Data Enhancing for Machine Learning

Paper ID: 1146

ABSTRACT

This paper studies enhancement of training data $\mathcal{D}$ to improve the robustness of machine learning (ML) classifiers $\mathcal{M}$ against adversarial attacks. Data enhancing aims to (a) defuse poisoned imperceptible features embedded in $\mathcal{D}$, and (b) defend against attacks at prediction time that are unseen in $\mathcal{D}$. We show that while there exists an inherent tradeoff between the accuracy and robustness of $\mathcal{M}$ in case (b), data enhancing can improve both the accuracy and robustness at the same time in case (a). We formulate two data enhancing problems accordingly, and show that both problems are intractable. Despite the hardness, we propose a framework that integrates model training and data enhancing. Moreover, we develop algorithms for (a) detecting and debugging corrupted imperceptible features in training data, and (b) selecting and adding adversarial examples to training data to defend against unseen attacks at prediction time. Using real-life datasets, we empirically verify that the method is at least 20.4% more robust and 2.02X faster than SOTA methods for classifiers $\mathcal{M}$, without degrading the accuracy of $\mathcal{M}$.

1 INTRODUCTION

Machine learning (ML) models are vulnerable to adversarial attacks [77]. Such attacks generate inputs that are almost indistinguishable from natural data and seem fine to a human eye, but cause the models to make highly-confident but erroneous predictions [8, 35, 86]. The poisoned data attempts to get unwarranted advantageous decisions, and may even come from malicious attacks of fraudsters [13]. As shown in a recent survey [56], (a) the ML systems of Google, Amazon, Microsoft and Tesla experienced adversarial attacks; and (b) data poisoning was ranked as the #1 (out of 11) attacks affecting business. Indeed, 13.6% of the adversarial tuples crafted in [13] were mis-predicated by a real-world production system. In 2018, adversarial credit fraud incurred loss of $24.26 billion [6].

Example 1: Consider Table 1 from dataset German [89], which has attributes age, credit, savingaccount, housing, education and purpose; its classification label is risk. It is partitioned into training data (tuples $t_1, t_2, \ldots$) and data for prediction (tuples $t_a, t_b, \ldots$). Each tuple in the training table represents a person who is assigned a credit by a bank, and is classified as high or low risk. Here attributes colored in green (resp. gray) are the ones that are very likely (resp. unlikely) checked by the bank; since manual inspection is costly, it is often unlikely to check all attributes in a table, *e.g.,* a bank usually checks only key attributes of a loan application, rather than all [8]. Values colored in red are corrupted by an attacker.

Below we show two cases of adversarial attacks, injected into the attributes of the training and prediction data, respectively.

(1) Adversarial attacks in training data. In the training data, tuple t_2 (resp. t_4) is corrupted at attribute purpose (resp. education), *e.g.,* t_2[purpose] (resp. t_4[education]) is modified from `car` (resp. `bachelor`) to `education` (resp. `Ph.D`). Such perturbations may mislead ML models $\mathcal{M}$ to learn that a loan for `education` (resp. `Ph.D`) has a high risk, and make bad prediction, *e.g.,* it may classify tuple t_a in the prediction table as high risk. This is not true.

tid	age (A_1)	credit (A_2)	savingaccount (A_3)	housing (A_4)	education (A_5)	purpose (A_6)	risk (Y)
t_1	23	medium	moderate	rent	Ph.D	car	low
t_2	20	low	little	rent	bachelor	education	high
t_3	30	medium	quite-rich	own	Ph.D	repairs	low
t_4	25	medium	moderate	rent	Ph.D	education	high
...	...	...	...	...	...	...	...
			the training dataset				
t_a	20	low	little	rent	bachelor	education	?
t_b	21	medium	moderate	rent	Ph.D	car	?
t_c	22	medium	moderate	rent	Ph.D	car	?
t_d	24	medium	moderate	rent	Ph.D	car	?
...	...	...	...	...	...	...	...
			the prediction dataset				

Table 1: The German **Dataset**

(2) Adversarial data at prediction time. After $\mathcal{M}$ is trained, when applied to corrupted tuples t_b and t_d in which attribute education is changed from `bachelor` to `PhD`, $\mathcal{M}$ may mis-classify them as low risk, as a young Ph.D can get a well-paid job and pay off the loan. □

No matter how important, it is nontrivial to mitigate the impact of adversarial attacks. There has been a host of work on generating adversarial data with a high fooling rate at training time [34, 45, 46, 81, 97] or prediction time [8, 13, 25, 42, 67, 87]; the former aims to mislead the training of ML models by poisoning training data $\mathcal{D}_{\text{train}}$, while the latter focuses on fooling a trained model by tampering data $\mathcal{D}_{\text{pred}}$ for prediction. In contrast, much less is known about how to make models robust against adversarial attacks, and the prior work has mostly focused on image models [12, 23, 36, 57, 70, 80, 85, 90, 91]. Against attacks in relational data, a method proposed in [62] suggests to (a) directly remove poisoned tuples from $\mathcal{D}_{\text{train}}$ to avoid misleading ML models $\mathcal{M}$ at training time, and (b) flag suspicious tuples in $\mathcal{D}_{\text{pred}}$ that may misguide $\mathcal{M}$ at prediction time.

