CleanML: A Benchmark for Joint Data Cleaning and Machine Learning [Experiments and Analysis]

Peng Li
Georgia Institute of Technology
pengli@gatech.edu

Xi Rao
Systems Group, ETH Zurich; KOF Swiss Economic Institute
rao@kof.ethz.ch

Jennifer Blase
Georgia Institute of Technology
jblase@gatech.edu

Yue Zhang
Georgia Institute of Technology
yzhang3271@gatech.edu

Xu Chu
Georgia Institute of Technology
xu.chu@cc.gatech.edu

Ce Zhang
Systems Group, ETH Zurich
ce.zhang@inf.ethz.ch

ABSTRACT

It is widely recognized that the data quality affects machine learning (ML) model performances, and data scientists spend considerable amount of time on data cleaning before model training. However, to date, there does not exist a rigorous study on how exactly does cleaning affect ML — ML community usually focuses on the effects of specific types of noises of certain distributions (e.g., mislabels) on certain ML models, while database (DB) community has been mostly studying the problem of data cleaning alone without considering how data is consumed by downstream analytics.

We propose the CleanML benchmark that systematically investigates the impact of data cleaning on downstream ML models. The CleanML benchmark currently includes 13 real-world datasets with real errors, five common error types, and seven different ML models. To ensure that our findings are statistically significant, CleanML carefully controls the randomness in ML experiments using statistical hypothesis testing, and also uses the Benjamini-Yekutieli (BY) procedure to control potential false discoveries due to many hypotheses in the benchmark. We obtain many interesting and non-trivial insights, and identify multiple open research directions. We also release the benchmark and hope to invite future studies on the important problems of joint data cleaning and ML.

1. INTRODUCTION

The quality of machine learning (ML) applications is only as good as the quality of the data it trained on, and data cleaning has been the cornerstone of building high-quality ML models for decades. Not surprisingly, both ML and database (DB) communities have been working on problems associated with dirty data over the last decades:

- **ML community** has been focusing on understanding the impact of noises to ML models without actually doing data cleaning. On one hand, many ML algorithms are *robust to noises* — there has been research showing that the noise introduced during the *training process*, via e.g., asynchronous execution and lossy data compression and quantization, can have negligible effect in the final accuracy, both empirically and theoretically [16, 27, 45, 46, 55, 68, 69]. On the other hand, ML algorithms can also be sensitive to other types of noises, especially those non-white noises that are in the input data and labels [32]. As a result, the community has been focusing on designing ML algorithms that are *robust* to noises, such as noise-robust decision trees [53], the use of regularization for improving robustness [61], and model bagging to reduce the variability of model performances caused by dirty data.
- **DB community** has been mostly focusing on understanding the fundamental process of data cleaning without considering its impact on ML models. Data cleaning activities usually consist of two phases: *error detection*, where various errors and violations are identied and possibly validated by experts; and *error repair*, where updates to the database are applied (or suggested to human experts) to bring the data to a cleaner state. Many different techniques have been proposed to detect different types of errors, for example, by designing integrity constraints to capture data inconsistencies [23], by using quantitative and statistical techniques to detect outliers [39], and by building ML models to detect duplicates [30]. Various heuristics and techniques have also been proposed to suggest data repairs, for example, by finding the minimal set of updates to resolve violations [24], by performing data transformations [38], by consulting external knowledge bases [25], and by using probabilistic graphical models to reason about errors holistically [56].

(The Emerging Problem of Joint Cleaning/ML) In real applications, these two angles as less segregated — In many, if not most, real-world applications, neither does the ML nor the data cleaning task appears on its own — instead, the most common paradigm is to have a *data cleaning component followed by a ML model learning phase*. As the majority of previous work has been focusing on solving each individual task independently, we believe that the joint data cleaning/ML problem is an interesting research topic for the DB community in the years to come. This paper is by no means the first to recognize the opportunities in performing data cleaning and model learning jointly — We are inspired by a range of recent work conducted by the DB community, which are discussed in Section 6.

(Our Scope) Many of the early work that tries to jointly deal with data cleaning and ML only focus on a specific subset of data fallacies and a specific set of ML models and tasks. For example, ActiveClean [44] only considers convex models that are trained using gradient descent methods, and also assumes that there is a data cleaning oracle to clean the mini-batches during training. BoostClean [43] selects a predefined set of error detection and repair combinations using statistical boosting. While it shows promising results in terms of increased predictions after cleaning, it only considers *domain value violations* when an attribute value is outside of its value domain and only tests the random forest model.

The goal of this paper is not to propose a new algorithm, instead, our goal is to (1) conduct a systematic study of the impact of various types of errors and cleaning methods on downstream various ML models; (2) construct the first benchmark for joint cleaning and ML tasks, in terms of both collecting real-world datasets with real errors and designing methods to evaluate the impacts; (3) provide a systematic, qualitative statement of the challenges and potentials; and (4) provide a starting point for follow-up studies by the community in the future.

(Challenges) Conducting such a systematic study and constructing benchmark datasets is not a trivial task. The difficulties hinge on dealing with the following challenges:

- **Realistic noise patterns.** Taking a standard ML benchmarks dataset and simply simulating data fallacies trivially (e.g., by randomly removing values to mimic missing values) might under/overestimate the impact of data cleaning on ML. For our study to reflect the real impacts, we have to work with data that has realistic noise patterns, for which we usually do not have ground truth.
- **Comprehensive ML models.** Studying a single ML algorithm (e.g., logistic regression as [60] did) is enough to evaluate a new algorithm, and yet, is not enough to provide a qualitative assessment on the impact of data cleaning on ML — Maybe there are other, simpler or more complicated, ML algorithms that are more robust to noises and are performing relatively well, hence eliminating the need for cleaning.
- **Statistically significant results.** ML models are inherently probabilistic — a different train/test split on the same dataset might produce entirely different results. This is further complicated by the many uncertainties and choices in the data cleaning algorithms (e.g., different imputation methods for missing value). While it might be easy to find a dataset, a cleaning method, and a ML model that jointly show the benefits of joining cleaning and learning, how to ensure that our findings are statistically significant presents a major challenge.

(Contributions) We focus on five types of errors, including outliers, duplicates, inconsistencies, mislabels and missing values. These errors are prevalent in the real-world datasets and frequently considered in research. We select seven classification algorithms, including Logistic Regression, Decision Tree, Random Forest, Adaboost, XGBoost, k-Nearest Neighbors (KNN) and Naive Bayes. These algorithms are chosen because they are classical and competitive classification models and are frequently used in practice. Our major contributions are summarized as follows:

- We collected 13 real-world datasets containing different types of errors, applied common data cleaning techniques, and devised classification tasks that were meaningful in the context of the data.
- We trained and evaluated ML models on dirty and cleaned versions of each dataset. We then compared model performance to show the effects of cleaning dirty data on ML models, in terms of both cleaning training set and cleaning test set.
- We controlled randomness in ML experiments using paired sample t-test, and also leveraged multiple hypothesis testing procedures to control false discoveries. We also explained our findings whenever possible, by examining the dataset error distributions, the ML model structures, and the properties of different cleaning algorithms.
- We made our benchmark publicly available for reproducibility (`https://chu-data-lab.github.io/CleanML/`). In addition, our benchmark can be easily extended by adding new datasets, new error types, new cleaning algorithms, or new ML models.

(Summary of the Insights) We present our detailed findings in Section 5. Here we summarize the following major insights from our benchmark as follows.

- Data cleaning does not necessarily improve the quality of downstream ML models. In fact, applying cleaning methods blindly may negatively impact model performances. This is because, with the lack of ground truth, cleaning algorithms may actually introduce more biases into the data than the original biases caused by data dirtiness.
- In general, the impacts of cleaning on ML depend on the errors and their distributions on the datasets (which are unknown), the correctness of the cleaning algorithms (which are also unknown without ground truth), and the internal structures of the ML models (which can be hard to interpret for some models). While some of our experimental results can be clearly attributed to one or two of these factors, many results remain hard to interpret as they are affected by these factors in combination.
- Performing model selection can significantly increase the robustness of impacts of cleaning on ML. This suggests that the models with higher validation accuracy scores (i.e., those are chosen as the best models) are often more robust in terms of the improvements.
- Performing cleaning algorithm selection further increases the robustness of impacts of cleaning on ML. This suggests that no single cleaning algorithm is the best, and any future joint cleaning and ML work must devise "adaptive" cleaning solutions.

The rest of the paper is organized as follows. We introduce our experimental and analysis methodologies, including CleanML relational schema and its queries in Section 2, which we are using to organize our experimental study. In Section 3, we explain the domains of each attribute in our CleanML relational model. In Section 4, we discuss how we actually conduct the experiments to populate the CleanML database, which includes ways to control randomness and false discoveries. In Section 5, we analyze the CleanML database using SQL queries to group by various attributes to obtain insights. We discuss related work in both ML and DB communities in Section 6. We conclude our work and discuss open research directions in Section 7.

2. EXPERIMENTAL METHODOLOGY

The impact of dirty data and data cleaning on ML in a dataset depends on a number of factors — some factors depend on the data cleaning process, i.e., the error types to be cleaned and the cleaning methods; some factors depend on the ML process, i.e., the model types used; and some factors depend on where the cleaning is performed during the ML process (training set or test set). Hence, in order to comprehensively investigate the impacts, we need to consider data cleaning and ML jointly in our experiments.

In Section 2.1, we first introduce the CleanML relational schema, where every tuple corresponds to a unique experiment or hypothesis to be tested (e.g., *how does filling in missing values using median imputation on the training set affect logistic regression in the dataset X?*). In Section 2.2, we present our approach for analyzing the impacts of data cleaning on ML using CleanML relations.

2.1 CleanML Schema

The CleanML relational schema consists of three relations, as shown in Table 1. We first introduce the attributes of our CleanML relational models, and then explain the differences between these three relations.

- **Attributes for Dataset.** The first attribute is *dataset*, which is the input to the data cleaning and ML pipeline. Each dataset can have multiple types of errors and has an associated ML task. Even

Table 1: CleanML Relational Schema

R1 (Vanilla)

Dataset	Error Type	Detection	Repair	ML Model	Scenario	Flag

R2 (With Model Section)

Dataset	Error Type	Detection	Repair	Scenario	Flag

R3 (With Model Selection and Cleaning Method Selection)

Dataset	Error Type	Scenario	Flag

for one error type, they might appear in a dataset in various distributions and hence affect ML models in complicated ways. To study the impact of real-world error types and distributions on ML models, we mostly use real-world datasets with real errors, and we apply various cleaning methods to detect and repair the errors in them (with the exception of mislabels for which we use the real-world datasets with injected errors as there is no known cleaning method to detect mislabels, as discussed in Section 3.1). We list all the datasets we use in Section 3.2.

