Conflict Resolution for Improving ML Accuracy

Wenfei Fan[1,2], Xiaoyu Han[3], Hufsa Khan[1], Weilong Ren[1], Yaoshu Wang[1], Min Xie[1], Zihuan Xu[1]
[1]Shenzhen Institute of Computing Sciences [2]University of Edinburgh [3]Fudan University
{wenfei, hufsa, renweilong, yaoshuw, xiemin, xuzihuan}@sics.ac.cn, xyhan22@m.fudan.edu.cn

***Abstract*—This paper investigates how to make practical use of conflict resolution (CR) to enhance the accuracy of ML classifiers $\mathcal{M}$ on relational data. We show that applying CR to influential attributes and/or tuples can substantially improve the performance of downstream model $\mathcal{M}$. Based on this, we formulate two problems for identifying influential attributes and tuples in the data to maximize model accuracy. Although we show that both problems are intractable, we develop effective algorithms to pinpoint these critical factors. To mitigate the impact of noise introduced by imprecise CR methods, we propose a creator-critic framework that iteratively applies CR and trains $\mathcal{M}$ with the corrected data. We prove that the creator-critic process guarantees convergence to a more accurate model. Using real-life datasets, we experimentally verify that our approach improves the relative accuracy of various ML classifiers by an average of 40.5%.**

I. Introduction

Given a dataset $\mathcal{D}$, *conflict resolution* (*CR*) [26] aims to detect and correct inconsistencies within a single tuple and across multiple tuples, *e.g.,* mismatching area code and city in an address, or an actor born after the movie's release. It returns an improved dataset $\mathcal{D}_c$ for subsequent applications. There has been a large body of work on CR, by using logic rules [26]–[38], ML [14], [39]–[42], and hybrid of the two [43], [44].

There has been recent interest in *data cleaning for machine learning (ML)* [17], [45]–[59]. The process takes an ML model $\mathcal{M}$ and a dataset $\mathcal{D}$ as input, where $\mathcal{D}$ is used to train and evaluate $\mathcal{M}$. The goal is to derive a "cleaned" dataset $\mathcal{D}_{\mathcal{M}}$ from $\mathcal{D}$ such that the accuracy of the downstream model $\mathcal{M}$, when trained with $\mathcal{D}_{\mathcal{M}}$, is maximally improved compared to training with $\mathcal{D}$. The need for this is evident. It is reported that 80% of data scientists' time is spent on cleaning datasets for ML applications [60]–[62]. Using the Cleanlab data cleaning tool [63], a Berkeley lab was able to improve the accuracy of ML models by 15% and reduce the training time by 33%.

Unlike traditional cleaning, which produces a one-size-fits-all dataset $\mathcal{D}_c$, data cleaning for ML tailors a dataset $\mathcal{D}_{\mathcal{M}}$ specifically to improve the accuracy of a given model $\mathcal{M}$, by fixing inconsistencies in the training data. However, computing $\mathcal{D}_{\mathcal{M}}$ is a delicate process, as it depends on factors such as the type of model $\mathcal{M}$, the distribution of data in $\mathcal{D}$, and the effectiveness of the CR methods used. It is claimed in [17] that CR "is more likely to have an insignificant impact and unlikely to have a negative impact on ML classification models".

We replicated the experiments in [17] and found that (1) CR only slightly improves $\mathcal{M}$'s accuracy when being applied to the entire training data since it may introduce noise while fixing inconsistencies; however, (2) CR can significantly enhance $\mathcal{M}$'s accuracy when applied to a carefully selected subset of noisy data that can be confidently fixed by CR and is predicted to benefit $\mathcal{M}$ after the fixes are made (see Section VI).

tid	category (A_1)	gender (A_2)	education (A_3)	workhour (A_4)	occupation (A_5)	income (Y)
t_1	Never-worked	M	HS-grad	0	Exec-managerial	$< 50K$
t_2	Private	M	Bachelor	30	Transport-moving	$\leq 50K$
t_3	Local-gov	F	PhD	70	Prof.	$> 50K$
t_4	Local-gov	F	HS-grad	40	Prof.	$> 50K$
t_5	State-gov	F	Bachelor	40	Adm-clerical	$\leq 50K$
t_6	State-gov	M	Bachelor	5	Tech-support	$> 50K$
...	...	...	...	...	...	...

TABLE I: An Adult Table

Example 1: Consider Table I of real-life schema Adult [19] with category, gender, education, workhour, occupation as the attributes and income as the label. Values in red are erroneous, *e.g.,* t_1[occupation] should be `None` instead of `Exec-managerial`, t_4[education] should be `PhD`, not `HS-grad`, and t_6[workhour] should be `40`, not `5`. Such erroneous values appear to conflict with other correct values in either the same tuple or across multiple tuples, *e.g.,* (a) category and workhour of t_6 are conflicting since `State-gov` employees usually work >30 hours (similarly for category and occupation of t_1); and (b) t_3 and t_4 are inconsistent on education, given their similar behavior in other attributes. These conflicts can mislead the model and deteriorate its accuracy, *e.g.,* it may learn that `Exec-managerial` comes with low salary (by t_1), which is usually not true. As will be seen in Section VI, fixing carefully selected data can improve the average (relative) accuracy of some classifiers $\mathcal{M}$ by 40.5%. In contrast, if we apply CR to the entire data, it improves only by 27.7%. □

This raises several questions. How can we effectively use CR to enhance ML model accuracy? Which conflicts should we fix to achieve this? When a CR method is *imprecise* (*i.e.,* its accuracy is not 100% and may introduce noise during cleaning), can we still utilize it while mitigating its negative impact?

Contributions & Organization. This paper addresses these questions, to make practical use of CR to improve ML model accuracy, focusing on differential classifiers for relational data.

(1) A framework (Section III). To utilize (imprecise) CR, we propose CR4ML, a creator-critic framework. Given a model $\mathcal{M}$ and a dataset $\mathcal{D}$, CR4ML iteratively cleans $\mathcal{D}$ and trains $\mathcal{M}$. In the k-th round, the critic identifies influential attributes and tuples (see below), resolves conflicts, and obtains a cleaner dataset $\mathcal{D}_k$. The creator then incrementally trains $\mathcal{M}$ with $\mathcal{D}_k$.

One may plug any (not-perfect) CR method in CR4ML, as CR4ML can mitigate the impact of noise introduced by CR.

(2) Influential attributes (Section IV-A). We formulate a problem that, given a dataset $\mathcal{D}$ of schema $\mathcal{R}$, a model $\mathcal{M}$ and a

validation set $\mathcal{C}$ (selected by the critic of CR4ML), identifies a set of influential attributes of $\mathcal{R}$ such that resolving their conflicts maximumly improves $\mathcal{M}$'s accuracy on $\mathcal{C}$. We show that its decision problem is intractable. Nonetheless, we develop a genetic algorithm that employs the boosting strategy [64] to select a sub-optimal solution, without frequently retraining $\mathcal{M}$.

(3) Influential tuples (Section IV-B). We formulate another problem that, given $\mathcal{M}$ and $\mathcal{C}$, pinpoints tuples of $\mathcal{D} \setminus \mathcal{C}$ on which CR can maximumly improve the accuracy of $\mathcal{M}$ on $\mathcal{C}$. Although the problem is also intractable, we identify a set $\mathcal{T}$ of influential tuples by extending influence functions [3], [19]. We resolve conflicts in $\mathcal{T}$, estimate the update to $\mathcal{M}$ and validate the model on $\mathcal{C}$, without actually retraining $\mathcal{M}$.

(4) Performance guarantee and adaptations (Section V). We show that CR4ML converges to a stable state with a cleaned set $\mathcal{D}_{\mathcal{M}}$, and provides an accuracy approximation bound for $\mathcal{M}$ trained on $\mathcal{D}_{\mathcal{M}}$. Moreover, we discuss its adaptations to regression, multi-table, and streaming settings.

(5) Experimental evaluation (Section VI). Using different datasets, classification models and CR methods, we find the following. On average, (a) CR4ML improves the (relative) accuracy of various models by 40.5%. (b) By resolving conflicts in both influential attributes and tuples, CR4ML is more accurate than cleaning the entire $\mathcal{D}$, with 12.8% higher (relative) accuracy. (c) It is fast given an efficient and accurate CR, *e.g.,* it takes 451.6s on $\mathcal{D}$ of 13K tuples with CR method Rock [15]. (d) The more accurate CR is, the better CR4ML improves the accuracy of $\mathcal{M}$; nonetheless, CR4ML also works with not-so-accurate CR and can mitigate the impact of its noises.

We discuss preliminaries in Section II, related work in Section VII and future work in Section VIII; proofs are in [6].

Novelty. The novelty of CR4ML lies in its practical approach to applying imprecise CR to enhance the ML accuracy when training data is dirty, including (a) a creator-critic framework that, given a dirty dataset $\mathcal{D}$ without clean validation data, a classifier $\mathcal{M}$ and a (possibly imprecise) CR method $\mathcal{A}_{\text{CR}}$, automatically identifies impactful data that can be confidently fixed by $\mathcal{A}_{\text{CR}}$, to improve $\mathcal{M}$'s accuracy, with theoretical guarantees, (b) a model-pair design in the creator where an assistant model accumulates cleaned data from $\mathcal{D}$ and the target model $\mathcal{M}$ is only trained on the progressively cleaned data, (c) a two-step fixing strategy in the critic that mitigates the impact of imprecise $\mathcal{A}_{\text{CR}}$ by applying CR only to partial data predicted to benefit $\mathcal{M}$, (d) the formulation of two problems for identifying influential attributes and tuples for $\mathcal{A}_{\text{CR}}$ to clean, together with the intractability proofs of both problems; and (e) effective methods for identifying such attributes and tuples, respectively.

II. Influential Attributes and Tuples

We start with the notations (summarized in Table V in [6]). Consider a database schema $\mathcal{R} = (R_1, \ldots, R_m)$, where each R_j is a relation schema $R(A_1, \ldots, A_n, L)$, each A_i is an attribute, and L is the classification label. A dataset $\mathcal{D}$ of $\mathcal{R}$ is $(D_1, \ldots, D_m)$, where D_j is a relation of R_j. Following [65], we assume *w.l.o.g.* that each tuple t in $\mathcal{D}$ represents an entity.

Accuracy. For an ML classification model $\mathcal{M}$ and a dataset $\mathcal{D}$, we use $\text{acc}(\mathcal{M}, \mathcal{D})$ to denote the accuracy of $\mathcal{M}$ that is trained (resp. evaluated) with a training set $\mathcal{D}_{\text{train}}$ (resp. testing set $\mathcal{D}_{\text{test}}$) of $\mathcal{D}$; here $\mathcal{D}_{\text{test}}$ will not be used in training $\mathcal{M}$ (thus no data leakage). Following [46], [55], [56], we assume that $\mathcal{D}_{\text{test}}$ is clean, while $\mathcal{D}_{\text{train}}$ may be erroneous; this is a common setting as $\mathcal{D}_{\text{test}}$ is relatively small and can be manually checked by humans while $\mathcal{D}_{\text{train}}$ may be too large for humans to handle.

We measure $\text{acc}(\mathcal{M}, \mathcal{D})$ by macro-averaged F_1 over all classes [66], *i.e.,* $\text{acc}(\mathcal{M}, \mathcal{D}) = \frac{1}{N} \sum_l 2 \times \frac{\text{rec}_l \times \text{pre}_l}{\text{rec}_l + \text{pre}_l}$, where N is the number of classes, l is a classification label, rec_l is the ratio of l-class tuples (*i.e.,* tuples with label l) correctly predicted by $\mathcal{M}$ to all l-class tuples, and pre_l is the ratio of l-class tuples correctly predicted by $\mathcal{M}$ to all tuples with predicted class l.

Influential data. Unlike previous studies (*e.g.,* [17], [50]) that apply CR to the entire $\mathcal{D}$, we clean $\mathcal{D}$ more selectively. As will be shown in Section VI, applying CR to a subset of impactful data in $\mathcal{D}$ improves $\mathcal{M}$'s accuracy more effectively than cleaning the full dataset. This is because: (a) CR can be imprecise and its confidence/accuracy may vary across different conflicts in $\mathcal{D}$, and (b) resolving certain conflicts impacts $\mathcal{M}$'s training differently. We argue that both $\mathcal{M}$ and CR should be considered when cleaning $\mathcal{D}$ to improve $\mathcal{M}$'s accuracy.

Given a dataset $\mathcal{D}$ of $\mathcal{R}$, a classifier $\mathcal{M}$ and a CR method $\mathcal{A}_{\text{CR}}$, we denote by $\mathcal{D}_{\text{CR}}$ the cleaned $\mathcal{D}$ via $\mathcal{A}_{\text{CR}}$. The difference between $\text{acc}(\mathcal{M}, \mathcal{D})$ and $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{CR}})$ is the accuracy improvement of $\mathcal{M}$. In particular, if $\mathcal{A}_{\text{CR}}$ only cleans a subset $\mathcal{S}$ of attributes in all tuples of $\mathcal{D}$ (resp. all attributes of a subset $\mathcal{T}$ of tuples in $\mathcal{D}$), we denote the cleaned set by $\mathcal{D}^{\mathcal{S}}_{\text{CR}}$ (resp. $\mathcal{D}^{\mathcal{T}}_{\text{CR}}$).

Below we define *influential attributes* and *influential tuples*.

Influential attributes. Given a dataset $\mathcal{D}$ of schema $\mathcal{R}$, we say that $\mathcal{S} \subseteq \mathcal{R}$ is a set of *influential attributes* for $\mathcal{M}$ *w.r.t.* $\mathcal{D}$ if for any set $\mathcal{S}' \subseteq \mathcal{R}$, $\text{acc}(\mathcal{M}, \mathcal{D}^{\mathcal{S}}_{\text{CR}}) \geq \text{acc}(\mathcal{M}, \mathcal{D}^{\mathcal{S}'}_{\text{CR}})$. Intuitively, it means that cleaning the $\mathcal{S}$-attribute values in $\mathcal{D}$ maximumly improves the accuracy of $\mathcal{M}$, compared with other attributes.

In Example 1, $\mathcal{S}$ consists of education (A_3) and occupation (A_5), since they have impactful erroneous values, *e.g.,* $t_1[A_5]$ may mislead $\mathcal{M}$ that `Exec-managerial` is low-paying; and $t_4[A_3]$ may deceive $\mathcal{M}$ that `HS-grad` has high income.

Influential tuples. A subset $\mathcal{T}$ of $\mathcal{D}$ is a set of *influential tuples* for $\mathcal{M}$ *w.r.t.* $\mathcal{D}$ if for any $\mathcal{T}' \subseteq \mathcal{D}$, $\text{acc}(\mathcal{M}, \mathcal{D}^{\mathcal{T}}_{\text{CR}}) \geq \text{acc}(\mathcal{M}, \mathcal{D}^{\mathcal{T}'}_{\text{CR}})$. Intuitively, cleaning tuples in $\mathcal{T}$ maximumly improves the accuracy of $\mathcal{M}$, compared with other tuples in $\mathcal{D}$.

In Example 1, erroneous values in $\mathcal{T} = \{t_1, t_6\}$ reduce the accuracy of $\mathcal{M}$, regardless of whether they are in influential attributes (*e.g.,* t_1) or non-influential ones (*e.g.,* t_6), *e.g.,* $\mathcal{M}$ may be misled that `Exec-managerial` are poorly paid by t_1, and inactive workers get high salary by t_6. In practice, tuples from under-represented classes are more likely to be influential, since disturbance on them mostly affects the data distribution.

III. A Creator-Critic Framework

This section proposes CR4ML, a creator-critic framework that integrates ML training and conflict resolution, to ensure that the ML model is more robust and accurate on unseen data.

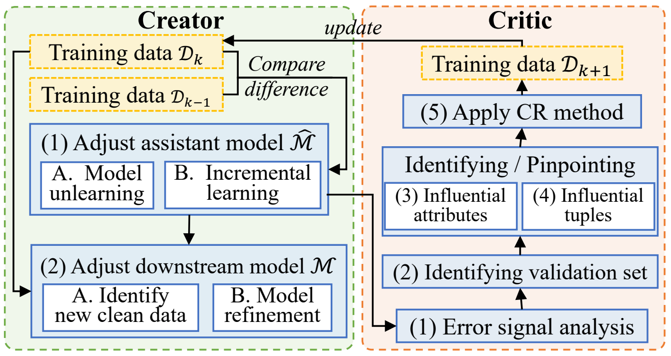

Creator
Critic
update
Training data $\mathcal{D}_k$
Training data $\mathcal{D}_{k-1}$
Compare difference
(1) Adjust assistant model $\hat{\mathcal{M}}$
A. Model unlearning
B. Incremental learning
(2) Adjust downstream model $\mathcal{M}$
A. Identify new clean data
B. Model refinement
Training data $\mathcal{D}_{k+1}$
(5) Apply CR method
Identifying / Pinpointing
(3) Influential attributes
(4) Influential tuples
(2) Identifying validation set
(1) Error signal analysis

Fig. 1: Overview of CR4ML

Overview. CR4ML takes as input a differentiable classifier $\mathcal{M}$, a (possibly imprecise) CR method $\mathcal{A}_{\mathsf{CR}}$, a schema $\mathcal{R}$, a dirty dataset $\mathcal{D}$ of $\mathcal{R}$, and a number e of epochs. CR4ML is developed to mitigate the impact of noise introduced by $\mathcal{A}_{\mathsf{CR}}$.

CR4ML iterates a creator and a critic in *rounds* (Figure 1), by integrating CR and ML training. In the k-th round, it works on a dirty set $\mathcal{D}_k$ and aims to produce cleaner $\mathcal{D}_{k+1}$ from $\mathcal{D}_k$ via $\mathcal{A}_{\mathsf{CR}}$, so that $\mathcal{D}_{k+1}$ provides more clean tuples to train $\mathcal{M}$. Initially, $\mathcal{D}_0 = \mathcal{D}$ and we maintain the following during the iterative process of CR4ML, by adopting a model-pair design:

- A set $\mathcal{D}_{\mathsf{clean}}$, accumulating clean tuples obtained so far.
- The target model $\mathcal{M}$. It is trained on $\mathcal{D}_{\mathsf{clean}}$ in each round.
- An assistant model $\hat{\mathcal{M}}$ initialized from $\mathcal{M}$. It is incrementally trained on (dirty) $\mathcal{D}_k$ to find (a) influential data in $\mathcal{D}_k$ to clean and (b) new clean data to add to $\mathcal{D}_{\mathsf{clean}}$.

This process continues until $\mathcal{D}_k$ cannot be further cleaned; $\mathcal{D}_k$ and the tuned $\mathcal{M}$ are returned (complete workflow in [6]).

In each round, the creator focuses on ML training and the critic targets conflict resolution. Below we present their details.

A. Creator for Model Training

The creator is responsible for (a) updating $\hat{\mathcal{M}}$ (that has been trained on data $\mathcal{D}_{k-1}$ in the (k-1)-th round) on $\mathcal{D}_k$ (*i.e.,* the dataset cleaned from $\mathcal{D}_{k-1}$) such that $\hat{\mathcal{M}}$ can better distinguish between dirty and clean data, and (b) refining $\mathcal{M}$ with newly identified clean data so that it can generalize to unseen data.

- *Input*: Models $\hat{\mathcal{M}}$ and $\mathcal{M}$ trained in (k-1)-th round, $\mathcal{D}_k$ and $\mathcal{D}_{k\text{-}1}$ from the (k-1)-th and (k-2)-th rounds, respectively, $\mathcal{D}_{\mathsf{clean}}$ accumulated so far, and a number e of epochs.
- *Output*: An extended $\mathcal{D}_{\mathsf{clean}}$ with more clean data, and updated model $\hat{\mathcal{M}}$ (resp. $\mathcal{M}$) trained on $\mathcal{D}_k$ (resp. $\mathcal{D}_{\mathsf{clean}}$).

Below we outline the model training (details are in [6]).

(1) Updating assistant model $\hat{\mathcal{M}}$. Denote by $\Delta\mathcal{D}_k = \{t \mid t \in \mathcal{D}_{k-1}$ and $t \notin \mathcal{D}_k\}$ the identified dirty tuples that were cleaned in round $k-1$. The training for $\hat{\mathcal{M}}$ can be sub-divided into two parts: (A) *model unlearning* to eliminate the negative effect of tuples in $\Delta\mathcal{D}_k$, on which $\hat{\mathcal{M}}$ was previously trained; and (B) *incremental learning* to update $\hat{\mathcal{M}}$ with cleaned tuples in $\mathcal{D}_k$.

(1A) Model unlearning. We adapt the influence function [3] to debias negative effect of $\Delta\mathcal{D}_k$, *i.e.,* mimic the effect of training $\hat{\mathcal{M}}$ on $\mathcal{D}_{k-1} \setminus \Delta\mathcal{D}_k$, without actually conducting retraining.

More specifically, for each tuple $t \in \Delta\mathcal{D}_k$, we can upweight t by an infinitesimal amount ϵ and the effect of unlearning t from $\hat{\mathcal{M}}$ is equivalent to upweighting it by $\epsilon = -\frac{1}{|\mathcal{D}_{k-1}|}$. Then, the parameter θ_k of $\hat{\mathcal{M}}$ in the k-th round can be updated as

$$\theta_k = \theta_{k-1} - \frac{1}{|\mathcal{D}_{k-1}|} \sum_{t \in \Delta\mathcal{D}_k} \mathcal{I}_{\mathsf{up,params}}(t),$$

where $\mathcal{I}_{\mathsf{up,params}}(t)$ denotes the influence of upweighting t on the parameters of $\hat{\mathcal{M}}$ (see [6] for its formal definition).

(1B) Incremental learning. Prior incremental strategies [67], [68] aim to preserve the performance of $\hat{\mathcal{M}}$ on both old and new data, *i.e.,* $\mathcal{D}_{k-1}$ and $\mathcal{D}_k$. In contrast, since $\mathcal{D}_k$ is derived from cleaning $\mathcal{D}_{k-1}$, we adopt a simple strategy that continuously fine-tunes $\hat{\mathcal{M}}$ with updated $\mathcal{D}_k$ in e epochs, with $\hat{\mathcal{M}}$ initialized from model unlearning. It works well since the negative effects of dirty data in $\mathcal{D}_{k-1}$ have already been removed.

(2) Refining target model $\mathcal{M}$. We train $\mathcal{M}$ by (A) *identifying new clean data* and (B) *refining model* on accumulated $\mathcal{D}_{\mathsf{clean}}$.

(2A) Identifying clean data. We find new clean data $\Delta\mathcal{D}_{\mathsf{clean}}$ from $\mathcal{D}_k$ to be accumulated in $\mathcal{D}_{\mathsf{clean}}$, with the help of $\hat{\mathcal{M}}$. Initially, $\Delta\mathcal{D}_{\mathsf{clean}} = \emptyset$ and we compute the dynamic loss $L_k(t)$ for each t in $\mathcal{D}_k$ [69], using exponential moving average (EMA):

$$L_k(t) = \lambda \cdot l(\hat{\mathcal{M}}(x;\theta_k), y) + (1-\lambda) \cdot L_{k-1}(t),$$

where $t = (x, y) \in \mathcal{D}_k$ (*i.e.,* t has attributes x and label y), $\lambda \in [0,1]$ is a discounting factor, $l(\hat{\mathcal{M}}(x;\theta_k), y)$ is the loss of t in the k-th round, and $L_k(t_i)$ is the accumulated loss of t from 0 to k epochs. The larger $L_k(t)$, the more likely t is dirty.

