Datenbank Spektrum (2022) 22:121–130
https://doi.org/10.1007/s13222-022-00413-2

SCHWERPUNKTBEITRAG

Data Cleaning and AutoML: Would an Optimizer Choose to Clean?

Felix Neutatz[1] · Binger Chen[1] · Yazan Alkhatib[2] · Jingwen Ye[2] · Ziawasch Abedjan[2,3]

Received: 31 January 2022 / Accepted: 20 April 2022 / Published online: 13 May 2022
© The Author(s) 2022

Abstract
Data cleaning is widely acknowledged as an important yet tedious task when dealing with large amounts of data. Thus, there is always a cost-benefit trade-off to consider. In particular, it is important to assess this trade-off when not every data point and data error is equally important for a task. This is often the case when statistical analysis or machine learning (ML) models derive knowledge about data. If we only care about maximizing the utility score of the applications, such as accuracy or F1 scores, many tasks can afford some degree of data quality problems. Recent studies analyzed the impact of various data error types on vanilla ML tasks, showing that missing values and outliers significantly impact the outcome of such models. In this paper, we expand the setting to one where data cleaning is not considered in isolation but as an equal parameter among many other hyper-parameters that influence feature selection, regularization, and model selection. In particular, we use state-of-the-art AutoML frameworks to automatically learn the parameters that benefit a particular ML binary classification task. In our study, we see that specific cleaning routines still play a significant role but can also be entirely avoided if the choice of a specific model or the filtering of specific features diminishes the overall impact.

Keywords Machine Learning · Data Cleaning · AutoML

1 Introduction

Data cleaning is considered to be an essential pre-processing step to ensure that subsequent data analysis is correct [18, 21]. Addressing data quality issues, such as mislabels or distribution skew can significantly improve model accuracy. In the data-centric AI competition, data scientists were able to improve the overall accuracy by more than 20% only through data engineering steps, such as removing mislabeled instances or adding augmented instances [22]. Furthermore, previous studies showed that cleaning a dataset from outliers and missing values (MV) can improve downstream ML classification performance [15]. The results of these studies suggest that pre-processing data and removing data quality issues is necessary for any pipeline. Yet, these studies do not consider the impact of data cleaning in conjunction with other optimizations. For example, it is well known that specific models are more robust towards MV and mislabels, or that feature engineering filters irrelevant data for a task. The aforementioned study evaluated the impact of data errors on single models without considering other parts of the ML pipeline, such as data encoding and other pre-processing steps. As a result, the overall impact of data cleaning on the entire process is maximally noticeable. As most real-world ML pipelines include other pre-processing steps, it is worth studying the overall impact of data cleaning routines in a holistic setting. Typically, such a composition requires a careful analysis of the task and dataset at hand and can be very tedious for an ML engineer. With the help of AutoML it is possible to obtain a composition of an ML pipeline that optimizes the accuracy metric of the ML task at hand by choosing the most fitting hyperparameters (HP) of such a process, including choosing the data encoding,

✉ Felix Neutatz
f.neutatz@tu-berlin.de

Binger Chen
chen@tu-berlin.de

Yazan Alkhatib
y.alkhatib@stud.uni-hannover.de

Jingwen Ye
jingwen.ye@stud.uni-hannover.de

Ziawasch Abedjan
abedjan@dbs.uni-hannover.de

[1] TU Berlin, Berlin, Germany

[2] Leibniz Universität Hannover, Hannover, Germany

[3] L3S Research Center, Hannover, Germany

the feature preprocessing, and the ML model, and all the corresponding HPs.

In this paper, we want to explore the interaction between data cleaning and other hyper-parameters of ML pipelines for supervised binary classification tasks. First, we reevaluate the CleanML benchmark using state-of-the-art AutoML systems [6]. We run AutoML for both the dirty and the clean state of all CleanML datasets. As AutoML optimizes the whole pipeline, we avoid measurement artifacts of static preprocessing. For instance, the CleanML benchmark always uses standardization based on the mean and the standard deviation that is prone to outliers. An AutoML system might choose a different standardization. Also, a dataset might have a large number of errors, but if these errors only appear in unimportant features, the ML models will ignore the features. Second, we present our first approach to bringing more advanced cleaning operations to AutoML. Current AutoML systems offer only limited cleaning capabilities, such as mean imputation. Therefore, we extend the state-of-the-art AutoML system AutoSklearn [6] with more advanced cleaning algorithms, which we call AutoClean (source code: [19]). Specifically, we add four outlier detection strategies and two advanced MV imputation strategies. To achieve this, we address the following challenges:

Data-dependent HP Space. AutoSklearn only supports data-independent HPs. However, for outlier detection, we need to train one outlier detection strategy per feature because each feature might require different outlier thresholds. This challenge is aggravated by the huge number of HPs because, for each numerical feature, we need to identify whether to detect outliers, which strategy to use, and which parameters for each strategy.

Cleaning Overhead. The additional time for training more advanced cleaning strategies is competing with the time needed to train the ML models. So, if data cleaning takes too much time, the model training is neglected. This issue is aggravated in outlier detection because we need to train an outlier detection model for every feature.