- **Attributes for Data Cleaning.** The *error type* attribute describes which type of dirtiness we are testing. We consider five most common types of dirtiness that are considered in the ML and DB communities: missing values, outliers, duplicates, inconsistencies, and mislabels. For each error type, there exist multiple cleaning methods, and each cleaning method includes an error *detection* component and an error *repair* component. We discuss the error types and their cleaning methods in Section 3.1.
- **Attributes for ML.** The ML model describes the algorithm of training and prediction. Different ML models may have different robustness or sensitivity to dirty data. We give a description of the chosen ML models in Section 3.3.
- **Attributes for Cleaning Scenario.** The *scenario* attribute indicates whether the cleaning operation is applied in the training set or the test set, in other words, whether cleaning benefits ML during the model training process or the model evaluation process. We explain this further in Section 3.4.
- **Flag Attribute.** The *flag* attribute summarizes the impact of data cleaning on ML of an experiment into three categories, "P (positive)", "N (negative)" or "S (insignificant)", indicating whether the cleaning has a positive, negative, or insignificant impact on the ML performance respectively.

Given these attributes, we designed three relations as shown in Table 1. $R1$ shows the vanilla version of the a CleanML relation schema with the key {*dataset, error type, detect, repair, ML model, scenario, flag*}. Every tuple of $R1$ represents a hypothesis or an experiment: *how does cleaning some type of error using a detection method and a repair method affect a ML model for a given dataset?* We also consider other two versions of relations in CleanML. Compared with $R1$, $R2$ eliminates the ML model attribute. In this case, we try different models during training and select the model that has the best validation accuracy (or F1 score) as the model to be considered in an experiment in $R2$. Every tuple of $R2$ represents a hypothesis or an experiment: *how does cleaning some type of error using a detection method and a repair method affect the best ML model for a given dataset?* $R3$ further eliminates the cleaning method (detection and repair) attributes. In this case, in addition to model selection, we also try different cleaning methods and select the cleaning method that results in the best validation accuracy as the cleaning method to be considered in an experiment in $R3$. Every tuple of $R3$ represents a hypothesis or an experiment: *how does the best cleaning method affect the predictive performance of the best model for a given dataset?*

All three relations can also be extended with other attributes that are associated with an experiment, which may help obtain insights and interpret the results, such as the accuracy scores before and after cleaning and p-values of hypothesis testing associated with each experiment (c.f. Section 4.1 to 4.3).

2.2 Analysis Methodology

The general analysis strategy is to query the tuples in the relational schema shown in Table 1. We first fix the error type and group by flags. The percentage of each type of flag indicates the general impact of cleaning this type of error on ML. For example, if flag "P" dominates in the error type *outliers*, it indicates cleaning outliers generally improves the performance of ML. Then we group by the other attributes (e.g., ML models, datasets, etc.) in addition to flags to see how each attribute affects the impact. For example, if flag "S" dominates under the ML algorithm decision tree, it indicates decision tree is insensitive to this error.

We also investigate the changes of percentage in each type of flag when moving from $R1$, to $R2$ and $R3$. This indicates how model selection and cleaning algorithm selection affect the impacts. For example, if the percentage of flag "N" significantly decreases from $R1$ to $R2$, it indicates the model selection reduces the negative impact of data cleaning on ML.

We investigate the impact of dirty data on ML models by simply running SQL queries on three relations $R1$, $R2$ and $R3$. We formally present the SQL query templates as follows, where $E \in$ {inconsistencies, duplicates, mislabels, outliers, missing values}.

Q1: Flag

```
SELECT flag, COUNT(*)
FROM R
WHERE error_type = E
GROUP BY flag
```

Q2: Scenario

```
SELECT scenario, flag, COUNT(*)
FROM R
WHERE error_type = E
GROUP BY scenario, flag
```

Q3: ML Model. (Not applicable to R2, R3)

```
SELECT model, flag, COUNT(*)
FROM R
WHERE error_type = E
GROUP BY model, flag
```

Q4: Clean Method (Not applicable to R3 or $E \in$ { inconsistencies, duplicates, mislabels}, where only one cleaning method is applied)

Q4.1:
```
SELECT detect_method, flag, COUNT(*)
FROM R
WHERE error_type = E
GROUP BY detect_method, flag
```

Q4.2:
```
SELECT repair_method, flag, COUNT(*)
FROM R
WHERE error_type = E
GROUP BY repair_method, flag
```

Q5: Dataset

```
SELECT dataset, flag, COUNT(*)
FROM R
WHERE error_type = E
GROUP BY dataset, flag
```

3. DESIGN OF BENCHMARK

Based on CleanML Relational Schema, we design CleanML Benchmark by specifying the domain of each key attribute. In Sections 3.1 to 3.4, we present all datasets, error types, cleaning methods, ML models, and cleaning scenarios we consider in the benchmark.

3.1 Error Types and Cleaning Methods

We consider five error types that are prevalent in the real-world datasets, including missing values, outliers, duplicates, inconsistencies and mislabels. While there are many cleaning methods in the literature (c.f. Section 6), we consider the most straightforward one in this study. The definition and the cleaning methods of each error type are described below and summarized in Table 2.

Table 2: Cleaning Methods

Error Type	Detect Method	Repair Method
Missing Values	Empty Entry	Deletion
		Imputation: Mean Mode Median Mode Mode Mode Mean Dummy Mode Dummy Median Dummy
Outliers	SD	Deletion
	IQR	Imputation: Mean Median Mode
	IF	
Duplicates	Key Collision	Deletion
Inconsistencies	OpenRefine	Merge
Mislabels	Ground Truth	Flip Labels

3.1.1 Missing Values

Missing values occur when no value is stored for some attribute in an observation. We detect missing values by finding empty or NaN entries in datasets. We use two methods to repair missing values:

- **Deletion:** Delete records with missing values.
- **Imputation:** For numerical missing values, we consider three types of imputation methods: mean, median and mode. For categorical missing values, we use two types of imputation methods: the mode (most frequent class) or a dummy variable named "missing". Therefore, we have six imputation methods. In Table 2, we denote each imputation method by the numerical imputation and categorical imputation (e.g. "Mean Dummy" represents imputing numerical missing values by mean and imputing categorical missing values by dummy variables).

3.1.2 Outliers

An outlier is an observation that is distant from other observations. We only consider numerical outliers in our experiments. We use three methods to detect numerical outliers.

- **Standard Deviation Method (SD):** A value is considered to be an outlier if it is n numbers of standard deviations away from the mean of the attribute it belongs to. We use $n = 3$.
- **Interquartile Range Method (IQR):** Let Q_1 and Q_3 be the 25th and the 75th percentiles of an attribute. Then, the interquartile range $IQR = Q_3 - Q_1$. A value is considered to be an outlier if it is outside the range of $[Q_1 - k \times IQR, Q_3 + k \times IQR]$. We use $k = 1.5$.
- **Isolation Forest Method (IF):** The isolation forest isolates observations by randomly selecting a feature and randomly selecting a split value of the selected feature. This partition can be represented by a tree structure and outliers will have noticeably shorter paths in the random trees. We use the scikit-learn IsolationForest and set the contamination parameter to be 0.01.

We use two methods to repair numerical outliers in the datasets.

- **Deletion:** Records with outliers entries are removed from the datasets.
- **Imputation:** Outliers entries are imputed. We consider three types of imputation: mean, median and mode.

3.1.3 Duplicates

Duplicates refer to the records that correspond to the identical real-world entity. We detect duplicates by defining the key attribute that is unique for each entity in the dataset. If two records have an identical value on the key attribute, they will be considered as duplicates. We repair the duplicates by keeping the first and deleting all the others.

3.1.4 Inconsistencies

Inconsistencies occur when two values correspond to the identical real-world entity but have different representations. We detect inconsistencies in datasets using *OpenRefine* [63] and repair them by merging different representations into one in *OpenRefine*.

3.1.5 Mislabels

Mislabels occur when an observation is incorrectly labeled. Since mislabels in our datasets come from error injection, we already know which label is incorrect. We repair mislabels by flipping the label. Our protocol in injecting class noise follows the recommendation in [34, pg. 25]: (1) uniform class noise: flip 5% in each class, in total 5% of the labels are changed; (2) pairwise class noise: in binary classification, flip 5% of the labels in class 0 and keep the labels for class 1 or flip 5% of the labels in class 1 and keep the labels for class 0.

3.2 Datasets

We collected 13 real-world datasets from different sources to conduct our experiments. Each dataset contains one or more types of error summarized in Table 3. For mislabels, we cannot find existing real-world datasets with both mislabels and ground truth. Since it is difficult to clean mislabels without ground truth (we do not know whether a label is mislabeled unless we have ground truth or domain knowledge), we injected mislabels in three real-world datasets.

Table 3: Dataset and Error Types

Datasets	Error Types				
	Inconsistencies	Duplicates	Missing Values	Outliers	*Mislabels Data
Airbnb		x	x	x	
Citation		x			
Company	x				
Credit			x	x	
EEG				x	x
KDD			x	x	x
Marketing			x		
Movie	x	x			
Restaurant	x	x			
Sensor				x	
Titanic			x		
University	x				
USCensus			x		x

*Note: Mislabels in datasets come from error injection.

Airbnb: This dataset contains 42,492 records on hotels in the top 10 tourist destinations and major US metropolitan areas scraped from Airbnb.com. Each record has 40 attributes including the number of bedrooms, price, location, etc. Demographic and economic attributes were scraped from city-data.com. The classification task is to determine whether the rating of each hotel is 5 or not. This dataset contains missing values, numerical outliers and duplicates.
Citation: [1] This dataset consists of titles of 5,005 publications

from Google Scholar and DBLP. Given a publication title, the classification task is to determine whether the paper is related to Computer Science or not. This dataset contains duplicates.

Company: [2] The original dataset contains over 2.5 million records about companies including company names and locations. Because of its large size, we randomly sampled 5% records (128,889 records) from the original dataset. Each record has seven attributes including company name, country, city, etc. The classification task is to predict whether the company sentiment is negative or not. This dataset contains inconsistent company names.

Credit: [3] This dataset consists of 150,000 credit data with 10 attributes including monthly income, age, then number of dependents, etc. The classification task for this dataset is to predict whether a client will experience financial distress in the next two years. This dataset has a class imbalance problem. There are only 6.7% records in minority class. This dataset primarily contains missing values and numerical outliers.