However, there are questions about robust $\mathcal{M}$ on relational data.

(1) Robustness vs. accuracy. Accuracy measures how often an ML model $\mathcal{M}$ makes correct predictions on unpoisoned data. Robustness evaluates the ability of $\mathcal{M}$ to resist being fooled, *i.e.,* how indifferent its accuracy is to various types of attacks. It is known that inherent tradeoff exists between the accuracy and robustness of $\mathcal{M}$ on image data [86]. The first question asks whether for any attacker on relational data, the tradeoff is avoidable? If so, when? If not, why? To the best of our knowledge, these questions have not been settled.

(2) Attack detection and diffusion in training data. Adversarial attacks carefully craft perturbations to misguide $\mathcal{M}$ for specific purposes. The perturbations are usually conducted on imperceptible features that are not likely to be inspected by human experts; worse still, the imperceptible features of different tuples may vary, *e.g.,* attributes purpose in tuple t_2 and education of t_4 in Table 1. The second question asks *how to distinguish attacks from common noise in the data*? Moreover, simply removing the poisoned tuples may reduce training data and hurt the accuracy of $\mathcal{M}$. *How can we defuse the attacks in training data without degrading the accuracy of* $\mathcal{M}$?

(3) Robustness against unseen attacks at prediction time. Even if we could accurately detect and defuse adversarial attacks in our training data, our models may make incorrect decisions at prediction

time when adversarial attacks are embedded into the input, especially when the attacks are not seen in the training data. Simply flagging suspicious tuples does not suffice to make our models robust against attacks that are not embedded in the training data. The third question is *how to fine-tune our trained models such that they are not only immune to the known attacks in our corrupted training data, but are also robust against other possible adversarial attacks*?

Contributions & Organization. In response to the questions, we propose DE4ML (Data Enhancing for ML), a framework to make ML models $\mathcal{M}$ robust against adversarial attacks. When training $\mathcal{M}$ with dataset $\mathcal{D}$, we enhance $\mathcal{D}$ by (1) identifying and debugging its tuples in which adversarial attacks $\mathcal{A}_1$ are embedded, and (2) adding adversarial example tuples from other possible attacks $\mathcal{A}_2$, to make $\mathcal{M}$ robust against both $\mathcal{A}_1$ and $\mathcal{A}_2$ at prediction time. That is, we aim to train $\mathcal{M}$ that is robust to attacks $\mathcal{A}_1$ and $\mathcal{A}_2$ at the same time, without degrading the accuracy of $\mathcal{M}$. To simplify the discussion, we focus on binary ML classifiers on relational data.

(1) Problem and complexity (Section 2). We show that defusing attacks in the training data does not reduce the accuracy, *i.e.,* both the accuracy and robustness of $\mathcal{M}$ can be achieved at the same time. To do this, we select and fix poisoned cells/tuples instead of all errors, to (a) reduce errors introduced by data cleaning tools and (b) retain the accuracy by purposely leaving non-influential errors unfixed.

When adding tuples to defend against attacks that corrupt the data for prediction but are unseen in the training data, we show that there exists an unavoidable tradeoff between the accuracy and robustness of $\mathcal{M}$ on relational data, for any model and attacker.

Hence, we formulate two data enhancing problems for defusing attacks injected into training data and unseen attacks at prediction data, respectively. We show that both problems are intractable.

(2) A framework (Section 3). We propose a two-phase framework DE4ML that integrates ML training and data enhancing. In phase 1, DE4ML takes as input a classifier $\mathcal{M}$, a training dataset $\mathcal{D}_{\text{train}}$ of a database schema $\mathcal{R}$, and a data cleaning tool C. It iteratively selects (imperceptible) attributes to fix using C, and trains $\mathcal{M}$ with the enhanced $\mathcal{D}_{\text{train}}$. In phase 2, DE4ML takes as input the trained $\mathcal{M}$, the enhanced $\mathcal{D}_{\text{train}}$ and possible attackers $\mathcal{A}$; it generates a set S_{at} of adversarial tuples via $\mathcal{A}$, iteratively enhances $\mathcal{D}_{\text{train}}$ with the data in S_{at}, and fine-tunes $\mathcal{M}$. One may opt to use DE4ML to enhance an arbitrary classifier $\mathcal{M}$ for defending against (a) injected attacks in training data, (b) possible attacks on data $\mathcal{D}_{\text{pred}}$ at prediction time, or (c) both of them.