To reduce these challenges, we extend AutoSklearn with the following contributions:

Hierarchical Search Space. As we add data-dependent HPs in AutoSklearn, we structure the HP as hierarchical as possible to prune as many HPs as early as possible. The reason is that many HPs are dependent on each other. For instance, the HP of the number of k neighbors depends on whether the KNN classifier was chosen in the first place. So, if we structure the search space hierarchically, we do not need to draw values for the k neighbors HP, if the KNN classifier is not chosen.

Sampling HP. To avoid the significant overhead of data cleaning operations, we add a sampling HP for each cleaning component. This way we only train the cleaning operations on a subset of the data if necessary. Instead of picking a sampling threshold ad hoc, we let the AutoML system choose the best sampling degree. Instead of drawing values for this HP from a uniform distribution, we draw them from a logarithmic one to push the system to prefer small subsets.

Main Findings. Our study leads to the following conclusions: First, we did not find a significant impact of data errors present in the CleanML benchmark when applying AutoML. AutoML can adjust the ML pipeline in various ways to work around or ignore the data errors, even for label errors. In the worst case, for 10% mislabels, we saw a 5% balanced accuracy decrease. Second, more advanced data cleaning preprocessors, such as outlier detection, did not improve the ML performance. However, the newly added imputation strategy based on k-nearest neighbors (KNN) was chosen frequently and could be a valuable imputation alternative for state-of-the-art AutoML systems. Third, most current benchmark datasets only contain few errors without impact. Therefore, we need to design benchmarks that contain impactful real-world errors and analyze the results on the feature level.

2 Related Work

In this section, we discuss existing entanglements of data cleaning in ML and robust ML that addresses data errors without cleaning.

Data Cleaning for AutoML. AutoML systems automatically search for ML pipelines that maximize the validation accuracy [6]. Thus, it can be used to automatically evaluate the overall impact of data cleaning routines in the whole ML pipeline. Some existing AutoML frameworks already include data cleaning mechanisms. For example, AutoGluon is a Python library for AutoML with tabular data [5]. It automatically performs MV imputation and outlier detection before the ML pipeline in both a model-agnostic and model-specific way. The employed methods are limited to basic methods, such as median imputation, creating an "Unknown" category, quantile normalization, and mean-zero unit-variance rescaling. Besides, in AutoGluon, data cleaning is only performed before the ML process, so it cannot further adapt the cleaning based on feedback from the ML process. The AutoML framework AutoSklearn considers data cleaning as part of the ML pipeline [6]. They use the model validation accuracy as the primary cleaning signal to assess the fitness of an ML pipeline. A pipeline consists of preprocessing operations, HP selection, and model training. To restrict the search space, the preprocessors are limited to value imputation based on mean, median, or most-frequent values, and are either applied to the whole dataset or not at all (e.g., no conditional repairs). We extend these frameworks with more advanced data cleaning techniques

to capture other types of dirty data and study the impact of data quality in conjunction with all other optimization parameters.

Data Cleaning for ML. The common strategy to tailor cleaning for an ML task is to leverage the downstream model or application to obtain signals with higher-level semantics that serve as data quality assessment. For example, ActiveClean [14] is a data cleaning system for models with convex loss functions. It treats cleaning as a stochastic gradient descent (SGD) problem, where in each step, the system samples and asks the user to clean records that are expected to shift the model along the steepest gradient. CPClean [11] is designed for KNN models. It incrementally cleans a training set until it is certain that no more repairs can change the model predictions. It uses the validation set and the counting query as the primary cleaning signals but relies on the user to perform the actual repairs. BoostClean [13] treats cleaning as a boosting problem and outputs a cleaning program that can be applied to training or test records. It is specialized in conditional value errors. BoostClean diverges from prior cleaning systems in that users do not need to manually repair records. It leverages the model's training or validation accuracy as the primary cleaning signal. Rain [35] executes relational workflows consisting of relational operators and inferences made by a differentiable model. By supporting complaints over workflow outputs, it empowers users to specify constraints within the context of the application's downstream semantics.

Robust ML. Some ML models are robust to small amounts of random noises [1, 36] and can also be sensitive to other types of noise, especially non-white noise in the input data [14] and labels [7]. These approaches focus on ML algorithms that are robust to noise in certain distributions rather than directly performing data cleaning. Examples include noise-robust decision trees [25], regularization for improving robustness [31], and model bagging to reduce the model variability caused by dirty data [12]. Other robust estimation methods re-weight, filter, and otherwise adapt the estimation procedure to be insensitive to outliers in the training data [30, 34]. For instance, models that perform local averaging, such as weighted and interpolated nearest neighbors, are naturally robust to noisy training data [2]. Further, there is extensive research on approaches to make ML more robust against adversarial examples, which can be seen as errors that are introduced on purpose [17, 26].

3 Extending AutoML with Cleaning Preprocessors

To assess how cleaning interacts with other parts of the ML pipeline, we include cleaning preprocessors in the optimization process. We could leverage state-of-the-art AutoML systems, such as AutoSklearn [6], and analyze which cleaning component is chosen how often. According to the CleanML benchmark, compared to other error types, MVs are more likely to negatively impact ML performance. Parametric statistics, such as the mean and standard deviation, are sensitive to outliers. Models that leverage such statistics, such as linear regression, are negatively affected. However, other error types, such as inconsistencies, e.g. functional dependency violations, or duplicates are less likely to affect ML performance. Thus, we built AutoClean, which extends AutoSklearn with a more sophisticated outlier detection component and context-dependent imputation strategies.

