imputation for heterogeneous data as a central component in our cleaning approach.

Label error In our work, we focus on errors in the input data. Complementary approaches focus on errors corrupting the label data. For instance, Northcutt et al. (2021a) demonstrate that mislabeled examples are common in computer vision, natural language processing, and audio processing benchmarks. Therefore, different methods are available that detect these label errors (e.g., Pleiss et al., 2020; Northcutt et al., 2021b; Zhou et al., 2023; Chen et al., 2021; Zhou et al., 2023) or to correct for shifts in the label distribution (Lipton et al., 2018). This type of cleaning is important to obtain a trustworthy training dataset. It usually requires manual inspection of the cleaning results to ensure responsible usage of the models trained on the presumably clean training data.

To summarize, our work complements existing work on outlier removal, cleaning approaches developed in the database management community as well as specialized cleaning approaches for, e.g., label errors in that we focus on a high degree of automation while accounting for the statistical dependencies in the input variables as well as the errors corrupting the data.

3 Methodology

In this section, we describe the foundations of modern ML data imputation approaches and the conformal framework and how we combined both into the proposed Conformal Data Cleaning (CDC).

3.1 ML-based Data Imputation

Missing values is one of the most frequent data quality problems (Kumar et al., 2017). In recent years, ML approaches have been increasingly used to model the statistical dependencies between columns in a table to impute missing values conditioned on values in other columns of the same row. Consider a dataset represented as a table or matrix $X_{n\times d}$. The main idea behind many ML-based imputation approaches is to train an imputation model $\hat{f}_c$ for each column $c \in \{1, ..., d\}$ that predicts the value in cell i, c given the values in row i except for the value in column c, i.e.,

$$X_{i,c} = \hat{f}_c(X_{\{1,...,d\}\setminus\{c\}}) \tag{1}$$

In other words, one column of a table is considered the label and, the remaining columns serve as input data to any supervised ML model. The model classes used for $\hat{f}_c$ include k-nearest neighbors (k-NN) (Batista and Monard, 2003), random forest-based approaches (Stekhoven and Buhlmann, 2012), or discriminative deep learning methods (Biessmann et al., 2018). These imputation models can be trained along with the ML models, consuming the matrix X as input data. At inference time, when data might be corrupted, the imputation models can be applied before the downstream ML model. In practice, these models need to be well-calibrated to allow for automated cleaning. This calibration can be challenging when dealing with heterogeneous data. In the following, we discuss how to adopt conformal predictors to alleviate this problem.

3.2 Conformal Predictors

Conformal predictors are uncertainty quantification methods that allow the calculation of statistically rigorous confidence intervals (regression) or sets (classification) from any point estimator for a user-defined error rate Vovk et al. (2005). Here, we focus on inductive conformal predictors, which are computationally efficient and were quickly adapted for machine learning (ML) applications (e.g., Papadopoulos, 2008; Balasubramanian et al., 2014). For better readability, we use confidence sets and intervals interchangeably. Assume $\mathcal{D} := \mathcal{X} \times \mathcal{Y}$ is a tabular dataset for a regression or classification problem. We sample training dataset $D_{train} := \{X \times Y\}$ and calibration dataset $D_{calib} := \{X_{n\times d} \times Y_n\}$. The two datasets are independent and identically distributed (i.i.d.). We define $\hat{f}$ as a (arbitrary) predictor fitted on the training data D_{train}. To conformalize $\hat{f}$, we compute *calibration nonconformity scores* for the calibration data D_{calib} using a *nonconformity score function* S.

$$\begin{aligned} \hat{y}_{calib} &= \hat{f}(X_{calib}) \\ R_{calib} &= S(\hat{y}_{calib}, y_{calib}) \end{aligned} \tag{2}$$

Intuitionally, nonconformity scores represent how different a data point (x_i, y_i) is from what the fitted predictor $\hat{f}$ expects it to be $(x_i, \hat{y}_i)$. Since smaller scores are better, the calibration nonconformity scores R_{calib} will be relatively small if $\hat{f}$ represents D_{train} well.

Then, we compute $\hat{q}$, the k-th empirical quantile of R_{calib}, as follows:

$$k = \frac{\lceil (n+1)(1-\alpha) \rceil}{n}$$
$$\hat{q} = quantile(R_{calib}, k), \tag{3}$$

where $\alpha \in [0, 1]$ is the user-chosen error rate.[1]

Lastly, for new and unseen test data X_{test}, we need to construct the prediction set $\mathcal{C}$, which depends on the chosen nonconformity score function S. Here, for classification tasks, we use class-conditional conformal prediction (Angelopoulos and Bates, 2022) and conformalized quantile regression (Romano et al., 2019) for regression. For more details, we refer the reader to Appendix B.

Since $\hat{q}$ is based on the calibration nonconformity scores and the error rate α, the conformal framework guarantees that $\mathcal{C}(X_{test})$ contains y_{test} (the true label) with at least probability $1-\alpha$, or in other words, with a confidence level of $1-\alpha$. More formally conformal predictors (CPs) ensure that:

$$\mathbb{P}(y_{test} \in \mathcal{C}(X_{test})) \geq 1 - \alpha. \tag{4}$$

If the model $\hat{f}$ fits the data D_{train} well, the prediction sets $\mathcal{C}$ will be small. However, if $\hat{f}$ performs poorly, the prediction sets will be larger to satisfy Statement (4). This property is known as *(marginal) coverage* (Lei and Wasserman, 2014; Lei et al., 2018).

3.3 Conformal Data Cleaning (CDC)

In the following, we demonstrate that the concepts of conformal prediction can help to automate data cleaning routines. We follow an ML-based approach, introduced by van Buuren and Oudshoorn (1999), allowing us to exploit the columns' dependencies and use conformal predictors to detect which cells are erroneous, given the information of all other cells in that row. Therefore, during training, we fit an ML model for each column, where all other columns are the model's features, and calibrate its output. During deployment, we (column-wise) test which values are erroneous, meaning which value does not belong to the prediction sets, and replace them with the underlying ML point prediction.