(3) Debugging training data (Section 4). For phase 1, we provide an algorithm that, given corrupted training data $\mathcal{D}_{\text{train}}$, identifies poisoned features that are (a) abnormal, *i.e.,* they incur a high training loss, (b) influential, *i.e.,* they highly impact the prediction of $\mathcal{M}$ on other tuples, and (c) imperceptible, *i.e.,* they appear in imperceptible attributes. We employ datamodel [44] to distinguish poisoned data from noise, to defuse adversarial attacks to $\mathcal{D}_{\text{train}}$, with enhanced training data, without degrading the accuracy of $\mathcal{M}$.

(4) Adding supplement tuples (Section 5). For phase 2, we use datamodel to project data into a unified embedding space, based on which we select adversarial examples created by existing methods, *e.g.,* [8, 13, 34, 45, 46, 81, 87, 97], where these examples are (a) influential, *i.e.,* they can make $\mathcal{M}$ robust against poisoned tuples in prediction data $\mathcal{D}_{\text{pred}}$, (b) harmless, *i.e.,* they do not largely degrade the accuracy of $\mathcal{M}$ on tuples in unpoisoned $\mathcal{D}_{\text{pred}}$, and (c) diversified, *i.e.,* they are from different groups (*e.g.,* bachelor and Ph.D) to defend against poisoned tuples in different groups. We add such tuples to the training data of $\mathcal{D}$, and incrementally train $\mathcal{M}$ with the correctly labeled data. Moreover, we propose an iterative approach that enhances $\mathcal{D}$ with a small number of critical adversarial examples, to strike a balance between the robustness and accuracy of $\mathcal{M}$.

(5) Experimental study (Section 6). Using real-life data, we evaluated DE4ML against 9 baselines. We empirically find the following. (a) DE4ML outperforms the SOTA (state-of-the-art) Picket [62] by 7% and 33.8% for defusing attacks in the training data and defending against unseen attacks at prediction time, respectively, 20.4% on average in each phase. (b) DE4ML improves the robustness of various ML models $\mathcal{M}$ with robustness 0.72 on average, 27.4% better than baselines in the two phases together. (c) DE4ML is consistently effective for different types of attacks, with a maximal robustness gap of 0.06. (d) DE4ML is efficient; *e.g.,* it is 2.04X and 2.02X faster than Picket for defusing the attacks in $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$, respectively.

We discuss related work in Section 7 and future work in Section 8. The proofs, technical details, code and data of the paper are in [2].

Data enhancing for ML is quite different from data cleaning for AI [4, 5, 27, 30, 40, 53–55, 62]. The latter aims to detect and fix errors (outliers, conflicts, missing data, wrong labels) in the training data for $\mathcal{M}$, to improve *the accuracy* of $\mathcal{M}$. In contrast, the former (a) targets training data corrupted by adversarial attacks by identifying and fixing imperceptible features, and (b) adds supplement tuples to the training data to defend against attacks at prediction time, to improve *the robustness* of $\mathcal{M}$ without degrading the accuracy. It is known that automated data cleaning may even hurt the fairness of $\mathcal{M}$ [39]. Since there is also inherent tradeoff between the robustness and accuracy, data enhancing cannot be replaced by data cleaning.

2 PROBLEM AND COMPLEXITY

This section studies the tradeoff between accuracy and robustness of ML classifiers (Section 2.1). We formulate two data enhancing problems against attacks in training and prediction data, respectively, and show that both problems are intractable (Section 2.2).

2.1 Accuracy and Robustness

We first define accuracy and robustness for classification models.

Preliminaries. We start with basic notations.

Datasets. A database schema is $\mathcal{R} = (R_1, \ldots, R_m)$, where R_i is a relation schema $R(A_1, \ldots, A_k, Y)$, each A_j is an attribute (feature), and Y is the label attribute for the classification of its tuples. An instance $\mathcal{D}$ of $\mathcal{R}$ is $(D_1, \ldots, D_m)$, where D_i is a relation of R_i.

ML classifiers. Following [51, 59], we consider *binary classifier* $\mathcal{M}(x) : \mathbb{R}^k \rightarrow \{0, 1\}$, which is a function that maps an attribute (feature) vector $x = [A_1, \ldots, A_k] \in \mathbb{R}^k$ to a class label $y \in \{0, 1\}$. For $\mathcal{M}$, we denote by $\mathcal{D}_{\text{train}}$ a dataset of schema $\mathcal{R}$ for training $\mathcal{M}$, and by $\mathcal{D}_{\text{pred}}$ a dataset of $\mathcal{R}$ for prediction by $\mathcal{M}$ after $\mathcal{M}$ is trained.