For each tuple $t \in \mathcal{D}_k \setminus \mathcal{D}_{\mathsf{clean}}$, we monitor its dynamic loss $L_k(t)$ of $\hat{\mathcal{M}}$. Given two thresholds η_1 and η_2 (see below), we add a tuple t into $\Delta\mathcal{D}_{\mathsf{clean}}$ if one of the following is satisfied:

- $L_{k-1}(t) - L_k(t) \geq \eta_1$, indicating that $\mathcal{A}_{\mathsf{CR}}$ has effectively enhanced t's quality and thus t can be considered as clean.
- $L_k(t) \leq \eta_2$, indicating that regardless of whether t has been cleaned by $\mathcal{A}_{\mathsf{CR}}$, it is likely to be clean due to its low $L_k(t)$.

Threshold setting. Here η_1 captures tuples "greatly improved after CR", and η_2 represents those "already good before CR". Let Q_1^1 and Q_3^1 (resp. Q_1^2 and Q_3^2) be the first and third quartiles of the loss distribution of $L_{k-1}(t)-L_k(t)$ (resp. $L_k(t)$) in $\mathcal{D}_k$, respectively. Following the Tukey IQR rule [25], we set $\eta_1 = Q_3^1 + 1.5 \cdot \mathsf{IQR}^1$ and $\eta_2 = Q_1^2 - 1.5 \cdot \mathsf{IQR}^2$, where $\mathsf{IQR}^1 = Q_3^1 - Q_1^1$, $\mathsf{IQR}^2 = Q_3^2 - Q_1^2$, ensuring stability against outliers.

(2B) Model refinement. CR4ML adapts Stochastic Gradient Descent [70] to iteratively refine the target model $\mathcal{M}$ using both new clean data $\Delta\mathcal{D}_{\mathsf{clean}}$ and accumulated clean data $\mathcal{D}_{\mathsf{clean}}$ (details in [6]). Finally, we set $\mathcal{D}_{\mathsf{clean}} = \Delta\mathcal{D}_{\mathsf{clean}} \cup \mathcal{D}_{\mathsf{clean}}$.

B. Critic for Conflict Resolution

Given updated $\hat{\mathcal{M}}$ from the creator, the critic performs *conflict resolution.* It (1) analyzes the error signals and (2) retrieves a relatively clean validation set $\mathcal{C}_k$ from $\mathcal{D}_k$ by adapting the idea of coreset selection [10], [71]. Using $\mathcal{C}_k$, it identifies (3) influential attributes and (4) influential tuples, on which (5) the CR method $\mathcal{A}_{\mathsf{CR}}$ is applied. It returns a cleaner set $\mathcal{D}_{k+1}$ for incrementally training $\hat{\mathcal{M}}$ and $\mathcal{M}$ in the next round.

- *Input:* Updated $\hat{\mathcal{M}}$, $\mathcal{D}_k$ from the $(k-1)$-th round, and $\mathcal{A}_{\mathsf{CR}}$.
- *Output:* A cleaner set $\mathcal{D}_{k+1}$ for the next round.

Below we outline the above five steps of the critic.

(1) Error signal analysis. Intuitively, the error signal of a tuple $t \in \mathcal{D}_k$, denoted by $\mathsf{err}(t)$, reflects the potential inaccuracy possibly introduced by training $\hat{\mathcal{M}}$ with t. To compute $\mathsf{err}(t)$, we transform dynamic loss $L_k(t)$ to a softmax probability, *i.e.,*

$$\mathsf{err}(t) = \mathcal{P}(t) = \frac{\exp[-\pi L_k(t)]}{\sum_{t' \in \mathcal{D}_k} \exp[-\pi L_k(t')]},$$

where π is a parameter controlling the probability distribution.

(2) Identifying validation set. Based on the error signals, the critic identifies a validation set $\mathcal{C}_k \subseteq \mathcal{D}_k$. Following [69], we add a tuple $t \in \mathcal{D}_k$ into $\mathcal{C}_k$ with probability $\mathcal{P}(t) = \mathsf{err}(t)$, such that $\mathcal{C}_k$ retains the underlying distribution and the diversity of $\mathcal{D}_k$. Note that if a dirty tuple is added into $\mathcal{C}_k$, it has a high probability to be excluded from $\mathcal{C}_{k+1}$ in the next round, and therefore, it can be eventually detected and cleaned.

(3) Identifying influential attributes. Given $\mathcal{C}_k$, we find a set of influential attributes in $\mathcal{R}$ so that if we resolve conflicts in those attributes using $\mathcal{A}_{\mathsf{CR}}$ and fine-tune $\hat{\mathcal{M}}$ on $\mathcal{D}_k \setminus \mathcal{C}_k$, the accuracy of $\hat{\mathcal{M}}$ is maximumly improved on $\mathcal{C}_k$. However, this problem is NP-hard and frequently fine-tuning $\hat{\mathcal{M}}$ is costly. We will develop an effective solution in Section IV-A.

(4) Pinpointing influential tuples. Detecting influential tuples is also NP-hard, which is costly since we need to frequently update the model $\hat{\mathcal{M}}$ (due to data modification) to decide what tuples are truly influential. We will tackle this in Section IV-B.

(5) CR. We adopt $\mathcal{A}_{\mathsf{CR}}$ to resolve the conflicts in the influential attributes of all tuples and all the attributes of influential tuples, yielding a cleaner set $\mathcal{D}_{k+1}$ for model training in next round.

Example 2: Consider $\mathcal{D} = \{t_1\text{-}t_6\}$ in Table I. In the first round, the creator trains $\hat{\mathcal{M}}$ using $\mathcal{D}_0 = \mathcal{D}$ and outputs $\hat{\mathcal{M}}$ to the critic. The creator (1) computes $\mathsf{err}(t)$ of each tuple t and (2) samples tuples with probability $\mathsf{err}(t)$. After $\mathcal{C}_0 = \{t_2, t_3, t_5\}$ is identified as the validation set, the critic (3) identifies influential attributes, *e.g.,* $\{A_3, A_5\}$, and (4) influential tuples, *e.g.,* $\mathcal{T} = \{t_1\}$, using $\mathcal{C}_0$. Finally, (5) the critic cleans influential data via $\mathcal{A}_{\mathsf{CR}}$ and gets cleaner $\mathcal{D}_1$ by updating $t_1[A_5] = \texttt{None}$ and $t_4[A_3] = \texttt{PhD}$. □

Remark. (1) CR4ML targets settings with *dirty data* and *imprecise CR*: (a) the creator adopts a model-pair design; it uses an assistant model $\hat{\mathcal{M}}$ to accumulate clean data, on which the target model $\mathcal{M}$ is trained, making $\mathcal{M}$ free from contamination by dirty data; and (b) the critic adopts a two-step fixing strategy to mitigate the impact of imprecise CR; it first pre-cleans candidate data so that influential attributes and tuples can be selected based on estimated gains (see Section IV), and then applies actual fixes on selected data, *i.e.,* cleaning predicted to be non-beneficial to $\mathcal{M}$ will not be actually applied. (2) Since a clean validation set is typically not offered in the dirty data setting, CR4ML retrieves a relatively clean set from $\mathcal{C}_k$ as the validation set, by adapting the idea of coreset selection [10] from images to relations. This differs from previous work (*e.g.,* [8], [9], [47]) that extracts coreset as the training data.

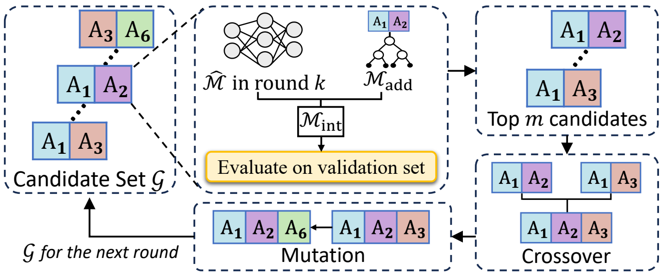

A3 A6
A1 A2
A1 A3
Candidate Set $\mathcal{G}$
$\hat{\mathcal{M}}$ in round k
A1 A2
$\mathcal{M}_{\text{add}}$
$\mathcal{M}_{\text{int}}$
Evaluate on validation set
A1 A2
A1 A3
Top m candidates
A1 A2
A1 A3
A1 A2 A3
Crossover
A1 A2 A6
A1 A2 A3
Mutation
$\mathcal{G}$ for the next round

Fig. 2: Process of AlgIA

IV. Algorithms for Identifying Influential Data

This section formulates the problems of identifying influential attributes (Sec. IV-A) and tuples (Sec. IV-B), proves their intractability (complete proofs in [6]), and develops solutions.

A. Identifying Influential Attributes

For each dataset $\mathcal{D}_k$, we split it into a training set $\mathcal{B}_k$ and a validation set $\mathcal{C}_k$, where $\mathcal{C}_k$ is identified via error signals, and $\mathcal{B}_k = \mathcal{D}_k \backslash \mathcal{C}_k$. Abusing notation, we use $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k)$ to denote the accuracy of $\hat{\mathcal{M}}$ trained (resp. evaluated) with $\mathcal{B}_k$ (resp. $\mathcal{C}_k$).

Given $\mathcal{D}_k = (\mathcal{B}_k, \mathcal{C}_k)$, a CR method $\mathcal{A}_{\mathsf{CR}}$, and an assistant model $\hat{\mathcal{M}}$, we identify a set $\mathcal{S}$ of influential attributes, such that if we resolve conflicts in the $\mathcal{S}$-attributes in $\mathcal{B}_k$, the accuracy of $\hat{\mathcal{M}}$ on $\mathcal{C}_k$ is maximumly improved. Denote the cleaned $\mathcal{B}_k$ by $\mathcal{B}_k^{\mathcal{S}}$. Then we obtain a cleaned version $\mathcal{D}_k^{\mathcal{S}} = (\mathcal{B}_k^{\mathcal{S}}, \mathcal{C}_k)$ of $\mathcal{D}_k$.

Problem. The *problem of finding influential attributes* (IA) is:

- *Input*: $\mathcal{R}, \mathcal{D}_k = (\mathcal{B}_k, \mathcal{C}_k), \hat{\mathcal{M}}$ and $\mathcal{A}_{\mathsf{CR}}$ as described above.
- *Output*: A set $\mathcal{S}$ of influential attributes from schema $\mathcal{R}$.
- *Objective*: Maximize $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{S}}) - \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k)$.

NP-hardness. It is hard to find an optimal set of influential attributes. The decision problem, also referred to as IA, is to decide, given $\mathcal{R}, \mathcal{D}_k, \hat{\mathcal{M}}, \mathcal{A}_{\mathsf{CR}}$ and a bound δ, whether there is a set $\mathcal{S}$ of attributes with $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{S}}) - \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) \geq \delta$. This is intractable, even when training/testing $\hat{\mathcal{M}}$ takes PTIME.

Theorem 1: *Problem* IA *is* NP*-hard.* □

Proof sketch: We prove the NP-hardness of IA by reduction from 3SAT, which is known to be NP-complete [72]. □

Naive approach. A naive approach is to greedily and iteratively add attributes A in $\mathcal{R}$ into $\mathcal{S}$, one attribute per round, so that cleaning the A-attribute of $\mathcal{B}_k$ via $\mathcal{A}_{\mathsf{CR}}$ can maximumly improve $\hat{\mathcal{M}}$'s accuracy on $\mathcal{C}_k$ in each round. Unfortunately, this method is costly, since $\hat{\mathcal{M}}$ needs to be retrained $O(|\mathcal{R}||\mathcal{S}|)$ times, where $|\mathcal{S}|$ (resp. $|\mathcal{R}|$) is the size of $\mathcal{S}$ (resp. $\mathcal{R}$).

Our approach. We develop Algorithm AlgIA, whose pseudo code is shown in Figure 3. Figure 2 visualizes its process. Unlike the naive approach, we do not retrain $\hat{\mathcal{M}}$ whenever an attribute is added, by adapting the boosting strategy [64].

More specifically, we maintain a set $\mathcal{G}$ of candidate attribute sets and record the set $\mathcal{S}_{\text{best}}$ (initially empty) with the highest

```
Input: A schema $\mathcal{R}$, a dataset $\mathcal{D}_k = (\mathcal{B}_k, \mathcal{C}_k)$ of schema $\mathcal{R}$,
       an ML model $\hat{\mathcal{M}}$, a CR method $\mathcal{A}_{\mathsf{CR}}$, the clean data $\mathcal{D}_{\mathsf{clean}}$,
       and thresholds $\mathsf{ite}_{\mathsf{max}}$ and $m$.
Output: A set $\mathcal{S}$ of influential attributes.
1.  $\mathcal{G} := \{\{A\} \mid A \in \mathcal{R}\}$; $\mathsf{idx} := 1$; $\mathcal{S}_{\mathsf{best}} := \emptyset$; $\mathsf{acc}_{\mathsf{best}} := 0$;
2.  $\mathcal{B}_k^{\mathcal{R}}$ := cleaning tuples in $\mathcal{B}_k \setminus \mathcal{D}_{\mathsf{clean}}$ via $\mathcal{A}_{\mathsf{CR}}$;
3.  while $\mathsf{idx} \leq \mathsf{ite}_{\mathsf{max}}$ do
4.      for each $\mathcal{S}_i$ in $\mathcal{G}$ do
5.          $\hat{\mathcal{M}}_{\mathsf{add}} := \mathsf{train}(\mathcal{B}_k^{\mathcal{R}}[\mathcal{S}_i])$;
6.          $\hat{\mathcal{M}}_{\mathsf{int}} := \mathsf{boosting}(\hat{\mathcal{M}}, \hat{\mathcal{M}}_{\mathsf{add}})$;
7.          if $\mathsf{acc}(\hat{\mathcal{M}}_{\mathsf{int}}, \mathcal{C}_k) > \mathsf{acc}_{\mathsf{best}}$ do
8.              $\mathcal{S}_{\mathsf{best}} := \mathcal{S}_i$; $\mathsf{acc}_{\mathsf{best}} := \mathsf{acc}(\hat{\mathcal{M}}_{\mathsf{int}}, \mathcal{C}_k)$;
9.      if $\mathcal{S}_{\mathsf{best}}$ is not updated do
10.         break
        /* candidate set maintenance */
11.     $\mathcal{G} := \mathsf{select}(\mathcal{G}, m)$;
12.     $\mathcal{G} := \mathsf{generate}(\mathcal{G}, m)$; $\mathsf{idx} := \mathsf{idx} + 1$;
13. return $\mathcal{S}_{\mathsf{best}}$;
```

Fig. 3: Algorithm AlgIA

accuracy $\mathsf{acc}_{\mathsf{best}}$ observed so far (line 1). Then we pre-clean the entire $\mathcal{B}_k$ as $\mathcal{B}_k^{\mathcal{R}}$ (line 2). For each attribute set $\mathcal{S}$ in $\mathcal{G}$, we train a lightweight tree-based model $\hat{\mathcal{M}}_{\mathsf{add}}$ (*e.g.,* GBDT [64]) with a projected set of $\mathcal{B}_k^{\mathcal{R}}$ with schema $\mathcal{S}$ (line 5), where $\hat{\mathcal{M}}_{\mathsf{add}}$ aims to fit the misalignment between the actual labels and the predicted labels by $\hat{\mathcal{M}}$. We then combine (two weaker) $\hat{\mathcal{M}}$ and $\hat{\mathcal{M}}_{\mathsf{add}}$ as an integrated/stronger model $\hat{\mathcal{M}}_{\mathsf{int}}$ by weighted averaging of predicted labels from $\hat{\mathcal{M}}$ and $\hat{\mathcal{M}}_{\mathsf{add}}$ (line 6), and validate $\hat{\mathcal{M}}_{\mathsf{int}}$ in $\mathcal{C}_k$ without retraining $\hat{\mathcal{M}}$ (line 7). The set $\mathcal{S}$ that gives the most accurate $\hat{\mathcal{M}}_{\mathsf{int}}$ (*i.e.,* recorded in $\mathcal{S}_{\mathsf{best}}$, lines 8-10) on $\mathcal{C}_k$ is returned as the solution (line 13).

However, since there are $|\mathcal{R}|!$ combinations for $\mathcal{S}$, it is still costly to train numerous $\hat{\mathcal{M}}_{\mathsf{add}}$. To tackle this, we give a genetic algorithm to maintain $\mathcal{G}$, by iterating a selection step and a generation step. Initially, $\mathcal{G}$ consists of all singleton attribute sets in $\mathcal{R}$, while in subsequent rounds, $\mathcal{G}$ is generated from "promising" sets selected earlier. This process stops when either it reaches the maximum round $\mathsf{ite}_{\mathsf{max}}$ or $\mathcal{S}_{\mathsf{best}}$ stabilizes.

(a) Set selection (line 11). Given an integer m for the number of sets that can be selected as seeds from the current $\mathcal{G}$ in each round, we adopt the boosting strategy [64] to evaluate each set in $\mathcal{G}$ and select m sets, whose corresponding $\hat{\mathcal{M}}_{\mathsf{int}}$ has the highest accuracy, as the seeds for set generation in next round.

(b) Set generation (line 12). In each round, we generate a new set $\mathcal{G}$ of at most $\frac{m\cdot(m-1)}{2}$ attribute sets, via two operators: (a) crossover that merges two sets (*e.g.,* merge $\{A_1, A_2\}$ and $\{A_1, A_3\}$ into $\{A_1, A_2, A_3\}$), and (b) mutation that randomly replaces an attribute (*e.g.,* replace A_3 in $\{A_1, A_2, A_3\}$ with A_6 and get $\{A_1, A_2, A_6\}$). Following [73], [74], each iteration starts with crossover and invokes mutation with a probability.

Example 3: Consider $\mathcal{D}_k = (\mathcal{B}_k, \mathcal{C}_k)$ in Table I, where $m = 8$. Initially, $\mathcal{G} = \{\{A_i\}\}_{i=1}^{5}$ and $\mathcal{S}_{\mathsf{best}} = \emptyset$. We first pre-clean the entire $\mathcal{B}_k$ via $\mathcal{A}_{\mathsf{CR}}$ to get $\mathcal{B}_k^{\mathcal{R}}$. In the first iteration, AlgIA trains a model $\hat{\mathcal{M}}_{\mathsf{add}}$ for each projected set of $\mathcal{B}_k^{\mathcal{R}}$ on $\{A_i\}$ ($1 \leq i \leq 5$). Assume that cleaning A_5 gives the best $\mathsf{acc}_{\mathsf{best}}$. We update $\mathcal{S}_{\mathsf{best}}$ as $\{A_5\}$. All sets in $\mathcal{G}$ are selected for set generation in next iteration, *e.g.,* $\{A_3\}$ and $\{A_5\}$ are merged into $\{A_3, A_5\}$. If cleaning $\{A_3, A_5\}$ gives an accuracy higher than $\mathsf{acc}_{\mathsf{best}}$, we update $\mathcal{S}_{\mathsf{best}} = \{A_3, A_5\}$. This process proceeds until either $\mathcal{S}_{\mathsf{best}}$ stabilizes or AlgIA reaches $\mathsf{ite}_{\mathsf{max}}$ rounds. □

```
Input: A schema $\mathcal{R}$, a dataset $\mathcal{D}_k = (\mathcal{B}_k, \mathcal{C}_k)$ of $\mathcal{R}$, an ML model
       $\hat{\mathcal{M}}$, a CR method $\mathcal{A}_{\mathsf{CR}}$, the cleaned $\mathcal{D}_{\mathsf{clean}}$ and a number $\mathcal{T}_{\mathsf{max}}$.
Output: A set $\mathcal{T}$ of at most $\mathcal{T}_{\mathsf{max}}$ influential tuples.
1.  $\mathcal{T} := \emptyset$; $\mathcal{B}_k$ := cleaning all tuples in $\mathcal{B}_k$ via $\mathcal{A}_{\mathsf{CR}}$;
2.  for $t \in \mathcal{B}_k \setminus \mathcal{D}_{\mathsf{clean}}$ do
3.      $t_{\mathsf{old}}$ := the version of $t$ with which $\mathcal{M}$ is trained;
4.      $t_{\mathsf{new}}$ := the new version of $t$ after pre-cleaning via $\mathcal{A}_{\mathsf{CR}}$;
5.      if $t_{\mathsf{old}} \neq t_{\mathsf{new}}$ do
6.          Estimate $\Delta\theta_t = \theta_{\mathsf{new}} - \theta_{\mathsf{old}}$ in a closed form;
7.          $\hat{\mathcal{M}}_t := \mathsf{update}(\hat{\mathcal{M}}, \Delta\theta_t)$;
            $\Delta\mathsf{acc}_t := \mathsf{acc}(\hat{\mathcal{M}}_t, \mathcal{C}_k) - \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{C}_k)$;
8.          if $\Delta\mathsf{acc}_t > 0$ do
9.              $\mathcal{T} := \mathsf{update}(\mathcal{T}, t, \mathcal{T}_{\mathsf{max}})$;
10. return $\mathcal{T}$;
```

Fig. 4: Algorithm AlgIT

Complexity. AlgIA takes $O(c_{\mathsf{CR}} + \mathsf{ite}_{\mathsf{max}} \cdot m^2 \cdot (c_{\mathsf{train}} + c_{\mathsf{val}} + \log(m)) + c_{\mathsf{cross}} + c_{\mathsf{muta}}))$ time, where $\mathsf{ite}_{\mathsf{max}}$ and m are given above, c_{CR} is the cost of cleaning $\mathcal{B}_k$ by $\mathcal{A}_{\mathsf{CR}}$, c_{train} is for training $\hat{\mathcal{M}}_{\mathsf{add}}$, c_{val} is for validating $\hat{\mathcal{M}}_{\mathsf{int}}$ on $\mathcal{C}_k$, $O(m^2 \cdot \log(m))$ is for selecting m sets from $\mathcal{G}$, and c_{cross} (resp. c_{muta}) is for a crossover (resp. mutation) operation in $\mathcal{G}$. AlgIA is fast (see Section VI); one may use a smaller $\mathsf{ite}_{\mathsf{max}}$ for better efficiency, without degrading much the accuracy.

B. Pinpointing Influential Tuples

Given $\mathcal{D}_k = (\mathcal{B}_k, \mathcal{C}_k)$, a CR method $\mathcal{A}_{\mathsf{CR}}$ and a model $\hat{\mathcal{M}}$, we aim to pinpoint the set $\mathcal{T}$ of *influential tuples* of $\mathcal{B}_k$, such that if we resolve conflicts in such tuples via $\mathcal{A}_{\mathsf{CR}}$, the accuracy of $\hat{\mathcal{M}}$ is maximumly improved on $\mathcal{C}_k$. Denote the cleaned $\mathcal{B}_k$ by $\mathcal{B}_k^{\mathcal{T}}$. We obtain a cleaned version $\mathcal{D}_k^{\mathcal{T}} = (\mathcal{B}_k^{\mathcal{T}}, \mathcal{C}_k)$ of $\mathcal{D}_k$.

Problem. The *problem of finding influential tuples* (IT) is:
- *Input*: $\mathcal{R}, \mathcal{D}_k = (\mathcal{B}_k, \mathcal{C}_k), \hat{\mathcal{M}}, \mathcal{A}_{\mathsf{CR}}, \mathcal{D}_{\mathsf{clean}}$ as above.
- *Output*: A set $\mathcal{T}$ of influential tuples of $\mathcal{B}_k$.
- *Objective*: Maximize $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{T}}) - \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k))$.