EEG: [4] This is a dataset of 14,980 EEG recordings. Each record has 14 EEG attributes. The classification task is to predict whether the eye-state is closed or open. This dataset contains numerical outliers. We inject mislabels into this dataset by randomly flip labels.

KDD: [6] This dataset contains 131,329 records about projects and donations from DonorsChoose.org. Each record has 100 attributes. The classification task is to predict whether a project is "exciting". This dataset has a class imbalance problem. There are 11% records in the minority class. This dataset contains missing values and numerical outliers. We inject mislabels into this dataset by randomly flip labels.

Marketing: This dataset consists of 8,993 data about household income from a survey. Each record has 14 demographic attributes including sex, age, education, etc. The classification task is to predict if the annual household income is less than $25000. This dataset contains missing values.

Movie: [10, 5] This dataset consists of 9,329 movie reviews, which we obtained by merging data from IMDB and TMDB datasets. Each record has seven attributes including title, language, score, etc. The classification task is to predict the genre of the movie (romance or comedy). It contains duplicates and inconsistent representations of languages.

Restaurant: [7] This dataset contains 12,007 records about restaurants, which we obtained by merging data from the Yelp and Yellowpages datasets. Each record has 10 attributes including city, category, rating, etc. The classification task is to predict whether the categorical price range of a restaurant is "$" or not. This dataset consists of duplicates and inconsistent restaurant names and categories.

Titanic: [9] This dataset contains 891 records and 11 attributes from the Titanic including name, sex, age, etc. The classification task is to determine whether the passenger survived or not. This dataset has a significant number of missing values.

Sensor: [8] The original sensor dataset contains 928,991 sensor recordings with eight attributes including temperature, humidity, light, etc. Because of the large size, we only used recordings from sensor 1 and sensor 2 and sampled the dataset to include 1 observation per hour, and day for each sensor. The sampled dataset contains 62,076 records. The classification task is to predict whether the readings came from a particular sensor (sensor 1 or sensor 2). This dataset contains outliers.

University: [11] This dataset contains 286 records about universities. Each record has 17 attributes including state, university name, SAT scores, etc. The classification task is to predict whether the expenses are greater than $7,000$ for each university. This dataset contains inconsistent representations for states and locations.

USCensus: [12] This dataset contains 32,561 US Census records for adults. Each record has 14 attributes including age, education, sex, etc. The classification goal is to predict whether the adult earns more than $50,000. This dataset contains missing values.

3.3 ML Models

We select seven classical and competitive classification algorithms in our experiments, including Logistic Regression, KNN, Decision Tree, Random Forest, Adaboost, XGBoost and Naive Bayes. We used scikit-learn [50] to train models. Each model is described below.

Logistic Regression: Logistic regression is a binary classifier that uses a Sigmoid function to create a linear classification boundary. Logistic regression uses optimization methods to determine the best regression coefficient of the function based on the training data [48].

KNN Classifier: KNN classifier uses a distance metric (Euclidean distance in our experiments), and the number of nearest neighbors (k) to calculate the distance between records in the training set. After calculating the distance it then retrieves the k nearest neighbors. Once the algorithm has found those neighbors, it can classify a record in the test set by computing its distance to other training records and using the class labels of the nearest neighbors to determine the label class of the unknown record [37].

Decision Tree: CART (Classification & Regression Trees) decision trees were used in this analysis. During training, the decision tree splits the data based on homogeneity. Gini index is used to measure node impurity and the attribute with minimum Gini index is used as the split node. This algorithm recursively partitions data until the splitting is completed [52].

Random Forest: Random forests are an ensemble learning method for classification. During training, the random forests algorithm constructs several decision trees and outputs an aggregated prediction (often the mode of the classes). Predictions in the test set are then made using this output [26].

Adaboost: Adaboost, also known as "Adaptive boost", is a meta-learning algorithm with high theoretical and empirical performance. It uses weak learners and transforms them into high performance learners by repeatedly emphasizing mispredicted instances. In experiments, the decision tree is our base learner. [36]

XGBoost: XGBoost is short for "Extreme Gradient Boosting". It is an implementation of gradient tree boosting system designed to be highly efficient and scalable. It is one of the most popular packages used by competitors to win ML challenges [21]. We use gradient boosted tree as our base learner in the experiments.

Naive Bayes: Naive Bayes predicts a class given a set of features using Bayes Theorem. This algorithm assumes independence among all attributes when the class is given [57].

We preprocess features before training ML models following these common practice: (1) Categorical variables were encoded using one hot encoding. (2) Text embeddings were used for non-categorical text attributes. We computed their tf-idf matrix using TfidfVectorizer from scikit-learn [50]. (3) Data were standardized to a mean of 0 and variance of 1. (4) Class-imbalanced datasets were downsampled, i.e., for every observation of the minor class, we randomly sample from the major class without replacement, to make the number of the instances in the major class equal to that in the minor class during the training.

3.4 Scenarios

Given a dataset with a train/test split, and a cleaning method, we can have different model performance depending on where the cleaning is performed. Table 4 shows the four cases: Case A repre-

sents a model built using the original dirty training set and tested on the original dirty test set; Case B represents a model built using the original dirty training set and tested on the cleaned test set; Case C represents a model built using the cleaned training set and tested on the original dirty test set; and Case D represents a model built using the cleaned training set and tested on the cleaned test set. Our goal is compare two of them to evaluate how cleaning affects model performance, and a chosen comparison between two cases is what we call a scenario in Table 1 (e.g., "BD" is a scenario). Do we then have a total of $C_4^2 = 6$ scenarios? The answer is no, and in fact, only two of them ("BD" and "CD") make sense as explained as follows:

- Scenario "BD". This shows the effects of cleaning dirty data in the training set when models are evaluated on the clean test set. This is reflective of the real-world model building phase, where we are given a dirty training set and we would like to know whether we need to clean the training set. Of course, in order to test whether cleaning training set helps, the two models need to be evaluated on the same cleaned test set.
- Scenario "CD". This shows the effects of cleaning dirty data in the test set when models are trained on the clean training set. This is reflective of the real-world model deployment phase, where the model is deployed and is being used for new test data, and we would like to know whether cleaning incoming dirty test data helps with the predictive performance.
- Scenario "AB". We do not compare the entries A and B because in real-world data cleaning, we do not consider evaluating a test set on a model trained with dirtiness, especially we are not interested in the performance improvement/degradation when swapping dirty with clean test sets.
- Scenario "AC". The comparison between A and C is also not reported because in production we are not interested in the performance of models evaluated on a dirty test set. It is common practice to ensure the test set is clean for evaluating the model performance.
- Scenario "AD" and "BC". The two scenarios are based on two different settings, which are not directly comparable.

Table 4: Where Cleaning is Performed

	Dirty Test Set	Cleaned Test Set
Dirty Training Set	A	B
Cleaned Training Set	C	D

Table 5: Where Cleaning is Performed (Missing Values)

	Deletion Test set	Imputation Test set
Deletion Training set	A	B
Imputation Training set	C	D

Special Handling for Missing Values. Missing values need special attention; they occur when no value is stored for a variable in an observation. We cannot train models when some data is missing. Thus, Cases A and B in Table 4 are not available for missing values. Instead of comparing dirty and cleaned datasets, we compare the difference between a deletion dataset (a dataset with missing values deleted) and an imputation dataset (a dataset with missing values imputed). The four cases for missing values are shown in Table 5. We only consider the "BD" scenario for missing values, which captures the difference of imputing and deleting training samples while testing the model performance in the imputed test set. This scenario is the authentic scenario we encounter in production, where it is not allowed to delete instances from the test set, so missing values in the test set have to be repaired using imputation methods.

4. RUNNING THE BENCHMARK

We have defined the domain of each key attribute. We call each valid assignment of key attributes an *experiment specification*. By definition, the experiment specification for a tuple t in relation R is $t[R - Flag]$, where $R \in \{R1, R2, R3\}$. For example, in Table 6, s_1, s_2 and s_3 are three experiment specifications in $R1$, $R2$ and $R3$, respectively.

In this section, we present how to run the benchmark to generate flags for each experiment specification in $R1$, $R2$ and $R3$. First, given an experiment specification, we generate a pair of performance metrics that will be used to determine the flag (Section 4.1). We then present our approach for controlling randomness in our experiments (Section 4.2). Finally, we show how to control false discoveries (Section 4.3).

Table 6: Example of Experiment specifications

s_1

Dataset	Error Type	Detection	Repair	ML Model	Scenario
EEG	Outliers	IQR	Mean Imputation	Logistic Regression	BD

s_2

Dataset	Error Type	Detection	Repair	Scenario
EEG	Outliers	IQR	Mean Imputation	BD

s_3

Dataset	Error Type	Scenario
EEG	Outliers	BD

4.1 Generating One Metric Pair

Given an experiment specification in R1, we can generate a pair of metrics through following steps:

(1) Split dataset. We first split dataset randomly into a training and a test set with a 70/30 ratio.
(2) Clean dataset. We clean the error in the training set and test set with the specific cleaning methods. To avoid the data leakage problem, all statistics needed for data cleaning, such as mean and standard deviation, are computed only on the training set and used to clean both training and test set.
(3) Training ML models. If the scenario is BD, we train two ML models, one on dirty training set and one on clean training set. If the scenario is CD, we only train one ML model on the clean training set. We tune hyper-parameters of ML models using random search and 5-fold cross validation.
(4) Evaluating ML models. If the scenario is BD, we evaluate two ML models (one trained on a dirty training set, anther trained on a clean training set) on the clean test set to get a pair of metrics. If the scenario is CD, the model trained on clean training set will be evaluated on dirty test set and clean test set respectively to get a pair of metrics. The evaluation metric is selected based on class imbalance. For class-imbalanced datasets (i.e., KDD and Credit), we use F1 score to evaluate the performance of models but for all the other datasets, we use accuracy as the evaluation metric.

For experiment specifications in $R2$, the difference is that we train various ML models at step (3) and select the model with the best validation accuracy from cross validation as the model evaluated in step (4). For R3, in addition to model selection, we use various cleaning methods to clean dataset at step (2) and select the cleaning method resulting in the best validation accuracy. At step (4), the test set cleaned by the best cleaning method is used to evaluate the best model.

EXAMPLE 4.1. *We take the specifications in Table 6 as an example to show how to generate one metric pair for each specification.*

To generate metric pair for s_1, we first split EEG into training and test datasets. We detect outliers in the training set and test set using IQR detection and repair them with mean imputation. The quantiles used in detection and mean used in repair are computed on the training set. Then, since the scenario here is BD, we train two logistic regression models on a dirty training set and a cleaned training set respectively. Finally, we evaluate two models on the cleaned test set and get two test accuracy scores to form a metric pair as shown in Table 7.