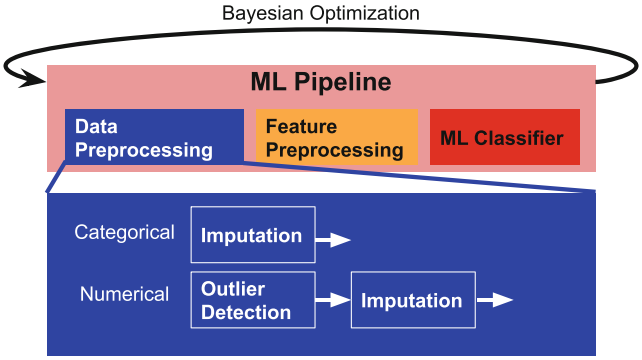


Fig. 1 AutoML with data cleaning pre-processors

First, we extend AutoSklearn to support data-dependent HPs. For instance, if we apply an outlier threshold X to feature A, this threshold might be a poor choice for another feature B. Thus, we need a threshold for each feature. This way, the HP search space quickly grows as there is not only one HP per feature. We address this problem by structuring the search space as hierarchical as possible to prune HP early on.

Second, data cleaning preprocessors might introduce significant overhead and therefore compete with the training time of ML models. This problem is aggravated by feature-specific data cleaning, e.g., training one outlier detection model for each feature. To reduce this overhead, we introduce an additional HP that allows sampling. This way the outlier detection model is not trained on the full dataset but only on a stratified sample.

Fig. 1 shows the workflow of AutoClean. It leverages Bayesian optimization (BO) to iteratively search for ML pipelines including data preprocessors, feature preprocessors, and an ML classification model. Our implementation supports additional imputation strategies and additional outlier detection in the data preprocessing step. In the following, we describe how we incorporate our data cleaning preprocessors in AutoSklearn. Then, we describe details of our extension for both MV imputation and outlier detection.

3.1 MV Imputation

Currently, AutoSklearn only supports constant value imputation for categorical data and mean/median/most-frequent imputation for numerical data. For categorical data, we add most-frequent value imputation and KNN imputation [32]. KNN imputation identifies the KNN to replace the MV based on the most-frequent value of these neighbors. The KNN imputation model has two HPs – the number of neighbors used for imputation and the weight function. To reduce the computational overhead of the KNN model, we define an HP to control stratified sampling. We sample values for this HP from a logarithmic distribution because we expect that lower values are computationally beneficial. We use the Scikit-learn [24] for both imputers. For numerical data, we add KNN imputation and iterative imputation [33]. Iterative imputation models each feature with MVs as a function of all other features iteratively. Therefore, we train one regression model for each feature with MVs. This approach requires much more compute resources than, e.g., mean value imputation where we compute only the mean for each feature. Therefore, we again sample values for this HP from a logarithmic distribution to allow stratified sampling because a subset of the data might be sufficient to predict the MVs.

3.2 Outlier Detection

For numerical data, we add four outlier detection strategies: Local Outlier Factor (LOF) [4], one-class Support Vector Machines (SVM) [29], Isolation Forest [16], and Elliptic Envelope [28]. For all outlier detection algorithms, we leverage the Scikit-learn implementation [24].

LOF measures the local deviation of the density of a given instance compared to its neighbors. It considers instances with a significantly lower density as outliers. One-class SVM is similar to the common SVM but instead of separating instances into two classes by a hyperplane, it encompasses as many instances as possible within a hypersphere. It considers all instances outside of this hypersphere as outliers. Isolation Forest randomly partitions the instances and measures how many times it had to partition to fully isolate a given instance. It considers instances that require significantly fewer partitions to be isolated as outliers. Elliptic Envelope considers all instances that violate a Gaussian distribution as outliers.

We choose one outlier detection strategy per column. This leads to a huge HP search space as we need to optimize the HPs of four outlier detection strategies for each feature. To reduce HP space, we structure it hierarchically as shown in Fig. 2. For each feature, we start with one HP to identify whether this feature benefits from outlier detection. If yes, we evaluate which outlier detection strategy fits the feature, then optimize the HPs of the corresponding strategy so that we can prune HP space early on. However, even if we fit an outlier detection model for a subset of the features, it still requires large computational resources. As shown in Fig. 2, we defined a sampling HP that – independent of the outlier detection strategy – decides how many instances are used for outlier detection because we assume that the strategies need a similar number of instances to estimate the underlying distributions to identify the outliers. Once an outlier is detected, we replace it with *NAN* as an MV. The MV component directly follows the outlier detection component and will impute the values accordingly.