Here $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{T}})$ (resp. $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k)$) is the accuracy of $\hat{\mathcal{M}}$ that is trained with $\mathcal{B}_k^{\mathcal{T}}$ (resp. $\mathcal{B}_k$) and is evaluated with $\mathcal{C}_k$.

NP-hardness. The problem is hard. The decision problem, also referred to as IT, is to determine, given $\mathcal{R}, \mathcal{D}_k = (\mathcal{B}_k, \mathcal{C}_k), \hat{\mathcal{M}}, \mathcal{A}_{\mathsf{CR}}$ and a bound δ, whether there exists a set $\mathcal{T}$ of tuples such that $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{T}}) - \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) \geq \delta$.

Theorem 2: *Problem* IT *is* NP*-hard.* □

Proof sketch: We also prove this by reduction from 3SAT. □

Naive approach. One might want to adopt a greedy algorithm to solve IT as follows. (1) Clean a dirty tuple t in $\mathcal{B}_k$ via $\mathcal{A}_{\mathsf{CR}}$, and denote the cleaned $\mathcal{B}_k$ as $\mathcal{B}_k^t$. (2) Retrain a model $\hat{\mathcal{M}}$, denoted by $\hat{\mathcal{M}}_t$, on each $\mathcal{B}_k^t$. (3) Iteratively find the most influential tuple t' in $\mathcal{B}_k$ such that cleaning t' in $\mathcal{B}_k$ improves the accuracy of $\hat{\mathcal{M}}$ on $\mathcal{C}_k$ the most, add t' to $\mathcal{T}$, and update $\mathcal{D}_k$ and $\hat{\mathcal{M}}$ to be $\mathcal{D}_k^{t'}$ and $\hat{\mathcal{M}}_{t'}$, respectively, until $\hat{\mathcal{M}}$ cannot be further improved. (4) Set $\mathcal{T}$ is returned as the set of influential tuples.

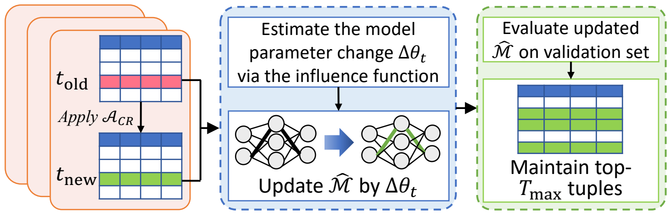

t_{old}
Apply $\mathcal{A}_{CR}$
t_{new}
Estimate the model parameter change $\Delta\theta_t$ via the influence function
Update $\hat{\mathcal{M}}$ by $\Delta\theta_t$
Evaluate updated $\hat{\mathcal{M}}$ on validation set
Maintain top-T_{max} tuples

Fig. 5: Process of AlgIT

However, this algorithm needs $O(c_{\mathsf{CR}}+|\mathcal{T}|\cdot(|\mathcal{D}_k|\cdot|\theta|\cdot c_{\mathsf{iter}}+c_{\Delta\mathsf{acc}_t}+\log(\mathcal{T})))$ time, where $|\theta|$ and c_{iter} represent the number of $\hat{\mathcal{M}}$'s parameters and training iterations, respectively, c_{CR} is the cost of cleaning $\mathcal{B}_k$ by $\mathcal{A}_{\mathsf{CR}}$, $c_{\Delta\mathsf{acc}_t}$ is for computing the accuracy improvement of $\hat{\mathcal{M}}$ on $\mathcal{C}_k$ by cleaning a tuple $t \in \mathcal{B}_k$, and $O(\log(\mathcal{T}))$ is the cost for maintaining the tuples in $\mathcal{T}$.

Our approach. We develop Algorithm AlgIT for IT, which returns a set $\mathcal{T}$ of at most T_{max} influential tuples. Its pseudo code (resp. visualization) is given in Figure 4 (resp. Figure 5).

The core is to estimate the effect of cleaning tuples in $\mathcal{B}_k$ without retraining $\hat{\mathcal{M}}$. To this end, we adapt the influence function [3], [19] to estimate the change to the parameters of $\hat{\mathcal{M}}$ when a tuple t in $\mathcal{B}_k$ is cleaned. Intuitively, this can be seen as removing an old version t_{old} of t from $\mathcal{B}_k$ and adding a new version t_{new} to $\mathcal{B}_k$. Denote by θ_{old} and θ_{new} the parameters of $\hat{\mathcal{M}}$ trained on the set with t_{old} and t_{new}, respectively. As shown in [3], the change of $\hat{\mathcal{M}}$'s parameters, *i.e.,* $\Delta\theta_t = \theta_{\mathsf{new}} - \theta_{\mathsf{old}}$, by cleaning t can be estimated in a closed form (details in [6]).

When a tuple t in $\mathcal{B}_k$ is cleaned via $\mathcal{A}_{\mathsf{CR}}$, we approximate its impact on $\hat{\mathcal{M}}$ by adjusting the parameters of $\hat{\mathcal{M}}$ as $\theta_{\mathsf{new}}(=\theta_{\mathsf{old}}+\Delta\theta_t)$ and then validate $\hat{\mathcal{M}}$ on $\mathcal{C}_k$, without retraining $\hat{\mathcal{M}}$.

More specifically, AlgIT initializes $\mathcal{T}$ as an empty set, and cleans all tuples in $\mathcal{B}_k \setminus \mathcal{D}_{\mathsf{clean}}$ via $\mathcal{A}_{\mathsf{CR}}$ (line 1). For each tuple t in $\mathcal{B}_k$, it checks whether t is cleaned by $\mathcal{A}_{\mathsf{CR}}$ (lines 3-5). If so (line 5), it uses the influence function to estimate the impact of cleaning $t \in \mathcal{B}_k$ on $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{C}_k)$. To do this, AlgIT produces a copy $\hat{\mathcal{M}}_t$ of $\hat{\mathcal{M}}$ by updating the parameters of $\hat{\mathcal{M}}$ with the change $\Delta\theta_t$, estimated with the influence function (line 6). Then it computes $\Delta\mathsf{acc}_t = \mathsf{acc}(\hat{\mathcal{M}}_t, C_k) - \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{C}_k)$ (line 7), the accuracy improvement of cleaning t on $\mathcal{C}_k$, and adds t to $\mathcal{T}$ if $\Delta\mathsf{acc}_t > 0$ and $\Delta\mathsf{acc}_t$ is among the top-T_{max} highest for tuples in $\mathcal{T}$ (lines 8-9). Finally, $\mathcal{T}$ is returned (line 10).

Note that there is a trade-off between the efficiency and accuracy with T_{max} (see Sec. VI). Given smaller T_{max}, $\hat{\mathcal{M}}$ is often more accurate. This is because influence function estimates impact to $\hat{\mathcal{M}}$ when a single tuple in $\mathcal{D}_k$ is modified [3], but not the joint influence when multiple tuples in $\mathcal{D}_k$ are changed. Thus a smaller T_{max} indicates more fine-tuning rounds of CR4ML, yielding more accurate joint influence estimates.

Example 4: Assume that $\mathcal{D}_{\mathsf{clean}} = \emptyset$, $T_{\mathsf{max}} = 1$ and $\hat{\mathcal{M}}$ is trained with $\mathcal{B}_k = \{t_1\text{-}t_6\}$. If t_1 and t_4 are cleaned, whose $\Delta\mathsf{acc}$ are 0.03 and 0.01, respectively, we return $\mathcal{T} = \{t_1\}$. □

Complexity. AlgIT takes $O(c_{\mathsf{CR}}+|\mathcal{T}|\cdot(c_{\Delta\theta_t}+c_{\hat{\mathcal{M}}_t}+c_{\Delta\mathsf{acc}_t}+\log(T_{\mathsf{max}})))$ time, where c_{CR} and $c_{\Delta\mathsf{acc}_t}$ are described above, $c_{\Delta\theta_t}$ is for assessing $\hat{\mathcal{M}}$'s parameter changes by influence function (*e.g.,* $O(r\cdot|\theta|)$ [4]), $c_{\hat{\mathcal{M}}_t}$ is for updating $\hat{\mathcal{M}}$ to $\hat{\mathcal{M}}_t$, and $O(\log(T_{\mathsf{max}}))$ is for maintaining the T_{max} tuples in $\mathcal{T}$.

Remark. we remark the following about algorithm AlgIT.

(1) *Two-step fixing.* It selects influential tuples by data pre-cleaning for evaluating performance gains. It adopts a two-step fixing strategy, *i.e.,* the fixes obtained from pre-cleaning are not applied to the entire dataset but only to selected tuples/attributes (Section III); similarly for Algorithm AlgIA.

(2) *Scalability.* Since algorithm AlgIT adapts influence functions, a scalability bottleneck in handling large datasets arises from computing the inverse of a *Hessian Matrix* [3] for estimating $\Delta\theta_t$, whose complexity is $O(|\mathcal{D}_k|\cdot|\theta|^2+|\theta|^3)$. We can reduce this to $O(|\mathcal{D}_k|\cdot|\theta|)$ by estimating Hessian-Vector Products (HVP) via iterative solvers [3], [4], or even to $O(r\cdot|\theta|)$ via recent works [4], where r is a small constant. Besides, we can restrict the computation of $\Delta\theta_t$ to only a few layers responsible for the predictions, further reducing $|\theta|$ and thus the total cost. In practice, we adopt micro-batch and serial HVP with activation checkpointing to cap GPU memory [5].

(3) *Non-convex model handling.* When $\mathcal{M}$ is non-convex, we follow [3] and add a small damping term to the Hessian diagonal to enable influence functions. We also show in Section V that CR4ML converges and reduces dynamic loss (see [6] for the complete proofs considering non-convex setting).

(4) *Small-update regime.* Rather than using influence function (IF) to rank tuples as in prior studies, CR4ML applies it to estimate model parameter changes, based on which CR4ML conducts model unlearning (Section III) and identifies influential tuples. To ensure the validity of IF, we enforce a small-update regime by admitting at most T_{max} tuples per round.

V. Performance Guarantees and Adaptations

We show that CR4ML converges with approximation bound.

Termination. CR4ML converges at a stable state if $\mathcal{A}_{\mathsf{CR}}$ is α-accurate ($\alpha \in (0.5, 1]$), *i.e.,* it correctly resolves conflicts with probability α; the cleaned data does not raise an error signal.

Theorem 3: *Given an α-accurate $\mathcal{A}_{\mathsf{CR}}$,* CR4ML *guarantees to return (a) a cleaned set $\mathcal{D}_{\mathsf{clean}}$ as $\mathcal{D}_{\mathcal{M}}$, and (b) a model $\mathcal{M}$ trained with $\mathcal{D}_{\mathsf{clean}}$ such that* $\mathsf{acc}(\mathcal{M}, \mathcal{D}_{\mathsf{clean}}) > \mathsf{acc}(\mathcal{M}, \mathcal{D})$. □

Proof sketch: Observe the following: (1) $\mathcal{A}_{\mathsf{CR}}$ is α-accurate ($\alpha > 50\%$), meaning that it cleans data and approaches error-free CR that resolves all conflicts through iterations; (2) the convergence of SGD is ensured by unbiased gradient estimates and Robbins-Monro step-size conditions; and (3) training models with clean data, rather than dirty data, approximates the true data distribution and improves test accuracy [45], [75], [76]. Taken together, these guarantee that by iteratively applying $\mathcal{A}_{\mathsf{CR}}$ and optimizing model parameters, CR4ML terminates with $\mathcal{D}_{\mathcal{M}}$, which improves the accuracy of $\mathcal{M}$. □

Remark about Theorem 3. It shows that CR4ML works with any $\mathcal{A}_{\mathsf{CR}}$ as long as $\mathcal{A}_{\mathsf{CR}}$ is α-accurate and $\alpha > 0.5$, which is common in practice (see Section VI). When multiple CR methods are available, one can assess the accuracy of $\mathcal{A}_{\mathsf{CR}}$ (*e.g.,* the value of α) on a small set of manually labeled data and select the one with the best accuracy. Users can configure each CR method by adapting their default settings, *e.g.,* lightweight human verification of discovered rules in Rock [15], and manual repair of selected tuples in Raha [40] and Baran [14].

Approximation of expected accuracy. While the problems of identifying the optimal influential attributes and tuples are NP-hard, CR4ML ensures an approximation bound on the expected accuracy of the model trained on the cleaned data $\mathcal{D}_{\text{clean}}$.

Theorem 4: *Given an α-accurate $\mathcal{A}_{\text{CR}}$ ($\alpha > 0.5$),* CR4ML *achieves an expected submodular improvement in model accuracy. The expected accuracy $\mathbb{E}(\text{acc}(\mathcal{M}, \mathcal{D}_{\text{clean}}))$ of the model $\mathcal{M}$ trained on the cleaned data $\mathcal{D}_{\text{clean}}$ is guaranteed to reach at least 63% of the optimal expected accuracy.* □

Proof sketch: (1) Since $\mathcal{A}_{\text{CR}}$ is α-accurate ($\alpha > 50\%$), each iteration resolves conflicts and progressively yields a cleaner dataset, such that the model's accuracy improves as more data is cleaned. (2) The accuracy gains remain stable across iterations, guaranteeing consistent improvement, as supported by the convergence of the SGD training process [77], [78]. (3) The attribute and tuple selection methods in CR4ML align with the per-round greedy objective that provides provable guarantees. (4) The gain functions are monotonic (*i.e.,* more cleaning always improves accuracy) and submodular (*i.e.,* marginal gains diminish as more data is cleaned). These enable the use of submodular optimization [79], ensuring that a greedy algorithm achieves at least 63% of the optimal accuracy. □

Remark about Theorem 4. (1) Theorem 4 holds for the overall framework, relying on its round-wise design that incrementally improves model accuracy through validation-guided cleaning. (2) The analysis critically depends on the marginal gains using the auxiliary models on a relatively clean validation set, which enables accurate and submodularity-aware selection.

Complexity. CR4ML takes $O(|\mathcal{R}| \cdot |\mathcal{D}_k| \cdot (|\mathcal{D}_k| \cdot c_{\text{ul}} + 2e \cdot c_{\text{ft}} + 5|\mathcal{D}_k| + c_{\text{IA}} + c_{\text{IT}} + c_{\text{CR}}) + e \cdot c_{\text{ft}})$ time, where c_{ul} is the cost of eliminating the effect of a dirty tuple in $\mathcal{D}_{k-1}$ from $\hat{\mathcal{M}}$, c_{ft} is the cost of fine-tuning $\hat{\mathcal{M}}$ (resp. $\mathcal{M}$) with $\mathcal{D}_k$ (resp. $\mathcal{D}_{\text{clean}} \cup \Delta\mathcal{D}_{\text{clean}}$) in one epoch, e is the number of epochs, c_{IA} is the cost of finding a set of influential attributes in $\mathcal{R}$, c_{IT} is the cost of getting a set of influential tuples in $\mathcal{D}_k$, and c_{CR} is the cost of cleaning $\mathcal{D}_k$ by $\mathcal{A}_{\text{CR}}$. Here c_{CR} is often the most costly factor. We will see in Section VI that CR4ML is fast when the embedded $\mathcal{A}_{\text{CR}}$ and $\mathcal{M}$ are efficient.

The worst-case complexity above assumes that CR4ML takes $|\mathcal{R}| \cdot |\mathcal{D}_k|$ rounds, cleaning one value per round. In practice, CR4ML needs much less rounds, *e.g.,* 7 rounds on average (Section VI), since in each round it cleans all dirty values for newly identified influential data, not just one.

Adaptations. Although CR4ML targets differential classifiers on relational data, it can be easily adapted to the following.

Regression tasks. CR4ML can be adapted to handle regression with the following changes: (a) replace the loss of ML models (resp. evaluation metric) with mean squared error (MSE) (resp. root MSE (RMSE)); and (b) apply min-max scaling to normalize per-tuple losses before computing the error signals. We leave further regression-specific optimizations to future work.

Datasets	$\|\mathcal{D}\|$	$\|\mathcal{R}\|$	#classes
Adult [19]	48,842	15	2
Nursery [80]	12,958	8	4
Bank [81]	49,732	16	2
Default [81]	30,000	24	2
German [81], [82]	1,000	20	2
RoadSafety [83]	363,243	67	3

TABLE II: The tested datasets

Method	CR4ML$_{10\%}$	CR4ML$_{40\%}$	CR4ML$_{70\%}$	CR4ML$_{100\%}$ (*i.e.,*CR4ML)
ACC$_{\text{re}}$	13%	38%	45%	54.8%

TABLE III: Ablation study

Multi-table conflicts. Conflicts can appear across tables. As most mainstream tabular ML libraries (*e.g.,* scikit-learn [11]) expect input as a single feature matrix, we follow standard practice [12]: merge multiple tables into one using entity resolution methods (*e.g.,* [13]) to resolve join-key conflicts, and apply CR4ML to the merged table to tackle multi-table conflicts.

Streaming settings. We can extend CR4ML with a two-trigger strategy in streaming settings. (1) *Periodic trigger.* When the amount of new data arrived exceeds a user-defined fraction (*e.g.,* 20%$|\mathcal{D}|$), we can warm-start models $\hat{\mathcal{M}}$ and $\mathcal{M}$ from the last checkpoint, reuse the accumulated $\mathcal{D}_{\text{clean}}$, and rerun the workflow of CR4ML in Figure 1. This yields incremental updates with amortized cost proportional to the increment. (2) *Drift trigger.* When validation accuracy falls below a threshold, we execute CR4ML on a recent sliding window, while optionally incorporating historical data if the drift is partial or cyclical, and discard outdated $\hat{\mathcal{M}}$ and $\mathcal{M}$.

VI. Experimental Study

This section experimentally evaluates the (1) effectiveness and (2) efficiency of CR4ML across different (a) ML classifiers, (b) CR methods, and (c) configurable parameters.

Experimental setting. We start with our settings.

Datasets. As shown in Table II, we used 6 real-life datasets for classifications (details in [6]), each randomly split into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ in 8 : 2 ratio for training and testing, respectively.

Following [19], we treat the released version of each dataset as "clean" $\mathcal{D}$, and inject noises (conflicts) in all attributes of $\mathcal{D}$ (influential or not). This is because: (a) there is no "good" public benchmark with ground truth of real inconsistencies, *e.g.,* [18] tested CleanML benchmarks [84] and found that they contain few real inconsistencies that have significant impact on the downstream models and (b) labeling real datasets is costly, *e.g.,* "it takes 3 people 6 months to finish the label process" for 16 benchmarks of categorical deduplication [58].

We consider three noise injectors NI: (1) Self-injector (SI) that randomly injects conflicts. (2) DeepFool [85], [86] that injects conflicts targeted for degrading given ML models. (3) Pattern injector (PI) that injects conflicts following systematic patterns (*e.g.,* violating functional dependencies [87]). Since the average error rates used in [14], [37], [40], [58] are around 10%, we set the noise ratio as 10% following the prior work.

ML models $\mathcal{M}$. In addition to (a) logistic regression (LR) [11]

$\mathcal{M}$	$\mathcal{A}_{\mathsf{CR}}$	**German**	**Adult**	**Nursery**	**Default**	**Bank**
LR	Base	0.481	0.653	0.727	0.523	0.487
	$\mathsf{Oracle}_{\mathsf{full}}$	0.691	0.784	0.901	0.673	0.684
MLP	Base	0.578	0.617	0.775	0.478	0.53
	$\mathsf{Oracle}_{\mathsf{full}}$	0.788	0.801	1	0.678	0.746
FTT	Base	0.434	0.47	0.259	0.46	0.471
	$\mathsf{Oracle}_{\mathsf{full}}$	0.51	0.493	0.26	0.484	0.496

TABLE IV: acc (Base) and acc($\mathsf{Oracle}_{\mathsf{full}}$)

on which [45], [48], [59], [88] primarily focus, we evaluated two differential ML classification models: (b) a shallow neural network model MLPClassifier [11] and (c) a representative table representation model FT-Transformer (FTT) [89]. We use one-hot encoding to encode relational data for its popularity.

CR methods $\mathcal{A}_{\mathsf{CR}}$. We tested five CR methods $\mathcal{A}_{\mathsf{CR}}$: (1) RB that uses Raha [40] to detect conflicts and Baran [14] to correct conflicts, where we set its labeling budget to 20 as in [14]; (2) Rock, a data cleaning system [15] for conflict detection and correction; (3) HoloClean [37], a data cleaning system, where we fed it with the same set of rules (transformed to denial constraints) as Rock, and we randomly selected a maximal subset of rules to maximize its memory usage; (4) RT uses Raha [40] and T5 [90] (a pretrained language model) to detect and resolve conflicts, respectively; and (5) Mode [46], [48] fills in null values with the most frequent domain values. As reported in [14], the F_1 scores of RB and HoloClean on seven widely-used datasets were 0.8 and 0.25, respectively. Besides, when evaluated on two billion-scale real-world dirty datasets in [15], the F_1 scores of Rock, RB and RT were 0.92, 0.54, and 0.32, respectively. Thus, Rock and RB are regarded as more accurate $\mathcal{A}_{\mathsf{CR}}$ than the others methods.

Baselines. We implemented CR4ML in python and used the following baselines: (1) Base, a "lower-bound" baseline that cleans nothing. (2) $\mathsf{Oracle}_{\mathsf{full}}$, an "upper-bound" baseline that resolves conflicts by experts with 100% accuracy. (3) $\mathsf{CR}_{\mathsf{full}}$, an approach that applies $\mathcal{A}_{\mathsf{CR}}$ to the entire training data. (4) $\mathsf{Rank}_{\mathsf{top}}$, an approach that applies $\mathcal{A}_{\mathsf{CR}}$ to the top-5 (resp. top-30%) attributes (resp. tuples) in the training data ranked via feature importance [11] (resp. influence function [3]). (5) Picket [49], which detects corrupted tuples via reconstruction loss and directly removes dirty tuples from the training data. (6) DiffPrep-fix (resp. DiffPrep-flex) [88], a data preprocessing pipeline that selects suitable cleaning operators for differential ML models with (resp. without) pre-defined transformation orders. (7) SAGA [59], a framework that automatically generates the top-K effective data cleaning pipelines; we set K as 1. (8) ActiveClean [45], an approach that treats data cleaning and model training in a form of SGD. (9) CPClean [46], a data imputation method for KNN models. (10) BoostClean [48], a data cleaning approach that enhances ML models by boosting strategies. (11) CL [2], a label cleaning approach for improving model training without human intervention. (12) Glister [8], a coreset selection method that mitigates noise in training data.

Here (a) BoostClean and CPClean fix missing values only; thus we only tested their efficiency; and (b) we configured DiffPrep-fix, DiffPrep-flex, SAGA, CPClean, BoostClean and CL with selected cleaning tools as in their released codes.

While Base, $\mathsf{Oracle}_{\mathsf{full}}$, $\mathsf{CR}_{\mathsf{full}}$ and Picket are independent of $\mathcal{M}$, the others are $\mathcal{M}$-aware, *i.e.,* they target a given $\mathcal{M}$.

Configurable parameters. By default, we set (1) for CR4ML, $e = 2$ for the number of epochs; (2) for AlgIA, $\mathsf{ite}_{\mathsf{max}} = 2$ as the maximum number of rounds and $m = |\mathcal{R}|$ as the bound on candidate sets; and (3) for AlgIT, $T_{\mathsf{max}} = 2\%|\mathcal{D}|$. We used LR as $\mathcal{M}$, SI as the noise injector NI and Rock as $\mathcal{A}_{\mathsf{CR}}$. For baselines, we set their parameters following their recommendations in [2], [8], [45], [48], [49], [59], [88].