Formally, let D^{train} be a dataset and *cleaner* our proposed method. Then, *cleaner* fits d models on a subset of the data, where $X_c^{train} = D^{train}_{\{1,...,d\}\setminus\{c\}}$ is the training data and $y_c^{train} = D_c^{train}$ the labels to fit $cleaner_c$ to clean column $c \in \{1, ..., d\}$.

Error detection For new and unseen test data $D^{test}_{n\times d}$ and error rate, e.g., $\alpha = 0.01$, each of the fitted $\widehat{cleaner}_c$ predicts confidence sets $\mathcal{C}_{i,c}$, where $\forall i \in \{1, ..., n\}$ and $\forall c \in \{1, ..., d\}$, for every test example i of the corresponding column c as follows:

$$X_{i,c}^{test} = D^{test}_{i,\{1,...,d\}\setminus\{c\}}$$
$$\mathcal{C}_{i,c} = \widehat{cleaner}_c(X_{i,c}^{test}) \tag{5}$$

If D^{test} is drawn from the same distribution as both the training and calibration data, Statement (4) holds. Hence, if $D_{i,c}^{test}$ is correct, then $\widehat{cleaner}_c$ guarantees that $\mathcal{C}_{i,c}$ contains $D_{i,c}^{test}$ with at least probability $1-\alpha$. Otherwise, we assume $D_{i,c}^{test}$ as incorrect if $D_{i,c}^{test} \notin \mathcal{C}_{i,c}$. Finally, we end up with a boolean matrix $B^{test}_{n\times d} \subset \{0, 1\}$, which represents incorrect values of D^{test} as 1.

Error cleaning After detecting errors, i.e., computing matrix B^{test}, it is straightforward to clean them. Having B^{test} representing erroneous cells of D^{test}, we can remove those and treat the situation as a missing value problem. In fact, we use the $\widehat{cleaner}$'s underlying conformalized ML models to clean the erroneous values. Instead of using $\{0, 1\}$ for B^{test}, we use the *best* prediction of these ML models. Formally, $\widehat{cleaner}$'s CPs return not only a confidence set but also the best prediction for $X_{i,c}^{test}$:

$$\mathcal{C}_{i,c}, \hat{y}_{i,c} = \widehat{cleaner}_c(X_{i,c}^{test})$$
$$B_{i,c}^{test} = \begin{cases} \hat{y}_{i,c}, & \text{if } D_{i,c}^{test} \notin \mathcal{C}_{i,c} \\ NAN, & \text{otherwise} \end{cases}, \tag{6}$$

where NAN is a placeholder representing not existing values. Replacing all values of D^{test} with B^{test} if $B_{i,c}^{test} \notin \{NAN\}$ is effectively cleaning D^{test}.

4 Implementation and Experimental Setup

In this section, we give implementation details of CDC and our baselines and describe our experimental benchmark.

4.1 Conformal Data Cleaner Implementation

We use AutoGluon (Erickson et al., 2020), allowing us to test many ML model types and optimize their hyperparameters (HPO). Depending on the columns' data type, we optimize the *pinball loss* or *F1* metric for boolean columns or *macro F1* for more than

[1] Originally, Vovk et al. (2005) calculated *p-values* for each nonconformity score. However, it has been proven that relying on the fitted residual distribution and $\hat{q}$ is equivalent (Lei et al., 2018) and commonly used in modern ML applications (Zeni et al., 2020).

two categories, respectively. Internally, AutoGluon finds the best model from random forests, extremely randomized trees, and a FastAI-based neural network (NN) (Howard and Gugger, 2020). For NNs, it further uses 50 trials (random search) to optimize their hyperparameters, where we use the default search spaces. For tree-based model types, some predefined hyperparameter settings are tested. We disable model stacking and bagging and use the best-performing model (without model ensembles) for data cleaning. For the calibration, we reserve 1000 data points[2] (not used during training) as described in Section 3.2. If a CP predicts empty sets, the input data typically is very different from the training and calibration data, meaning the underlying stationarity assumption was likely violated. In such a scenario, the conformal framework is no longer valid and loses its guarantees, and Statement (6) would mark those values as erroneous. To avoid this, we only apply Statement (6) if the prediction sets are not empty. For consistency with the ML baseline, we expect the single parameter a confidence level, i.e., $1 - \alpha$.

4.2 Baseline Implementations

No Conformal Inference To assess the impact of calibration, we implemented a cleaning baseline that uses ML imputation models without conformal inference but otherwise applied in the same way, as described in Section 3.3. For a fair comparison, we apply alternative thresholding mechanisms often used in practice. Categorical variables were defined as errors if the imputation model did not reach a given precision threshold p. For numerical columns, we also use quantile regression to fit D_{train}'s *lower* $q_{lo} = \frac{1-p}{2}$ and *upper* $q_{up} = 1 - \frac{1-p}{2}$ empirical quantiles, where $p \in [0, 1]$ is a given parameter representing the (empirical) range defined as correct. Therefore, the hyperparameter represents how certain the cleaner has to be to mark a value as erroneous.

Garf Garf (Peng et al., 2022) uses logical data cleaning (functional dependencies), which is popular in the database management community. However, in particular, it uses a sequence of generative adversarial networks (SeqGAN) to learn dependencies between columns and generate data repair rules. Garf applies these learned rules iteratively to clean dirty data and, at the same time, uses the relative confidence between data and rules to improve both. Therefore, it does not need any user input and directly works on dirty datasets. For the implementation, we used default hyperparameters and made some minor adaptions to import/export our benchmark (cf. Section 4.3) into/from an SQLite database. Unfortunately, Garf casts the datasets into strings, i.e., categorical data, which potentially decreases its performance.