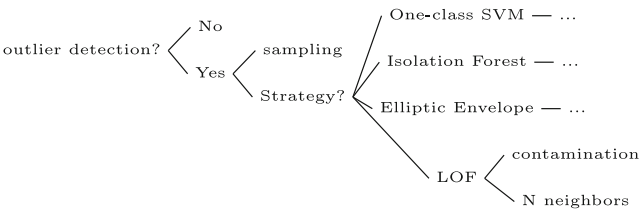


Fig. 2 Hierarchical HP space for outlier detection

4 Case Study

The CleanML benchmark [15] is one of the most comprehensive studies on the interplay of ML and data cleaning. It measures the impact of various cleaning methods for multiple datasets with five error types: duplicates, outliers, inconsistencies, MVs, and mislabels. However, they assume static preprocessing and do not consider model ensembles. We revisit the CleanML benchmark with state-of-the-art AutoML systems, AutoSklearn [6], and AutoGluon [5]. This way, we optimize the whole ML pipeline from preprocessing over modeling to ensembling. In contrast to CleanML, which analyzes the impact of errors on a data level, we follow a more fine-grained approach on the feature level using permutation importance [3].

For all experiments, we use the binary classification datasets provided by CleanML whose main statistics are shown in Table 1. Additionally, we evaluate the Flights dataset [20] with the task to predict whether a flight was over 5 minutes delayed. For each dataset, a clean and a dirty version is available. To measure prediction performance, we use balanced accuracy because it is robust against class imbalance. It is often used to benchmark AutoML systems for classification tasks [6, 10]. For instance, for two classes, it is calculated as follows:

$$\texttt{balanced accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right),$$

Table 1 Datases

Error	Dataset	Instances	Features	Minority Class
Outliers	Credit	8413	10	0.07
	Airbnb	10500	39	0.33
	EEG	10486	14	0.45
	Sensor	10500	8	0.48
Missing values	Airbnb	10500	39	0.32
	Credit	10500	10	0.07
	Marketing	6296	13	0.45
	Titanic	891	9	0.38
	US Census	10500	14	0.24
Inconsistencies	Company	10500	5	0.38
	Movie	6531	7	0.37
	Restaurant	8405	7	0.38
	University	147	15	0.46
Duplicates	Airbnb	7965	39	0.32
	Citation	2998	1	0.40
	Movie	4419	7	0.37
	Restaurant	7736	7	0.38
Multiple	Flights	2376	9	0.25

where *TP* is the number of true positives, *TN* the number of true negatives, *FN* the number of false negatives, and *FP* the number of false positives.

We use AutoSklearn [6] for numerical data and AutoGluon [5] for textual data because AutoSklearn does not provide off-the-shelf natural language processing capabilities like AutoGluon. Specifically, we leverage AutoGluon for the datasets Citation and Restaurant. For all experiments, we apply 5-fold cross-validation. In each cross-validation iteration, 4 folds are considered as the training data while the remaining fold is considered as the test data. On the training data, we apply AutoSklearn/AutoGluon for 20 minutes using 20 processors. Both AutoML systems split the training data internally into a smaller training and a validation set (hold-out validation). So, we use hold-out validation for hyperparameter optimization and cross validation for evaluation. This way, we avoid that any information about the test set is available during training (test snooping).

4.1 Impact of Error Types

To understand the impact of different error types, we apply the vanilla version of AutoSklearn for both the clean and dirty versions for five error types – duplicates, inconsistencies, outliers, MVs, and mislabels. We explain for each error type how it affects the ML performance when optimizing the complete ML pipeline.

Duplicates: One can consider duplicates as incorrect instance weighting. AutoML can adjust the models' HPs to counter this instance weighting. Table 2 shows that in the best case cleaning yields only 1% higher balanced accuracy for the Restaurant dataset. For the Movie dataset, deleting the duplicates even leads to 5% less balanced accuracy. Removing instances from the data might be more detrimental than having duplicate information because the ML model can choose to ignore data but it cannot make up for missing data. The CleanML benchmark [15] also found the same result for their setting.

Table 2 AutoML for dirty/clean data

Error Type	Dataset	Dirty	Clean
Duplicates	Airbnb	0.64 ± 0.01	0.63 ± 0.01
	Citation	0.68 ± 0.01	0.63 ± 0.02
	Movie	0.69 ± 0.04	0.64 ± 0.00
	Restaurant	0.73 ± 0.01	0.74 ± 0.01
Inconsistencies	Movie	0.74 ± 0.01	0.73 ± 0.02
	Company	0.67 ± 0.01	0.67 ± 0.01
	Restaurant	0.73 ± 0.01	0.73 ± 0.01
	University	0.77 ± 0.06	0.78 ± 0.04

Table 3 Feature importance for University

Feature Names	Importance Clean	Importance Dirty	Error Fraction
Control	1.00	1.00	0.00
Sat verbal	0.26	0.32	0.00
State	0.20	0.12	0.16
Student:faculty	0.18	0.18	0.00
No-of-students	0.14	0.13	0.00
Admittance	0.12	0.08	0.00
Location	0.11	0.00	0.16
Academics scale	0.10	0.12	0.00
Quality-of-life	0.08	0.07	0.00
Social scale	0.08	0.20	0.00
Male:female	0.08	0.15	0.00
Sat math	0.07	0.03	0.00
Financial-aid	0.06	0.17	0.00
Enrolled	0.04	0.13	0.00
Applicants	0.00	0.10	0.00

Inconsistencies: AutoML can also workaround inconsistencies, such as domain value violations. For instance, when we have both values *CA* and *California*, AutoML can apply one-hot-encoding and the model can learn that both categories are the same. Table 2 shows that cleaning yields 1% higher balanced accuracy for the dataset University. To better understand how AutoML addresses such inconsistencies, we can analyze the feature importance with and without cleaning. Table 3 showed a significant drop in feature importance for features with domain value violations. The University dataset contains such domain value violations (different capitalizations) in the location and the state feature. In cases of these domain value violations, the AutoML models depend more on features that do not