Configuration. We ran experiments on a machine powered by 504GB RAM and 104 Intel(R) Xeon(R) Gold 5320 CPUs @2.20GHz. Each test was run 3 times; the average is reported.

Experimental findings. We report our findings (more in [6]).

Exp-1 Effectiveness. We evaluated the accuracy of CR4ML and baselines with various models and CR methods. Following [46], we use the *relative accuracy* of a baseline X, defined as

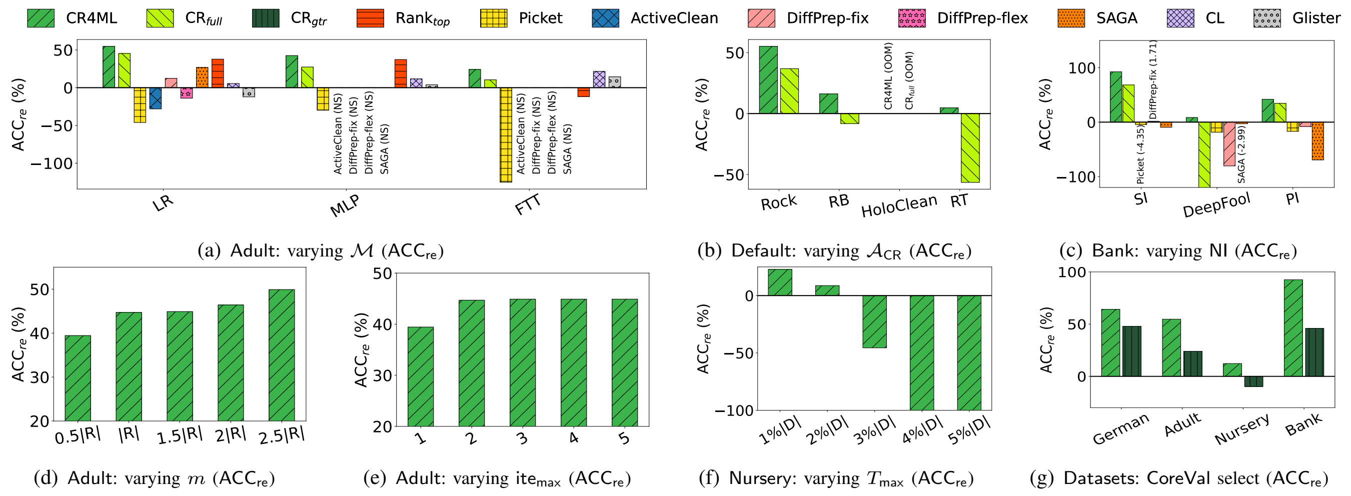
$$\mathsf{ACC}_{\mathsf{re}}(X) = \frac{\mathsf{acc}(X) - \mathsf{acc}(\mathsf{Base})}{\mathsf{acc}(\mathsf{Oracle}_{\mathsf{full}}) - \mathsf{acc}(\mathsf{Base})},$$

where $\mathsf{acc}(X) = \mathsf{acc}(\mathcal{M}, \mathcal{D}_{\mathsf{CR}})$ and $\mathcal{D}_{\mathsf{CR}}$ is the set cleaned by X, *e.g.,* applying a CR method on entire data (*e.g.,* $\mathsf{CR}_{\mathsf{full}}$) or only influential data (*e.g.,* CR4ML). Since $\mathsf{Oracle}_{\mathsf{full}}$ (resp. Base) is the "upper bound" (resp. "lower bound") of cleaning, (with accuracy in Table IV), $\mathsf{ACC}_{\mathsf{re}}(X) \in (-\infty, 1]$ quantifies the improvement in accuracy relative to the upper bound; here $\mathsf{acc}(\mathsf{Oracle}_{\mathsf{full}})$ for complex models (*e.g.,* FTT) may be low due to overfitting. Note that $\mathsf{ACC}_{\mathsf{re}}(X) < 0$ if the accuracy of cleaning via X is even worse than cleaning nothing, *i.e.,* Base.

Accuracy vs. baselines. As shown in Figure 6(a), CR4ML has the highest $\mathsf{ACC}_{\mathsf{re}}$. Besides, we find the following.

(1) The released code of all $\mathcal{M}$-aware baselines except CL and Glister can only support LR. CR4ML outperforms all these baselines for LR *e.g.,* its $\mathsf{ACC}_{\mathsf{re}}$ is 54.8%, as opposed to 12.3%, -13.9%, -28.1%, 26.8%, 5.5% and -12.2% of DiffPrep-fix, DiffPrep-flex, ActiveClean, SAGA, CL and Glister, respectively. This is because CR4ML (a) incrementally identifies dirty data by monitoring and analyzing fine-grained error signals, and (b) selectively resolves conflicts only in influential data that positively affect $\mathcal{M}$, *e.g.,* CR4ML fixed $1‰|\mathcal{D}|$ influential tuples on Adult, yielding a 2.63% improvement in $\mathsf{ACC}_{\mathsf{re}}$. In contrast, the $\mathcal{M}$-aware baselines attempt to clean the entire $\mathcal{D}$ without carefully considering how each correction affects the model performance. This justifies the need for cleaning influential data alone, instead of the entire $\mathcal{D}$.

(2) CR4ML beats $\mathsf{CR}_{\mathsf{full}}$, $\mathsf{Rank}_{\mathsf{top}}$, Picket, CL and Glister for all models; its $\mathsf{ACC}_{\mathsf{re}}$ is 40.5%, rather than 27.7%, 21.1%, -67%, 12.9% and 2% of the five baselines, respectively. These further show the benefits of cleaning influential data; moreover, guided by the influence function and boosting strategy, CR4ML selectively resolves conflicts that (a) it can confidently address, and (b) the cleaning of such conflicts will maximumly improve the accuracy of $\mathcal{M}$, *e.g.,* the $\mathsf{ACC}_{\mathsf{re}}$ of CR4ML and $\mathsf{CR}_{\mathsf{full}}$ is 40.5% and 27.7%, respectively. This is because CR4ML considers the effect of conflict



(a) Adult: varying $\mathcal{M}$ (ACC_{re}) (b) Default: varying $\mathcal{A}_{\text{CR}}$ (ACC_{re}) (c) Bank: varying NI (ACC_{re})

(d) Adult: varying m (ACC_{re}) (e) Adult: varying ite_{max} (ACC_{re}) (f) Nursery: varying T_{max} (ACC_{re}) (g) Datasets: CoreVal select (ACC_{re})

Fig. 6: Effectiveness evaluation ("NS" stands for "not support")

resolution and model performance in a fine-grained manner, and iteratively identifies influential data to fix, rather than relying on one-time bet as in Rank_{top}. It also mitigates the impact of noises introduced by imprecise CR methods, by considering their joint influence with model accuracy.

(3) The ACC_{re} of CR4ML for LR, MLP and FTT is 54.8%, 42.5% and 24.2%, by correcting 24.5%, 23.5% and 12.4% of the injected conflicts, respectively. Thus CR4ML is effective in both simple models like LR and complex models like FTT.

Case study on CleanML *benchmarks.* We evaluated CR4ML on CleanML benchmarks [84] (not shown), which contain real-world conflicts but lack ground truth (*i.e.,* $\text{Oracle}_{\text{full}}$). Thus, we report the accuracy of $\mathcal{M}$ instead of ACC_{re}. It improves the accuracy, *e.g.,* by 6.1% on Company, at least $2\times$ higher than the result reported in [18]. This shows that CR4ML is promising in improving $\mathcal{M}$'s accuracy by fixing real-world conflicts.

We next tested the impact of parameters on the accuracy.

Varying $\mathcal{A}_{\text{CR}}$. We tested the impact of $\mathcal{A}_{\text{CR}}$ on Default (except Holoclean, whose features cannot fit in memory). Besides, Picket, DiffPrep-fix, DiffPrep-flex, ActiveClean and SAGA are not compared since they do not rely on any particular CR.

As shown in Figure 6(b), (1) CR4ML performs better with more accurate $\mathcal{A}_{\text{CR}}$, *e.g.,* the ACC_{re} of CR4ML with Rock is 55.1%, as opposed to 4.7% with RT. This is because more accurate CR correctly resolves more influential conflicts *e.g.,* Rock (resp. RT) correctly resolves 39.4% (resp. 0.04%) of conflicts. (2) Given accurate $\mathcal{A}_{\text{CR}}$, CR4ML improves the ACC_{re} better than directly cleaning entire $\mathcal{D}$, *e.g.,* the ACC_{re} of CR4ML with Rock is 55.1% instead of 36.9% by cleaning the entire $\mathcal{D}$. This further verifies the effectiveness of selective conflict resolution. (3) CR4ML works with imprecise $\mathcal{A}_{\text{CR}}$ and mitigates its noises, *e.g.,* the ACC_{re} of CR4ML with RT is 4.7%, as opposed to -56.4% by cleaning entire $\mathcal{D}$ with RT.

Varying NI. We tested the impact of different noise injectors NI in Figure 6(c). The ACC_{re} of CR4ML is sensitive to NI, *e.g.,* it is 92.5%, 8.5% and 42% on Bank under SI, DeepFool and PI, respectively, since injectors like DeepFool are $\mathcal{M}$-aware and harm the accuracy of $\mathcal{M}$ by injecting noises with small training losses, such that the injected noises are hard to detect. Nevertheless, CR4ML consistently beats the baselines, *e.g.,* its ACC_{re} is 47.7% on average, as opposed to -13.3% of Picket, the highest one among baselines under various noise injectors. This is because CR4ML considers the joint influence of cleaned data and carefully identifies the influential data, instead of only focusing on the training loss of individual tuples.

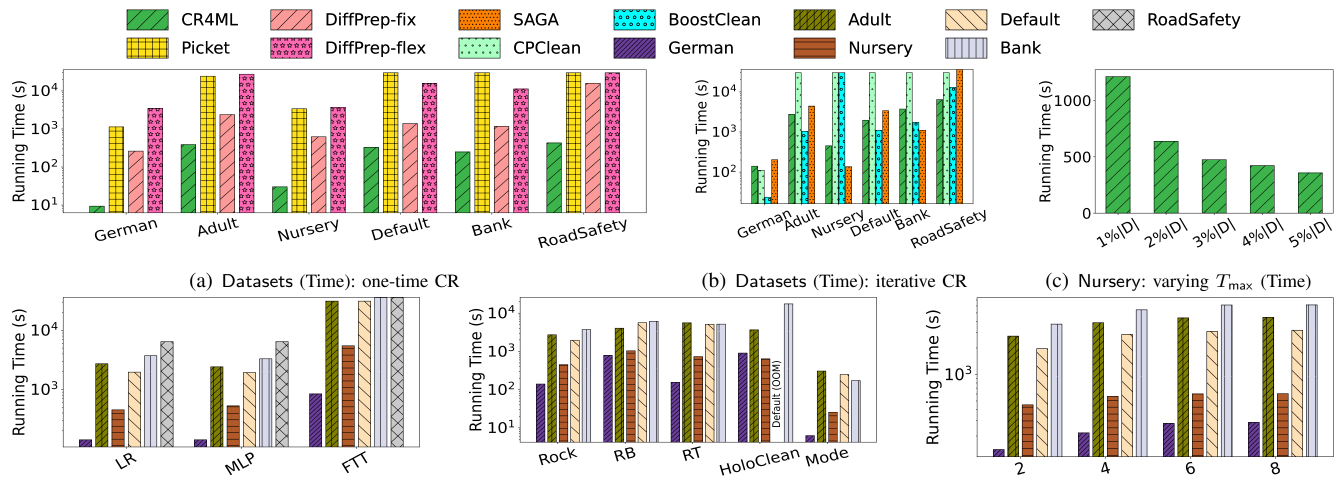
We also evaluated the impact of noises in influential vs. non-influential data, by injecting the same ratio (5%) of noises into influential (resp. non-influential) data in the "clean" version of $\mathcal{D}$, resulting in "dirty" $\mathcal{D}_{\text{inf}}$ (resp. $\mathcal{D}_{\text{non-inf}}$). Comparing the accuracy of $\mathcal{M}$ trained on $\mathcal{D}_{\text{inf}}$ and $\mathcal{D}_{\text{non-inf}}$, we find that noises in influential data significantly degrades the accuracy of $\mathcal{M}$, while noises in non-influential data have little impact on $\mathcal{M}$, *e.g.,* the accuracy drops by 5.4% (resp. 0%) on Bank when $\mathcal{M}$ is trained on $\mathcal{D}_{\text{inf}}$ (resp. $\mathcal{D}_{\text{non-inf}}$). This reinforces the importance of identifying and resolving conflicts in influential data.

Varying m. We varied the number m of attribute sets from $0.5|\mathcal{R}|$ to $2.5|\mathcal{R}|$ in Figure 6(d). To better illustrate, we set $e = 3$. The ACC_{re} of CR4ML increases when m gets larger, *e.g.,* 39.4% (resp. 46.6%) when $m = 0.5|\mathcal{R}|$ (resp. $2|\mathcal{R}|$). This is because a larger m indicates a larger search space, increasing the chance of identifying desired influential attributes. Nonetheless, a fairly small m often suffices, *e.g.,* the ACC_{re} of $\mathcal{M}$ with $m = |\mathcal{R}|$ is just 1.7% less than with $m = 2|\mathcal{R}|$.

Varying ite_{max}. We varied ite_{max} from 1 to 5 with $e = 3$. As shown in Figure 6(e), the ACC_{re} initially increases and then stabilizes when ite_{max} exceeds 2. Indeed, while CR4ML checks more candidate attribute sets when ite_{max} increases, a good set of influential attributes can be identified with a small ite_{max}.

Varying T_{max}. We tested the impact of the number T_{max} of influential tuples on CR4ML, varying it from $1\%|\mathcal{D}|$ to $5\%|\mathcal{D}|$ in Figure 6(f). To better illustrate its effect, we disabled Algorithm AlgIA (*i.e.,* we cleaned influential tuples alone).

As shown there, when T_{max} increases, the ACC_{re} of CR4ML decreases, *e.g.,* its ACC_{re} decreases from 22.9% to -45.6% when T_{max} varies from $1\%|\mathcal{D}|$ to $3\%|\mathcal{D}|$. This is because



(a) Datasets (Time): one-time CR (b) Datasets (Time): iterative CR (c) Nursery: varying $T_{\max}$ (Time)

(d) Datasets: varying $\mathcal{M}$ (Time) (e) Datasets: varying $\mathcal{A}_{\text{CR}}$ (Time) (f) Datasets: varying $\text{ite}_{\max}$ (Time)

Fig. 7: Efficiency evaluation

with a large $T_{\max}$, the estimation of joint influence of tuples in CR4ML is less accurate, and thus the assistant model $\hat{\mathcal{M}}$ may provide the target model $\mathcal{M}$ with inappropriate tuples for incremental training. However, since a small $T_{\max}$ also results in longer runtime (see Exp-2), we suggest to set $T_{\max} = 2\%|\mathcal{D}|$ that balances effectiveness and efficiency well.

Varying e. We varied the number e of epochs for incremental learning (not shown). The ACC_{re} of CR4ML is not very sensitive to e when it reaches a certain threshold (*e.g.,* 3). This is because $\mathcal{M}$ can fit $\mathcal{D}$ within a few epochs.

Ablation study. We also evaluated CR4ML via following tests.

(1) Recall that we retrieve a relatively clean set $\mathcal{C}_k$ from $\mathcal{D}_k$ as validation set. We tested its effectiveness in Figure 6(g). Denote by CR_{gtr} a variant of CR4ML by retrieving $\mathcal{C}_k$ with Glister [8] (a coreset selection method). CR4ML beats CR_{gtr}; on average its ACC_{re} is 55.9%, rather than 27% of CR_{gtr}. This is because CR4ML obtains $\mathcal{C}_k$ by continuously tracking per-tuple loss dynamics to identify tuples likely to be clean, while Glister is mainly designed to select tuples to accelerate model *training*; this deviates of the objective of $\mathcal{C}_k$ for *validation*.

(2) To test the usefulness of our influential data selection strategy, we compared Random, an approach that applies $\mathcal{A}_{\text{CR}}$ to 5 (resp. 30%) random attributes (resp. tuples). CR4ML achieves a higher ACC_{re} (54.8% vs. 30.8%), verifying that the data we select to fix are truly "influential" to model accuracy.

(3) We next evaluated the impact of CR accuracy on CR4ML. Denote by $\text{CR4ML}_{x\%}$ a CR4ML variant that applies only $x\%$ of correct fixes by Rock. As shown in Table III, the ACC_{re} of CR4ML increases with CR accuracy, *e.g.,* rising from 13% to 54.8% as x increases from 10 to 100. These results show that higher CR accuracy yields larger gains, as expected.

Adaptations to regression. We evaluated the effectiveness of CR4ML on regression tasks (not shown). Using the California housing dataset House [16] that predicts median house value from demographic/geographic features, we injected 10% noise with SI, applied MLPRegressor [11] as $\mathcal{M}$, RB as $\mathcal{A}_{\text{CR}}$, and measured performance with RMSE (the smaller, the better). The RMSE of CR4ML is 5.1% lower than Base. This shows the adaptability of CR4ML for regression with minor changes.

Exp-2 Efficiency. We next compared the efficiency of CR4ML and baselines. We evaluated CR4ML for both one-round and multi-round iterations, and compared it with the non-iterative baselines (Picket, DiffPrep-fix and DiffPrep-flex) and iterative ones (BoostClean, CPClean and SAGA), respectively; we did not include (a) ActiveClean, since it needs experts for CR and is hard to estimate its time and (b) CL and Glister, since they are not tailored for relational data. We omitted a baseline if it ran out of memory (OOM) or could not finish in 10 hours.

We compared CR4ML with baselines in Figures 7(a)-7(b).

Non-iterative baselines. One-round execution of CR4ML beats all non-iterative baselines, *e.g.,* 16.8X, 121.6X, and 96.5X faster than DiffPrep-fix, DiffPrep-flex and Picket on average, respectively. Moreover, its ACC_{re} is higher than the three for LR (see Exp-1), verifying that CR4ML can make practical use of CR. Better still, multi-round CR4ML is even comparable to some non-iterative baselines, *e.g.,* it takes 140.5s on German with 15 rounds, and is 8.1X, 1.9X and 24.8X faster than Picket, DiffPrep-fix and DiffPrep-flex, respectively.

Iterative baselines. Multi-round CR4ML is faster than or comparable to iterative baselines, *e.g.,* it only takes 451.6s on Nursery and is at least 66.4X faster than CPClean. This is because CR4ML only detects and resolves conflicts in influential data, and stops as soon as all such conflicts are resolved. Indeed, CR4ML converges after 7 rounds on average. Note that BoostClean is often the fastest one since it enhances $\mathcal{M}$ by training weaker classifiers, without directly cleaning $\mathcal{D}$ using $\mathcal{A}_{\text{CR}}$, which dominates the computational cost. This said, CR4ML is able to not only improve the accuracy of $\mathcal{M}$ but also tune datasets for other use, *e.g.,* training other models.

In short, CR4ML is practical since (a) it is fast given efficient $\mathcal{M}$ and $\mathcal{A}_{\text{CR}}$, and (b) its creator-critic framework stops as soon as $\mathcal{A}_{\text{CR}}$ resolves conflicts in all influential data.

Scalability on $|\mathcal{D}|$. To test the scalability of CR4ML on large datasets, we produced an enlarged Adult with 1M tuples by repeatedly sampling tuples in Adult. We find that CR4ML is able to handle large $\mathcal{D}$ with a reasonable T_{max}, *e.g.,* with T_{max} = $10\%|\mathcal{D}|$, CR4ML takes 14,909s to get $\mathsf{ACC}_{\mathsf{re}} = 66.9\%$, which is only 0.1% less than the $\mathsf{ACC}_{\mathsf{re}}$ of CR4ML with T_{max} = $2\%|\mathcal{D}|$, while all baselines fail due to OOM or time limitation.

Varying T_{max}. We disabled AlgIA and varied the number T_{max} of influential tuples, from $1\%|\mathcal{D}|$ to $5\%|\mathcal{D}|$. As shown in Figure 7(c), it takes less time when T_{max} gets larger, from 1211.6s to 356.9s on Nursery. This is because with a larger T_{max}, algorithm AlgIT may find more tuples to fix, and thus make $\mathcal{M}$ converge quicker to a local optima. However, larger T_{max} may make CR4ML less accurate as shown in Exp-1 (see also Section IV-B) and thus, we set $T_{\mathsf{max}} = 2\%|\mathcal{D}|$ in default setting.

Varying $\mathcal{M}$. We tested the impact of ML models $\mathcal{M}$ on the efficiency. As shown in Figure 7(d), CR4ML takes longer when $\mathcal{M}$ is more complex, *e.g.,* with FTT it takes 5,487.2s on Nursery, 12.1X longer than with LR. This is because with a complex $\mathcal{M}$, CR4ML needs more time to train $\mathcal{M}$, *e.g.,* to compute the Hessian matrix for $\mathcal{M}$. This said, CR4ML is fast given fast $\mathcal{M}$, *e.g.,* with LR it only takes 140.5s on German.

Varying $\mathcal{A}_{\mathsf{CR}}$. The efficiency of CR4ML is heavily dependent on embedded $\mathcal{A}_{\mathsf{CR}}$ as shown in Figure 7(e), *e.g.,* it takes 1056.1s (resp. 5,669.2s) on average when $\mathcal{A}_{\mathsf{CR}}$ is Rock (resp. HoloClean). When $\mathcal{A}_{\mathsf{CR}}$ is fast, so is CR4ML, *e.g.,* CR4ML with Mode takes only 6.3s and 307s on German and Adult, respectively, compared to 156.1s and 5,575.5s with RB.

Varying $\mathsf{ite}_{\mathsf{max}}$. We varied the maximum number $\mathsf{ite}_{\mathsf{max}}$ of iterations for algorithm AlgIA from 1 to 8 with $e = 3$. As shown in Figure 7(f), the cost of CR4ML first increases and then stabilizes, *e.g.,* it increases from 2,025s to 4,379s on Adult when $\mathsf{ite}_{\mathsf{max}}$ varies from 1 to 6, and then remains stable when $\mathsf{ite}_{\mathsf{max}} \geq 6$. This is because given a larger $\mathsf{ite}_{\mathsf{max}}$, CR4ML explores more attribute combinations to fine-tune the models, and takes longer as expected. This said, a small $\mathsf{ite}_{\mathsf{max}}$ suffices for AlgIA to find a good $\mathcal{S}$ and converge, as shown in Exp-1.

The impact of m is similar and hence is not shown.

Varying e. We also varied the number e of epochs for incremental learning from 1 to 5 (not shown). When e gets larger, the training time of $\mathcal{M}$ increases slightly, *e.g.,* when e varies from 1 to 5, CR4ML only takes 1.06X longer to converge. This tells us that it only needs a few epochs such that $\mathcal{M}$ fits the newly cleaned tuples in $\mathcal{D}$, *i.e.,* a small e suffices; moreover, the result also verifies that incremental training is efficient.