To generate the metric pair for s_2, we train various ML models. Table 8 shows that based on the validation accuracy, XGBoost is the best model trained on the dirty training set and KNN is the best model trained on the clean training set. We then evaluate two best models on the cleaned test set to get two test accuracy scores to form a metric pair.

To generate the metric pair for s_3, in addition to model selection, we use various cleaning methods. Table 9 shows the clean test accuracy of best models under each cleaning methods. Based on the validation accuracy, detecting outliers by SD and repairing by deletion is the best cleaning method. Hence, we use its metric pair as the metric pair for s_3.

4.2 Handling Randomness

The above procedure has randomness, which mainly comes two sources: (1) Search Randomness. This is introduced by random search in tuning hyper-parameters. Different search spaces will result in different performances, which may affect the evaluation of ML models. (2) Split Randomness. This is introduced by random train/test split. Different train/test splits may result in different error distribution, which may affect the quality of data cleaning.

4.2.1 Handling Search Randomness

To handle the randomness from random search, we repeat step (3) and step (4) 5 times with different seeds for random search. Each random search will generate a pair of metrics. To aggregate 5 pairs, for specifications in $R1$, since we care more about the performance of a model on average, we averages each metric in the pair over 5 random searches. For specifications in $R2$ and $R3$, similarly as we select the best model based on the validation accuracy, we select the one with the best validation accuracy from 5 random searches for each metric. After repeating experiments with 5 times random search, we still have one metric pair for each specification, but it provides a better evaluation of the model performance and reduces the effect of randomness caused by random search.

EXAMPLE 4.2. *Table 10 shows the five metric pairs we get from repeating random search with 5 different seeds. For s_1, we average over 5 random search for each metric. For s_2, as shown in Table 11, we select the one with the best validation accuracy from 5 random search for each metric. s_3 can be generated in a similar way.*

4.2.2 Handling Split Randomness

To avoid the occasionality caused by a train/test split, we randomly split each dataset 20 times with different seeds and repeat the experiments under the same protocol on each train/test split. Each split will generate one pair of metrics. Hence, we end up with 20 metric pairs for each specification.

EXAMPLE 4.3. *Table 12 shows 20 pairs of metrics from 20 different train/test splits for s_1.*

Given 20 pairs of metrics for each specification $s = t[R-Flag]$, we generate the flag $t[Flag]$ using paired sample t-test. We consider 20 metric pairs as two sets of 20 observations from the same dataset before and after data cleaning. Then paired sample t-test can determine whether the mean difference between two sets of observations is zero, positive or negative. The paired sample t-test is formally defined below.

For each specification $t[R-Flag]$, let μ^t be the mean difference of the metrics of the ML model before and after we clean the error in the dataset with the detection and repair method. We define null and alternative hypotheses for three types of paired sample t-test as:

Hypothesis	Two-tailed t-test	Upper-tailed t-test	Lower-tailed t-test
Null	$H_0^t : \mu^t = 0$	$H_1^t : \mu^t \leq 0$	$H_2^t : \mu^t \geq 0$
Alternative	$H_a^t : \mu^t \neq 0$	$H_b^t : \mu^t > 0$	$H_c^t : \mu^t < 0$

Table 7: s_1 Metric Pairs

	Train on Dirty Training Set		Train on Clean Training Set	
Model	Validation Accuracy	Clean Test Accuracy	Validation Accuracy	Clean Test Accuracy
Logistic Regression	0.638849	0.634179	0.673467	0.668892
Metric Pair: (0.634179, 0.668892)				

Table 8: s_2 Metric Pairs

	Train on Dirty Training Set		Train on Clean Training Set	
Model	Validation Accuracy	Clean Test Accuracy	Validation Accuracy	Clean Test Accuracy
AdaBoost	0.763205	0.711393	0.718193	0.715176
Decision Tree	0.822621	0.754784	0.796487	0.810414
KNN	0.895481	0.821095	0.948312	0.956386
Logistic Regression	0.638849	0.634179	0.673467	0.668892
Naive Bayes	0.453365	0.457276	0.634745	0.638407
Random Forest	0.918556	0.854695	0.903680	0.907210
XGBoost	0.932098	0.862706	0.920369	0.922786
Metric Pair: (0.862706, 0.956386)				

Table 9: s_3 Metric Pairs

Detect Method	Repair Method	Validation Accuracy of Best Model on Clean Training set	Clean Test Accuracy of Best Model on Dirty Training set	Clean Test Accuracy of Best Model on Clean Training set
SD	Delete	0.959370	0.937612	0.969928
SD	Mean Imputation	0.955179	0.938140	0.964174
SD	Median Imputation	0.955179	0.938140	0.964174
SD	Mode Imputation	0.955179	0.937917	0.964174
IQR	Delete	0.958072	0.929052	0.967190
IQR	Mean Imputation	0.948312	0.862706	0.956386
IQR	Median Imputation	0.944115	0.868046	0.951268
IQR	Mode Imputation	0.946308	0.870049	0.957499
IF	Delete	0.959250	0.935250	0.969846
IF	Mean Imputation	0.957466	0.925456	0.966845
IF	Median Imputation	0.957371	0.924789	0.966177
IF	Mode Imputation	0.956990	0.927236	0.966400
Metric Pair: (0.937612, 0.969928)				

Table 10: Aggregate Five Random Search For s_1

Train on Dirty Training Set			Train on Clean Training Set	
Random Search Seed	Validation Accuracy	Clean Test Accuracy	Validation Accuracy	Clean Test Accuracy
8006	0.638849	0.634179	0.673467	0.668892
6130	0.638849	0.635292	0.673562	0.667557
5824	0.638849	0.634846	0.673372	0.668669
3659	0.638754	0.635291	0.672323	0.668892
3239	0.639040	0.634179	0.672323	0.669114
Average		0.634767		0.668625
Aggregated Metric Pair: (0.634767, 0.668625)				

Table 11: Aggregate Five Random Search For s_2

Train on Dirty Training Set			Train on Clean Training Set	
Random Search Seed	Validation Accuracy of the Best Model	Clean Test Accuracy of the Best Model	Validation Accuracy of the Best Model	Clean Test Accuracy of the Best Model
8006	0.932098	0.862706	0.948312	0.956386
6130	0.930381	0.868046	0.948312	0.956386
5824	0.932098	0.862706	0.920369	0.922786
3659	0.930381	0.868046	0.948312	0.956386
3239	0.932098	0.862706	0.948312	0.956386
Metric Pair: (0.862706, 0.956386)				

Table 12: Accuracy Evaluated on the Clean Test Set

Split Seed	B	D	Split Seed	B	D
v144	0.632488	0.657321	v5192	0.631954	0.67401
v235	0.634757	0.668625	v5374	0.638362	0.676992
v2516	0.625812	0.666266	v5396	0.641032	0.672452
v2895	0.636404	0.662394	v6542	0.63992	0.670049
v2962	0.637161	0.674633	v7751	0.640098	0.669871
v3462	0.644726	0.673654	v7813	0.634535	0.676591
v3562	0.635514	0.67401	v8093	0.636271	0.666489
v4225	0.641478	0.674989	v8444	0.632443	0.673431
v4764	0.649177	0.680196	v905	0.636671	0.673565
v5056	0.629773	0.669381	v9394	0.632176	0.668803

We run three types of paired sample t-test on 20 metric pairs. Let p_0, p_1, p_2 be the p-values of two-tailed t-test, upper-tailed t-test and lower-tailed t-test respectively. Let α be the significant level. The procedure for determining flags using paired sample t-test is described below:

(1) if $p_0 \geq \alpha$, t[Flag] = "S".
(2) if $p_0 < \alpha$ and $p_1 < \alpha$, t[Flag] = "P".
(3) if $p_0 < \alpha$ and $p_2 < \alpha$, t[Flag] = "N".

The intricacy of conducting two-tailed test and one-tailed test together lies in the fact that if the test statistics distribution is symmetric (e.g., Gaussian), the p-value in one of one-tailed tests is half of the p-value in a two-tailed test. Hence, a two-tailed test with significance does imply that the one-tailed test under the same distribution is also significant; yet if the one-tailed test is significant, the two-tailed one is not necessarily significant. What is criticized often is that people only report a one-tailed test p-value because the two-tailed test is insignificant. However, in our case, we do not face this claim because we conduct three tests and only report the one-tailed test results if its corresponding two-tailed test is significant.

EXAMPLE 4.4. *For s_1, we run two-tailed, upper-tailed and lower-tailed sample t-test on 20 metric pairs (Table 12). Assume $\alpha = 0.05$. Table 13 shows $p_0 < \alpha$ and $p_1 < \alpha$. Thus, the flag of s_1 is determined to be "P".*

Table 13: p-values in t-test and Hypothesis Testing

Type	p-value
Two-tailed (p_0)	3.82E-17
Upper-tailed (p_1)	1.91E-17
Lower-tailed (p_2)	1

4.3 Controlling False Discoveries

Since we aim at studying the significant impacts of data cleaning techniques on ML performances in spite of the search and split randomness, we face the challenges that not all statistically significant results in individual hypothesis tests are true discoveries. Some results are significant simply due to the large number of tests examined.

This is commonly known as the *multiple hypothesis testing* or the *multiple comparisons* problem in the statistics literature [58]. To see the effect of multiple testing, consider a case where there are 20 hypotheses to test and we set a significance level of $\alpha = 0.05$. The probability of observing at least one significant result just due to chance is $1 - (1 - \alpha)^{20} \approx 0.64$. With just 20 tests considered, we have a 64% chance of observing at least one significant result, even if all of the tests are actually not significant.

With $3,990,570$ and 150 hypotheses in our relations $R1$, $R2$ and $R3$, respectively, it is highly likely that our results contain many false discoveries by chance. Strategies to control the false discoveries caused by the multiple hypothesis testing problem usually adjust the significance level α in some way [58, 18, 19]. For example, a simple way to adjust α is called the *Bonferroni correction* [20][1], which tests each hypothesis at the significance level of $\frac{\alpha}{m}$ instead of α, where m is the number of tests. Unfortunately, this correction also significantly increases the number of false negatives because it can miss a lot of true significant tests. In practice, it is usually very hard to determine the desirable significance level for every test so that both false positives and false negatives are minimized [58].