4.3 Experimental Setup

We use 16 openly available datasets from OpenML (Vanschoren et al., 2014) for the three most common task types: regression, binary classification, and multi-class classification that fulfill the criteria for tabular datasets formulated by Grinsztajn et al. (2022). Appendix A lists our datasets and gives information about their size and columns. We split them 80/20 into training and test sets and applied *Jenga*[3] (Schelter et al., 2021) to each of the test sets multiple times to create several corrupted test sets. We use four error types (swapping values between columns, random scaling, Gaussian noise, and shifts of categorical value distributions) with 1%, 5%, 10%, 30%, and 50% error fractions, which results in 20 corrupted test sets for each dataset. Since we do not mix the error types, we end up with a wide range of error fractions (not every error type can be applied to each column) ranging from 0% to about 41% with $11\% \pm 14$ errors on average.

To simulate a real-world scenario, we train the cleaning methods and a downstream model[4] on the high-quality training datasets. For CDC and the ML baseline, we experiments with different parameters (*confidence level* of CDC and p for ML baseline). Since Garf is designed to work on corrupted data, we only train the downstream model on the clean training data and use Garf to clean the corrupted test sets. All experiments are repeated three times with different seeds – for CDC, this leads to resampling the 1000 calibration data points for each repetition.

5 Results

We evaluate the cleaning performance as the predictive performance on a variety of downstream tasks after applying the cleaning procedure to a corrupted test data set. Depending on the task, we use *F1* for binary classification, *macro F1* for multi-class classification,

[2] Angelopoulos and Bates (2022) discussed the effect of the calibration set size and show that 1000 data points works well.

[3] "Jenga is an experimentation library that allows data science practitioners and researchers to study the effect of common data corruptions". GitHub: `https://github.com/schelterlabs/jenga`

[4] We use Jenga, which builds up on scikit-learn's `SGDClassifier` for classification or `SGDRegressor` for regression tasks, pre-processes the columns (scaling, missing value imputation, and one-hot encoding), and optimizes hyperparameters (grid search).

and *root mean square error* (*RMSE*) for regression datasets to report the performance of the downstream model. To compare the overall performance of the cleaning methods, we calculate the downstream performance improvement (corrupted vs. cleaned) relative to the corrupted performance. The average coefficient of variation ($\frac{\sigma}{\mu}$) over the three experiment repetitions for the downstream performance is $1.84\% \pm 2.84$. If not stated otherwise, we report the averaged results over the three repetitions (with different seeds). For five datasets (see Appendix A), Garf does not produce valid cleaned datasets. Either it introduces new symbols that lead to failing downstream models, or it removes data points from the dirty data. For this reason, we removed those results from the following evaluations.

5.1 Error Detection

To report the outlier detection performance, we use the *true positive rate (TPR)*, i.e., probability of detection, and the *false positive rate (FPR)*, i.e., probability of false alarm. Figure 1 visualizes the outlier detection performance. We binned the error fractions for better visualization and averaged the results.

True positive rate (↑) With lower values for ML's and CDC's hyperparameters, they can detect more erroneous values. In almost all cases, CDC has higher TPR and is more robust against the error fraction of the datasets. In detail, CDC detects in $\sim 43\%$ of the experiments more erroneous values than the ML baseline. On the other hand, Garf is in all cases worse than CDC and ML.

False positive rate (↓) In general, CDC's and ML's hyperparameters have more influence on the FPR, which is why their results are further apart. This is more pronounced for ML than for CDC, and, again, its results depend more on the error fraction. More precisely, CDC produces in $\sim 80\%$ of the experiments fewer false alarms than the ML baseline. In contrast to the TPR, Garf is among the best methods concerning to the FPR.

An optimal error detector would detect all erroneous values, i.e., $TPR = 1$, and does not produce any false alarms, i.e., $FPR = 0$. Figure 1 illustrates the trade-off between TPR and FPR.

5.2 Error Cleaning

Figure 2 shows the downstream performance improvement (↑) in percent relative to the corrupted performance. For better visualization, we show the median values of the results that belong to the same range of error fractions. Median values give insight into the minimal downstream performance improvement we can expect in the majority of experiments[5].

The more errors exist, the higher the potential improvement of downstream performance in contrast to not using any cleaning method. Similarly to the error detection performance, CDC is more robust regarding the error fraction. Especially with $(30 - 40]\%$ errors, where the majority of ML results degrade the downstream performance, CDC maintains more than 15% downstream improvements. Overall, in $\sim 61\%$ of the experiments, CDC leads to better downstream improvements than ML. Garf does not lead to improvements or degrade the downstream performance.

5.3 Confidence Set Size

As described in Section 3.2, conformal predictors vary the confidence set size to express their prediction uncertainties. Here, we normalize the set sizes to the valid range, i.e., the cardinality of categorical or the range of numerical columns, and compute the average *relative confidence set size* (↓) for each experiment. Figure 3 shows CDC's downstream improvement, similar to Figure 2, but visualizes the 20% easiest (smallest sets), 60% moderately difficult, and the 20% most difficult (largest sets) of the experiments separately. To make the results comparable, we controlled for the confidence level because it directly influences the confidence set size.

The 20% most difficult experiments mostly degrade the downstream performance, often by several tens of percent. This is more pronounced with smaller confidence levels, e.g., CDC with a confidence level of 0.999, leads in $\sim 71\%$ of the experiments to improvements, with 0.8 in $\sim 18\%$ and with 0.5 in $\sim 15\%$. Similarly, for the easiest 20% of experiments, with a confidence level of 0.999, we see better improvements. However, in the midfield, this effect vanishes.

6 Discussion

We investigate the performance of CDC and the baselines on many datasets and a wide range of artificially created but realistic errors. In the following, we highlight some of the key findings. Evaluations of the underlying model types can be found in Appendix C.