Summary. We find the following. (1) On average, CR4ML improves $\mathsf{ACC}_{\mathsf{re}}$ of differential classification models by 40.5% *e.g.,* it improves the $\mathsf{ACC}_{\mathsf{re}}$ of LR by 54.8%, as opposed to -28.1%, 12.3%, -13.9%, 26.8%, 5.5% and -12.2% of ActiveClean, DiffPrep-fix, DiffPrep-flex, SAGA, CL and Glister, respectively. This verifies that CR can indeed improve the model accuracy. (2) CR4ML is more accurate than cleaning the entire dataset (resp. the top-ranked attributes/tuples), with 12.8% (resp. 19.4%) higher $\mathsf{ACC}_{\mathsf{re}}$. These justify the need for selectively cleaning. (3) CR4ML improves $\mathcal{M}$ better with more accurate CR, *e.g.,* the $\mathsf{ACC}_{\mathsf{re}}$ of CR4ML with LR is 55.1%, as opposed to 4.7% with RT on Default for LR. This said, it works with imprecise CR by mitigating the impact of noises that CR may introduce. (4) CR4ML is fast when equipped with an efficient CR method, *e.g.,* with Rock, CR4ML is faster than most non-iterative and iterative baselines. It scales well, *e.g.,* it takes 14,909s on a dataset with 1M tuples for LR, with $\mathsf{ACC}_{\mathsf{re}} = 66.9\%$.

VII. Related Work

We categorize the related work as follows.

Data cleaning for ML. The prior work improves the accuracy of white-box and black-box models by cleaning training data.

White-box. ActiveClean [45] treats data cleaning and model training as a form of stochastic gradient descent. CPClean [46] studies the impact of data incompleteness on the quality of KNN models. [91] investigates whether Naive Bayes classifiers trained on possible worlds of incomplete data produce consistent predictions on a fixed test dataset. [92] learns an accurate ML model on incomplete training data without imputing missing values. ZORRO [93] learns linear models over possible worlds of uncertain data. CleanML [17] empirically studies the impact of data cleaning on ML classification on structured data. It claims that CR "is more likely to have insignificant impact and unlikely to have negative impact".

Black-box. BoostClean [48] automatically selects an ensemble of detecting and repairing operations from a library. Picket [49] proposes a framework to safeguard against data corruption during both training and deployment of ML models. MLClean [50] proposes a framework to simultaneously handle data cleaning, unfairness mitigation and sanitation. Snorkel [51] allows users to train ML models via data programming with labeling functions. Coco [52], [53] assesses unfriendly tuples for models by using constraints defined with arithmetic expressions on numerical attributes. Complaint-driven methods [55], [56] leverage users' complaints to remove/revise a minimum set of training examples. CategDedup [58] empirically studies the impact of categorical duplicates on ML models. DLearn [94] employs integrity constraints to learn accurate pattern models on dirty data without cleaning the data. COMET [95] incrementally selects features to clean by simulating errors and estimating their impact on model accuracy. CL [2] improves training by removing mislabeled data using model-predicted probabilities. CHEF [7] iteratively relabels influential examples with human annotators and updates parameters via influence functions. CDC [20] automates imputation with conformal prediction to enhance ML performance.

This work differs from prior work on data cleaning in the following. (1) Instead of one-shot cleaning for $\mathcal{M}$ (*e.g.,* MLClean [50] and CleanML [17]), we propose CR4ML, a creator-critic framework for differentiable models $\mathcal{M}$, to iteratively conduct CR and model training. It selectively resolves conflicts on influential data, rather than indistinguishably applying

CR to the entire dataset as practiced by all previous works except [95]. (2) CR4ML differs from the selective cleaning in [95] as follows: (a) we do not require a noise injector that replicates the noise pattern of dirty data, (b) we account for the impact of noise that may be introduced by imprecise CR methods, and (c) we perform CR at both tuple and attribute levels, not just the attribute level as in [95]. (3) CR4ML gives a concrete solution to improve $\mathcal{M}$ via CR, beyond an empirical analysis as in [58]. (4) CR4ML does not restrict the choice of (perfect) CR methods as in ActiveClean [45], BoostClean [48] and COMET [95]; in fact, it mitigates the impact of noise introduced by imprecise CR via a two-step fixing strategy at both attribute and tuple levels. (5) CR4ML detects and resolves conflicts in the data, rather than deleting dirty tuples entirely as in [49]. (6) CR4ML does not require user interaction in the cleaning/training process, as opposed to Snorkel [51] and complaint-driven methods [55], [56]. (7) CR4ML aims to enhance the accuracy of differential models $\mathcal{M}$ by detecting and resolving conflicts in dirty data, rather than investigating the stability of $\mathcal{M}$ trained on incomplete data as in [91], [92], or learning linear models on uncertain data as in [93]. (8) CR4ML focuses on improving the accuracy of a specific downstream ML model, rather than learning patterns over possible worlds as in [94]. (9) Compared to CL [2], CR4ML: (a) considers conflicts in all attributes, while CL only targets errors in classification labels; (b) detects erroneous tuples by iteratively monitoring training losses, while CL relies on model-predicted class probabilities; and (c) resolves conflicts in the data, whereas CL directly discards suspicious samples. (10) CR4ML is fully human-free, unlike CHEF [7] which depends on manual relabeling and only targets errors in classification labels; moreover, CR4ML selects influential tuples based on their impact on validation accuracy, rather than validation loss as in CHEF.

Data cleaning for AutoML. A related topic is data cleaning for AutoML, that selects ML pipelines to maximumly improve ML models, in which data cleaning serves as a component.

White-box. DiffML [96] enables differentiable ML pipelines by assigning adjustable parameters to each training record and each error detection/repairing approach in a pool of data cleaning tools. AutoClean [18] re-evaluates several CleanML benchmarks using AutoML systems and extends AutoSklearn [97] with more advanced cleaning strategies for both white-box and black-box models; it concludes that most existing benchmarks contain few errors that can substantially impact the downstream models, and thus, calls on the community for benchmarks with impactful real-world errors so as to study data cleaning for ML/AutoML on the attribute level.

Black-box. AlphaClean [98] studies parameter tuning for data cleaning pipelines to optimize an objective function. AutoSklearn [97] improves AutoML by assessing its historical performance on similar datasets. DiffPrep [88] proposes an automatic data preprocessing method for any dataset and differentiable model. SAGA [59] automatically generates the top-K most effective data cleaning pipelines.

More generally, [99] shows how different aspects of data quality propagate through various stages of ML development; [22], [100] review recent progress in data cleaning for ML; Rein [19] introduces a comprehensive benchmark to evaluate the impact of data cleaning methods on various ML models; [24] evaluates 12 data repair algorithms across diverse datasets; DataPerf [1] defines benchmark tasks that assess the quality of training data with fixed models and testing sets to advance data-centric AI; and CtxPipe [21] uses pretrained embeddings and reinforcement learning to automate AutoML pipeline construction. None of the previous work considers how to make practical use of CR methods in a fine-grained manner to improve the accuracy of ML classifications.

This work differs from the prior work on AutoML in the following. (1) Rather than replying on other components (*e.g.,* feature engineering [18], [97]) in AutoML, we study the impact of CR itself on ML; moreover, our results can help AutoML prepare data for feature engineering and smoothing, beyond serving as embedded data cleaning. (2) CR4ML allows users to plug in any CR method with $\alpha > 0.5$, as opposed to those (*e.g.,* [88], [96]) that only support restrictive CR tools.

Other optimizations for ML/AutoML. There are also studies that boost ML models through strategies beyond data cleaning.

Glister [8] shows that coreset selection can mitigate noise in model training by framing it as a submodular optimization problem and selecting examples via a Taylor approximation of validation loss reduction. [10] and [9] further improve coreset selection in streaming and large-model settings through bilevel optimization and lightweight proxy models. GoodCore [47] selects a coreset over incomplete data through gradient approximation. DE4ML [23] augments relational datasets to improve model robustness against adversarial attacks. Amalur [54] unifies data integration and ML. Explanation-based models [57] treat query explanation as a hyper-parameter tuning problem; it adopts Random tree and Bayesian optimization to solve it.

This work differs from the prior works on other optimizations in the following. (1) CR4ML adapts the idea of coreset selection to extract a relatively clean validation set from noisy data to guide influential tuple/attribute selection, not for training efficiency as in [8]–[10]. (2) CR4ML improves ML accuracy by cleaning influential data, rather than enhancing robustness against attacks via data augmentation as in [23].

VIII. Conclusion

We have shown how CR can be effectively utilized to improve the accuracy of downstream ML models. (1) CR can substantially enhance differential classification models when applied correctly, only to influential attributes and tuples. (2) We propose CR4ML, a creator-critic framework that can work even with not-so-accurate CR tools $\mathcal{A}_{\mathsf{CR}}$, mitigating the impact of noise introduced by $\mathcal{A}_{\mathsf{CR}}$. (3) Although identifying influential attributes and tuples is intractable, CR4ML provides effective methods with overall approximation bounds. Our experiments have verified that CR4ML is promising in practice.

Topics for future work include extending CR4ML to (a) improve non-differential classifiers, and (b) support other data cleaning operations, *e.g.,* label cleaning and outlier repairing.

References

[1] M. Mazumder, C. Banbury, X. Yao, B. Karlaš, W. Gaviria Rojas, S. Diamos, G. Diamos, L. He, A. Parrish, H. R. Kirk *et al.*, "DataPerf: Benchmarks for data-centric ai development," *NIPS*, pp. 5320–5347, 2023.

[2] C. Northcutt, L. Jiang, and I. Chuang, "Confident learning: Estimating uncertainty in dataset labels," *JAIR*, pp. 1373–1411, 2021.

[3] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 1885–1894.

[4] Z. Hammoudeh and D. Lowd, "Training data influence analysis and estimation: A survey," *Machine Learning*, 2024.

[5] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *NIPS*, 2019.

[6] "Code, datasets and full version," 2025, https://github.com/HatsuneHan/CR4ML-ICDE26.

[7] Y. Wu, J. Weimer, and S. B. Davidson, "CHEF: A cheap and fast pipeline for iteratively cleaning label uncertainties (technical report)," *VLDB*, 2021.

[8] K. Killamsetty, D. Sivasubramanian, G. Ramakrishnan, and R. Iyer, "Glister: Generalization based data subset selection for efficient and robust learning," in *AAAI*, 2021, pp. 8110–8118.

[9] C. Coleman, C. Yeh, S. Mussmann, B. Mirzasoleiman, P. Bailis, P. Liang, J. Leskovec, and M. Zaharia, "Selection via proxy: Efficient data selection for deep learning," *ICLR*, 2020.

[10] Z. Borsos, M. Mutny, and A. Krause, "Coresets via bilevel optimization for continual learning and streaming," in *NeurIPS*, 2020.

[11] 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*, vol. 12, pp. 2825–2830, 2011.

[12] H. T. Lam, J.-M. Thiebaut, M. Sinn, B. Chen, T. Mai, and O. Alkan, "One button machine for automating feature engineering in relational databases," *arXiv preprint arXiv:1706.00327*, 2017.

[13] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan, "Deep entity matching with pre-trained language models," *PVLDB*, vol. 14, no. 1, pp. 50–60, 2020.

[14] M. Mahdavi and Z. Abedjan, "Baran: Effective error correction via a unified context representation and transfer learning," *PVLDB*, vol. 13, no. 12, pp. 1948–1961, 2020.

[15] X. Bao, Z. Bao, Q. Duan, W. Fan, H. Lei, D. Li, W. Lin, P. Liu, Z. Lv, M. Ouyang, J. Peng, J. Zhang, R. Zhao, S. Tang, S. Zhou, Y. Wang, Q. Wei, M. Xie, J. Zhang, X. Zhang, R. Zhao, and S. Zhou, "Rock: Cleaning data by embedding ML in logic rules," in *SIGMOD (industrial track)*, 2024.

[16] "California housing prices," 2025, https://www.kaggle.com/datasets/camnugent/california-housing-prices.

[17] P. Li, X. Rao, J. Blase, Y. Zhang, X. Chu, and C. Zhang, "CleanML: A study for evaluating the impact of data cleaning on ML classification tasks," in *ICDE*. IEEE, 2021, pp. 13–24.

[18] F. Neutatz, B. Chen, Y. Alkhatib, J. Ye, and Z. Abedjan, "Data cleaning and AutoML: Would an optimizer choose to clean?" *Datenbank-Spektrum*, 2022.

[19] M. Abdelaal, C. Hammacher, and H. Schöning, "REIN: A comprehensive benchmark framework for data cleaning methods in ML pipelines," in *EDBT*. OpenProceedings.org, 2023, pp. 499–511.

[20] S. Jäger and F. Biessmann, "From data imputation to data cleaning - automated cleaning of tabular data improves downstream predictive performance," in *International Conference on Artificial Intelligence and Statistics*, ser. PMLR, 2024.

[21] H. Gao, S. Cai, T. T. A. Dinh, Z. Huang, and B. C. Ooi, "Ctxpipe: Context-aware data preparation pipeline construction for machine learning," *SIGMOD*, 2024.

[22] P.-O. Côté, A. Nikanjam, N. Ahmed, D. Humeniuk, and F. Khomh, "Data cleaning and machine learning: A systematic literature review," *Automated Software Engineering*, 2024.

[23] W. Fan, X. Han, W. Ren, and Z. Xu, "Data enhancement for binary classification of relational data," *SIGMOD*, pp. 1–28, 2025.

[24] W. Ni, X. Miao, X. Zhao, Y. Wu, and J. Yin, "Automatic data repair: Are we ready to deploy?" *VLDB*, 2024.

[25] D. C. Hoaglin and B. Iglewicz, "Fine-tuning some resistant rules for outlier labeling," *Journal of the American statistical Association*, vol. 82, no. 400, pp. 1147–1149, 1987.

[26] W. Fan, F. Geerts, N. Tang, and W. Yu, "Conflict resolution with data currency and consistency," *J. Data and Information Quality*, vol. 5, no. 1-2, pp. 6:1–6:37, 2014.

[27] M. Arenas, L. Bertossi, and J. Chomicki, "Consistent query answers in inconsistent databases," in *PODS*, 1999, pp. 68–79.

[28] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," *ACM Trans. Database Syst.*, vol. 33, no. 2, pp. 6:1–6:48, 2008.

[29] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, "Improving data quality: Consistency and accuracy," in *VLDB*, 2007, pp. 315–326.

[30] S. Giannakopoulou, M. Karpathiotakis, and A. Ailamaki, "Cleaning denial constraint violations through relaxation," in *SIGMOD*, 2020, pp. 805–815.

[31] G. Beskales, I. F. Ilyas, L. Golab, and A. Galiullin, "On the relative trust between inconsistent data and inaccurate constraints," in *ICDE*. IEEE, 2013, pp. 541–552.

[32] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas, "Guided data repair," *PVLDB*, vol. 4, no. 5, pp. 279–289, 2011.

[33] L. Bertossi, *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers, 2011.

[34] X. Ding, H. Wang, J. Su, M. Wang, J. Li, and H. Gao, "Leveraging currency for repairing inconsistent and incomplete data," *TKDE*, 2020.

[35] F. Geerts, G. Mecca, P. Papotti, and D. Santoro, "The llunatic data-cleaning framework," *PVLDB*, vol. 6, no. 9, pp. 625–636, 2013.

[36] A. Gilad, D. Deutch, and S. Roy, "On multiple semantics for declarative database repairs," in *SIGMOD*, 2020, pp. 817–831.

[37] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "HoloClean: Holistic data repairs with probabilistic inference," *PVLDB*, vol. 10, no. 11, pp. 1190–1201, 2017.

[38] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, "Towards certain fixes with editing rules and master data," *VLDB J.*, vol. 21, no. 2, pp. 213–238, 2012.

[39] A. Heidari, J. McGrath, I. F. Ilyas, and T. Rekatsinas, "HoloDetect: Few-shot learning for error detection," in *SIGMOD*, 2019, pp. 829–846.

[40] M. Mahdavi, Z. Abedjan, R. Castro Fernandez, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, "Raha: A configuration-free error detection system," in *SIGMOD*, 2019, pp. 865–882.

[41] L. Visengeriyeva and Z. Abedjan, "Metadata-driven error detection," in *SSDBM*, 2018, pp. 1:1–1:12.

[42] M. Yakout, L. Berti-Équille, and A. K. Elmagarmid, "Don't be scared: Use scalable automatic repairing with maximal likelihood and bounded changes," in *SIGMOD*. ACM, 2013.

[43] W. Fan, P. Lu, and C. Tian, "Unifying logic rules and machine learning for entity enhancing," *Sci. China Inf. Sci.*, vol. 63, no. 7, 2020.

[44] W. Fan, Z. Han, W. Ren, D. Wang, Y. Wang, M. Xie, and M. Yan, "Splitting tuples of mismatched entities," *Proc. ACM Manag. Data*, 2024.

[45] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg, "ActiveClean: Interactive data cleaning for statistical modeling," *PVLDB*, vol. 9, no. 12, pp. 948–959, 2016.

[46] B. Karlas, P. Li, R. Wu, N. M. Gürel, X. Chu, W. Wu, and C. Zhang, "Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions," *CoRR*, vol. abs/2005.05117, 2020. [Online]. Available: https://arxiv.org/abs/2005.05117

[47] C. Chai, J. Liu, N. Tang, J. Fan, D. Miao, J. Wang, Y. Luo, and G. Li, "GoodCore: Data-effective and data-efficient machine learning through coreset selection over incomplete data," *Proc. ACM Manag. Data*, 2023.

[48] S. Krishnan, M. J. Franklin, K. Goldberg, and E. Wu, "BoostClean: Automated error detection and repair for machine learning," *CoRR*, vol. abs/1711.01299, 2017.

[49] Z. Liu, Z. Zhou, and T. Rekatsinas, "Picket: Self-supervised data diagnostics for ML pipelines," *CoRR*, vol. abs/2006.04730, 2020. [Online]. Available: https://arxiv.org/abs/2006.04730

[50] K. H. Tae, Y. Roh, Y. H. Oh, H. Kim, and S. E. Whang, "Data cleaning for accurate, fair, and robust models: A big data - AI integration approach," in *DEEM@SIGMOD 2019*. ACM, 2019, pp. 5:1–5:4.

[51] A. Ratner, S. H. Bach, H. R. Ehrenberg, J. A. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *PVLDB*, vol. 11, no. 3, pp. 269–282, 2017.

[52] A. Fariha, A. Tiwari, A. Meliou, A. Radhakrishna, and S. Gulwani, "Coco: Interactive exploration of conformance constraints for data understanding and data cleaning," in *SIGMOD*. ACM, 2021, pp. 2706–2710.

[53] A. Fariha, A. Tiwari, A. Radhakrishna, S. Gulwani, and A. Meliou, "Conformance constraint discovery: Measuring trust in data-driven systems," in *SIGMOD*. ACM, 2021, pp. 499–512.

[54] R. Hai, C. Koutras, A. Ionescu, Z. Li, W. Sun, J. van Schijndel, Y. Kang, and A. Katsifodimos, "Amalur: Data integration meets machine learning," in *ICDE*. IEEE, 2023, pp. 3729–3739.

[55] W. Wu, L. Flokas, E. Wu, and J. Wang, "Complaint-driven training data debugging for query 2.0," in *SIGMOD*. ACM, 2020, pp. 1317–1334.

[56] L. Flokas, W. Wu, Y. Liu, J. Wang, N. Verma, and E. Wu, "Complaint-driven training data debugging at interactive speeds," in *SIGMOD*, 2022, pp. 369–383.

[57] B. Lockhart, J. Peng, W. Wu, J. Wang, and E. Wu, "Explaining inference queries with Bayesian optimization," *PVLDB*, vol. 14, no. 11, pp. 2576–2585, 2021.

[58] V. Shah, T. Parashos, and A. Kumar, "How do categorical duplicates affect ML? A new benchmark and empirical analyses," *PVLDB*, vol. 17, no. 6, pp. 1391–1404, 2024.

[59] S. Siddiqi, R. Kern, and M. Boehm, "SAGA: A scalable framework for optimizing data cleaning pipelines for machine learning applications," *Proc. ACM Manag. Data*, 2024.

[60] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang, "The data civilizer system," in *CIDR*. www.cidrdb.org, 2017.

[61] A. Agrawal, R. Chatterjee, C. Curino, A. Floratou, N. Godwal, M. Interlandi, A. Jindal, K. Karanasos, S. Krishnan, B. Kroth, J. Leeka, K. Park, H. Patel, O. Poppe, F. Psallidas, R. Ramakrishnan, A. Roy, K. Saur, R. Sen, M. Weimer, T. Wright, and Y. Zhu, "Cloudy with high chance of DBMS: A 10-year prediction for enterprise-grade ML," *CoRR*, vol. abs/1909.00084, 2019.

[62] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer, "Enterprise data analysis and visualization: An interview study," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 12, pp. 2917–2926, 2012.

[63] "Why users love Cleanlab," 2025, https://cleanlab.ai/love/.

[64] T. Zhang, Z. A. Zhang, Z. Fan, H. Luo, F. Liu, Q. Liu, W. Cao, and L. Jian, "OpenFE: Automated feature generation with expert-level performance," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 41 880–41 901.

[65] E. F. Codd, "Extending the database relational model to capture more meaning," *TODS*, vol. 4, no. 4, pp. 397–434, 1979.

[66] "Scikit-learn metrics," 2025, https://scikit-learn.org/stable/modules/model_evaluation.html.

[67] Z. Wang and Y. Shen, "Incremental learning for multi-interest sequential recommendation," in *ICDE*. IEEE, 2023, pp. 1071–1083.

[68] H. Liu, S. Di, and L. Chen, "Incremental tabular learning on heterogeneous feature space," *Proc. ACM Manag. Data*, vol. 1, no. 1, pp. 18:1–18:18, 2023.

[69] T. Zhou, S. Wang, and J. A. Bilmes, "Robust curriculum learning: from clean label detection to noisy label self-correction," in *ICLR*, 2021.

[70] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[71] B. Mirzasoleiman, J. A. Bilmes, and J. Leskovec, "Coresets for data-efficient training of machine learning models," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 6950–6960.

[72] S. A. Cook, "The complexity of theorem-proving procedures," in *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, 2023, pp. 143–152.

[73] C. Darwin, "Origin of the species," in *British Politics and the Environment in the Long Nineteenth Century*. Routledge, 2023, pp. 47–55.

[74] O. H. Babatunde, L. Armstrong, J. Leng, and D. Diepeveen, "A genetic algorithm-based feature selection," Edith Cowan University, Tech. Rep., 2014.

[75] B. Mirzasoleiman, K. Cao, and J. Leskovec, "Coresets for robust training of deep neural networks against noisy labels," in *NeurIPS*, 2020.

[76] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2018.

[77] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches." *Journal of Machine Learning Research*, vol. 13, no. 1, 2012.

[78] P. Toulis, T. Horel, and E. M. Airoldi, "The proximal robbins–monro method," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 83, no. 1, pp. 188–212, 2021.

[79] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Mathematical programming*, vol. 14, pp. 265–294, 1978.

[80] Y. Seldin and N. Tishby, "Multi-classification by categorical features via clustering," in *ICML*, ser. Proceedings of Machine Learning Research. PMLR, 2008, pp. 920–927.

[81] Z. Wang, Y. Zhou, M. Qiu, I. Haque, L. Brown, Y. He, J. Wang, D. Lo, and W. Zhang, "Towards fair machine learning software: Understanding and addressing model bias through counterfactual thinking," *arXiv preprint arXiv:2302.08018*, 2023.

[82] S. Guha, F. A. Khan, J. Stoyanovich, and S. Schelter, "Automated data cleaning can hurt fairness in machine learning-based decision making," in *ICDE*. IEEE, 2023, pp. 3747–3754.

[83] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" in *Advances in Neural Information Processing Systems*, 2022.

[84] P. Li, X. Rao, J. Blase, Y. Zhang, X. Chu, and C. Zhang, "CleanML: A benchmark for joint data cleaning and machine learning [experiments and analysis]," *CoRR*, vol. abs/1904.09483, 2019.