Attribute importance vector [8]. An attribute importance vector $\mathcal{V}$ of a schema R has the form of $[v_1, \ldots, v_k, v_Y]$, where $v_j \in [0, 1]$ is the likelihood of an attribute $A_j \in R$ to be manually inspected. The higher v_j is, the more likely attribute A_j is checked. Vector $\mathcal{V}$ is usually summed up by experts and is available; different sectors adhere to different $\mathcal{V}$, *e.g.,* Wolters Kluwer [1] (a global information leader) ranked and listed the key attributes of loan applications (*e.g.,* credit history and cash flow history) considered by banks.

Imperceptible attributes. For a relation schema $\mathcal{R}$ and its attribute importance vector $\mathcal{V}$, an attribute $A_j \in \mathcal{R}$ is *imperceptible* if its importance score v_j is below a user-defined threshold $\mu \in (0, 1]$. However, "an imperceptible attribute" does not imply that it is non-decisive for an $\mathcal{M}$, *i.e.,* it may have a strong correlation with the label of its tuple, since the $\mathcal{V}$ is subjectively summed up by human.

Attacker model. Following [8, 13], we assume that an attacker tends to inject more noises in imperceptible attributes that are less likely to be inspected. To corrupt a tuple $t \in \mathcal{D}$, an attacker interpolates its attributes A_j following the probability $1 - v_j$. The attacker corrupts a selected cell $t[A_j]$ by replacing its original value c with value c' ($\neq c$), following an adversarial distribution. We assume *w.l.o.g.* that the fraction of corrupted data λ in a dataset, referred to as the *power* of an attacker, is less than 50% [62, 82]. This is the attacker model commonly adopted on relations, *e.g.,* [8, 13, 81].

Following [8, 13, 25, 42, 67, 87], we consider two types of attackers $\mathcal{A}_1$ and $\mathcal{A}_2$, where (a) $\mathcal{A}_1$ attacks the training data [34, 45, 46, 81, 97], *i.e.,* corrupting the attributes and labels of $\mathcal{D}_{\text{train}}$, and (b) $\mathcal{A}_2$ attacks the attributes of the prediction data $\mathcal{D}_{\text{pred}}$ [8, 13, 25, 42, 67, 87]; we denote prediction data $\mathcal{D}_{\text{pred}}$ poisoned by $\mathcal{A}_2$ as $\mathcal{D}^{-}_{\text{pred}}$.

Perturbations to $\mathcal{D}_{\text{train}}$. To defuse attacks injected by $\mathcal{A}_1$ and $\mathcal{A}_2$, we enhance the training data $\mathcal{D}_{\text{train}}$ with *perturbations*, which are either (a) value perturbations $\Delta\mathcal{D}_V$, *i.e.,* modifications of attribute value $t[A]$ of tuples $t \in \mathcal{D}_{\text{train}}$; or (b) tuple perturbations $\Delta\mathcal{D}_T$, *i.e.,* insertions of tuples into $\mathcal{D}_{\text{train}}$. We do not consider deletions of adversarial tuples from $\mathcal{D}_{\text{train}}$. Instead, we fix such tuples in place to retain the accuracy. Denote by $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$ (resp. $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$) the enhanced $\mathcal{D}_{\text{train}}$ by applying $\Delta\mathcal{D}_V$ (resp. $\Delta\mathcal{D}_T$) to $\mathcal{D}_{\text{train}}$.

We next formalize metrics for measuring the accuracy and robustness of $\mathcal{M}$ that is trained with enhanced $\mathcal{D}_{\text{train}}$ via perturbations.

Accuracy. Denote by $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{pred}})$ *the accuracy of* $\mathcal{M}$ that is trained with (possibly enhanced) $\mathcal{D}_{\text{train}}$ and evaluated with unpoisoned predication dataset $\mathcal{D}_{\text{pred}}$ of $\mathcal{M}$. Following [86], we measure $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{pred}})$ in terms of the *expected loss*:

$$\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{pred}}) = 1 - \mathop{\mathbb{E}}_{(x,y) \in \mathcal{D}_{\text{pred}}} [L(\mathcal{M}(x), y)], \quad (1)$$

where (x, y) is a tuple in $\mathcal{D}_{\text{pred}}$, $\mathcal{M}(x)$ is the prediction of $\mathcal{M}$ at the tuple based on the attribute vector x, and $L(\mathcal{M}(x), y)$ is the 0-1 loss between the prediction $\mathcal{M}(x)$ and its label y (*i.e.,* assigning 0/1 for a prediction correctly/wrongly matching the label).

Robustness. When $\mathcal{D}_{\text{train}}$ is enhanced with value (resp. tuple) perturbations to defuse attacks in $\mathcal{D}_{\text{train}}$ (resp. $\mathcal{D}_{\text{pred}}$), the accuracy of $\mathcal{M}$ on prediction data $\mathcal{D}_{\text{pred}}$ (resp. $\mathcal{D}^{-}_{\text{pred}}$) may be affected. We use robustness to measure to which extent (a) training of $\mathcal{M}$ is insensitive to attacks in $\mathcal{D}_{\text{train}}$ by $\mathcal{A}_1$, and/or (b) trained $\mathcal{M}$ can identify and correctly classify corrupted tuples in $\mathcal{D}^{-}_{\text{pred}}$ when $\mathcal{D}_{\text{pred}}$ is attacked by $\mathcal{A}_2$. It evaluates the impact of attacks on the accuracy of $\mathcal{M}$, and is thus also referred to as adversarially robust accuracy [86].