[5]Additional visualizations and results can be found in Appendix D

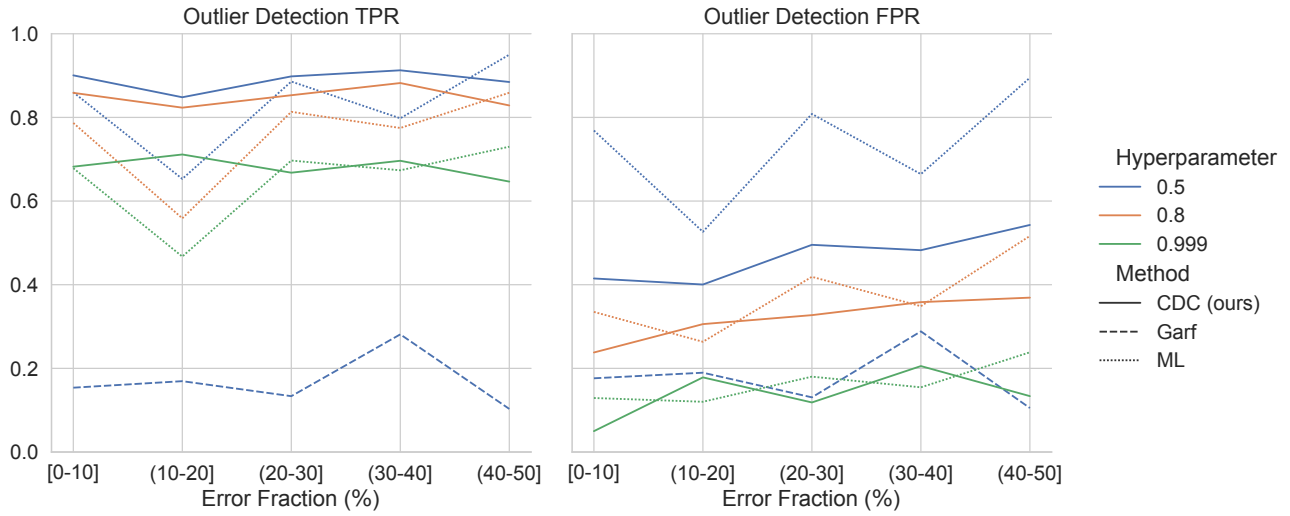

Figure 1: (*Left*) Outlier Detection TPR (↑) vs. (*Right*) FPR (↓). Shows the average values for all experiments, which fall into the binned error fractions. The lower the hyperparameter for ML and CDC, the more erroneous values they detect – but CDC is more robust against the error fraction. On the other hand, larger hyperparameters lead to smaller FPR.

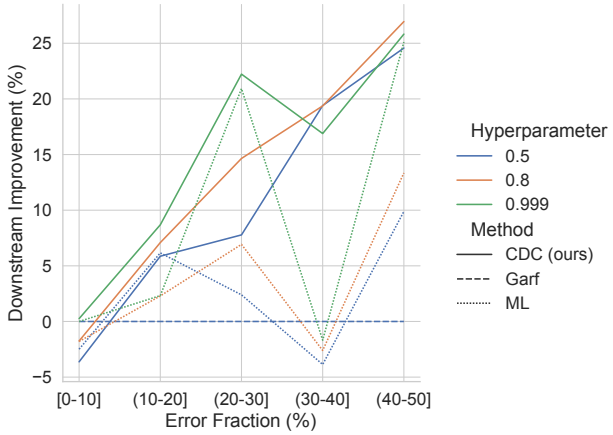

Figure 2: Downstream Improvement. Shows the median of the results that belong to the same range of error fractions to visualize the expected improvement in the majority of experiments. CDC with high hyperparameters outperforms others in most cases. Garf does not lead to worth mentioning improvements.

6.1 CDC Outperforms Baselines in Most Experiments

Garf Since our benchmark consists of heterogeneous datasets, but Garf treats the data as categorical, it is expected to perform much worse than the others (cf. Figure 1). On the other hand, it does not introduce too many errors (small FPR) and, therefore, does not degrade the downstream performance (cf. Figure 2).

ML baseline Figures 1 and 2 indicate that CDC performs in most experiments better than the ML baseline. Comparing them one by one shows this is true in more than 61% of the experiments. Besides this, CDC is more robust regarding the error fraction and its hyperparameter, as shown by less dispersed results in Figure 2. Interestingly, for both methods, their hyperparameter influences the detection performance (ML more than CDC), but the amount of erroneous values only slightly influences the detection performance. This is why the downstream improvement increases with the amount of erroneous values.

6.2 Influence of CDC's Hyperparameter *Confidence Level*

The user-defined confidence level directly influences the confidence prediction set sizes. High confidence levels typically lead to larger prediction sets to satisfy its coverage guarantee. Therefore, the higher the confidence level, the fewer errors are detected (decreasing TPR), but it also prevents false alarms (smaller FPR) (cf. Figure 1). Even though detecting erroneous values is only the first step and does not guarantee downstream improvements, in many scenarios (cf. Figure 2), higher confidence levels lead to better downstream improvements. To be precise, on average, in more than 66% of the experiments, increasing the confidence level improves the downstream performance. Especially based on Figure 3, we argue that CDC with high confidence levels mitigates the worst cases. The higher the confidence level, the better the downstream improvements, we can expect, which es-