[85] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *ICCV*, 2016.

[86] Z. He, C. Ouyang, L. Alzubaidi, A. Barros, and C. Moreira, "Investigating imperceptibility of adversarial attacks on tabular data: An empirical analysis," *arXiv preprint arXiv:2407.11463*, 2024.

[87] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for capturing data inconsistencies," *TODS*, 2008.

[88] P. Li, Z. Chen, X. Chu, and K. Rong, "DiffPrep: Differentiable data preprocessing pipeline search for learning over tabular data," *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 183:1–183:26, 2023.

[89] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," *NeurIPS*, vol. 34, pp. 18 932–18 943, 2021.

[90] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2020.

[91] S. Bian, X. Ouyang, Z. Fan, and P. Koutris, "Naive Bayes classifiers over missing data: Decision and poisoning," *arXiv preprint arXiv:2303.04811*, 2023.

[92] C. Zhen, N. Aryal, A. Termehchy, and A. S. Chabada, "Certain and approximately certain models for statistical learning," *SIGMOD*, vol. 2, no. 3, pp. 1–25, 2024.

[93] J. Zhu, S. Feng, B. Glavic, and B. Salimi, "Learning from uncertain data: From possible worlds to possible models," *arXiv preprint arXiv:2405.18549*, 2024.

[94] J. Picado, J. Davis, A. Termehchy, and G. Y. Lee, "Learning over dirty data without cleaning," in *SIGMOD*, 2020, pp. 1301–1316.

[95] S. Mohammed, F. Naumann, and H. Harmouch, "Step-by-step data cleaning recommendations to improve ML prediction accuracy," *arXiv preprint arXiv:2503.11366*, 2025.

[96] B. Hilprecht, C. Hammacher, E. S. dos Reis, M. Abdelaal, and C. Binnig, "DiffML: End-to-end differentiable ML pipelines," in *DEEM@SIGMOD*. ACM, 2023, pp. 7:1–7:7.

[97] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *NIPS*, 2015, pp. 2962–2970.

[98] S. Krishnan and E. Wu, "AlphaClean: Automatic generation of data cleaning pipelines," *CoRR*, vol. abs/1904.11827, 2019. [Online]. Available: http://arxiv.org/abs/1904.11827

[99] C. Renggli, L. Rimanic, N. M. Gürel, B. Karlas, W. Wu, and C. Zhang, "A data quality-driven view of MLOps," *IEEE Data Eng. Bull.*, vol. 44, no. 1, pp. 11–23, 2021.

[100] F. Neutatz, B. Chen, Z. Abedjan, and E. Wu, "From cleaning before ML to cleaning for ML," *IEEE Data Eng. Bull.*, vol. 44, no. 1, pp. 24–41, 2021.

[101] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

```
Input: An ML model $\mathcal{M}$, a data schema $\mathcal{R}$, a set $\mathcal{D}$ of $\mathcal{R}$,
        a CR method $\mathcal{A}_{\mathsf{CR}}$, the number $e$ of epochs.
Output: A fine-tuned ML model $\mathcal{M}$ and a cleaned set $\mathcal{D}_{\mathsf{clean}}$.
1.  $k := 0; \mathcal{D}_0 := \mathcal{D}; \hat{\mathcal{M}} := \mathcal{M}; L_k := \emptyset; \{\mathcal{D}_{\mathsf{clean}}, \Delta\mathcal{D}_{\mathsf{clean}}\} := \emptyset$;
2.  while true do
      /* Creator */
3.      $\Delta\mathcal{D}_k := \mathsf{CheckUpdates}(\mathcal{D}_k, \mathcal{D}_{k-1})$;
4.      Model unlearning of $\hat{\mathcal{M}}$ with $\Delta\mathcal{D}_k$;
5.      Incrementally fine-tune $\hat{\mathcal{M}}$ with $\mathcal{D}_k$ in $e$ epochs;
6.      Compute $L_k(t)$ for all $t \in \mathcal{D}_k/\mathcal{D}_{\mathsf{clean}}$;
7.      $\Delta\mathcal{D}_{\mathsf{clean}} := \mathsf{CleanDataIdentify}(\mathcal{D}_k/\mathcal{D}_{\mathsf{clean}}, L_k, L_{k-1})$;
8.      if $\Delta\mathcal{D}_{\mathsf{clean}} \neq \emptyset$ then
9.          Train $\mathcal{M}$ with $\mathcal{D}_{\mathsf{clean}}$ and $\Delta\mathcal{D}_{\mathsf{clean}}$ in $e$ epochs;
10.     $\mathcal{D}_{\mathsf{clean}} := \mathcal{D}_{\mathsf{clean}} \cup \Delta\mathcal{D}_{\mathsf{clean}}$;     $\Delta\mathcal{D}_{\mathsf{clean}} := \emptyset$;
     /* Critic */
11.     err := $\mathsf{ErrSignalsAnalysis}(\hat{\mathcal{M}}, \mathcal{D}_k)$;
12.     $\mathcal{C}_k := \mathsf{ValidationSetIdentify}(\mathcal{D}_k, \mathrm{err})$;
13.     $\mathcal{S} := \mathsf{InfluentialAttributes}(\mathcal{D}_k, \mathcal{R}, \hat{\mathcal{M}}, \mathcal{A}_{\mathsf{CR}}, \mathcal{C}_k, \mathcal{D}_{\mathsf{clean}})$;
14.     $\mathcal{T} := \mathsf{InfluentialTuples}(\mathcal{D}_k, \hat{\mathcal{M}}, \mathcal{A}_{\mathsf{CR}}, \mathcal{C}_k, \mathcal{D}_{\mathsf{clean}})$;
15.     $\mathcal{D}_{k+1} := \mathcal{A}_{\mathsf{CR}}(\mathcal{D}_k, \mathcal{T}, \mathcal{S})$;
16.     if $\mathcal{D}_{k+1}$ remains unchanged from $\mathcal{D}_k$ then
17.         $\mathcal{D}_{\mathsf{clean}} := \mathcal{D}_k$;   Train $\mathcal{M}$ with $\mathcal{D}_{\mathsf{clean}}$ in $e$ epochs;
18.         return $(\mathcal{M}, \mathcal{D}_{\mathsf{clean}})$;
19.     $k := k + 1$;
```

Fig. 8: The workflow of CR4ML

Appendix

A. *Workflow of* CR4ML

Workflow. As shown in Figure 8, CR4ML takes as input $\mathcal{M}$, $\mathcal{R}$, $\mathcal{D}$, $\mathcal{A}_{\mathsf{CR}}$, and the number e of epochs. It outputs a cleaned $\mathcal{D}_{\mathcal{M}}$ (*i.e.,* $\mathcal{D}_{\mathsf{clean}}$) and a fine-tuned $\mathcal{M}$ with $\mathcal{D}_{\mathcal{M}}$. It initializes $\mathcal{D}_0 = \mathcal{D}$, $\hat{\mathcal{M}} = \mathcal{M}$, $\mathcal{D}_{\mathsf{clean}} = \Delta\mathcal{D}_{\mathsf{clean}} = \emptyset$ (line 1). It trains $\hat{\mathcal{M}}$ and $\mathcal{M}$ with the creator, and cleans $\mathcal{D}$ with the critic, in rounds (lines 2-15). In each round, the creator first computes $\Delta\mathcal{D}_k$, the differences between $\mathcal{D}_k$ and $\mathcal{D}_{k-1}$ (line 3). Then it eliminates the effect of $\Delta\mathcal{D}_k$ from $\hat{\mathcal{M}}$ via model unlearning (line 4) and incrementally trains $\hat{\mathcal{M}}$ with $\mathcal{D}_k$ in e epochs (line 5). It computes the dynamic loss (line 6), and identifies new clean data $\Delta\mathcal{D}_{\mathsf{clean}}$ (line 7). If $\Delta\mathcal{D}_{\mathsf{clean}} \neq \emptyset$, it trains $\mathcal{M}$ with $\mathcal{D}_{\mathsf{clean}}$ and $\Delta\mathcal{D}_{\mathsf{clean}}$ in e epochs (lines 8-9). It updates $\mathcal{D}_{\mathsf{clean}}=\mathcal{D}_{\mathsf{clean}}\cup\Delta\mathcal{D}_{\mathsf{clean}}$, and sets $\Delta\mathcal{D}_{\mathsf{clean}}=\emptyset$ (line 10).

The critic is then executed to find influential data in $\mathcal{D}_k$. It first analyzes the error signals (line 11), based on which it identifies a relatively clean subset $\mathcal{C}_k$ of $\mathcal{D}_k$ as the validation set $\mathcal{C}_k$ (line 12); $\mathcal{C}_k$ is used to identify a subset $\mathcal{S}$ of influential attributes and a subset $\mathcal{T}$ of influential tuples (lines 13-14). The CR method $\mathcal{A}_{\mathsf{CR}}$ is then applied on these attributes and tuples to get a better dataset $\mathcal{D}_{k+1}$ for the next round (line 15).

The process proceeds until it is *stable*, *i.e.,* $\mathcal{D}_k$ stabilizes (lines 16-18). We take $\mathcal{D}_k$ as $\mathcal{D}_{\mathsf{clean}}$ and fine-tune $\mathcal{M}$ on $\mathcal{D}_{\mathsf{clean}}$ (line 17) and finally, return improved $\mathcal{M}$ and $\mathcal{D}_{\mathsf{clean}}$ (line 18).

Example 5: Continuing with Example 2, given that that tuples t_1 and t_4 are updated in the second round ($\Delta\mathcal{D}_1 = \{t_1, t_4\}$), the creator eliminates their effect from $\hat{\mathcal{M}}$ by model unlearning and incrementally learns $\hat{\mathcal{M}}$ on $\mathcal{D}_1$. Given $\hat{\mathcal{M}}$, the critic re-computes error signals, based on which a new $\mathcal{C}_1$ is identified. Assume that η_1=0.04, η_2=0.02, $L_1(t_1)$=0.04, $L_0(t_1)$=0.08, $L_1(t_5)$=0.01. Then t_1 and t_5 are added to $\Delta\mathcal{D}_{\mathsf{clean}}$ since (a) t_1 is enhanced by $\mathcal{A}_{\mathsf{CR}}$ with high loss reduction (*i.e.,* $L_1(t_1) - L_0(t_1) \geq \eta_1$), and (b) t_5 has a low loss (*i.e.,* $L_1(t_5) \leq \eta_2$). After identifying $\Delta\mathcal{D}_{\mathsf{clean}}$, $\mathcal{M}$ is trained with $\Delta\mathcal{D}_{\mathsf{clean}}$ and $\mathcal{D}_{\mathsf{clean}}$ (=$\emptyset$), and we update $\mathcal{D}_{\mathsf{clean}} = \mathcal{D}_{\mathsf{clean}}\cup\Delta\mathcal{D}_{\mathsf{clean}}$. After identifying influential data as before, we obtain a cleaned $\mathcal{D}_2$ by fixing $t_6[A_4] = \mathtt{40}$. Since all conflicts are now resolved, $\mathcal{D}_2$ is stable and we incrementally train $\mathcal{M}$ on $\mathcal{D}_{\mathcal{M}} = \mathcal{D}_2$. □

B. *More about model training*

Below we present the incremental training for $\hat{\mathcal{M}}$ and $\mathcal{M}$.

(1) Assistant model $\hat{\mathcal{M}}$. The training for $\hat{\mathcal{M}}$ consists of two parts: (1) *model unlearning* and (2) *incremental learning*, to eliminate the negative effect of dirty tuples in $\mathcal{D}_{k-1}$ (on which the assistant model $\hat{\mathcal{M}}$ is previously trained) and to update $\hat{\mathcal{M}}$ with cleaned tuples in $\mathcal{D}_k$, respectively.

(1A) Model unlearning. Denote by $\Delta\mathcal{D}_k$ the set of tuples in dataset $\mathcal{D}_{k-1}$ that are cleaned by $\mathcal{A}_{\mathsf{CR}}$ in $\mathcal{D}_k$: $\Delta\mathcal{D}_k = \{t \mid t \in \mathcal{D}_{k-1} \text{ and } t \notin \mathcal{D}_k\}$; *i.e.,* $\Delta\mathcal{D}_k$ contains biased/dirty tuples. To debias the negative effects of $\Delta\mathcal{D}_k$ from $\hat{\mathcal{M}}$, we adapt the influence function of [3], an effective tool to estimate parameter change of a model when tuples are removed/modified in the training data. Intuitively, it mimics the scenario that we retrain $\hat{\mathcal{M}}$ with a smaller training data, $\mathcal{D}_{k-1} \setminus \Delta\mathcal{D}_k$, in the $(k-1)$-th round, by removing dirty tuples in $\Delta\mathcal{D}_k$ from $\mathcal{D}_{k-1}$. To achieve this, for each $t = (x, y) \in \Delta\mathcal{D}_k$, where x and y are the attributes and label of t, respectively, we upweight t by an infinitesimal amount ϵ; the influence of upweighting t on parameters of $\hat{\mathcal{M}}$ is

$$\mathcal{I}_{\mathsf{up,params}}(t) \overset{\text{def}}{=} \frac{\mathrm{d}\theta_{\epsilon,t}}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} = -H_\theta^{-1}\nabla_{\theta_{k-1}} l(\hat{\mathcal{M}}(x;\theta_{k-1}), y),$$

where θ_k is the parameter of $\hat{\mathcal{M}}$ in the k-th round, $\nabla_{\theta_{k-1}}$ is the partial derivative of θ_{k-1} *w.r.t.* ϵ, H_θ is the Hessian Matrix, and $l(\hat{\mathcal{M}}(x;\theta_k), y)$ is the loss of tuple t in the k-th round. Note that removing tuple t from the training set is the same as upweighting it by $\epsilon = -\frac{1}{|\mathcal{D}_{k-1}|}$, such that the parameter θ_k of $\hat{\mathcal{M}}$ in the k-th round can be updated as $\theta_k = \theta_{k-1} - \frac{1}{|\mathcal{D}_{k-1}|}\sum_{t\in\Delta\mathcal{D}_k}\mathcal{I}_{\mathsf{up,params}}(t)$.

In particular, when $\hat{\mathcal{M}}$ is non-convex, H_θ may contain negative eigenvalues and H_θ^{-1} does not exist. To cope with this, we follow [3] and add a small dampening coefficient to the matrix's diagonal to ensure H_θ^{-1} exists.

(1B) Incremental learning. Previous incremental strategies [67], [68] mainly focus on retaining the performance of $\hat{\mathcal{M}}$ on both old and new datasets, *i.e.,* $\mathcal{D}_{k-1}$ and $\mathcal{D}_k$. However, since $\mathcal{D}_k$ is cleaned from $\mathcal{D}_{k-1}$, we adopt a simple strategy that continuously fine-tunes $\hat{\mathcal{M}}$ with updated $\mathcal{D}_k$ in e epochs, where $\hat{\mathcal{M}}$ inherits the parameters from the model unlearning step. This strategy works well since the negative effects of dirty data in $\mathcal{D}_{k-1}$ have already been removed.

(2) Downstream model $\mathcal{M}$. We train model $\mathcal{M}$ on accumulated clean data $\mathcal{D}_{\mathsf{clean}}$ by means of (1) *identifying new clean data* and (2) *model refinement*, as follows.

Notations	Definitions
$\mathcal{R}$, $\mathcal{D}$, $\mathcal{A}_{\text{CR}}$	database schema, database, conflict resolution method
$\mathcal{D}_{\text{CR}}$ (resp. $\mathcal{D}_{\mathcal{M}}$)	database fixed by $\mathcal{A}_{\text{CR}}$ directly (resp. CR4ML with $\mathcal{A}_{\text{CR}}$)
$\mathcal{S}$, $\mathcal{T}$	influential attributes, influential tuples
$\mathcal{D}_{\text{clean}}$, $\Delta\mathcal{D}_{\text{clean}}$	already cleaned tuples, newly cleaned tuples
$\mathcal{D}_k$, $\mathcal{B}_k$, $\mathcal{C}_k$	database, training set, validation set in round k of CR4ML
$\mathcal{M}$, $\hat{\mathcal{M}}$	downstream model, assistant model
$L_k(t)$	dynamic loss for each tuple t trained with $\hat{\mathcal{M}}$
$\text{acc}(\mathcal{M}, \mathcal{D})$	accuracy of $\mathcal{M}$ trained (resp. evaluated) with subsets of $\mathcal{D}$
NI	noise injector
ite_{max}	the maximal round in genetic algorithm for identifying $\mathcal{S}$
m	the number of seed sets in genetic algorithm for identifying $\mathcal{S}$
$\mathcal{T}_{\text{max}}$	the largest number of $\mathcal{T}$ identified by Algorithm AlgIT
e	the number of epochs for incremental learning

TABLE V: Notation table

(2A) Identifying new clean data. The creator first identifies new clean data $\Delta\mathcal{D}_{\text{clean}}$ from $\mathcal{D}_k$ with the help of the assistant model $\hat{\mathcal{M}}$. To do this, we initialize $\Delta\mathcal{D}_{\text{clean}}$ to empty and compute the dynamic loss $L_k(t)$ for each tuple t in $\mathcal{D}_k$ [69], using the exponential moving average (EMA):

$$L_k(t) = \lambda \cdot l(\hat{\mathcal{M}}(x;\theta_k), y) + (1-\lambda)\cdot L_{k-1}(t),$$

where $t = (x, y) \in \mathcal{D}_k$, $\lambda \in [0, 1]$ is a discounting factor, and $L_k(t_i)$ is the accumulated loss of t from 0 to k epochs. Intuitively, the larger $L_k(t)$, the more likely t is dirty. For each tuple $t \in \mathcal{D}_k$, we monitor the dynamic loss $L_k(t)$ of $\hat{\mathcal{M}}$. Given thresholds η_1 and η_2, we add a tuple t into $\Delta\mathcal{D}_{\text{clean}}$ if one of the following conditions is satisfied:

- $L_{k-1}(t) - L_k(t) \geq \eta_1$, *i.e.,* $\mathcal{A}_{\text{CR}}$ has effectively enhanced t's quality and thus t can be considered as a clean tuple.
- $L_k(t) \leq \eta_2$, *i.e.,* no matter whether t has been cleaned by $\mathcal{A}_{\text{CR}}$ or not, it is likely to be clean due to its low $L_k(t)$.

(2B) Model refinement. CR4ML adopts Stochastic Gradient Descent (SGD) [70] for iterative model refinement, to leverage insights from both new clean data $\Delta\mathcal{D}_{\text{clean}}$ and the accumulated clean data $\mathcal{D}_{\text{clean}}$. We adapt SGD for continuous integration of clean data, such that the accuracy of $\mathcal{M}$ can be progressively improved with more clean data accumulated.

We apply full-gradient computation to $\mathcal{D}_{\text{clean}}$, denoted as $g_{\text{clean}}(\cdot)$, taking into account all data points in $\mathcal{D}_{\text{clean}}$:

$$g_{\text{clean}}(\theta_{k-1}) = \frac{1}{|\mathcal{D}_{\text{clean}}|}\sum_{(x,y)\in\mathcal{D}_{\text{clean}}} \nabla l(\mathcal{M}(x;\theta_{k-1}), y).$$

The idea is to ensure a continuous training of $\mathcal{M}$ by effectively incorporating the data cleaned after the critic (see below).

We calculate the gradient from the newly cleaned data $\Delta\mathcal{D}_{\text{clean}}$ after each round of critic, denoted by $g_{\Delta\text{clean}}(\cdot)$:

$$g_{\Delta\text{clean}}(\theta_{k-1}) = \frac{1}{|\Delta\mathcal{D}_{\text{clean}}|}\sum_{(x,y)\in\Delta\mathcal{D}_{\text{clean}}} \frac{\nabla l(\mathcal{M}(x;\theta_{k-1}), y)}{p(x,y)}.$$

Here $p(x, y)$ denotes the probability for tuple (x, y) in $\mathcal{D}_k$ to be picked to clean (see below). We use $p(x, y)$ to adjust any bias that may arise from non-uniform sampling or selective cleaning. By weighting the gradient contribution of each newly cleaned tuple based on the inverse of its selection probability, the gradient computation accurately reflects the entire dataset, and thus provides an unbiased foundation for the subsequent updates of model parameters.

We compute the parameter updates by taking into account the gradients from both data sources, which are weighted by their respective proportions in the entire dataset $\mathcal{D}_k$, as follows:

$$g(\theta_{k-1}) = \frac{|\mathcal{D}_{\text{clean}}|}{|\mathcal{D}_k|} g_{\text{clean}}(\theta_{k-1}) + \frac{|\mathcal{D}_k| - |\mathcal{D}_{\text{clean}}|}{|\mathcal{D}_k|} g_{\Delta\text{clean}}(\theta_{k-1}).$$

Intuitively, the weighting scheme balances the influence of (a) the stable, already cleaned $\mathcal{D}_{\text{clean}}$, and (b) the dynamic, newly cleaned data sampled from $\mathcal{D}_k \setminus \mathcal{D}_{\text{clean}}$ on the model evolution.

Using the combined gradient g, we update $\mathcal{M}$'s parameters:

$$\theta_k = \theta_{k-1} - \zeta_k g(\theta_{k-1}),$$

where ζ_k denotes the learning rate for round k, modulating the extent of each update in response to the calculated gradient.

To ensure the convergence of SGD, the learning rate ζ_k is adjusted at each round following the Robbins-Monro conditions [70], *i.e.,* on the one hand, the rate ζ_k does not decrease too rapidly to allow sufficient exploration of the parameter space, and on the other hand, the rate ζ_k decreases fast enough to ensure convergence to an optimum.

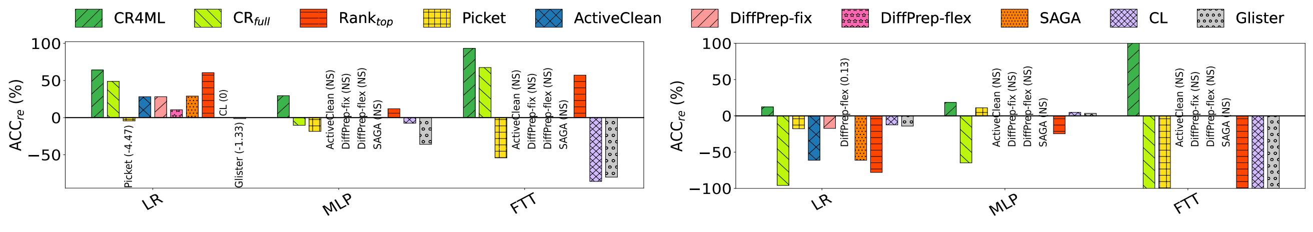
Finally we accumulate $\mathcal{D}_{\text{clean}}$, *i.e.,* $\mathcal{D}_{\text{clean}} = \Delta\mathcal{D}_{\text{clean}} \cup \mathcal{D}_{\text{clean}}$.

C. Closed form estimation of parameter changes

For each $t = (x, y)$, where x and y are the attributes and label of t, respectively, we can upweight t by an infinitesimal amount ϵ, such that removing (resp. adding) t from $\mathcal{D}$ is the same as upweighting its old version t_{old} (resp. t_{new}) by $-\epsilon = -\frac{1}{|\mathcal{D}_k|}$ (resp. ϵ). As shown in [3], the change of $\hat{\mathcal{M}}$'s parameters, *i.e.,* $\Delta\theta_t = \theta_{\text{new}} - \theta_{\text{old}}$, by cleaning t via $\mathcal{A}_{\text{CR}}$ can be estimated in a closed form: $-H_\theta^{-1}(\nabla_\theta l(t_{\text{new}}, \theta) - \nabla_\theta l(t_{\text{old}}, \theta))$, where $\theta = \theta_{\text{old}}$, ∇_θ is the partial derivative of θ *w.r.t.* ϵ, H_θ is the Hessian Matrix, and $l(t, \theta)$ is the loss of tuple t.