(1) Corrupted $\mathcal{D}_{\text{train}}$ *by* $\mathcal{A}_1$. When attacker $\mathcal{A}_1$ corrupts $\mathcal{D}_{\text{train}}$ to mislead the training of $\mathcal{M}$, we enhance $\mathcal{D}_{\text{train}}$ with value perturbations $\Delta\mathcal{D}_V$ to defuse $\mathcal{A}_1$. Denote by $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}})$ *the robustness of* $\mathcal{M}$ *trained with enhanced* $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$ and evaluated on unattacked $\mathcal{D}_{\text{pred}}$. Following [86], we define

$$\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) = 1 - \mathop{\mathbb{E}}_{(x,y) \in \mathcal{D}_{\text{pred}}} [L(\mathcal{M}(x), y)], \quad (2)$$

where (x, y), $\mathcal{M}(\cdot)$ and $L(\cdot, \cdot)$ are the same as in Equation 1. Intuitively, the robustness measures how well the enhanced $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$ defuses attacks of $\mathcal{A}_1$ and improves $\mathcal{M}$'s accuracy on the unpoisoned $\mathcal{D}_{\text{pred}}$. Thus, a model $\mathcal{M}$ with the higher $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}})$ indicates that the training of $\mathcal{M}$ is more robust.

(2) Corrupted $\mathcal{D}_{\text{pred}}$ *by* $\mathcal{A}_2$. Attacker $\mathcal{A}_2$ aims to make a trained $\mathcal{M}$ misjudge on poisoned tuples in an attacked $\mathcal{D}^{-}_{\text{pred}}$ of $\mathcal{D}_{\text{pred}}$. When $\mathcal{D}_{\text{pred}}$ is corrupted by $\mathcal{A}_2$ but $\mathcal{D}_{\text{train}}$ is uncorrupted or attack-defused, we enhance $\mathcal{D}_{\text{train}}$ with adversarial tuple perturbations $\Delta\mathcal{D}_T$ to fine-tune $\mathcal{M}$, such that $\mathcal{M}$ can defend against attacks by $\mathcal{A}_2$ that are unseen in $\mathcal{D}_{\text{train}}$. Denote by $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}})$ *the robustness of* $\mathcal{M}$ *fine-tuned with enhanced* $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ and evaluated on the attacked $\mathcal{D}^{-}_{\text{pred}}$. Following [86], we define

$$\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^{-}_{\text{pred}}) = 1 - \mathop{\mathbb{E}}_{(x,y) \in \mathcal{D}^{-}_{\text{pred}}} [L(\mathcal{M}(x), y)], \quad (3)$$

where (x, y), $\mathcal{M}(\cdot)$ and $L(\cdot, \cdot)$ are the same as in Equation 2, but $\mathcal{D}^{-}_{\text{pred}}$ is a poisoned version of $\mathcal{D}_{\text{pred}}$ by $\mathcal{A}_2$. Intuitively, here the robustness measures how well the model $\mathcal{M}$ fine-tuned with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ can identify and correctly classify the adversarial tuples in $\mathcal{D}^{-}_{\text{pred}}$. A large (resp. small) robustness indicates that a trained $\mathcal{M}$ is (resp. is not) able to resist the attacks in $\mathcal{D}^{-}_{\text{pred}}$.

Accuracy vs. robustness. We now study the tradeoff between the accuracy and robustness of model $\mathcal{M}$ when its training dataset $\mathcal{D}_{\text{train}}$ is enhanced. Recall that we enhance $\mathcal{D}_{\text{train}}$ with value (resp. tuple) perturbation to defend against attacks in $\mathcal{D}_{\text{train}}$ (resp. $\mathcal{D}_{\text{pred}}$), where the injected attacks in $\mathcal{D}^{-}_{\text{pred}}$ are unseen in $\mathcal{D}_{\text{train}}$.

Corrupted $\mathcal{D}_{\text{train}}$. When $\mathcal{D}_{\text{train}}$ is corrupted but $\mathcal{D}_{\text{pred}}$ is not, the accuracy and robustness of $\mathcal{M}$ can be improved at the same time. We fix $\mathcal{D}_{\text{train}}$ with value perturbations $\Delta\mathcal{D}_V$, and train $\mathcal{M}$ with the enhanced $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$. The perturbations $\Delta\mathcal{D}_V$ defuse the attacks and make the distribution of $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$ closer to the "unpoisoned" true distribution. Hence, the model trained with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$ has a higher accuracy than with the poisoned $\mathcal{D}_{\text{train}}$. By Equations 1 and 2, both accuracy and robustness aim to minimize the same expected loss function. Correcting adversarial data in $\mathcal{D}_{\text{train}}$ by $\Delta\mathcal{D}_V$ aligns the model's decision boundary with the true data distribution, improving $\mathcal{M}$'s performance on unpoisoned $\mathcal{D}_{\text{pred}}$ and increasing its resilience to attacks in $\mathcal{D}_{\text{train}}$. This alignment optimizes both accuracy and robustness simultaneously.