Instead of adjusting the significance level α for every test, another strategy is to rank the tests by their *p-values*, which indicate the statistical significance levels of tests [65]. This is called the FDR approach [33, pg. 687] where we ensure that in expectation, $\frac{V}{R} = FDR$, where R is the total number of rejections, and V the number of false rejections. Common FDR approaches include Benjamini-Hochberg (BH) and Benjamini-Yekutieli (BY) procedures, where we try to control the FDR that is "(upper) bounded by a user-defined level α" ([33, pg. 688]). We employed the BY procedure since it controls the FDR under arbitrary dependence assumptions[2]. For each relation, we conduct a separate multiple hypothesis testing. We assign α to be 0.05 in our experiments.

EXAMPLE 4.5. *P-values for s_1 (Table 13) are corrected in a multiple testing setting where we run BY-Procedure on p-values of all of hypotheses in R1. Table 14 shows the corrected p-values for s_1. Since $p_0 < \alpha$ and $p_1 < \alpha$. Thus, the flag of s_1 is finally determined to be "P".*

Table 14: Corrected p-values of the Example Analysis (Outlier)

Test Type	Corrected p-value
Two-tailed (p_0)	6.28E-17
Upper-tailed (p_1)	3.25E-17
Lower-tailed (p_2)	1

5. ANALYZING BENCHMARK RESULTS

We inspect the correlation between the test accuracy scores in the scenarios BD and CD. In Figure 1, the scatter plots are generated based on R3. We plot the test accuracy scores of 20 splits in each dataset, given the best model and the best data cleaning method. Different colors correspond to various datasets. The visualization shows that: (1) cleaning does not always improve the ML performance; (2) cleaning can help improve accuracy scores up to 10% (e.g., cleaning outliers in the scenario CD); (3) the improvements vary largely from one error type to another; (4) we need a systematic and principled approach to analyze the results.

Table 15 presents the results according to the query templates we define in Section 2.2. In Section 5.1 to 5.5, we present the impacts of each type of error on ML by analyzing the query results. In Section 5.6, we summarize the key insights.

5.1 Inconsistencies

Insight #1: Cleaning inconsistencies is more likely to have insignificant impact and unlikely to have negative impact on ML. Insight #2: Model selection increases the probability of having positive impacts on ML.

Q1: We first group by flags on the tuples in R1, R2 and R3. Table 15:Q1(E=Inconsistencies) shows no negative flags ("N") in the impact directionality. For every relation, the insignificant changes ("S") have the largest likelihood. This implies that cleaning inconsistency in both training and test sets is unlikely to produce negative impacts on the ML model performance. Furthermore, comparing the percentages of "P" among all the relations, selecting the best ML model and the best data cleaning strategy helps to gradually introduce positive changes to ML performances after data cleaning.

Q2: Table 15: Q2(E=Inconsistencies) shows the query results of grouping by flags and scenarios, which follows the tendency we observe in Q1, i.e., no negative impacts on ML performance after cleaning inconsistency in both scenarios. Adding an auto ML model/cleaning tuner increases the changes of positive impacts.

[1]Bonferroni correction is one of many familywise error rate (FWER) methods, for an extensive survey, c.f. [70].

[2]https://en.wikipedia.org/wiki/False_discovery_rate (last accessed: February 12, 2019).

Table 15: Benchmark Results(Organized by Query)

Q1

Q1(E=Inconsistencies)

R	P	S	N
R1	14.29% (8)	85.71% (48)	0%(0)
R2	25.0% (2)	75.0% (6)	0%(0)
R3	37.5% (3)	62.5% (5)	0%(0)

Q1(E=Duplicates)

R	P	S	N
R1	17.86% (10)	71.43% (40)	10.71% (6)
R2	12.5% (1)	62.5% (5)	25.0% (2)
R3	25.0% (2)	50.0% (4)	25.0% (2)

Q1(E=Mislabels)

R	P	S	N
R1	59.52% (75)	26.19% (33)	14.29% (18)
R2	61.11% (11)	27.78% (5)	11.11% (2)
R3	61.11% (11)	27.78% (5)	11.11% (2)

Q1(E=Outliers)

R	P	S	N
R1	31.55% (265)	57.02% (479)	11.43% (96)
R2	33.33% (40)	60% (72)	6.67% (8)
R3	30% (3)	70% (7)	0% (0)

Q1(E=Missing Values)

	P	S	N
R1	61.51% (155)	34.92% (88)	3.57% (9)
R2	50.00% (18)	50.00% (18)	0.00% (0)
R3	50.00% (3)	50.00% (3)	0.00% (0)

Q2

Q2(E=Inconsistencies)

R	Scenario	P	S
R1	BD	7.14% (2)	92.86% (26)
	CD	21.43% (6)	78.57% (22)
R2	BD	25.0% (1)	75.0% (3)
	CD	25.0% (1)	75.0% (3)
R3	BD	25.0% (1)	75.0% (3)
	CD	50.0% (2)	50.0% (2)

Q2(E=Duplicates)

R	Scenario	P	S	N
R1	BD	10.71% (3)	75.0% (21)	14.29% (4)
	CD	25.0% (7)	67.86% (19)	7.14% (2)
R2	BD	0.0% (0)	50.0% (2)	50.0% (2)
	CD	25.0% (1)	75.0% (3)	0.0% (0)
R3	BD	0.0% (0)	50.0% (2)	50.0% (2)
	CD	50.0% (2)	50.0% (2)	0.0% (0)

Q2(E=Mislabels)

R	Scenario	P	S	N
R1	BD	50.79% (32)	49.21% (31)	0.0% (0)
	CD	68.25% (43)	3.17% (2)	28.57% (18)
R2	BD	44.44% (4)	55.56% (5)	0.0% (0)
	CD	77.78% (7)	0.0% (0)	22.22% (2)
R3	BD	44.44% (4)	55.56% (5)	0.0% (0)
	CD	77.78% (7)	0.0% (0)	22.22% (2)

Q2(E=Outliers)

R	Scenario	P	S	N
R1	BD	50.79% (32)	49.21% (31)	0.0% (0)
	CD	68.25% (43)	3.17% (2)	28.57% (18)
R2	BD	44.44% (4)	55.56% (5)	0.0% (0)
	CD	77.78% (7)	0.0% (0)	22.22% (2)
R3	BD	44.44% (4)	55.56% (5)	0.0% (0)
	CD	77.78% (7)	0.0% (0)	22.22% (2)

Q3

Q3(E=Inconsistencies)

R	Model	P	S
R1	AdaBoost	12.5% (1)	87.5% (7)
	Decision Tree	0.0% (0)	100.0% (8)
	KNN	25.0% (2)	75.0% (6)
	Logistic Regression	12.5% (1)	87.5% (7)
	Naive Bayes	12.5% (1)	87.5% (7)
	Random Forest	25.0% (2)	75.0% (6)
	XGBoost	12.5% (1)	87.5% (7)

Q3(E=Duplicates)

R	Model	P	S	N
R1	AdaBoost	25.0% (2)	75.0% (6)	0.0% (0)
	Decision Tree	0.0% (0)	100.0% (8)	0.0% (0)
	KNN	12.5% (1)	87.5% (7)	0.0% (0)
	Logistic Regression	25.0% (2)	62.5% (5)	12.5% (1)
	Naive Bayes	37.5% (3)	50.0% (4)	12.5% (1)
	Random Forest	12.5% (1)	62.5% (5)	25.0% (2)
	XGBoost	12.5% (1)	62.5% (5)	25.0% (2)

Q3(E=Mislabels)

R	Model	P	S	N
R1	AdaBoost	77.78% (14)	11.11% (2)	11.11% (2)
	Decision Tree	66.67% (12)	22.22% (4)	11.11% (2)
	KNN	50.0% (9)	38.89% (7)	11.11% (2)
	Logistic Regression	55.56% (10)	33.33% (6)	11.11% (2)
	Naive Bayes	27.78% (5)	38.89% (7)	33.33% (6)
	Random Forest	61.11% (11)	27.78% (5)	11.11% (2)
	XGBoost	77.78% (14)	11.11% (2)	11.11% (2)

Q3(E=Outliers)

R	Model	P	S	N
R1	AdaBoost	20% (24)	62.50% (75)	17.50% (21)
	Decision Tree	27.50% (33)	65.83% (79)	6.67% (8)
	KNN	50% (60)	44.17% (53)	5.83% (7)
	Logistic Regression	28.33% (34)	56.67% (68)	15% (18)
	Naive Bayes	34.17% (41)	58.33% (70)	7.5% (9)
	Random Forest	29.17% (35)	56.67% (68)	14.17% (17)
	XGBoost	31.67% (38)	55% (66)	13.33% (16)

Q3(E=Missing Values)

R	Model	P	S	N
R1	AdaBoost	20% (24)	62.50% (75)	17.50% (21)
	Decision Tree	27.50% (33)	65.83% (79)	6.67% (8)
	KNN	50% (60)	44.17% (53)	5.83% (7)
	Logistic Regression	28.33% (34)	56.67% (68)	15% (18)
	Naive Bayes	34.17% (41)	58.33% (70)	7.5% (9)
	Random Forest	29.17% (35)	56.67% (68)	14.17% (17)
	XGBoost	31.67% (38)	55% (66)	13.33% (16)

Q4

Q4.1(E=Outliers)

R	Detect Method	P	S	N
R1	IF	32.14% (90)	47.14% (132)	20.71% (58)
	IQR	50.71% (142)	38.21% (107)	11.07% (31)
	SD	11.79% (33)	85.71% (240)	2.50% (7)
R2	IF	32.50% (13)	60% (24)	7.50% (3)
	IQR	52.50% (21)	35% (14)	12.50% (5)
	SD	15% (6)	85% (34)	0% (0)

Q4.2(E=Outliers)

R	Repair Method	P	S	N
R1	Delete	32.86% (69)	50.48% (106)	16.67% (35)
	Mean Imputation	31.90% (67)	60.95% (128)	7.14% (15)
	Mode Imputation	30.95% (65)	57.14% (120)	11.90% (25)
	Median Imputation	30.48% (64)	59.52% (125)	10.00% (21)
R2	Delete	33.33% (10)	53.33% (16)	13.33% (4)
	Mean Imputation	33.33% (10)	63.33% (19)	3.33% (1)
	Mode Imputation	30.00% (9)	63.33% (19)	6.67% (2)
	Median Imputation	36.67% (11)	60.00% (18)	3.33% (1)

Q4.2(E=Missing Values)