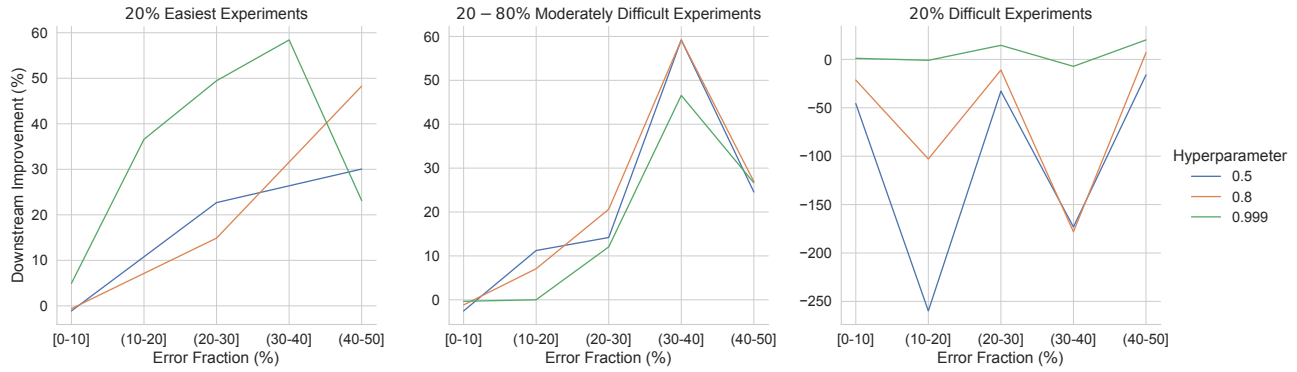

Figure 3: CDC Downstream Improvement depends on the experiment's difficulty. We show the median results to visualize the expected improvement in the majority of experiments. If experiments are relatively difficult (*Right*), cleaning degrades the downstream performance. On the other hand, for moderately difficult (*Middle*) or easy (*Left*) experiments, we can expect downstream improvements (cf. Figure 2).

pecially applies to the more difficult experiments.

6.3 CDC's Relative Confidence Set Size

In contrast to the ML baseline, we can detect which experiments (or single data points) are more difficult than others by sorting them according to their relative confidence set size. The relative confidence set sizes and the downstream improvements correlate negatively. More precisely, about -0.61 for confidence level 0.5 until about -0.17 for 0.999. This supports our argument and shows that CDC mitigates bad results with high confidence levels. Further, this allows us to detect data points (or whole experiments) that potentially lead to worse downstream performance. Therefore, besides automated cleaning, CDC can be used for advanced monitoring, e.g., whether the data shifts (cf. Schelter et al., 2020). However, this aspect is not part of this study and will be addressed in future work.

6.4 Limitations

In this study, we use tabular datasets as defined by Grinsztajn et al. (2022) with five to 15 columns (categorical, numerical, and mixed) and about $4,800$ to $89,000$ rows without missing values to benchmark three common downstream tasks: regression, binary classification, and multi-class classification. Thus, our approach and results can not be transferred to other data modalities, such as text- or image-based datasets. However, since we benchmarked CDC on a wide range of dataset sizes, we assume that it generalizes well to larger or smaller tabular datasets. As described in Section 4.3, we assume that high-quality training data is available and data errors are tackled at inference time. While this assumption is often violated, we believe that it is a sensible simplification of the problem: First, it allows for substantial improvements in the degree of automation, and our results demonstrate that cleaning will improve the downstream predictive performance. Second, the curation of training data is often a necessary part of model development cycles and an essential step to ensure responsible usage of ML components. The curated training data can usually be used for training the cleaning models without additional curation efforts. Other approaches focus on application scenarios where there is no high-quality training data available to calibrate the cleaning models. Such cleaning systems often circumvent the assumption of high-quality training data by requiring additional user input (e.g., Mahdavi et al., 2019; Mahdavi and Abedjan, 2020; Neutatz et al., 2019; Krishnan et al., 2017, 2016).

7 Conclusion and Future Work

In this study, we present how conformalized machine learning models (CDC) can be used to detect and clean erroneous values of heterogeneous tabular data without requiring user input. We benchmark CDC on 16 datasets in a comprehensive suite of experiments with realistic data corruptions. Our results show that in about 61% of our experiments, CDC outperforms the baselines, and using high confidence levels improves the downstream performance in about 60% of the cases without any manual effort in the data cleaning procedure. These results highlight the potential of automated imputation methods combined with modern calibration methods.

In future work, it could be interesting to test an iterative cleaning approach similar to multiple imputation Rubin (1987), which has the potential to further increase CDC's performance, especially with many er-

roneous values. Further, we plan to investigate how CDC can be applied for both performance monitoring and data cleaning.

Acknowledgements

We thank the anonymous reviewers for their helpful and constructive feedback and Philipp Jung for valuable discussions. This research was supported by the German Federal Ministry of the Environment grant number 67KI2022A and the German Federal Ministry of Education and Research grant number 16SV8856.

References

Abdelaal, M., Hammacher, C., and Schoening, H. (2023). Rein: A comprehensive benchmark framework for data cleaning methods in ml pipelines. *Proceedings of the VLDB Endowment (PVLDB)*.

Angelopoulos, A. N. and Bates, S. (2022). A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. arXiv:2107.07511 [cs, math, stat].

Balasubramanian, V., Ho, S.-S., and Vovk, V. (2014). *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Batista, G. E. A. P. A. and Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6):519–533.

Biessmann, F., Golebiowski, J., Rukat, T., Lange, D., and Schmidt, P. (2021). Automated data validation in machine learning systems. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*.

Biessmann, F., Rukat, T., Schmidt, P., Naidu, P., Schelter, S., Taptunov, A., Lange, D., and Salinas, D. (2019). DataWig: Missing Value Imputation for Tables. *Journal of Machine Learning Research*, 20(175):1–6.

Biessmann, F., Salinas, D., Schelter, S., Schmidt, P., and Lange, D. (2018). "Deep" Learning for Missing Value Imputationin Tables with Non-Numerical Data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 2017–2025, New York, NY, USA. Association for Computing Machinery.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. (2022). Deep Neural Networks and Tabular Data: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

Breck, E., Polyzotis, N., Roy, S., Whang, S., and Zinkevich, M. (2019). Data Validation for Machine Learning. *Proceedings of Machine Learning and Systems*, 1:334–347.

Camino, R. D., Hammerschmidt, C. A., and State, R. (2019). Improving Missing Data Imputation with Deep Generative Models. *arXiv:1902.10666 [cs, stat]*. arXiv: 1902.10666.