Corrupted $\mathcal{D}_{\text{pred}}$ *(i.e.,* $\mathcal{D}^{-}_{\text{pred}}$*)*. In contrast, to defend against attacks in $\mathcal{D}^{-}_{\text{pred}}$ unseen in $\mathcal{D}_{\text{train}}$, we generate a set $\Delta\mathcal{D}_T$ of tuple perturbations and fine-tune $\mathcal{M}$ with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$. This introduces inherent trade-off between the accuracy (Equation 1) and robustness (Equation 3) of $\mathcal{M}$, as they focus on different objectives. Fine-tuning $\mathcal{M}$ with the enhanced $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ may change the distribution of $\mathcal{M}$ learned towards the poisoned $\mathcal{D}^{-}_{\text{pred}}$ (*i.e.,* improving robustness in Equation 3), and unavoidably steer the distribution away from the unpoisoned $\mathcal{D}_{\text{pred}}$ (*i.e.,* degrading accuracy in Equation 1), unless

$\mathcal{D}^{-}_{\text{pred}}$ and $\mathcal{D}_{\text{pred}}$ share the same distribution.

Theorem 1: *There exists an inherent tradeoff between the accuracy and robustness of any ML classifier $\mathcal{M}$, when $\mathcal{M}$ is fine-tuned with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ and evaluated with the tuples in unpoisoned $\mathcal{D}_{\text{pred}}$ (resp. poisoned $\mathcal{D}^{-}_{\text{pred}}$) for accuracy (resp. robustness).*

Specifically, under any attacker with power $\lambda \in (0, 0.5)$, any ML classifier $\mathcal{M}$ with accuracy $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}}) = 1 - \eta$ for $\eta \in [0, 1]$, its robustness $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^{-}_{\text{pred}})$ is at most $(1 - \lambda) \cdot (1 - \eta) + \frac{\lambda \cdot (p + v_d - 2 \cdot v_d \cdot p) \cdot \eta}{(1-p)}$, where $p \in [0.5, 1]$ quantifies the correlation between decisive attributes and labels in $\mathcal{D}_{\text{pred}}$, and v_d is the probability for an attacker to poison decisive attributes. □

Observe the following about Theorem 1. (1) The result holds for any ML model $\mathcal{M}$ and any attacker with power $\lambda \in (0, 0.5)$. (2) When the accuracy is lower bounded, the robustness is upper bouneded. (3) The tradeoff is indicated by η, *e.g.,* with $\eta \rightarrow 0$, when accuracy approaches 1, the robustness cannot exceed $1 - \lambda$; the improved robustness comes with reduced accuracy.

Proof sketch: Consider a schema with a decisive attribute A_1 that aligns with label y with probability $p \geq 0.5$ in unpoisoned data. For any ML classifier $\mathcal{M}$, the expected loss of $\mathcal{M}$ relies primarily on A_1 but $\mathcal{M}$ also uses non-decisive attributes for higher accuracy [86]. An attacker with power λ corrupts parts of the data; it flips decisive attribute A_1 with probability v_d, and alters the distributions of non-decisive attributes that impact the label. The adversarial attacks disrupt data of both types of attributes, making it hard for $\mathcal{M}$ to distinguish between unpoisoned and poisoned data. By analyzing different cases of decisive and non-decisive attributes when $\mathcal{M}$ predicts $y = 1$, we show that when we retain the accuracy of $\mathcal{M}$ on unpoisoned data above a bound, the robustness of $\mathcal{M}$ on poisoned data is necessarily upper bounded; hence the inherent tradeoff. □

Example 2: Continuing with Example 1, consider training data $\mathcal{D}_{\text{train}}$ in Table 1 enhanced by value perturbations, *e.g.,* the purpose (resp. education) of t_2 (resp. t_4) is fixed from `education` (resp. `Ph.D`) to `car` (resp. `bachelor`). Denote by $\mathcal{D}^{-}_{\text{pred}}$ (resp. $\mathcal{D}_{\text{pred}}$) the attacked (resp. unattacked) prediction dataset. Intuitively, model $\mathcal{M}$ trained with the enhanced $\mathcal{D}_{\text{train}}$ has a good accuracy (*i.e.,* Equation 1) on the unattacked $\mathcal{D}_{\text{pred}}$, *e.g.,* it correctly predicts tuples t_a-t_d, but is less accurate on the corrupted $\mathcal{D}^{-}_{\text{pred}}$ with a bad robustness (*i.e.,* Equation 3), *e.g.,* it wrongly predicts t_b and t_d. Hence the robustness of $\mathcal{M}$ is hampered as indicated by its low accuracy on $\mathcal{D}^{-}_{\text{pred}}$.