R	Imputation Method	P	S	N
R1	Mean Dummy	54.76% (23)	40.48% (17)	4.76% (2)
	Median Dummy	59.52% (25)	35.71% (15)	4.76% (2)
	Mode Dummy	64.29% (27)	30.95% (13)	4.76% (2)
	Mean Mode	64.29% (27)	33.33% (14)	2.38% (1)
	Median Mode	61.90% (26)	35.71% (15)	2.38% (1)
	Mode Mode	64.29% (27)	33.33% (14)	2.38% (1)
R2	Mean Dummy	50.00% (3)	50.00% (3)	0.00% (0)
	Median Dummy	50.00% (3)	50.00% (3)	0.00% (0)
	Mode Dummy	50.00% (3)	50.00% (3)	0.00% (0)
	Mean Mode	50.00% (3)	50.00% (3)	0.00% (0)
	Median Mode	50.00% (3)	50.00% (3)	0.00% (0)
	Mode Mode	50.00% (3)	50.00% (3)	0.00% (0)

Q5

Q5(E=Inconsistencies)

R	Dataset	P	S
R1	Company	28.57% (4)	71.43% (10)
	Movie	21.43% (3)	78.57% (11)
	Restaurant	0.0% (0)	100.0% (14)
	University	7.14% (1)	92.86% (13)
R2	Company	0.0% (0)	100.0% (2)
	Movie	100.0% (2)	0.0% (0)
	Restaurant	0.0% (0)	100.0% (2)
	University	0.0% (0)	100.0% (2)
R3	Company	0.0% (0)	100.0% (2)
	Movie	100.0% (2)	0.0% (0)
	Restaurant	50.0% (1)	50.0% (1)
	University	0.0% (0)	100.0% (2)

Q5(E=Duplicates)

R	Dataset	P	S	N
R1	Airbnb	0.0% (0)	85.71% (12)	14.29% (2)
	Citation	7.14% (1)	85.71% (12)	7.14% (1)
	Movie	64.29% (9)	28.57% (4)	7.14% (1)
	Restaurant	0.0% (0)	85.71% (12)	14.29% (2)
R2	Airbnb	0.0% (0)	100.0% (2)	0.0% (0)
	Citation	0.0% (0)	100.0% (2)	0.0% (0)
	Movie	50.0% (1)	0.0% (0)	50.0% (1)
	Restaurant	0.0% (0)	50.0% (1)	50.0% (1)
R3	Airbnb	0.0% (0)	100.0% (2)	0.0% (0)
	Citation	50.0% (1)	50.0% (1)	0.0% (0)
	Movie	50.0% (1)	0.0% (0)	50.0% (1)
	Restaurant	0.0% (0)	50.0% (1)	50.0% (1)

Q5(E=Mislabels)

R	Dataset	P	S	N
R1	EEG_major	78.57% (11)	14.29% (2)	7.14% (1)
	EEG_minor	78.57% (11)	21.43% (3)	0.0% (0)
	EEG_uniform	78.57% (11)	14.29% (2)	7.14% (1)
	KDD_major	28.57% (4)	21.43% (3)	50.0% (7)
	KDD_minor	50.0% (7)	50.0% (7)	0.0% (0)
	KDD_uniform	35.71% (5)	14.29% (2)	50.0% (7)
	USCensus_major	50.0% (7)	42.86% (6)	7.14% (1)
	USCensus_minor	64.29% (9)	35.71% (5)	0.0% (0)
	USCensus_uniform	71.43% (10)	21.43% (3)	7.14% (1)
	EEG_major	100.0% (2)	0.0% (0)	0.0% (0)
	EEG_minor	100.0% (2)	0.0% (0)	0.0% (0)
	EEG_uniform	100.0% (2)	0.0% (0)	0.0% (0)
	KDD_major	0.0% (0)	50.0% (1)	50.0% (1)
	KDD_minor	50.0% (1)	50.0% (1)	0.0% (0)
	KDD_uniform	0.0% (0)	50.0% (1)	50.0% (1)
	USCensus_major	50.0% (1)	50.0% (1)	0.0% (0)
	USCensus_minor	50.0% (1)	50.0% (1)	0.0% (0)
	USCensus_uniform	100.0% (2)	0.0% (0)	0.0% (0)
R3	EEG_major	100.0% (2)	0.0% (0)	0.0% (0)
	EEG_minor	100.0% (2)	0.0% (0)	0.0% (0)
	EEG_uniform	100.0% (2)	0.0% (0)	0.0% (0)
	KDD_major	0.0% (0)	50.0% (1)	50.0% (1)
	KDD_minor	50.0% (1)	50.0% (1)	0.0% (0)
	KDD_uniform	0.0% (0)	50.0% (1)	50.0% (1)
	USCensus_major	50.0% (1)	50.0% (1)	0.0% (0)
	USCensus_minor	50.0% (1)	50.0% (1)	0.0% (0)
	USCensus_uniform	100.0% (2)	0.0% (0)	0.0% (0)

Q5(E=Outliers)

R	Dataset	P	S	N
R1	Airbnb	0.0% (0)	85.71% (12)	14.29% (2)
	Citation	7.14% (1)	85.71% (12)	7.14% (1)
	Movie	64.29% (9)	28.57% (4)	7.14% (1)
	Restaurant	0.0% (0)	85.71% (12)	14.29% (2)
R2	Airbnb	0.0% (0)	100.0% (2)	0.0% (0)
	Citation	0.0% (0)	100.0% (2)	0.0% (0)
	Movie	50.0% (1)	0.0% (0)	50.0% (1)
	Restaurant	0.0% (0)	50.0% (1)	50.0% (1)
R3	Airbnb	0.0% (0)	100.0% (2)	0.0% (0)
	Citation	50.0% (1)	50.0% (1)	0.0% (0)
	Movie	50.0% (1)	0.0% (0)	50.0% (1)
	Restaurant	0.0% (0)	50.0% (1)	50.0% (1)

Q5(E=Missing Values)

Relation	Dataset	P	S	N
R1	Airbnb	23.21% (39)	70.83% (119)	5.95% (10)
	Credit	22.62% (38)	43.35% (73)	33.93% (57)
	EEG	60.12% (101)	33.93% (57)	5.95% (10)
	KDD	3.57% (6)	93.45% (157)	2.98% (5)
	Sensor	48.21% (81)	43.45% (73)	8.33% (14)
R2	Airbnb	37.50% (9)	62.50%(15)	0% (0)
	Credit	0% (0)	66.67% (16)	33.33% (8)
	EEG	79.17% (19)	20.83% (5)	0% (0)
	KDD	0% (0)	100% (0)	0% (0)
	Sensor	50% (12)	50% (12)	0% (0)
R3	Airbnb	50% (1)	50% (1)	0% (0)
	Credit	0% (0)	100% (2)	0% (0)
	EEG	50% (1)	50% (1)	0% (0)
	KDD	0%(0)	100% (2)	0% (0)
	Sensor	50% (1)	50% (1)	0% (0)

Q3: Grouped by ML models, it is noticeable that all the ML models have demonstrated the same tendency in the impacts of data cleaning, as shown in Table 15: Q3(E=Inconsistencies). Again, there is no negative impact on ML performances that cleaning inconsistency can induce.

Q5: At last, we group by datasets and provide a view on dataset choice and its influence on ML performance with data cleaning strategy incorporated. As shown in Table 15:(E=Inconsistencies), in general, the pattern holds that insignificant impact of cleaning inconsistency prevails; no negative impacts of cleaning inconsistency was found. It is probably due to the idiosyncrasy in the Movie dataset, which has 48% of inconsistencies[3], the improvement an

[3]We calculate the inconsistency of the datasets using the percentage of minority class. For instance, in the dataset Movie, we correct the

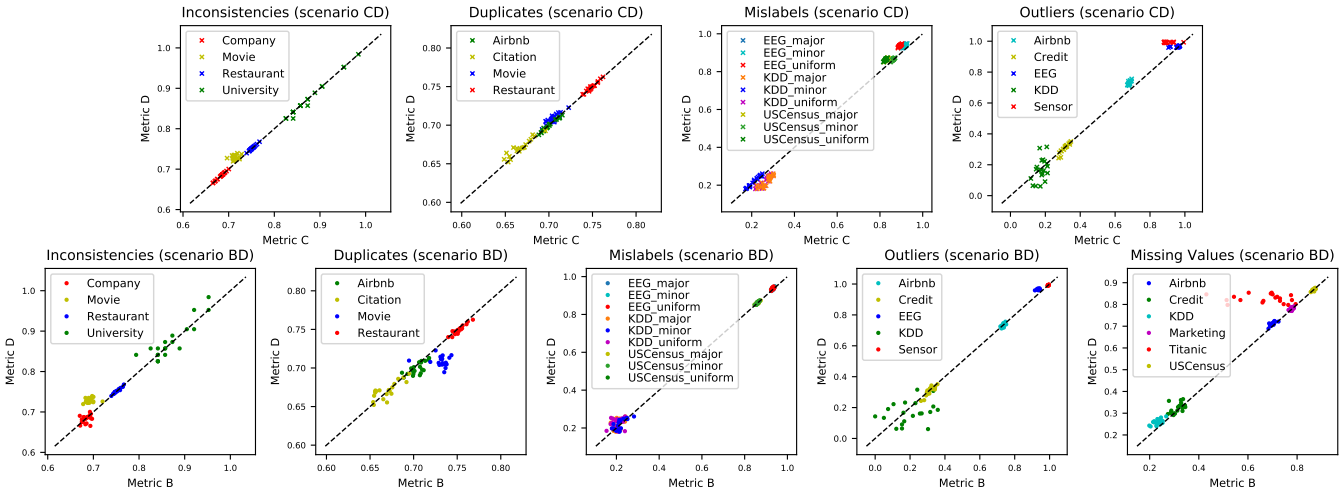

Figure 1: Test Accuracy Scores in Scenarios BD and CD of 20 Splits for Five Error Types

auto ML/cleaning strategy tuner brings is 78.57% in the direction of positive changes.

5.2 Duplicates

Insight #1: Cleaning duplicates is more likely to have insignificant impacts on ML and it is possible to produce negative impacts. Insights #2: With model selection, negative impacts caused by cleaning test set can be eliminated.

Q1: From the query on flags shown in Table 15:Q1(E=Duplicates), it is unclear that if cleaning duplicates could bring either positive or negative impacts. In all the relations, the number of insignificant flags has the largest percentage compared with the positive and negative flags.

Q2: Looking scenarios grouped by flags in Table 15: Q2 (E= Duplicates), using a clean test set with a clean training model (CD) is highly unlikely to result in negative changes of ML performance. Especially when we utilize an auto ML/data cleaning tuner, the negative impacts disappear. In the scenario BD, utilizing an ML/cleaning tuner does not decrease the chance of yielding negative impacts when we clean the duplicates. This observation is also made from the scatter plot on BD in Figure 1, where there are more points with the "N" flag than the other two flags. The reason is due to the various duplication rates of features in the datasets, as we see when analyzing the results of Q5 below.