Chen, D., Yu, Z., and Bowman, S. R. (2021). Clean or Annotate: How to Spend a Limited Data Collection Budget.

Chen, J., Sathe, S., Aggarwal, C., and Turaga, D. (2017). Outlier Detection with Autoencoder Ensembles. In *Proceedings of the 2017 SIAM International Conference on Data Mining (SDM)*, Proceedings, pages 90–98. Society for Industrial and Applied Mathematics.

Dallachiesa, M., Ebaid, A., Eldawy, A., Elmagarmid, A., Ilyas, I. F., Ouzzani, M., and Tang, N. (2013). NADEEF: a commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 541–552, New York, NY, USA. Association for Computing Machinery.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., and Smola, A. (2020). AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. arXiv:2003.06505 [cs, stat].

Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In *NeurIPS 2022 Datasets and Benchmarks Track*, New Orleans, United States.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330. PMLR. ISSN: 2640-3498.

Ham, K. (2013). OpenRefine (version 2.5). http://openrefine.org. Free, open-source tool for cleaning and transforming data. *Journal of the Medical Library Association : JMLA*, 101(3):233. Publisher: Medical Library Association.

Hawkins, S., He, H., Williams, G., and Baxter, R. (2002). Outlier Detection Using Replicator Neural Networks. In Kambayashi, Y., Winiwarter, W., and Arikawa, M., editors, *Data Warehousing and Knowledge Discovery*, Lecture Notes in Computer Science, pages 170–180, Berlin, Heidelberg. Springer.

Howard, J. and Gugger, S. (2020). Fastai: A Layered API for Deep Learning. *Information*, 11(2):108.

Jäger, S., Allhorn, A., and Bießmann, F. (2021). A benchmark for data imputation methods. *Frontiers in Big Data*, 4.

Kadra, A., Lindauer, M., Hutter, F., and Grabocka, J. (2021). Well-tuned Simple Nets Excel on Tabular Datasets. In *Advances in Neural Information Processing Systems*, volume 34, pages 23928–23941. Curran Associates, Inc.

Krishnan, S., Franklin, M. J., Goldberg, K., and Wu, E. (2017). BoostClean: Automated Error Detection and Repair for Machine Learning. arXiv:1711.01299 [cs].

Krishnan, S., Wang, J., Wu, E., Franklin, M. J., and Goldberg, K. (2016). ActiveClean: interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, 9(12):948–959.

Kumar, A., Boehm, M., and Yang, J. (2017). Data Management in Machine Learning: Challenges, Techniques, and Systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1717–1722, New York, NY, USA. Association for Computing Machinery.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., and Wasserman, L. (2018). Distribution-Free Predictive Inference for Regression. *Journal of the American Statistical Association*, 113(523):1094–1111. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01621459.2017.1307116.

Lei, J. and Wasserman, L. (2014). Distribution-free prediction bands for non-parametric regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1):71–96.

Li, Z., Zhao, Y., Botta, N., Ionescu, C., and Hu, X. (2020). COPOD: Copula-Based Outlier Detection. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1118–1123. ISSN: 2374-8486.

Li, Z., Zhao, Y., Hu, X., Botta, N., Ionescu, C., and Chen, G. (2022). ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

Lipton, Z., Wang, Y.-X., and Smola, A. (2018). Detecting and correcting for label shift with black box predictors. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3122–3130. PMLR.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. ISSN: 2374-8486.

Liu, Z., Zhou, Z., and Rekatsinas, T. (2022). Picket: guarding against corrupted data in tabular data during learning and inference. *The VLDB Journal*, 31(5):927–955.

Machado, P., Fernandes, B., and Novais, P. (2022). Benchmarking Data Augmentation Techniques for Tabular Data. In Yin, H., Camacho, D., and Tino, P., editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2022*, Lecture Notes in Computer Science, pages 104–112, Cham. Springer International Publishing.

Mahdavi, M. and Abedjan, Z. (2020). Baran: effective error correction via a unified context representation and transfer learning. *Proceedings of the VLDB Endowment*, 13(12):1948–1961.

Mahdavi, M., Abedjan, Z., Castro Fernandez, R., Madden, S., Ouzzani, M., Stonebraker, M., and Tang, N. (2019). Raha: A Configuration-Free Error Detection System. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 865–882, New York, NY, USA. Association for Computing Machinery.

Nazábal, A., Olmos, P. M., Ghahramani, Z., and Valera, I. (2020). Handling incomplete heterogeneous data using VAEs. *Pattern Recognition*, 107:107501.

Neutatz, F., Mahdavi, M., and Abedjan, Z. (2019). ED2: A Case for Active Learning in Error Detection. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 2249–2252, New York, NY, USA. Association for Computing Machinery.

Northcutt, C., Athalye, A., and Mueller, J. (2021a). Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1.

Northcutt, C., Jiang, L., and Chuang, I. (2021b). Confident Learning: Estimating Uncertainty in Dataset Labels. *Journal of Artificial Intelligence Research*, 70:1373–1411.

Papadopoulos, H. (2008). *Inductive Conformal Prediction: Theory and Application to Neural Networks*. IntechOpen. Publication Title: Tools in Artificial Intelligence.

Peng, J., Shen, D., Tang, N., Liu, T., Kou, Y., Nie, T., Cui, H., and Yu, G. (2022). Self-Supervised and Interpretable Data Cleaning with Sequence Generative Adversarial Networks. *Proceedings of the VLDB Endowment*, 16(3):433–446.

Pleiss, G., Zhang, T., Elenberg, E., and Weinberger, K. Q. (2020). Identifying Mislabeled Data using the

Area Under the Margin Ranking. In *Advances in Neural Information Processing Systems*, volume 33, pages 17044–17056. Curran Associates, Inc.