To improve the robustness of $\mathcal{M}$, we need to make $\mathcal{M}$ more accurate on the poisoned $\mathcal{D}^{-}_{\text{pred}}$. We enhance $\mathcal{D}_{\text{train}}$ with tuple perturbations, *e.g.,* adding to $\mathcal{D}_{\text{train}}$ two adversarial tuples $t_b^{'}$ and $t_d^{'}$ with the same attributes as t_b and t_d, respectively, but with labels `high`, and fine-tune $\mathcal{M}$ with the enhanced $\mathcal{D}_{\text{train}}$, such that its robustness is improved, *e.g.,* it correctly predict more tuples (*i.e.,* t_a, t_b and t_d) on poisoned $\mathcal{D}^{-}_{\text{pred}}$. However, its accuracy on unpoisoned $\mathcal{D}_{\text{pred}}$ is degraded, *e.g.,* it misclassifies t_c after adding adversarial tuple $t_b^{'}$ and $t_d^{'}$ to $\mathcal{D}_{\text{train}}$. In short, improving the robustness of $\mathcal{M}$ on poisoned $\mathcal{D}^{-}_{\text{pred}}$ with tuple perturbations unavoidably affects the learned distribution (accuracy) of $\mathcal{M}$ on the unpoisoned $\mathcal{D}_{\text{pred}}$. □

The notations of the paper are summarized in Table 2.

Notations	Definitions
$\mathcal{R}$, $\mathcal{V}$	database schema, attribute importance vector of $\mathcal{R}$
$\mathcal{M}$, $\hat{\mathcal{M}}_t$	ML model, datamodel of $\mathcal{M}$ for predicting a tuple t
$\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{pred}}$	dataset for training $\mathcal{M}$, dataset for prediction by $\mathcal{M}$
α, $\hat{\theta}_t$	sampling ratio for training $\hat{\mathcal{M}}_t$, parameter vector of $\hat{\mathcal{M}}_t$
$\mathcal{A}$, $\mathcal{D}^{-}_{\text{pred}}$, C	attacker, poisoned $\mathcal{D}_{\text{pred}}$ by $\mathcal{A}$, data cleaning tool
$\Delta\mathcal{D}_V$, $\Delta\mathcal{D}_T$	value perturbations, tuple perturbations
$\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}$	enhanced $\mathcal{D}_{\text{train}}$ with value/tuple perturbations $\Delta\mathcal{D}$
$\text{acc}(\cdot,\cdot,\cdot)$, $\text{rob}(\cdot,\cdot,\cdot)$	the accuracy of $\mathcal{M}$, the robustness of $\mathcal{M}$

Table 2: Notations

2.2 Data Enhancing Problems

We next formulate two data enhancing problems, for defusing attacks in corrupted $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ (*i.e.,* $\mathcal{D}^{-}_{\text{pred}}$), respectively.

Data enhancing against attacks in $\mathcal{D}_{\text{train}}$. This is to train an ML classifier $\mathcal{M}$ with dataset $\mathcal{D}_{\text{train}}$ possibly poisoned by adversary $\mathcal{A}_1$, to maximize the robustness and accuracy of $\mathcal{M}$. We evaluate the accuracy/robustness of $\mathcal{M}$ using a set $\mathcal{D}_{\text{pred}}$ of data for prediction, which is disjoint from $\mathcal{D}_{\text{train}}$. We enhance $\mathcal{D}_{\text{train}}$ with value perturbations, *i.e.,* modifying attribute values $t[A]$ and labels $t[Y]$ of tuples t in $\mathcal{D}_{\text{train}}$. As remarked in Section 2.1, the robustness and accuracy of $\mathcal{M}$ can be improved at the same time in this case, *i.e.,* maximizing $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}})$ is equivalent to maximizing $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}})$.

More formally, the *problem of Data Enhancing Against Attacks in Training data*, denoted by DEAAT, is stated as follows.

- *Input*: A database schema $\mathcal{R}$, two datasets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ of $\mathcal{R}$, a data cleaning tool C, and an ML classifier $\mathcal{M}$.
- *Output*: A set $\Delta\mathcal{D}_V$ of value perturbations to $\mathcal{D}_{\text{train}}$.
- *Objective*: Maximize both $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}})$ and $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}})$ at the same time.

We want to identify a small set $\Delta\mathcal{D}_V$ to maximully improve the accuracy and robustness of $\mathcal{M}$ trained with $\mathcal{D}_{\text{train}} \oplus \Delta D_V$.