Table 4 AutoSklearn and AutoClean for dirty/clean data

Error Type	System Dataset	AutoSklearn Dirty	AutoSklearn Clean	AutoClean Dirty
Outliers	Airbnb	0.64 ± 0.01	0.64 ± 0.01	0.65 ± 0.00
	Credit	0.76 ± 0.01	0.76 ± 0.01	0.75 ± 0.01
	EEG	0.98 ± 0.00	0.97 ± 0.01	0.94 ± 0.01
	Sensor	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
MV	Airbnb	0.64 ± 0.01	0.65 ± 0.01	0.65 ± 0.01
	Credit	0.77 ± 0.00	0.77 ± 0.01	0.77 ± 0.01
	Marketing	0.79 ± 0.01	0.78 ± 0.02	0.79 ± 0.02
	Titanic	0.80 ± 0.04	0.81 ± 0.03	0.79 ± 0.03
	US Census	0.83 ± 0.01	0.84 ± 0.01	0.84 ± 0.00
Multiple	Flights	0.48 ± 0.05	0.53 ± 0.04	0.49 ± 0.02

Table 5 Feature Importance for Credit (MV)

Feature Names	Importance Clean	Importance Dirty	Error Fraction
30-59DaysPastDueNot-Worse	1.00	0.93	0.00
90DaysLate	0.87	1.00	0.00
60-89DaysPastDueNot-Worse	0.82	0.90	0.00
MonthlyIncome	0.24	0.17	0.20
OpenCreditLinesAndLoans	0.17	0.24	0.00
UtilizationOfUnsecuredLines	0.16	0.17	0.00
Dependents	0.15	0.17	0.03
DebtRatio	0.13	0.19	0.00
RealEstateLoansOrLines	0.04	0.17	0.00
Age	0.00	0.00	0.00

Table 6 Feature Importance for US Census (MV)

Feature Names	Importance Clean	Importance Dirty	Error Fraction
Capital-gain	1.00	1.00	0.00
Education-num	0.65	0.83	0.00
Age	0.46	0.44	0.00
Marital-status	0.45	0.70	0.00
Capital-loss	0.34	0.33	0.00
Hours/week	0.30	0.25	0.00
Occupation	0.27	0.17	0.06
Workclass	0.18	0.14	0.05
Race	0.17	0.11	0.00
Native-country	0.17	0.10	0.02
Fnlwgt	0.14	0.10	0.00
Education	0.08	0.06	0.00
Sex	0.06	0.07	0.00
Relationship	0.00	0.00	0.00

have these issues, e.g. *social scale* gains 12% in relative importance. Feature importance can also shift if functional dependencies exist. For instance, if feature A functionally determines B and A contains many errors, AutoML can shift the importance to feature B.

Outliers: Table 4 shows that cleaning outliers did not improve the balanced accuracy for any of the evaluated datasets. The reasons are manifold. ML models are designed to ignore errors. For instance, SVM support a dedicated regularization term to allow a small number of points to be within the SVM margin. Another example is the KNN classifier that is robust against outliers because it only considers the local neighborhood that by definition ignores outliers. Second, the AutoML's model selection automatically selects those models that are most robust for the given outliers. Third, AutoSklearn builds model ensembles that are known to be robust against errors.

Missing Values: The CleanML found that MV imputation is likely to improve ML performance. However, for our setting of optimizing the ML pipeline holistically, imputation causes no significant improvements. Table 4 shows that imputation yields at most 1% higher balanced accuracy. The reasons are manifold. First, most evaluated datasets contain very few MVs. For instance, US Census (Table 6) contains at most 5.5% MVs per feature. Second, for some

Table 7 Feature Importance for Titanic(MV)

Feature Names	Importance Clean	Importance Dirty	Error Fraction
Sex	1.00	1.00	0.00
Age	0.30	0.11	0.20
Pclass	0.22	0.08	0.00
Fare	0.09	0.12	0.00
Ticket	0.08	0.12	0.00
Cabin	0.05	0.08	0.77
SibSp	0.02	0.00	0.00
Parch	0.01	0.05	0.00
Embarked	0.00	0.02	0.00

Table 8 Mislabels: AutoSklearn for dirty/clean data

Dataset	Clean	1% dirty	5% dirty	10% dirty
Credit	0.77 ± 0.01	0.77 ± 0.01	0.74 ± 0.04	0.72 ± 0.03
EEG	0.97 ± 0.01	0.97 ± 0.01	0.96 ± 0.01	0.95 ± 0.01
US Census	0.84 ± 0.01	0.83 ± 0.01	0.83 ± 0.01	0.83 ± 0.01

datasets, the MVs predominantly affect features that are not important for the prediction. For instance, the Credit dataset contains MVs only for features that are less than 25% of the min-max-scaled permutation importance (Table 5). But even if the MVs affect an important feature, e.g., 20% of the values of the feature *age* in the Titanic dataset, the models will resort to other features (Table 7). The relative feature importance drops by 19% for the feature *age* but it increases for other features, such as *Fare* and *Ticket*.