Qahtan, A. A., Elmagarmid, A., Castro Fernandez, R., Ouzzani, M., and Tang, N. (2018). FAHES: A Robust Disguised Missing Values Detector. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 2100–2109, New York, NY, USA. Association for Computing Machinery.

Ramaswamy, S., Rastogi, R., and Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record*, 29(2):427–438.

Rekatsinas, T., Chu, X., Ilyas, I. F., and Ré, C. (2017). HoloClean: holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment*, 10(11):1190–1201.

Romano, Y., Patterson, E., and Candes, E. (2019). Conformalized Quantile Regression. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, 63(3):581–592.

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley.

Schelter, S., Biessmann, F., Januschowski, T., Salinas, D., Seufert, S., and Szarvas, G. (2018a). On Challenges in Machine Learning Model Management. *Bull. IEEE Comput. Soc. Tech. Comm. Data Eng.*, pages 5–13.

Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., and Grafberger, A. (2018b). Automating large-scale data quality verification. *Proc. VLDB Endow.*, 11(12):1781–1794.

Schelter, S., Rukat, T., and Biessmann, F. (2020). Learning to Validate the Predictions of Black Box Classifiers on Unseen Data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1289–1299, Portland OR USA. ACM.

Schelter, S., Rukat, T., and Biessmann, F. (2021). JENGA - A Framework to Study the Impact of Data Errors on the Predictions of Machine Learning Models. In *Proceedings of the 24th International Conference on Extending Database Technology, {EDBT} 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 529–534. OpenProceedings.org.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. *Adv. Neural Inf. Process. Syst.*, 2015-Janua:2503–2511.

Stekhoven, D. J. and Buhlmann, P. (2012). MissForest–non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118.

van Buuren, S. and Oudshoorn, K. (1999). *Flexible Multivariate Imputation by MICE*, volume (PG/VGZ/99.054). TNO Prevention and Health.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2014). OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60.

Vovk, V., Gammerman, A., and Shafer, G. (2005). *Algorithmic learning in a random world*. Springer, New York.

Wang, H., Pang, G., Shen, C., and Ma, C. (2020). Unsupervised Representation Learning by Predicting Random Distances. arXiv:1912.12186 [cs, stat].

Yoon, J., Jordon, J., and van der Schaar, M. (2018). GAIN: Missing Data Imputation using Generative Adversarial Nets. Technical Report arXiv:1806.02920, arXiv. arXiv:1806.02920 [cs, stat] type: article.

Yoon, S. and Sull, S. (2020). GAMIN: Generative Adversarial Multiple Imputation Network for Highly Missing Data. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8453–8461. ISSN: 2575-7075.

Zeni, G., Fontana, M., and Vantini, S. (2020). Conformal Prediction: a Unified Review of Theory and New Challenges. Technical Report arXiv:2005.07972, arXiv. arXiv:2005.07972 [cs, econ, stat] type: article.

Zhao, Y., Nasrullah, Z., and Li, Z. (2019). PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*, 20(96):1–7.

Zhou, H., Mueller, J., Kumar, M., Wang, J.-L., and Lei, J. (2023). Detecting Errors in Numerical Data via any Regression Model. arXiv:2305.16583 [cs, stat].

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. – Yes, we try to describe our setting as precise as possible.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. – Not Applicable.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. – Yes, the source code is part of our contribution and will be publicly available on GitHub.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. – Not Applicable.
   (b) Complete proofs of all theoretical results. – Not Applicable.
   (c) Clear explanations of any assumptions. – Not Applicable.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). – Yes, the source code is part of our contribution and will be publicly available on GitHub.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). – Yes, as described in Section 4.3.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). – Yes.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). – Yes, the source code is part of our contribution and will be publicly available on GitHub.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. – Yes, we use and cite these assets: OpenML (Vanschoren et al., 2014), AutoGluon (Erickson et al., 2020), Jenga (Schelter et al., 2021), and Garf (Peng et al., 2022)
   (b) The license information of the assets, if applicable. – Not Applicable, regulated by licenses. Please see the citations above.
   (c) New assets either in the supplemental material or as a URL, if applicable. – Not Applicable.
   (d) Information about consent from data providers/curators. – Not Applicable, regulated by licenses.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. – Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. – Not Applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. – Not Applicable.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. – Not Applicable.

A Datasets

Table 1: Datasets overview. *ID* is the assigned OpenML id, # means the number of, *Cat.* and *Num.* stand for categorical and numerical columns, and *Obs.* means observations, i.e., the number of rows of the tabular dataset. *Garf* shows whether or not Garf was able to clean the dataset.

ID	Task Type	#Cat.	#Num.	#Obs.	#Cells	Garf
251	Binary	1	8	39,366	354,294	✗
310	Binary	1	5	11,183	67,098	✓
725	Binary	1	7	8,192	65,536	✓
823	Binary	1	7	20,640	165,120	✓
1046	Binary	1	4	15,545	77,725	✗
4135	Binary	5	4	32,769	294,921	✗
42493	Binary	4	3	26,969	188,783	✓
30	Multi Class	1	9	5,473	54,730	✓
40498	Multi Class	9	2	4,898	53,878	✓
198	Regression	4	2	9,517	57,102	✓
218	Regression	0	8	22,784	182,272	✗
1193	Regression	7	2	31,104	279,936	✓
1199	Regression	3	6	17,496	157,464	✓
1200	Regression	0	9	59,049	531,441	✗
23515	Regression	0	6	10,081	60,486	✓
42225	Regression	6	3	53,940	485,460	✓

B Conditional Conformal Prediction

B.1 Class-Conditioned Conformal Classification

For classification problems (especially with skewed label spaces), it is desirable to get guaranteed coverage for each class, i.e., class-conditioned conformal classification[6]. In this approach, calibration nonconformity scores are stratified by class, and multiple $\hat{q}^{(c)}$ are calculated. The following equations represent the necessary adaptions, where $\bullet^{(c)}$ represents the subset for class c.