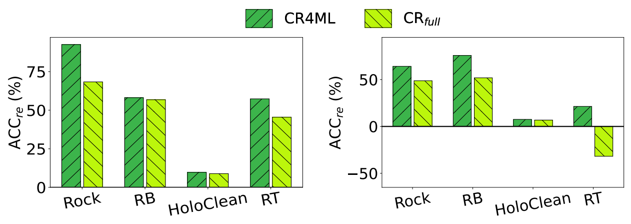
D. Datasets

- **Adult:** It contains demographic information of individuals. It is to determine whether a person's income exceeds 50k per year. The class labels are $\leq$50k or $>$50k.
- **Nursery:** It includes the family and financial circumstances of children. The prediction is to assign a ranking label to each child, including not recommend, strongly recommend, priority, and top priority.
- **Bank:** It contains the demographic, financial and social information of bank clients. The goal is to predict whether a client will apply for a term deposit (yes or no).
- **Default:** It records default payments of customers in Taiwan from April 2005 to September 2005. The ML prediction is to estimate whether a customer will default payments in the next month. The class labels are Yes or No.
- **German:** It contains financial information of bank account holders. The prediction is to determine the creditworthiness of a holder. The class labels are Good or Bad.
- **RoadSafety:** It contains detailed information on road traffic incidents in the UK from 1979 to 2015. The prediction task is to determine the severity of an accident. The class labels include slight, serious, and fatal.



(a) German (b) Nursery

Fig. 9: Datasets: varying $\mathcal{M}$ (ACC$_{\text{re}}$)



(a) Bank (b) German

Fig. 10: Datasets: varying $\mathcal{A}_{\text{CR}}$ (ACC$_{\text{re}}$)

E. More experimental results

Figure 9 (resp. 10) evaluates the performance of CR4ML with different ML models (resp. CR methods) on more datasets. The trends are consistent with Figure 6(a) (resp. 6(b)), and thus we omit further discussion here.

F. Proof of Theorem 1

Theorem 1: *Problem* IA *is NP-hard.* □

Proof: We prove the NP-hardness of problem IA by reduction from the 3-SAT problem, which is NP-complete (cf. [101]). The 3-SAT problem is to decide, given a Boolean formula ϕ in the conjunctive normal form with n variables and m clauses, each containing three literals (a variable x or its negation $\neg x$), whether there exists a truth value assignment to the variables that makes ϕ true. The reduction from a 3-SAT instance to an IA instance is given as follows.

(1) Schema $\mathcal{R}$: We use a relation schema $(A_1, A_2, \ldots, A_{2n}, \text{id}, \text{label})$, where for all $i \in [1, 2n]$, A_i is a Boolean attribute, id is in the range $[1, m]$, and label is in the range $[1, 2m]$.

(2) Dataset $D_k = (\mathcal{B}_k, \mathcal{C}_k)$: Construct relations $\mathcal{B}_k$ and $\mathcal{C}_k$ of the schema above, each having m tuples. For tuples t in $\mathcal{B}_k$, $t[\text{id}]$ takes values from 1 to m, respectively, $t[A_i] = 0$ $(i \in [1, n])$, $t[A_j] = 1$ $(j \in [n+1, 2n])$, and $t[\text{label}] = t[\text{id}] + m$. Intuitively, we use attributes $A_1, A_2, \ldots, A_n$ of tuples in $\mathcal{B}_k$ to encode the n variables of ϕ.

For tuples s in $\mathcal{C}_k$, their id attributes are instantiated in the same way as their counterparts in $\mathcal{B}_k$. However, we set $s[A_i] = 0$ $(i \in [1, 2n])$, and $s[\text{label}] = s[\text{id}] + e$, where e is the number of satisfied clauses in ϕ when all variables are set false.

Intuitively, we will use a CR method $\mathcal{A}_{\text{CR}}$ to simulate the truth assignment of ϕ, an ML model $\hat{\mathcal{M}}$ to check the satisfaction of the clauses of ϕ, and the correspondence between attributes A_i and A_{n+i} $(i \in [1, n])$ in $\mathcal{B}_k$ to identify influential attributes.

(3) CR method $\mathcal{A}_{\text{CR}}$: We use a rule-based method $\mathcal{A}_{\text{CR}}$ to adjust attributes in $\mathcal{B}_k$. It employs a single rule that makes use of a set $\mathcal{S}$ of influential attributes. The rule states that for each tuple $t \in \mathcal{B}_k$ and each $i \in [1, n]$, if the attribute A_i is identified as an influential attribute, then $t[A_i]$ takes the value of $t[A_{i+n}]$; while for $i > n$, $t[A_i]$ remains unchanged. That is, $\mathcal{A}_{\text{CR}}$ aligns correlated attributes such that the attribute pair A_i and A_{i+n} have the same value.

(4) Model $\hat{\mathcal{M}}$: We use a classification tree-based (regression) model $\hat{\mathcal{M}}$ to evaluate clause satisfaction based on the attribute values in $\mathcal{D}_k$. It consists of m decision trees, each for a clause C_i in ϕ. The i-th tree yields $w_i = 1$ if clause C_i is satisfied, otherwise $w_i = 0$. Figure 11 shows an example decision tree for the clause $C_i = (x_1 \vee \neg x_3 \vee x_5)$.

Specifically, the model $\hat{\mathcal{M}}$ predicts labels for each tuple $t \in D_k$ using the regression function $f(t) = \sum_{i=1}^{m} w_i + t[\text{id}] + w_t$, where w_t is a trainable parameter, making f differentiable on the input tuple t. The label is predicated as $\lfloor f(t) \rfloor$.

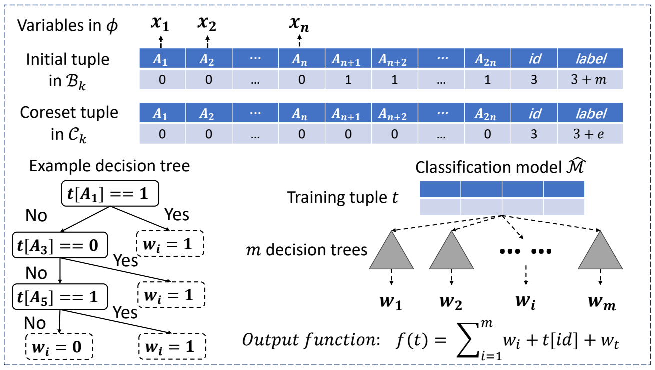
Observe that before applying $\mathcal{A}_{\text{CR}}$ to $\mathcal{B}_k$, for any tuple $t \in \mathcal{B}_k$, the function $f(t)$ is evaluated as $e + t[\text{id}] + w_t$. Since $t[\text{label}] = t[\text{id}] + m$ for tuples $t \in \mathcal{B}_k$, the training process adjusts w_t to $m - e$ such that $f(t) = t[\text{label}]$. When $\hat{\mathcal{M}}$ predicts any tuple $s \in \mathcal{C}_k$ in the validation process, $f(s) = e + s[\text{id}] + m - e$, which is not equal to $s[\text{label}] = s[\text{id}] + e$; as a consequence, initially $\text{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) = 0$.

(5) Parameter δ: Define $\delta = 1$, which is the maximum possible improvement under an initial $\text{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) = 0$.

The reduction can be obviously constructed in PTIME. We next show that there exists an attribute set $\mathcal{S}$ that makes $\text{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) = 1$ if and only if the 3-SAT instance ϕ is satisfiable.

$\Rightarrow$ First, we assume that there exists a set $\mathcal{S}$ such that $\mathcal{A}_{CR}$ on $\mathcal{S}$ makes $\text{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{S}}) \geq \text{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) + 1$. More specifically, the application of $\mathcal{A}_{CR}$ ensures that for any tuple t in $\mathcal{B}_k^{\mathcal{S}}$ and for any attribute $A_i \in \mathcal{S}$, $t[A_i] = 1$; and for any attribute $A_j \notin \mathcal{S}$, $t[A_j] = 0$. In fact, $\text{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{S}}) \geq \text{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) + 1$ implies that $\hat{\mathcal{M}}$ classifies all tuples in $\mathcal{C}_k$ correctly. Given $f(s) = \sum_{i=1}^{m} w_i + s[\text{id}] + w_t$ for each $s \in \mathcal{C}_k$ and $s[\text{label}] = e + s[\text{id}]$, it necessitates $w_t = 0$. Recall that during training on tuples t in $\mathcal{B}_k^{\mathcal{S}}$ using mean squared error, the optimal value for w_t is determined by the formula $w_t = \frac{\sum_{t \in \mathcal{B}_k^{\mathcal{S}}} \left(t[\text{label}] - t[\text{id}] - \sum_{i=1}^{m} w_i\right)}{m}$. If this optimal w_t is found to be 0, it implies that $\sum_{i=1}^{m} w_i = m$, indicating that each of the m clauses has been satisfied by the current attribute settings in $\mathcal{B}_k^{\mathcal{S}}$.

We give the truth assignment in ϕ as follows. For each tuple t in $\mathcal{B}_k^{\mathcal{S}}$ and $i \in [1, n]$, if $t[A_i] = 1$, then variable x_i is assigned



Variables in ϕ
x_1 x_2 x_n
Initial tuple in $\mathcal{B}_k$
A_1 A_2 … A_n A_{n+1} A_{n+2} … A_{2n} id label
0 0 … 0 1 1 … 1 3 3 + m
Coreset tuple in $\mathcal{C}_k$
A_1 A_2 … A_n A_{n+1} A_{n+2} … A_{2n} id label
0 0 … 0 0 0 … 0 3 3 + e
Example decision tree
t[A1] == 1
No
Yes
t[A3] == 0
wi = 1
Yes
No
t[A5] == 1
wi = 1
Yes
No
wi = 0
wi = 1
Classification model $\hat{\mathcal{M}}$
Training tuple t
m decision trees
w_1 w_2 w_i w_m
Output function: $f(t) = \sum_{i=1}^{m} w_i + t[id] + w_t$

Fig. 11: Problem IA reduction example

true. Conversely, if $t[A_i] = 0$, then x_i is set to false.

One can verify that this truth assignment satisfies ϕ. Observe that the truth assignment is determined by $\hat{\mathcal{M}}$'s decision trees, each aligned with a specific clause C_i from ϕ. A tree yields $w_i = 1$ iff for any tuple $t \in \mathcal{B}_k^{\mathcal{S}}$ and $i \in [1, n]$, at least one $t[A_i]$ value meets the decision criterion, equivalently satisfying C_i by ensuring a literal's truth. This leads to $\sum_{i=1}^{m} w_i = m$, with $w_i = 1$ for all trees, which verifies ϕ's satisfiability via the truth assignment derived from $\mathcal{S}$.

$\Leftarrow$ Conversely, we show that the existence of a satisfying truth assignment for ϕ guarantees the existence of a set $\mathcal{S}$ of attributes such that $\mathcal{A}_{\mathsf{CR}}$ on $\mathcal{S}$ in $\mathcal{B}_k$ ensures $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{S}}) \geq \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) + 1$. Given a satisfying truth assignment for ϕ, the set $\mathcal{S}$ is identified as follows: For each variable x_i in ϕ, if x_i is true in the truth assignment, then we include the attribute A_i in the influential attribute set $\mathcal{S}$. Then we perform $\mathcal{A}_{\mathsf{CR}}$ on $\mathcal{S}$ to obtain $\mathcal{D}_k^{\mathcal{S}}$, where for each $t \in \mathcal{B}_k^{\mathcal{S}}$ and $A_i \in \mathcal{S}$, $t[A_i] = 1$, while $t[A_j] = 0$ for $A_j \notin \mathcal{S}$. Suppose by contradiction that $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{S}}) < \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) + 1$, *i.e.,* some $\mathcal{C}_k$ tuples are incorrectly classified. Given $f(s) = e + s[\mathsf{id}] + w_t$ for $s \in \mathcal{C}_k$ where $s[\mathsf{label}] = e + s[\mathsf{id}]$, w_t cannot be 0 in order to produce incorrect predictions. A non-zero w_t contradicts the premise that ϕ is satisfiable. Indeed, during the training, for a tuple $t \in \mathcal{B}_k^{\mathcal{S}}$ with $t[\mathsf{label}] = t[\mathsf{id}] + m$, $\hat{\mathcal{M}}$ outputs $f(t) = m + t[\mathsf{id}] + w_t$; this indicates that w_t must be 0 to train an accurate classifier $\hat{\mathcal{M}}$. □

G. *Proof of Theorem 2*

Theorem 2: *Problem* IT *is NP-hard.* □

Proof: The NP-hardness of IT is also shown by reduction from the 3-SAT problem. The reduction is given as follows. Consider a 3-SAT instance ϕ that has n variables and m conjunctive clauses.

(1) Schema $\mathcal{R}$: We use a ternary schema $(A_1, \mathsf{id}, \mathsf{label})$, where A_1 is a Boolean, id is in the range $[1, n]$, and label is in the range $[1, n + m]$.

(2) Dataset $\mathcal{D}_k = (\mathcal{B}_k, \mathcal{C}_k)$. We create relations $\mathcal{B}_k$ and $\mathcal{C}_k$ of the schema above, each having n tuples. For tuples t in $\mathcal{B}_k$, $t[\mathsf{id}]$ takes values from 1 to n, respectively, $t[A_1] = 0$, and $t[\mathsf{label}] = t[\mathsf{id}] + m$. Intuitively, we use the n tuples in $\mathcal{B}_k$ to encode the n variables of ϕ.

For tuples s in $\mathcal{C}_k$, their id attributes are instantiated in the same way as their counterparts in $\mathcal{B}_k$. However, we set $s[A_1] = 1$, and $s[\mathsf{label}] = s[\mathsf{id}] + e$, where e is the number of satisfied clauses in ϕ when all variables in the 3-SAT instance ϕ are set true.

As will be seen shortly, we will use a CR method $\mathcal{A}_{\mathsf{CR}}$ to simulate the truth assignment of ϕ, an ML model $\hat{\mathcal{M}}$ to check the satisfaction of the clauses of ϕ, and the pairwise correspondence between tuples in $\mathcal{B}_k$ and those in $\mathcal{C}_k$ to identify influential tuples.

(3) CR method $\mathcal{A}_{\mathsf{CR}}$: We use $\mathcal{A}_{\mathsf{CR}}$ that, for each tuple t in $\mathcal{B}_k$, if t is identified as influential, then $t[A_1]$ takes the value of $s[A_1]$, where s is the tuple in $\mathcal{C}_k$ such that $t[\mathsf{id}] = s[\mathsf{id}]$. That is, $\mathcal{A}_{\mathsf{CR}}$ ensures that for any pair $t \in \mathcal{B}_k$ and $s \in \mathcal{C}_k$ of tuples, if they have the same id, then the two must have the same A_1-attribute value.

(4) Model $\hat{\mathcal{M}}$: Design a classification tree-based (regression) model $\hat{\mathcal{M}}$ that uses a decision tree for each 3-SAT clause, where each tree decides whether its clause is satisfied based on the A_1 values. The input to the model regression function is an array of A_1 values in the size of n, arranged in order by the tuple identifiers; that is, $\hat{\mathcal{M}}$ processes a batch d of n tuples simultaneously. The trees yield $w_i = 1$ for satisfied clauses C_i, 0 otherwise. Figure 12 shows an example for the clause $C_i = (x_1 \vee \neg x_3 \vee x_5)$, where $d[i]$ indicates the i-th A_1 value of the input d. The regression function $f(d) = \sum_{i=1}^{m} w_i + w_t$ aggregates the trees' outputs and $\hat{\mathcal{M}}$ predicts the label of each tuple t within the batch by $f(d) + t[\mathsf{id}]$, where w_t is the trainable parameter, making f differentiable on the input d.

Observe that before applying $\mathcal{A}_{\mathsf{CR}}$ on $\mathcal{B}_k$, for a tuple $t \in \mathcal{B}_k$ with label $t[\mathsf{id}] + m$, its predicted label is $f(d) + t[\mathsf{id}] = e' + w_t + t[\mathsf{id}]$. Here e' is the number of satisfied clauses of ϕ under a truth assignment that assigns false to all the variables of ϕ. Based on the mean squared error, the training adjusts w_t to $m - e' \neq 0$. When $\hat{\mathcal{M}}$ predicts any tuple $s \in \mathcal{C}_k$ by batching n tuples in $\mathcal{C}_k$ into an input d', the predicted label of s is $f(d') + s[\mathsf{id}] = e + m - e' + s[\mathsf{id}]$, which is not equal to $s[\mathsf{id}] + e$; as a result, we have that $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) = 0$.

(5) Parameter δ: Choose $\delta = 1$ as the target for maximum accuracy improvement under an initial $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) = 0$.

The reduction can be constructed in PTIME. We next show that there exists a set $\mathcal{T} \subseteq \mathcal{B}_k$ of influential tuples such that $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{T}}) \geq \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) + 1$ iff ϕ is satisfiable.

$\Rightarrow$ First, assume that there exists a set $\mathcal{T}$ of tuples in $\mathcal{B}_k$ such that $\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{T}}) \geq \mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) + 1$. Then the application of $\mathcal{A}_{\mathsf{CR}}$ ensures that for any tuple t in $\mathcal{T}$, if there exists a tuple s in $\mathcal{C}_k$ such that $t[\mathsf{id}] = s[\mathsf{id}]$, then $t[A_1]$ takes the value of $s[A_1]$. Since $\hat{\mathcal{M}}$ can correctly output all labels in $\mathcal{C}_k$, the parameter w_t must be 0. During training with mean squared error, the optimal w_t equals $\frac{\sum_{\forall t \in \mathcal{B}_k^{\mathcal{T}}} \left(t[\mathsf{label}] - t[\mathsf{id}] - \sum_{i=1}^{m} w_i\right)}{n} = \frac{\sum_{\forall t \in \mathcal{B}_k^{\mathcal{T}}} \left(m - \sum_{i=1}^{m} w_i\right)}{n}$. When $w_t = 0$, $\sum_{i=1}^{m} w_i = m$; this leads to a satisfying assignment for all m clauses.

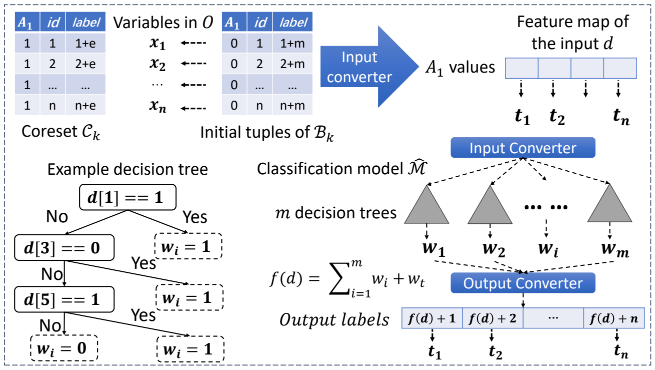


Fig. 12: Problem IT reduction example

Specifically, the truth assignment is defined as follows. For each tuple t in $\mathcal{B}_k^{\mathcal{T}}$ with t[id]=i, if $t[A_1]$=1, then x_i is assigned true. Conversely, if $t[A_1]$ remains 0, then x_i is set to false.

One can verify that this truth assignment satisfies ϕ. Indeed, observe that the truth assignment is determined by $\hat{\mathcal{M}}$'s decision trees, each corresponding to a clause C_i from ϕ. A tree yields $w_i = 1$ iff at least one A_1 value in the input d meets the decision criterion; in other words, it satisfies clause C_i. Since $\sum_{i=1}^{m} w_i = m$, all the clauses of ϕ are satisfied by the truth assignment.

$\Leftarrow$ Conversely, assume that ϕ is satisfied by a certain truth assignment for $x_1, \ldots, x_n$. Then we identify a set $\mathcal{T}$ of influential tuples in $\mathcal{B}_k$ as follows. If a variable x_i is true, we add $t \in \mathcal{B}_k$ to $\mathcal{T}$ if $t[\text{id}] = i$. Then we apply $\mathcal{A}_{\text{CR}}$ on $\mathcal{T}$ to obtain $\mathcal{D}_k^{\mathcal{T}}$, such that $t[A_1] = 1$ for all tuples t in $\mathcal{T}$. We next show that $\mathcal{T}$ is indeed the set of influential tuples desired. Suppose by contradiction that $\text{acc}(\hat{\mathcal{M}}, \mathcal{D}_k^{\mathcal{T}}) < \text{acc}(\hat{\mathcal{M}}, \mathcal{D}_k) + 1$. Then $\hat{\mathcal{M}}$ fails to predict all $\mathcal{C}_k$ labels correctly. Indeed, when predicting labels of a batch d' of n tuples in $\mathcal{C}_k$, since $f(d') = e + w_t$, the predicated label of each tuple $s \in \mathcal{C}_k$ is $f(d') + s[\text{id}] = e + w_t + s[\text{id}]$. Meanwhile, the truth label of tuple s is $s[\text{id}] + e$, indicating $w_t \neq 0$. However, if ϕ is satisfied by a truth assignment, consider the batch d that consists of all tuples $t \in \mathcal{B}_k^{\mathcal{T}}$; then $\hat{\mathcal{M}}$ predicts $t[\text{label}]$ as $f(d) + t[\text{id}] = m + w_t + t[\text{id}]$, and $t[\text{label}]$ is $t[\text{id}] + m$; as a result, during the training phase, the optimal w_t must be adjusted to 0; this contradicts that $w_t \neq 0$. □

H. Proof of Theorem 3

Theorem 3: *Given an α-accurate $\mathcal{A}_{\text{CR}}$,* CR4ML *guarantees to terminate and returns (a) a cleaned dataset $\mathcal{D}_{\text{clean}}$ as $\mathcal{D}_{\mathcal{M}}$, and (b) model $\mathcal{M}$ trained with $\mathcal{D}_{\text{clean}}$ such that* $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{clean}}) > \text{acc}(\mathcal{M}, \mathcal{D})$. □

Proof: We start with three observations.

- (F1). The CR method $\mathcal{A}_{\text{CR}}$ is α-accurate ($\alpha \in (0.5, 1]$), *i.e.,* it correctly fixes erroneous data with probability α. We assume that the data correctly fixed by $\mathcal{A}_{\text{CR}}$ raises no error signals by the ML model $\mathcal{M}$ as in practice, and hence remains unchanged. Intuitively, iteratively applying $\mathcal{A}_{\text{CR}}$ approaches the "perfect method" Oracle, which correctly fixes all errors in the data.
- (F2). The Stochastic Gradient Descent (SGD) process converges on unbiased gradient estimates with a step size sequence $\{\gamma_t\}$ under the Robbins-Monro conditions, as verified by [77], [78].
- (F3). Training $\mathcal{M}$ with clean data makes a closer approximation of the true data distribution and better test accuracy than training with dirty data, as observed in [45], [75], [76].

Based on the observations, the proof consists of three steps.