Mislabels: CleanML found that cleaning mislabels is likely to have a positive impact on ML performance. To understand how mislabels affect the ML performance when we optimize the ML pipeline holistically, we randomly switched a specified number of labels (evenly distributed across classes). Specifically, for each class, we randomly switch the same number of labels to another class. For binary classification, this approach preserves the class ratio. Table 8 shows the results of this experiment. For the Credit dataset, we see that mislabels significantly affect the balanced accuracy, which is decreased by 5%. One reason is that the minority class only represents 7% of the data. For the other two datasets, the impact is less significant because the classes are more balanced.

Multiple Errors: To evaluate the scenario of multiple errors, we evaluate the Flights dataset. It contains missing values (3 columns contain 22%, 35%, 37%) and 17% mislabels. The significant number of errors combined with the class imbalance allows data cleaning to improve the balanced accuracy by 5% as shown in Table 4. AutoClean only slightly improves upon AutoSklearn. It predominantly uses median imputation for the missing timestamps (see Table 12). However, it cannot compete with the cleaned version because AutoClean cannot clean mislabels yet.

In the datasets Sensor, EEG, and Company we observe less than 1% errors. This small portion of errors does not significantly impact the prediction performance.

4.2 Impact on HP Selection

To analyze how AutoML systems adjust their HP selection to the prevalence of different error types, we compute the fraction of times that an HP was chosen for the final ML ensemble. As AutoSklearn returns weighted pipeline ensembles, we compute the weighted fraction. For instance, the KNN classifier was chosen for two pipelines with a weight of 0.3 and 0.1 respectively. The sum of all pipeline weights in the ensemble is 1. Thus, the weighted fraction of KNN is 0.4. If we would only count the number of times that a specific component was chosen, we would consider an occurrence for an ML pipeline with weight 0.8 as important as weight 0.01 that barely contributes to the final prediction. Therefore, a weighted analysis is important to understand the contribution of a given component to the final ML predictions. As there are hundreds of HPs, we reduce the HPs to the main components in Tables 9 and 10. Then, we report only those that significantly differ between dirty and clean state.

Table 9 reports the HPs whose average weighted fraction changed the most between datasets with and without outliers (see Table 4). For instance, random kitchen sinks [27] create a feature map that is more robust against outliers. Further, it is well known that KNN classifiers are robust against outliers. This classifier was chosen four times more

Table 9 Outliers: Chosen pipeline components

Pipeline Components	Dirty average	Clean average
Kitchen sinks	0.039 ± 0.06	0.001 ± 0.00
KNN	0.250 ± 0.50	0.058 ± 0.12
SGD	0.151 ± 0.21	0.055 ± 0.08
Quantile Transformer	0.569 ± 0.42	0.244 ± 0.36
no preprocessing	0.048 ± 0.08	0.099 ± 0.06
no rescaling	0.104 ± 0.15	0.215 ± 0.18
robust scaling	0.044 ± 0.07	0.155 ± 0.14
Extra trees	0.053 ± 0.07	0.188 ± 0.28
Normalize	0.008 ± 0.02	0.045 ± 0.09
Feature agglomeration	0.018 ± 0.02	0.109 ± 0.08
Nystroem sampler	0.000 ± 0.00	0.021 ± 0.04

Table 10 MV: Chosen pipeline components

Pipeline Components	Dirty average	Clean average
Kitchen sinks	0.006 ± 0.01	0.001 ± 0.00
Feature agglomeration	0.206 ± 0.16	0.029 ± 0.03
LDA	0.044 ± 0.09	0.007 ± 0.01
Passive aggressive	0.054 ± 0.12	0.018 ± 0.02
normalize	0.094 ± 0.14	0.033 ± 0.05
liblinear svc	0.086 ± 0.07	0.038 ± 0.05
libsvm svc	0.082 ± 0.13	0.039 ± 0.09
kernel PCA	0.048 ± 0.08	0.023 ± 0.05
random trees embedding	0.005 ± 0.01	0.002 ± 0.01
polynomial	0.162 ± 0.19	0.339 ± 0.30
no class weighting	0.056 ± 0.08	0.139 ± 0.19
Extra trees	0.046 ± 0.05	0.134 ± 0.18
PCA	0.018 ± 0.04	0.057 ± 0.11

often for data with outliers compared to clean data. Further, components, such as feature agglomeration and normalization leverage parametric statistics and are sensitive to outliers. Therefore, both components are chosen six times less often for datasets with outliers.

Next, we compare the average weighted fraction of HPs chosen by the AutoML systems with regard to the prevalence of MVs. Table 10 aggregates the results across the five datasets once with and once without MVs. As expected, feature preprocessing strategies, such as random kitchen sinks, feature agglomeration, kernel principal component analysis (PCA), and random trees embedding, are chosen more often in case of MVs because they can extract information from features that miss values. The classifier extra trees [9] suffers from MVs because it leverages averaging which is affected by a large number of imputed values.