$$\begin{aligned} R_{calib}^{(c)} &= S\Big(\hat{y}_c^{(c)}\Big) \\ k^{(c)} &= \frac{\lceil (n^{(c)}+1)(1-\alpha)) \rceil}{n^{(c)}} \\ \hat{q}^{(c)} &= quantile\Big(R_{calib}^{(c)}, k^{(c)}\Big) \\ \mathcal{C}(X_{test}) &= \Big\{ c : S(\hat{y}_{test_c}) < \hat{q}^{(c)} \Big\} \end{aligned} \tag{7}$$

A class-conditioned conformal classifier is guaranteed to satisfy the stronger *class-conditioned* coverage:

$$\mathbb{P}(y_{test} \in \mathcal{C}(X_{test}) \mid y_{test} = y) \geq 1 - \alpha, \quad \forall y \in \mathcal{Y}. \tag{8}$$

B.2 Conformal Quantile Regression

Conformal Quantile Regression (CQR) predicts confidence intervals that can vary depending on x and, therefore, adapt to the aleatoric uncertainty of the data. It fits a model to D_{train}'s *lower* $q_{\alpha_{lo}} = \alpha/2$ and *upper* $q_{\alpha_{up}} = 1 - \alpha/2$ empirical quantiles. Using the nonconformity score function $S(\hat{y}, y) = max(\hat{y}_{\alpha_{lo}} - y, y - \hat{y}_{\alpha_{up}})$ in the above-described conformal framework and computing the prediction intervals as $\mathcal{C}(X_{test}) = [\hat{y}_{\alpha_{lo}} - \hat{q}, \hat{y}_{\alpha_{up}} + \hat{q}]$, it is possible to calibrate the predicted quantiles to satisfy Statement (4). For proof or intuitions about the score function, we refer the reader to Romano et al. (2019).

[6]Class-conditioned conformal predictors are also known as *mondrian conformal predictors*. For details and proofs, we refer the reader to Vovk et al. (2005).

C Tree-Based Models Perform Best in the Majority of Cases

To clean values, CDC and the ML baseline fit one ML model for every column. As mentioned in Section 4.1, we use AutoGluon for our experiments to find the best model-hyperparameter combination. In about 18% of these cases, AutoGluon finds a FastAI NN as best performing, in about 46% an extremely randomized tree (XT), and in about 37% a random forest (RF). Given the fact that for FastAI NN, we optimize 50 different hyperparameter settings (random search) but only three for RF and XF each, it is surprising that tree-based models (83%) outperform FastAI NN. However, for data imputation using the same approach, Jäger et al. (2021) already showed that RFs work well, which is in line with a study by Grinsztajn et al. (2022). They provide evidence that tree-based models still outperform neural networks on tabular data. In the future, CDC's performance could be increased by focusing on tree-based models and optimizing their hyperparameters, which we leave for future research.

D Error Cleaning Results

Besides the results described and shown in Section 5.2, Figure 4 also gives insights into the results' spread. Increasing hyperparameter values lead typically to better downstream improvements. In most cases, not only CDC's median improvement is higher than the ML baseline, also its lower quartile, while the boxes are generally shorter, meaning its results are more robust.

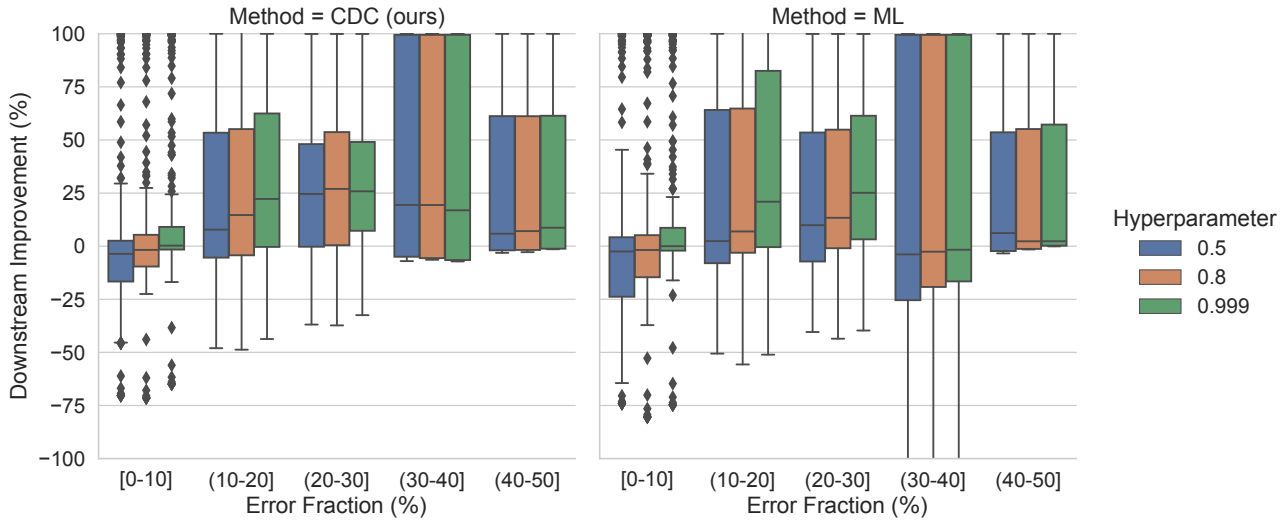

Figure 4: Downstream Improvement. Box plots visualize five summary statistics (min, max, first quartile, median, and second quartile) to give insights about the results' spread. Because Garf does not lead to worth mentioning improvements, its results are omitted.

The drop performance drop for the ML baseline with $(30 - 40]\%$ errors is due to the fact that here are only seven experiments, whereas for others are more than 14. For experiments with 0% errors, CDC leads to a slight degradation of downstream performance with a median of 0.25%, whereas the degradation for the ML baseline is 0.93%.