- (S1) **Iterative data cleaning**: For part (a) of Theorem 3, we show that iterative applications of $\mathcal{A}_{\text{CR}}$ reduce the error rate in the dataset (F1) such that $\mathcal{D}_{\mathcal{M}}$ obtained more accurately reflects the true data distribution and improves the accuracy of $\mathcal{M}$ (F3).
- (S2) **Convergence by unbiased gradient estimation**: For part (b) of the theorem, we prove the convergence of model refinement with progressively cleaned data by (F2); that is, CR4ML ensures unbiased gradient estimation in SGD when incorporating gradients from both the accumulated and newly cleaned datasets. CR4ML approaches this by adjusting gradients based on the probability of the data points being cleaned.
- (S3) **Accuracy improvement**: Also for part (b) of the theorem, we show that $\mathcal{M}$ trained with $\mathcal{D}_{\text{clean}}$ becomes more accurate. We first examine model convergence results in both convex and non-convex optimization scenarios. We then explore the impact of CR4ML's cleaning on the accuracy of $\mathcal{M}$ in terms of dynamic loss reduction. We show that the selection criteria of $\Delta\mathcal{D}_{\text{clean}}$ yield lower dynamic losses, thereby demonstrating the improved model accuracy of $\mathcal{M}$ trained with $\mathcal{D}_{\mathcal{M}}$.

Below we provide the details of each step in the proof.

(S1) **Iterative data cleaning**: Given an initial dataset $\mathcal{D}$ with an error rate e_0, we inductively show that iterative applications of α-accurate $\mathcal{A}_{\text{CR}}$ can reduce the error rate below a threshold ϵ in k iterations, where k is determined by α, ϵ and $\mathbb{E}[p(x, y)]$ (see below).

For a dataset $\mathcal{D}_k$ in the k-th iteration, the likelihood of selecting its erroneous data points for cleaning is:

$$\mathbb{E}[p(x, y)] = \frac{1}{|\mathcal{D}_k|} \sum_{(x,y) \in \mathcal{D}_k} p(x, y),$$

where $|\mathcal{D}_k|$ is the size of $\mathcal{D}_k$, and $p(x, y)$ is determined by AlgIA and AlgIT. Assuming that the error rate is e_k after k iterations, then by incorporating $\mathbb{E}[p(x, y)]$, the error rate e_{k+1} is

$$e_{k+1} = e_k \cdot (1 - \alpha \cdot \mathbb{E}[p(x, y)]),$$

where α is the accuracy of $\mathcal{A}_{\text{CR}}$. To satisfy $e_k < \epsilon$, the equation:

$$e_0 \cdot (1 - \alpha \cdot \mathbb{E}[p(x, y)])^k < \epsilon$$

determines the necessary number k of iterations. That is,

$$k > \frac{\log(\epsilon / e_0)}{\log(1 - \alpha \cdot \mathbb{E}[p(x, y)])}.$$

This shows the connection between the number k of iterations and (a) the average error selection probability $\mathbb{E}[p(x,y)]$, and (b) the accuracy α of CR method $\mathcal{A}_{\mathsf{CR}}$. For example, when $\alpha = 0.6$ and $\mathbb{E}[p(x,y)] = 0.6$, to reduce $e_0 = 0.1$ below $\epsilon = 0.01$, it typically requires about 4 iterations. That is, we need k rounds in the creator-critic framework of CR4ML to get the cleaned dataset $\mathcal{D}_{\mathcal{M}}$. In practice, CR4ML typically converges in much fewer rounds.

(S2) **Convergence via unbiased gradient estimation**: CR4ML converges when the gradient estimation during the training process is unbiased. The gradient from the already cleaned data $\mathcal{D}_{\mathsf{clean}}$ in the k-th round, denoted by $g_{\mathsf{clean}}(\theta_{k-1})$, is computed as follows:

$$g_{\mathsf{clean}}(\theta_{k-1}) = \frac{1}{|\mathcal{D}_{\mathsf{clean}}|} \sum_{(x,y)\in\mathcal{D}_{\mathsf{clean}}} \nabla l(\mathcal{M}(x;\theta_{k-1}), y),$$

where $\nabla l(\mathcal{M}(x;\theta_{k-1}), y)$ denotes the gradient of the loss function l *w.r.t.* model $\mathcal{M}$'s parameters θ_{k-1} in the (k-1)-th round, evaluated at the predicted label $\mathcal{M}(x;\theta_{k-1})$ and the truth label y of a data point (x,y). That is, the continuous training of $\mathcal{M}$ integrates all tuples in $\mathcal{D}_{\mathsf{clean}}$ and is unbiased by the use of the full-gradient method.

Denote by $g_{\Delta\mathsf{clean}}(\theta_{k-1})$ the gradient from the newly cleaned data $\Delta\mathcal{D}_{\mathsf{clean}}$ in the k-th round. Given probability $p(x,y)$ for selecting data to clean, CR4ML conducts gradient estimation as follows:

$$g_{\Delta\mathsf{clean}}(\theta_{k-1}) = \frac{1}{|\Delta\mathcal{D}_{\mathsf{clean}}|} \sum_{(x,y)\in\Delta\mathcal{D}_{\mathsf{clean}}} \frac{\nabla l(\mathcal{M}(x;\theta_{k-1}), y)}{p(x,y)}.$$

To show the unbiasedness of $g_{\Delta\mathsf{clean}}(\theta_{k-1})$, we consider its expectation *w.r.t.* the data selection process for cleaning:

$$\mathbb{E}[g_{\Delta\mathsf{clean}}(\theta_{k-1})] = \mathbb{E}\left[\frac{1}{|\Delta\mathcal{D}_{\mathsf{clean}}|} \sum_{(x,y)\in\Delta\mathcal{D}_{\mathsf{clean}}} \frac{\nabla l(\mathcal{M}(x;\theta_{k-1}), y)}{p(x,y)}\right].$$

Given that each data point (x,y) is independently chosen from $\mathcal{D}_k \setminus \mathcal{D}_{\mathsf{clean}}$ with probability $p(x,y)$, the expected contribution of the gradient for any selected data is adjusted by its selection probability. This ensures that the overall expectation reflects the true gradient contribution from the subset $\mathcal{D}_k \setminus \mathcal{D}_{\mathsf{clean}}$:

$$\mathbb{E}\left[\frac{\nabla l(\mathcal{M}(x;\theta_{k-1}), y)}{p(x,y)}\right] = \sum_{(x,y)\in\mathcal{D}_k\setminus\mathcal{D}_{\mathsf{clean}}} \nabla l(\mathcal{M}(x;\theta_{k-1}), y).$$

In light of this, the combined gradient $g(\theta_{k-1})$ for model parameter update integrates gradients from both sources, appropriately weighted by their proportions in the total dataset $\mathcal{D}_k$:

$$g(\theta_{k-1}) = \frac{|\mathcal{D}_{\mathsf{clean}}|}{|\mathcal{D}_k|} g_{\mathsf{clean}}(\theta_{k-1}) + \frac{|\mathcal{D}_k| - |\mathcal{D}_{\mathsf{clean}}|}{|\mathcal{D}_k|} g_{\Delta\mathsf{clean}}(\theta_{k-1}).$$

Given the unbiasedness of $g_{\mathsf{clean}}(\theta_{k-1})$ and the proven unbiased expectation of $g_{\Delta\mathsf{clean}}(\theta_{k-1})$, the expectation of the combined gradient $E[g(\theta_{k-1})]$ accurately reflects the true gradient of the entire dataset $\mathcal{D}_k$ (note that $|\mathcal{D}_k| = |\mathcal{D}_{\mathsf{clean}}| = |\mathcal{D}_{\mathcal{M}}|$ in the end). This ensures unbiased gradient estimation and model parameter updates:

$$\mathbb{E}[g(\theta_{k-1})] = \frac{1}{|\mathcal{D}_{\mathcal{M}}|} \sum_{(x,y)\in\mathcal{D}_{\mathcal{M}}} \nabla l(\mathcal{M}(x;\theta_{k-1}), y).$$

Consequently, model parameters are updated according to:

$$\theta_k = \theta_{k-1} - \zeta_k g(\theta_{k-1}),$$

where the learning rate ζ_k satisfies Robbins-Monro conditions. From this it follows CR4ML's convergence:

$$\sum_{k=1}^{\infty} \zeta_k = \infty, \quad \sum_{k=1}^{\infty} \zeta_k^2 < \infty.$$

That is, CR4ML ensures stable convergence in model training and data processing of the creator-critic process.

(S3) **Accuracy improvement**: We next show that when CR4ML converges, $\mathcal{M}$ is more accurate than $\mathcal{M}$ trained with the original dataset $\mathcal{D}$, given that $\mathcal{M}$ is trained with progressively cleaned datasets. We prove that the process reduces dynamic loss by considering convex and non-convex optimization cases.

Convex optimization. In convex optimization scenarios, CR4ML ensures that model $\mathcal{M}$ converges at the global optimum θ^* via gradient descent. Employing unbiased gradient updates $g(\theta_{k-1})$, $\mathcal{M}$ iteratively refines its parameters with the cleaned data in $\mathcal{D}_{\mathsf{clean}}$. As more data is cleaned and integrated into the training process, $\mathcal{M}$ gradually approaches θ^*, demonstrating convergence as:

$$\lim_{k\to\infty} \theta_k = \theta^*.$$

Non-convex optimization. In non-convex optimization, gradient descent on $\mathcal{M}$ typically converges at local optima or saddle points. However, by (F3), CR4ML enhances the exploration of the loss function's landscape by progressively incorporating cleaned data during training. This increases the chances of finding lower local minima, and improves the accuracy of $\mathcal{M}$ trained with uncleaned data in non-convex settings. We show this by dynamic loss analysis.

Given a dataset $\mathcal{D}$, let $D_{\mathsf{clean}}^{(k)}$ and $\Delta\mathcal{D}_{\mathsf{clean}}^{(k)}$ denote the dataset D_{clean} and $\Delta\mathcal{D}_{\mathsf{clean}}$ after k rounds, respectively. The conditions for a data point t to be added to $\Delta\mathcal{D}_{\mathsf{clean}}^{(k)}$ are either $L_{k-1}(t) - L_k(t) \geq \eta_1$ or $L_k(t) \leq \eta_2$, where $L_k(t)$ is the dynamic loss of t in round k. The total dynamic loss of $\mathcal{M}$ trained with $\mathcal{D}_{\mathsf{clean}}^{(k)}$ at the k-th round is defined as:

$$\mathcal{L}_{\mathsf{clean}}^{(k)} = \sum_{t\in\mathcal{D}_{\mathsf{clean}}^{(k)}} L_k(t),$$

while the total dynamic loss of $\mathcal{M}$ trained with $\mathcal{D}$ is:

$$\mathcal{L}_{\mathcal{D}}^{(k)} = \sum_{t\in\mathcal{D}} L_k(t).$$

In the dynamic loss analysis, $\mathcal{D}_{\mathsf{clean}}^{(k)} = \bigcup_{i=0}^{k-1} \Delta\mathcal{D}_{\mathsf{clean}}^{(i)}$, where $\Delta\mathcal{D}_{\mathsf{clean}}^{(i)}$ consists of tuples t categorized by their original state in dataset $\mathcal{D}$, either clean or dirty, processed in the i-th round.

Intuitively, inherently clean tuples are incorporated into $\Delta\mathcal{D}_{\mathsf{clean}}^{(k)}$ when they meet the condition $L_k(t) \leq \eta_2$, due to their inherent cleanliness and alignment with the model's accuracy criteria, thus resulting in a low dynamic loss. Their impact on the dynamic loss $\mathcal{L}_{\mathsf{clean}}^{(k)}$ aligns with their effect on the dynamic loss $\mathcal{L}_{\mathcal{D}}^{(k)}$, since their contribution to the model's learning is both stable and predictable.

On the other hand, newly-cleaned dirty tuples are selected for $\Delta\mathcal{D}^{(k)}_{\text{clean}}$ when they satisfy $L_{k-1}(t) - L_k(t) \geq \eta_1$, indicating a significant reduction in the dynamic loss by $\mathcal{A}_{\text{CR}}$. Such tuples decrease $\mathcal{L}^{(k)}_{\text{clean}}$ more substantially compared to $\mathcal{L}^{(k)}_{\mathcal{D}}$. Denote the quantity of such tuples by d. Its reduction in dynamic loss is quantified as:

$$\Delta\mathcal{L}^{(k)}_{\text{clean}} = \sum_{i=1}^{k} \sum_{t \in \Delta\mathcal{D}^{(i)}_{\text{clean}}} (L_{i-1}(t) - L_i(t)) \geq \eta_1 \cdot d \geq 0.$$

Therefore, suppose that CR4ML terminates at round k. Then we have the following, taking into account the reduction in cleaning loss and the contribution of inherently clean data:

$$\mathcal{L}^{(k)}_{\text{clean}} + \Delta\mathcal{L}^{(k)}_{\text{clean}} \leq \mathcal{L}^{(k)}_{\mathcal{D}}.$$

Thus $\mathcal{L}^{(k)}_{\text{clean}} \leq \mathcal{L}^{(k)}_{\mathcal{D}} - \eta_1 \cdot d$. From this it follows that $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{clean}}) > \text{acc}(\mathcal{M}, \mathcal{D})$, *i.e.,* part (b) of Theorem 3 holds. □

I. *Proof of Theorem 4*

Theorem 4: *Given an α-accurate $\mathcal{A}_{\text{CR}}$ with $\alpha > 0.5$, CR4ML achieves an expected submodular improvement in model accuracy. The expected accuracy $\mathbb{E}(\text{acc}(\mathcal{M}, \mathcal{D}_{\text{clean}}))$ of the model $\mathcal{M}$ trained on cleaned data $\mathcal{D}_{\text{clean}}$ is guaranteed to reach at least 63% of the optimal expected accuracy* □

Proof: We develop the proof under the following assumptions:

- (A1) **Data consistency of validation set $\mathcal{C}_k$**: As outlined in Section III, we assume that in each iteration, the $\mathcal{C}_k$ is a representative subset of the cleaned dataset $\mathcal{D}_{\text{clean}}$, sharing the same underlying data distribution. This consistency is supported by the α-accuracy of $\mathcal{A}_{\text{CR}}$ in providing more cleaned data and by the coreset selection strategy [69].
- (A2) **Stability of model accuracy gain**: We assume that the incremental accuracy gain of the ML models remains stable across iterations. This means that as influential attributes and tuples are progressively cleaned, the marginal improvement in accuracy does not fluctuate significantly due to training randomness. This assumption is supported by studies on the convergence and stability of Stochastic Gradient Descent (SGD) [77], [78].
- (A3) **Predictive accuracy of auxiliary models $\hat{\mathcal{M}}_{\text{int}}$ and $\hat{\mathcal{M}}_t$**: We assume that accuracy improvements in models $\hat{\mathcal{M}}_{\text{int}}$ and $\hat{\mathcal{M}}_t$ are proportional to those in the model $\hat{\mathcal{M}}$, *i.e.,* greater gains in $\hat{\mathcal{M}}_{\text{int}}$ or $\hat{\mathcal{M}}_t$ correspond to greater gains in $\hat{\mathcal{M}}$. This assumption relies on the robustness of boosting for effectively capturing influential attribute effects [64] and on the established accuracy of influence functions for assessing individual tuple impacts [3], [19].

We proceed with the proof through the following steps:

- (S1) **Define gain functions**: We define gain functions to measure the *expected accuracy improvement* in the auxiliary models when influential attributes or tuples are cleaned. By focusing on the expected gains, we capture the average impact of CR, with the stability from A2 ensuring the consistency despite randomness in training and data selection. Additionally, by Assumption A1, the validation set $\mathcal{C}_k$ reliably represents $\mathcal{D}_{\text{clean}}$, providing an effective basis for evaluating improvements in both $\hat{\mathcal{M}}$ and $\mathcal{M}$.
- (S2) **Prove monotonicity and submodularity**: We verify the monotonicity of the gain functions through the positive contributions of influential attributes and tuples selected by algorithms AlgIA and AlgIT, respectively, ensuring that the accuracy does not decrease as more data points are cleaned. The submodularity is verified by the diminishing returns property, *i.e.,* as additional tuples are cleaned, the marginal gain in accuracy decreases due to fewer remaining errors in the dataset for the $\mathcal{A}_{\text{CR}}$ to correct.
- (S3) **Approximation guarantee**: Utilizing the submodularity and monotonicity, we apply results from submodular optimization to obtain the approximation guarantee.

Below we provide the details of each step in the proof.

(S1) **Gain functions**. We define functions $g(\mathcal{S})$ and $g(\mathcal{T})$ to measure the expected accuracy improvement in the auxiliary models $\hat{\mathcal{M}}_{\text{int}}$ and $\hat{\mathcal{M}}_t$ when influential attributes $\mathcal{S}$ or tuples $\mathcal{T}$ are cleaned. Let G denote the expected accuracy improvement in $\hat{\mathcal{M}}$.

Attribute gain function. For a set of influential attributes $\mathcal{S}$ selected by AlgIA in the k-th round, the gain function $g(\mathcal{S})$ is defined as:

$$g(\mathcal{S}) = \mathbb{E}\left[\text{acc}\left(\hat{\mathcal{M}}_{\text{int}}(\mathcal{S}), \mathcal{C}_k\right) - \text{acc}\left(\hat{\mathcal{M}}_{\text{int}}, \mathcal{C}_k\right)\right],$$

where $\hat{\mathcal{M}}_{\text{int}}(\mathcal{S})$ denotes the model $\hat{\mathcal{M}}_{\text{int}}$ trained on data after cleaning $\mathcal{S}$, and $\text{acc}(\cdot, \mathcal{C}_k)$ is the model accuracy on $\mathcal{C}_k$.

Tuple gain function. Similarly, for an influential tuple set $\mathcal{T}$ selected by AlgIT in the k-th round, the gain function $g(\mathcal{T})$ is defined as:

$$g(\mathcal{T}) = \mathbb{E}\left[\text{acc}\left(\hat{\mathcal{M}}_t(\mathcal{T}), \mathcal{C}_k\right) - \text{acc}\left(\hat{\mathcal{M}}_t, \mathcal{C}_k\right)\right],$$

where $\hat{\mathcal{M}}_t(\mathcal{T})$ denotes $\hat{\mathcal{M}}_t$ trained after cleaning tuples in $\mathcal{T}$.

Gain function for $\hat{\mathcal{M}}$. By Assumption A3, the expected accuracy gain in $\hat{\mathcal{M}}$ is proportional to the gains in auxiliary models:

$$G(\mathcal{S}) \propto g(\mathcal{S}), \quad G(\mathcal{T}) \propto g(\mathcal{T}),$$

where $G(\mathcal{S})$ and $G(\mathcal{T})$ represent expected accuracy gains in $\hat{\mathcal{M}}$ from cleaning attributes in $\mathcal{S}$ and tuples in $\mathcal{T}$, respectively.

(S2) **The monotonicity and submodularity of** G. We analyze the behavior under incremental data cleaning and show that iterative cleaning leads to diminishing expected accuracy gains such that G satisfies the monotonicity and submodularity.

Monotonicity of G: For any sets $A \subseteq B \subseteq E$, where E represents all possible influential attributes or tuples, we show that:

$$G(A) \leq G(B).$$

Since the CR method $\mathcal{A}_{\text{CR}}$ is α-accurate with $\alpha > 0.5$, it ensures that the error rate in the dataset is non-increasing. As shown in Figures 3 and 4, AlgIA (resp. AlgIT) selects influential attributes (resp. tuples) that contribute positively to the accuracy of $\hat{\mathcal{M}}_{\text{int}}$ (resp. $\hat{\mathcal{M}}_t$). By S1, the auxiliary models

$\hat{\mathcal{M}}_{\mathsf{int}}$ and $\hat{\mathcal{M}}_t$ reliably reflect the accuracy improvement trends of $\hat{\mathcal{M}}$, maintaining an expected non-decreasing accuracy in $\hat{\mathcal{M}}$ when additional data points are cleaned. Therefore, cleaning subset B provides at least as much accuracy improvement as cleaning subset A, confirming the monotonicity of the gain function G in our framework.

Submodularity of G: We show that the gain of cleaning an additional influential attribute or tuple $t \notin B$ for subsets $A \subseteq B \subseteq E$ satisfies:

$$G(A \cup \{t\}) - G(A) \geq G(B \cup \{t\}) - G(B).$$

When t is an influential attribute, cleaning data points in $\mathcal{D} - \mathcal{D}_{\mathsf{clean}}$ based on the influential attributes identified in $B \cup \{t\}$ yields a non-increasing accuracy improvement compared to $A \cup \{t\}$. Since $A \subseteq B$ implies that $\mathcal{D}$ at the stage of B has fewer uncleaned tuples than at the stage of A, the impact of cleaning t is reduced since fewer errors remain in the attribute e for $\mathcal{A}_{\mathsf{CR}}$ to correct. When t is an influential tuple, cleaning data points in $\mathcal{D} - \mathcal{D}_{\mathsf{clean}}$ associated with the stage either $A \cup \{t\}$ or $B \cup \{t\}$ produces the same accuracy improvement. Since cleaning the same tuple results in a fixed accuracy gain, the improvement remains identical. Thus, in both cases, we have:

$$G(A \cup \{t\}) - G(A) \geq G(B \cup \{t\}) - G(B).$$

By Assumption A2 (The stability of model accuracy gain), these incremental gains remain stable across iterations, ensuring a consistent trend of diminishing returns of G, and thus is submodular.

(S3) **Approximation guarantee**. Since G is non-negative, monotonic, and submodular, classical submodular optimization results [79] guarantee a $(1 - 1/e) > 63\%$ approximation bound with the greedy strategies of AlgIA and AlgIT, where $e \approx 2.718$ is the base of the natural logarithm. This ensures that the expected improvement in accuracy achieved by $\hat{\mathcal{M}}$ on the cleaned dataset $\mathcal{D}_{\mathsf{clean}}$ is at least 63% of the optimal achievable improvement. Consequently, the expected accuracy of $\hat{\mathcal{M}}$ on $\mathcal{D}_{\mathsf{clean}}$, combining its pre-cleaning accuracy and the achieved improvement, satisfies:

$$\mathbb{E}(\mathsf{acc}(\hat{\mathcal{M}}, \mathcal{D}_{\mathsf{clean}})) \geq (1 - 1/e) \cdot \mathsf{acc}_{\mathsf{opt}}(\hat{\mathcal{M}}),$$

where $\mathsf{acc}_{\mathsf{opt}}(\hat{\mathcal{M}})$ denotes the optimal expected accuracy that is achievable by $\hat{\mathcal{M}}$ through data cleaning.

To extend this result to the target model $\mathcal{M}$, Assumption A1 ensures that the $\mathcal{C}_k$ reliably represents $\mathcal{D}_{\mathsf{clean}}$, allowing accuracy improvements observed in $\hat{\mathcal{M}}$ to approximate those expected on $\mathcal{D}_{\mathsf{clean}}$. Furthermore, Assumption A2 guarantees stability in incremental accuracy gains across iterations, such that these improvements generalize consistently to $\mathcal{M}$.

Thus, the expected accuracy of $\mathcal{M}$ satisfies the following:

$$\mathbb{E}(\mathsf{acc}(\mathcal{M}, \mathcal{D}_{\mathsf{clean}})) \geq (1-1/e) \cdot \mathsf{acc}_{\mathsf{opt}}(\mathcal{M}) > 63\% \cdot \mathsf{acc}_{\mathsf{opt}}(\mathcal{M}).$$

This result confirms that the expected accuracy of $\mathcal{M}$ trained on the cleaned dataset $\mathcal{D}_{\mathsf{clean}}$ achieves at least 63% of the optimal accuracy. This completes the proof of Theorem 4. □