4.3 Data Cleaning Extensions

To evaluate AutoSklearn with additional imputation and outlier detection, we evaluate it on the datasets with MVs, outliers, and multiple errors. For all datasets (see Table 4), AutoClean achieves similar or slightly lower balanced accuracy compared to vanilla AutoSklearn. The reason for lower accuracy for AutoClean is that the extended search space increases the likelihood of overfitting on the validation set. This result shows that very simple data cleaning preprocessors that are already present in AutoSklearn are enough to achieve high ML performance because it can build ML pipelines that include data preprocessors, feature processors, and models that can work around these issues as well.

We now analyze the cleaning components that were selected by AutoClean. Table 11 shows that our outlier detection component was active in the majority of the datasets, even if no outliers were present. In these cases, outlier detection acted as a regularizer to reduce noise from features.

Table 11 How often is outlier detection chosen and which sampling degree is chosen

Error type	Dataset	Detection	Outlier Strategy				Sampling
		Fraction	One-class SVM	LOF	Isolation Forest	Elliptic Envelope	Fraction
Missing values	Credit	0.82	0.18	0.43	0.10	0.11	0.15
	US Census	0.88	0.12	0.44	0.13	0.17	0.10
	Titanic	0.65	0.14	0.15	0.22	0.13	0.20
	Airbnb	0.42	0.08	0.17	0.08	0.09	0.06
	Marketing	0.00	0.00	0.00	0.00	0.00	–
Outliers	Airbnb	0.42	0.12	0.13	0.09	0.07	0.07
	EEG	0.64	0.11	0.25	0.14	0.14	0.24
	Credit	0.66	0.18	0.34	0.08	0.05	0.22
	Sensor	0.74	0.10	0.41	0.15	0.06	0.27
Multiple	Flights	0.65	0.07	0.29	0.11	0.16	0.11
Average		0.63	0.11	0.26	0.11	0.10	0.16

Table 12 How often is imputation chosen and which sampling degree is chosen

Error type	Dataset	Numerical Imputation					Categorical Imputation			Sampling
		KNN	Mean	Median	Most frequent	Iterative	KNN	Constant	Most frequent	fraction
Missing values	Credit	0.17	0.15	0.31	0.37	0.00	0.40	0.36	0.24	0.07
	US census	0.21	0.11	0.30	0.39	0.00	0.35	0.48	0.17	0.09
	Marketing	0.00	0.00	0.00	0.00	0.00	0.28	0.38	0.35	0.10
	Titanic	0.25	0.24	0.35	0.16	0.00	0.25	0.49	0.26	0.05
	Airbnb	0.14	0.28	0.41	0.17	0.00	0.20	0.58	0.23	0.06
Outliers	Airbnb	0.37	0.15	0.40	0.08	0.00	0.14	0.68	0.19	0.06
	EEG	0.22	0.22	0.28	0.28	0.00	0.27	0.32	0.41	0.14
	Credit	0.13	0.29	0.34	0.24	0.00	0.65	0.25	0.10	0.11
	Sensor	0.26	0.14	0.23	0.37	0.00	0.28	0.21	0.51	0.06
Multiple	Flights	0.17	0.22	0.34	0.26	0.00	0.54	0.23	0.23	0.14
Average		0.19	0.18	0.30	0.23	0.00	0.33	0.40	0.27	0.09

For the dataset Marketing, AutoClean did not detect outliers because it only contains categorical features. We implemented the outlier detection strategies only for numerical data. Further, Table 11 also reports the weighted fraction of how much each outlier detection strategy contributed to the final ensemble. LOF contributes the most because it has a low overhead as it uses the KNN approach and has fewer HPs than the alternatives. Additionally, Table 11 shows the average sampling fraction that was chosen by BO for each dataset. We see that in most cases only a small fraction of the dataset is considered for detection. This way, our approach saves significant computation.

Table 12 shows that the new imputation strategy based on KNN is effective because it was the second most chosen strategy for categorical data and the third most chosen strategy for numerical data. However, BO identifies that ML pipelines that leverage iterative imputation yield often no result due to exceeded time limits and therefore never choose this strategy for imputation. Similar to the detection strategies, BO chooses a small fraction of the data to train the imputation strategies.

To summarize advanced cleaning operations did not yield significant ML performance improvements. However, AutoML chooses in the majority of the cases to leverage outlier detection. Further, KNN-based imputation is a valuable addition to other simple imputation strategies.

5 Conclusions

We evaluated the importance of data cleaning in conjunction with other optimization possibilities using state-of-the-art AutoML systems. In contrast to previous studies that only consider data cleaning, we showed that data errors did not affect the ML performance significantly. AutoML systems reduce data error impact by falling back on robust ML models or reevaluating feature importance. Further, we analyzed which ML pipeline components are more beneficial in the case of outliers and missing values. Finally, we evaluated our with data cleaning preprocessors extended AutoSklearn and found that simple data cleaning strategies outperform more advanced ones because AutoML can adjust the entire ML pipeline to the data errors at hand. However, we found that KNN imputation benefits AutoML systems because it was one of the most chosen imputation strategies.

Our work opens up various exciting directions for future work. First, our analysis is limited to the datasets available to us. An expansion of the benchmark with real-world datasets and diverse error types is necessary to make educated judgments on the importance of other data cleaning preprocessors addressing mislabel correction [23], duplicate detection, and functional dependency violation correction. Second, we only optimized the ML performance using an AutoML system. One could also formulate the problem as an ML performance maximization problem subject to data quality constraints [8].