Q3: We observe from Table 15:Q3(E=Duplicates) that for all the ML models, AdaBoost, KNN and Decision Tree tend to have no negative impact.

Q5: The discrepancies of datasets affect the ML performances largely as we observe from Table 15:Q5(E=Duplicates). For the datasets "Airbnb" and "Citation", the negative impacts could be dampened to zero if we add an auto ML/cleaning tuner. Yet for the datasets "Movie" and "Restaurant", the negative impacts even increase after we use an auto ML/cleaning tuner. This might be due to the protocol of generating the duplicates, where we have combined the two sources of datasets and do not correlate the duplication rate with the class distribution. Finally, "Movie" has a duplication rate of 40%, "Citation" 10%, "Airbnb" 10%, and "Restaurant" 10%.

values *English* and *en* under the variable "Language". The value *English* takes 52% in all the attribute values; the value *en* 48%. We therefore replace *en* with *English*. This gives us an inconsistency rate of 48%.

5.3 Mislabels

Insight #1: Cleaning mislabels are highly likely to have positive impacts on ML. Insight #2: Cleaning mislabels for models that have bad performances may produce negative impacts.

Q1: Table 15:Q1(E=Mislabels) is generated by grouping flags, which demonstrates strong positive impacts of cleaning mislabels in all relations. The likelihood of improving ML model performances after cleaning mislabels (the "P" flag) is higher than that of insignificant changes (the "S" flag) , which is more likely than reducing the ML model performances (the "N" flag).

Q2: In the scenario CD shown in Table 15:Q2(E=Mislabels), cleaning mislabels has always a higher likelihood of rendering a positive impact on the ML performances, which corresponds to the observation from Figure 1. This could be understood by the fact that using a clean training set and a clean test set, i.e., sets without label flipping, ML performances are generally better than using a clean training set and a corrupted test set. There are no negative flags in the scenario BD under the error type "Mislabel". We find out that if we clean the dirtiness in the training set, it is highly unlikely that the cleaning method has a negative impact on the ML model performance.

Q3: Apart from Naive Bayes (Table 15:Q2(E=Mislabels)), all the other ML models have demonstrated a stronger tendency in producing more accurate predictions after cleaning the mislabels.

Q5: We observe that the negative impact only occurs when the model has a bad performance (accuracy $< 50\%$). This is because when we inject mislabels by flipping labels, we are more likely to flip a label that bad models predict incorrectly. Then accuracy of bad models is likely to be improved after injecting mislabels. Hence, cleaning mislabels may reduce the performance of bad models. With model selection, the negative impact is reduced, since we are less likely to have a bad model.

5.4 Outliers

Insight #1: Cleaning outliers is more likely to have insignificant and positive impacts, but it may produce negative impacts on ML. Insight #2: With model selection, the probability of having negative impacts can be reduced. With cleaning method and model selection, it is unlikely to have negative impacts. Insight #3: The probability of having positive and negative impacts is highly related to datasets and detection methods.

Q1: Table 15:Q1(E=Outliers) shows the results of Q1 for outliers. The results of R1 indicate that cleaning outliers mostly have no impact or positive impact on ML, but sometimes it may negatively affect ML. This is because outlier detection and repair are not completely accurate. The detection methods are usually based on the assumptions of the error distribution, which may not be the underlying authentic distribution. Some outliers may be true data instead of errors although they are distant from other instances. Cleaning such data will distort the true distribution of the dataset, which results in negative impact on ML.

The results of R2 and R3 show that with the model selection and with the cleaning method selection the percentage of flag "N" decreases to 0 and the percentage of "S" increases, while the flag "P" remains at the similar percentage. This indicates that using model selection and cleaning method selection can eliminate the negative impact of cleaning outliers and improve the robustness without losing too much benefit.

Q2: Table 15:Q1(E=Outliers) shows the results of Q2 for outliers. In R1, R2 and R3, BD and CD have similar percentage scores of "P", "S" and "N". This indicates that cleaning outliers in the training and test sets have similar impacts on ML models.

Q3: Table 15:Q3(E=Outliers) shows the results of Q3 for outliers. KNN has more "P" flags, less "N" and "S" flags than other models. Therefore, KNN is the most sensitive model to outliers and gains most benefit from cleaning outliers. Other models are affected similarly.

Q4.1: Table 15:Q4.1(E=Outliers) shows the results of Q4.1. In R1, IQR and IF have more "P" flags and "N" flags than the SD method. This indicates that IQR and IF are more aggressive than SD and SD is more conservative. In R2, the negative impact of IF and IQR is largely eliminated by model selection, but the positive impact remains.

Q4.2: Table 15:Q4.2(E=Outliers) shows that there is no significant difference between repair methods in both R1 and R2.

Q5: Table 15:Q5(E=Outliers) shows the result of Q5. In R1, most negative flags are from "Credit" and "KDD" datasets. In R2, all of negative flags are from "Credit". EEG and Sensor have more "P" flags than other datasets. This echoes our interpretation in Q1 that the impact of outliers on ML models largely depends on the error distribution in the dataset.

5.5 Missing Values

Insight #1: Cleaning missing values by imputation are more likely to improves the performance or achieves similar performance as deleting missing values.

Insight #2: With model selection and imputation method selection, ML models tend to be more robust to missing values.

As mentioned in section 3.4, we only consider one scenario (BD) for missing values. Therefore, we do not run Q2 for missing values.

Q1: Table 15:Q1(E=Missing Values) shows the results of Q1 for missing values. The result of R1 exhibits that cleaning missing values by imputation mostly improves the performance or achieves similar performance as deleting missing values. But there are still few "N" flags, which indicates that imputation can sometimes hurt the performance. The reason is that imputation is simply an approximation of ground truth. If the imputation is distant from ground truth, it may introduce bias to data and reduce the performance of ML.

The results of R2 and R3 show that, with model selection, "N" flags are eliminated and the percentage of "S" flags increases. Therefore, ML becomes more robust to missing values.

Q3: Table 15:Q3(E=Missing Values) presents the results of Q3. Only Naive Bayes has "N" flags. Therefore, Naive Bayes is the most vulnerable model to missing value imputation. Other models have similar results.

Q4.2: Table 15:Q4.2(E=Missing Values) shows the results for Q4.2. In both R1 and R2, different imputation methods have similar results. Therefore, different imputation methods have similar impacts.

Q5: Table 15:Q5(E=Missing Values) shows the results of Q5. In R1, "USCensus" has much less "P" flags and more "N" flags than other datasets. The reason may be that the imputation in this dataset is distant from ground truth. All of flags in "KDD" are "P", which indicates that imputation in this dataset may be close to ground truth. R2 and R3 show that with the model selection the negative impact in "UScensus" caused by missing value imputation is eliminated.

5.6 Summary of Key Insights

Data cleaning does not necessarily improve the quality of downstream ML models. We see from the result analyses that applying cleaning methods blindly could negatively impact model performances. Cleaning methods could introduce biases and sometimes this bias is larger than the original bias: (1) It is unclear that if cleaning *duplicates* could bring either positive or negative impacts, defining a better duplicate injection protocol is key. (2) Cleaning *outliers* mostly have no impact or positive impact on ML, but sometimes it may negatively affect ML. The effects are highly dependent on detection and repair techniques. (3) If the imputation of *missing values* is distant from ground truth, it may introduce bias to data and reduce the performance of ML.

Interpretation of the experimental results should take into the following factors: the errors and their distributions on the datasets (which are unknown), the correctness of the cleaning algorithms (which are also unknown without ground truth), and the internal structures of the ML models (which can be hard to interpret for some models). Since these factors are jointly at work, it could be hard to interpret the results: (1) Cleaning *inconsistency* in both training and test sets is unlikely to produce negative impacts on the ML model performance. Dataset choice and dirtiness in key features matter. (2) Dataset choices and how to inject *duplicates* when combining the datasets are crucial in the experiment setups of duplicates. (3) Class distribution has a huge impact on how *mislabels* should be cleaned.

Performing model selection can significantly increase the robustness of impacts of cleaning on ML. This effect has been identified in cleaning all error types. In particular, the negative effect of data cleaning can be eliminated by selecting the best ML model.

Performing cleaning algorithm selection further increases the robustness of impacts of cleaning on ML. This effect has been identified in cleaning all the error types. Since the data cleaning techniques are dependent of error distributions in datasets, no single cleaning algorithm is the best, and any future joint cleaning and ML work must devise "adaptive" cleaning solutions.

6. RELATED WORK

6.1 Work in DB Community

Despite the many fruitful research contributions in the general area of data cleaning, the dirty data problem remains challenging [22]. Recent study shows that even the combination of current error detection techniques can still miss many errors in real-world datasets [13]. We refer users to various surveys and tutorials on the broad topic of data cleaning [54, 39, 31, 40].

ML for data cleaning. Various ML techniques have been used in multiple data cleaning activities. Scared [66] and GDR [67] use

ML classification models to predict the likelihood of a candidate value update. Active learning is used to judiciously solicit human inputs to detect duplicate records [59, 17]. HoloClean [56] builds a graphical model to holistically reason about various signals to assign a probability to a candidate repair. Even deep learning techniques have been used for entity matching [29, 49].

Analytics-driven cleaning. As data cleaning itself is expensive and it is hard to reach ground truth, the DB community is starting to work on analytics-driven cleaning methods, which usually aim at reducing cleaning cost so that a given data analytical task can return better results. SampleClean [64] targets the problem of answering aggregate queries when the input data is dirty. SampleClean aims at answering a query only by cleaning a sample of the dirty dataset, and at the same time, providing confidence interval guarantees the query results. ActiveClean [44] is an example of cleaning data intelligently for convex models that are trained using gradient descent methods. The key insight of ActiveClean is that convex loss models (e.g., linear regression) can be trained and cleaned simultaneously. However, ActiveClean assumes that the cleaning is performed by an oracle. BoostClean [43] automatically selects an ensemble of error detection and repair combinations using statistical boosting. It does show promising results in terms of increased prediction in some datasets. However, BoostClean only considered *domain value violations* when an attribute value is outside of its value domain and only tested random forest model, while CleanML consider five error types and seven ML models. CleanML also features the use of (multiple) hypothesis testing to control randomness and to ensure that our findings are statistically significant.

Benchmarks. The DB community has the tradition of creating benchmarks, such as Wisconsin benchmark [28] and TPC[4]. The ML community has also contributed to benchmarking in various ways: MLBench, "the first benchmark for declarative ML on the cloud"; DAWNBench[5], a benchmark for end-to-end deep learning competition; MLPerf[6], a broad ML benchmark to measure ML system performance.

In this paper we create yet another benchmark to jointly study ML and data cleaning techniques. This work is inspired by benchmark efforts in the data cleaning and ML communities.

6.2 Work in ML Community

Impact of Noise on ML Models. Marlin [47] explores a variety of strategies for performing classification with missing feature values. García-Laencina et. al. [35] reviews a group of important missing data techniques in pattern classification. Acuña et. al. [15] study the effects of outliers on three classifiers (Linear discriminant analysis (LDA), KNN and decision trees). Kołcz et. al. [42] investigates the effects of class-biased duplicates using a spam-detection dataset. Artificial duplicates are only introduced into the spam class and two classifiers (Nave Bayes and Perceptron with Margins (PAM)) are tested. Frénay [32] outlines the effects of label noise including deterioration of classification performance and increased model complexity. Van Hulse et. al. [62] show that mislabeled minority class examples have a greater impact on classification performance than mislabeled majority class examples. Qi et. al. [51] conducts an experimental comparison of the effects of inconsistent data on classification, clustering, and regression algorithms. Two metrics (sensibility and keeping points) are proposed to measure effects. The results show that Bayesian Network is the most robust model, while random forest is the least robust. These studies are mostly isolated, and focuses on how particular kinds of noise with certain distributions affect certain ML models. In contrast, CleanML builds comprehensive a benchmark to study the effects of cleaning on ML, and carefully controls randomness to ensure the results are statistically significant, which is usually missing in previous studies.

Robust ML. Another line of work aims at developing ML algorithms that are robust to certain noises. Quinlan [53] proposes a learning algorithm for building noise-robust decision trees. Abellan and Moral [14] shows that using imprecision information gain can improve the accuracy of decision trees against the presence of label noise. Teng [61] demonstrates that over-fitting avoidance approaches, like regularization, can improve the robustness of ML models. Khoshgoftaar et al. [41] show that bagging can improve model performance due to the variability caused by dirty data.

7. CONCLUSIONS AND OPEN RESEARCH DIRECTIONS

This paper proposes the CleanML benchmark that studies the impact of data cleaning on ML, and we obtained many valuable insights as discussed in Section 1 and 5. We hope that this study will invite many future work on joint data cleaning and ML. We list some open research directions as follows.

1. *Extending the CleanML Benchmark*. While CleanML represents the most comprehensive study on the subject to date, there are many interesting future extensions. For example, we need more datasets with diverse error distributions, especially for missing values and outliers, as the current CleanML has no ground truth for these error types.
2. *Identifying the Biases Introduced in Data Cleaning.* As discussed, cleaning does not necessarily help and can sometimes even harm model performances. To decide whether a given cleaning method should be applied to an ML task, we need to detect whether the cleaning process introduces more biases than the dirtiness itself, which is an interesting research problem.
3. *Holistic Cleaning for ML.* CleanML currently considers each error type separately as a first step to understand how cleaning affects ML. Real-world datasets often exhibit multiple error types and they interact in non-trivial ways. Whether the benefits of cleaning are sub-linear, compositional, or super-linear remains an open question.
4. *Principled Approach for Joint Data Cleaning and ML.* We believe that there need to be principled approaches that jointly deal with dirty data and model training, so as to achieve the best performance. The cleaning process needs to be model-driven, and the training process needs to be error-aware.

[4]http://www.tpc.org/information/benchmarks.asp(last accessed: Feb 28, 2019).

[5]https://dawn.cs.stanford.edu/benchmark/(last accessed: Feb 28, 2019).

[6]https://www.mlperf.org/(last accessed: Feb 28, 2019).

8. REFERENCES

[1] Citation dataset. https://sites.google.com/site/anhaidgroup/useful-stuff/data. Accessed: April 29, 2019.

[2] Company dataset. https://www.kaggle.com/jacksapper/company-sentiment-by-location. Accessed: April 29, 2019.

[3] Credit dataset. https://www.kaggle.com/c/GiveMeSomeCredit/data. Accessed: April 29, 2019.

[4] EEG dataset. https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State. Accessed: April 29, 2019.

[5] IMDB movie dataset. https://data.world/popculture/imdb-5000-movie-dataset. Accessed: April 29, 2019.

[6] KDD dataset. https://www.kaggle.com/c/kdd-cup-2014-predicting-excitement-at-donors-choose/data. Accessed: April 29, 2019.

[7] Restaurant dataset. https://sites.google.com/site/anhaidgroup/useful-stuff/data. Accessed: April 29, 2019.

[8] Sensor dataset. http://db.csail.mit.edu/labdata/labdata.html. Accessed: April 29, 2019.

[9] Titanic dataset. https://www.kaggle.com/upendr/titanic-machine-learning-from-disaster/data. Accessed: April 29, 2019.

[10] TMDB movie dataset. https://www.kaggle.com/tmdb/tmdb-movie-metadata. Accessed: April 29, 2019.

[11] University dataset. https://archive.ics.uci.edu/ml/datasets/University. Accessed: April 29, 2019.

[12] USCensus dataset. https://archive.ics.uci.edu/ml/datasets/US+Census+Data+%281990%29. Accessed: April 29, 2019.

[13] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment*, 9(12):993–1004, 2016.

[14] J. Abellán and S. Moral. Building classification trees using the total uncertainty criterion. *International Journal of Intelligent Systems*, 18(12):1215–1225, 2003.

[15] E. Acuña and C. Rodríguez. An empirical study of the effect of outliers on the misclassification error rate. *Submitted to Transactions on Knowledge and Data Engineering*, 2005.

[16] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. QSGD: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

[17] A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 783–794. ACM, 2010.

[18] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

[19] C. Binnig, L. De Stefani, T. Kraska, E. Upfal, E. Zgraggen, and Z. Zhao. Toward sustainable insights, or why polygamy is bad for you. In *CIDR*, 2017.

[20] C. E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.

[21] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[22] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2201–2206. ACM, 2016.

[23] X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *Proceedings of the VLDB Endowment*, 6(13):1498–1509, 2013.

[24] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 458–469. IEEE, 2013.

[25] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1247–1261. ACM, 2015.

[26] Z. Cui, W. Chen, Y. He, and Y. Chen. Optimal action extraction for random forests and boosted trees. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 179–188. ACM, 2015.

[27] C. De Sa, M. Feldman, C. Ré, and K. Olukotun. Understanding and optimizing asynchronous low-precision stochastic gradient descent. In *ACM SIGARCH Computer Architecture News*, volume 45, pages 561–574. ACM, 2017.

[28] D. J. DeWitt. The wisconsin benchmark: Past, present, and future. the benchmark handbook, j. gray, ed, 1991.

[29] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

[30] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1):1–16, 2007.

[31] W. Fan and F. Geerts. Foundations of data quality management. *Synthesis Lectures on Data Management*, 4(5):1–217, 2012.

[32] B. Frénay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[33] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[34] S. García, J. Luengo, and F. Herrera. *Data preprocessing in data mining*. Springer, 2015.

[35] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2):263–282, 2010.

[36] T. Hastie, S. Rosset, J. Zhu, and H. Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[37] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification and regression. In *Advances in Neural*

Information Processing Systems, pages 409–415, 1996.

[38] Y. He, X. Chu, K. Ganjam, Y. Zheng, V. Narasayya, and S. Chaudhuri. Transform-data-by-example (tde): an extensible search engine for data transformations. *Proceedings of the VLDB Endowment*, 11(10):1165–1177, 2018.

[39] J. M. Hellerstein. Quantitative data cleaning for large databases. *United Nations Economic Commission for Europe (UNECE)*, 2008.

[40] I. F. Ilyas, X. Chu, et al. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends® in Databases*, 5(4):281–393, 2015.

[41] T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. Comparing boosting and bagging techniques with noisy and imbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(3):552–568, 2011.

[42] A. Kołcz, A. Chowdhury, and J. Alspector. Data duplication: An imbalance problem? 2003.

[43] S. Krishnan, M. J. Franklin, K. Goldberg, and E. Wu. Boostclean: Automated error detection and repair for machine learning. *arXiv preprint arXiv:1711.01299*, 2017.

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

[45] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

[46] T. Lin, S. U. Stich, and M. Jaggi. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217*, 2018.

[47] B. Marlin. *Missing data problems in machine learning*. PhD thesis, 2008.

[48] D. McFadden et al. Conditional logit analysis of qualitative choice behavior. 1973.

[49] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34. ACM, 2018.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[51] Z. Qi, H. Wang, J. Li, and H. Gao. Impacts of dirty data: and experimental evaluation. *arXiv preprint arXiv:1803.06071*, 2018.

[52] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[53] J. R. Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[54] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[55] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

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

[57] I. Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

[58] G. Rupert Jr et al. *Simultaneous statistical inference*. Springer Science & Business Media, 2012.

[59] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–278. ACM, 2002.

[60] S. K. Sarkar, H. Midi, and S. Rana. Detection of outliers and influential observations in binary logistic regression: An empirical study. *Journal of Applied Sciences*, 11(1):26–35, 2011.

[61] C. M. Teng. Evaluating noise correction. In *Pacific Rim International Conference on Artificial Intelligence*, pages 188–198. Springer, 2000.

[62] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano. Skewed class distributions and mislabeled examples. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 477–482. IEEE, 2007.

[63] R. Verborgh and M. De Wilde. *Using OpenRefine*. Packt Publishing Ltd, 2013.

[64] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 469–480. ACM, 2014.

[65] R. L. Wasserstein, N. A. Lazar, et al. The asas statement on p-values: context, process, and purpose. *The American Statistician*, 70(2):129–133, 2016.

[66] M. Yakout, L. Berti-Équille, and A. K. Elmagarmid. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 553–564. ACM, 2013.

[67] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *Proceedings of the VLDB Endowment*, 4(5):279–289, 2011.

[68] H. Zhang, J. Li, K. Kara, D. Alistarh, J. Liu, and C. Zhang. Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4035–4043. JMLR. org, 2017.

[69] W. Zhang, S. Gupta, X. Lian, and J. Liu. Staleness-aware async-sgd for distributed deep learning. *arXiv preprint arXiv:1511.05950*, 2015.

[70] Z. Zhao, L. De Stefani, E. Zgraggen, C. Binnig, E. Upfal, and T. Kraska. Controlling false discoveries during interactive data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 527–540. ACM, 2017.