Acknowledgements This work was funded by the German Ministry for Education and Research as BIFOLD – Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref. 01IS18037A). We thank Henrik Tipp.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Alistarh D, Grubic D, Li J, Tomioka R, Vojnovic M (2017) QSGD: communication-efficient SGD via gradient quantization and encoding. In: NeurIPS
2. Belkin M, Hsu DJ, Mitra P (2018) Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In: NeurIPS
3. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
4. Breunig MM, Kriegel H, Ng RT, Sander J (2000) LOF: identifying density-based local outliers. In: SIGMOD
5. Erickson N, Mueller J, Shirkov A, Zhang H, Larroy P, Li M, Smola AJ (2020) Autogluon-tabular: Robust and accurate automl for structured data. CoRR, abs/2003.06505
6. Feurer M, Klein A, Eggensperger K, Springenberg JT, Blum M, Hutter F (2015) Efficient and robust automated machine learning. In: NeurIPS
7. Frénay B, Verleysen M (2013) Classification in the presence of label noise: a survey. IEEE Trans Neural Netw Learning Syst 25(5):845–869
8. Gelbart MA, Snoek J, Adams RP (2014) Bayesian optimization with unknown constraints. In: UAI
9. Geurts P, Ernst D, Wehenkel L (2006) Extremely randomized trees. Mach Learn 63(1):3–42
10. Guyon I, Bennett KP, Cawley GC, Escalante HJ, Escalera S, Ho TK, Macià N, Ray B, Saeed M, Statnikov AR, Viegas E (2015) Design of the 2015 chalearn automl challenge. In: IJCNN, pp 1–8
11. Karlas B, Li P, Wu R, Gürel NM, Chu X, Wu W, Zhang C (2021) Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions. PVLDB 14:255–267
12. Khoshgoftaar TM, Hulse JV, Napolitano A (2011) Comparing boosting and bagging techniques with noisy and imbalanced data. IEEE Trans Syst Man Cybern Part A 41(3):552–568
13. Krishnan S, Franklin MJ, Goldberg K, Wu E (2017) BoostClean: automated error detection and repair for machine learning. CoRR, abs/1711.01299

14. Krishnan S, Wang J, Wu E, Franklin MJ, Goldberg K (2016) ActiveClean: interactive data cleaning for statistical modeling. PVLDB 9(12):948–959
15. Li P, Rao X, Blase J, Zhang Y, Chu X, Zhang C (2021) Cleanml: A study for evaluating the impact of data cleaning on ML classification tasks. In: ICDE
16. Liu FT, Ting KM, Zhou Z (2008) Isolation forest. In: ICDM
17. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Towards deep learning models resistant to adversarial attacks. In: ICLR
18. Mahdavi M, Abedjan Z (2020) Baran: Effective error correction via a unified context representation and transfer learning. PVLDB 13(11):1948–1961
19. Neutatz F (2022) Data-cleaning-specific AutoSklearn. https://github.com/LUH-DBS/AutoClean. Accessed 06 May 2022
20. Neutatz F, Chen B, Abedjan Z, Wu E (2021) From cleaning before ML to cleaning for ML. IEEE Data Eng Bull 44:24–41
21. Neutatz F, Mahdavi M, Abedjan Z (2019) ED2: A case for active learning in error detection. In: CIKM
22. Ng A (2021) The batch: regulating ai, where the mles are, real-time voice replacement, robot painter. https://read.deeplearning.ai/the-batch/issue-114/. Accessed 06 May 2022
23. Northcutt CG, Jiang L, Chuang IL (2021) Confident learning: Estimating uncertainty in dataset labels. JAIR 70:1373–1411
24. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. JMLR 12:2825–2830
25. Quinlan JR (1999) Simplifying decision trees. Int J Hum Comput Stud 51(2):497–510
26. Raghunathan A, Steinhardt J, Liang P (2018) Certified defenses against adversarial examples. In: ICLR
27. Rahimi A, Recht B (2007) Random features for large-scale kernel machines. NeurIPS 20:1177–1184
28. Rousseeuw PJ, van Driessen K (1999) A fast algorithm for the minimum covariance determinant estimator. Technometrics 41(3):212–223
29. Schölkopf B, Williamson RC, Smola AJ, Shawe-Taylor J, Platt JC (1999) Support vector method for novelty detection. In: NeurIPS
30. Steinhardt J (2018) Robust learning: Information theory and algorithms. Stanford University
31. Teng C (2000) Evaluating noise correction. In: PRICAI, vol 1886
32. Troyanskaya OG, Cantor MN, Sherlock G, Brown PO, Hastie T, Tibshirani R, Botstein D, Altman RB (2001) Missing value estimation methods for DNA microarrays. Bioinform 17(6):520–525
33. Van Buuren S, Groothuis-Oudshoorn K (2011) mice: Multivariate imputation by chained equations in r. J Stat Soft 45:1–67
34. Wilcox RR (2011) Introduction to robust estimation and hypothesis testing. Academic Press
35. Wu W, Flokas L, Wu E, Wang J (2020) Complaint-driven training data debugging for query 2.0. In: SIGMOD
36. Zhang H, Li J, Kara K, Alistarh D, Liu J, Zhang C (2017) Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning. In: ICML, vol 70