The decision problem of DEAAT, also denoted by DEAAT, is to decide, given $\mathcal{R}$, $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{pred}}$, C, $\mathcal{M}$ and a bound B_{train}, whether there exists a set $\Delta\mathcal{D}_V$ such that $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) \geq B_{\text{train}}$ and $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) \geq B_{\text{train}}$.

Data enhancing against attacks in $\mathcal{D}^{-}_{\text{pred}}$. Assume that we have defused attacks in $\mathcal{D}_{\text{train}}$ and obtain a model $\mathcal{M}$ trained with the updated $\mathcal{D}_{\text{train}}$. Assume that an adversary $\mathcal{A}_2$ attacks the attributes of tuples in $\mathcal{D}_{\text{pred}}$ with perturbations following a fixed but unknown distribution, generating a poisoned predication dataset $\mathcal{D}^{-}_{\text{pred}}$. We want to further enhance $\mathcal{D}_{\text{train}}$ with a set $\Delta\mathcal{D}_T$ of tuple perturbations and fine-tune $\mathcal{M}$ with the enhanced $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$, such that the fine-tuned model $\mathcal{M}$ can also guard against attacks in $\mathcal{D}^{-}_{\text{pred}}$. As for DEAAT, we use $\mathcal{D}_{\text{pred}}$ and $\mathcal{D}^{-}_{\text{pred}}$ only for evaluation.

The *problem of Data Enhancing Against Attacks at Prediction time*, denoted by DEAAP, is stated as follows.

- *Input*: $\mathcal{R}$, $\mathcal{M}$, $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{pred}}$ as above, an adversarial perturbation function $\mathcal{A}_2$, corrupted $\mathcal{D}^{-}_{\text{pred}}$ of $\mathcal{D}_{\text{pred}}$ by $\mathcal{A}_2$, and a bound B_{inf}.
- *Output*: A set $\Delta\mathcal{D}_T$ tuple perturbations to $\mathcal{D}_{\text{train}}$.
- *Objective*: Maximize $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^{-}_{\text{pred}})$, subject to $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}}) \geq B_{\text{inf}}$.

It is to improve the robustness of $\mathcal{M}$ while retaining the accuracy.

Its decision version, also denoted by DEAAP, is to decide, given

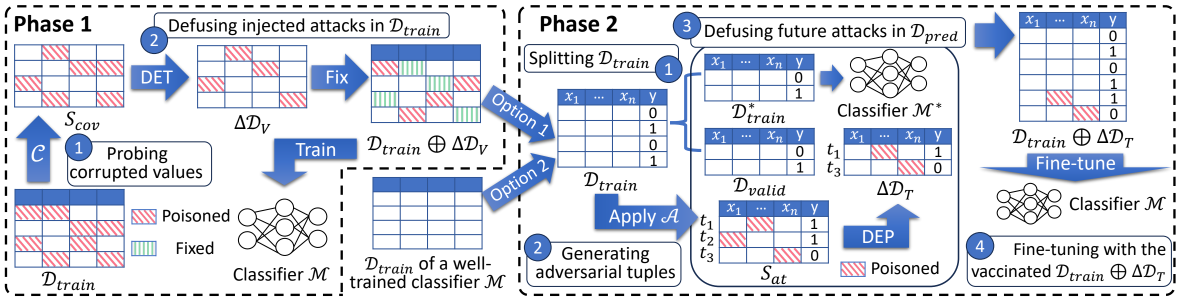


Figure 1: Overview of DE4ML

$\mathcal{R}, \mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{pred}}, \mathcal{D}^-_{\text{pred}}, \mathcal{A}_2, B_{\text{inf}}$ and a bound R_{inf}, whether there is a set $\Delta\mathcal{D}_T$ of tuple perturbations such that $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}}) \geq B_{\text{inf}}$ and $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}}) \geq R_{\text{inf}}$.

Complexity. We show that both problems are intractable, even when training the downstream ML models is in PTIME.

Theorem 2: *Both* DEAAT *and* DEAAP *are* NP*-hard.* □

Proof sketch: We show that DEAAT is NP-hard by reduction from the NP-complete problem 3SAT [33]. Given a 3SAT instance ϕ, we construct a DEAAT instance with (a) schema $\mathcal{R}$, (b) datasets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ of a single tuple of $\mathcal{R}$ such that each of its attributes encodes a variable in ϕ, (c) a model $\mathcal{M}$, and (d) a data cleaning tool C that flips the Boolean value of a corrupted attribute. We show that there exists a value perturbation set $\Delta\mathcal{D}_V$ such that $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) \geq B_{\text{train}}$ iff ϕ is satisfiable.

We show that DEAAP is NP-hard also by reduction from 3SAT. Given a 3SAT instance ϕ, we construct a DEAAP instance with (a) schema $\mathcal{R}$ with Boolean attributes and dataset $\mathcal{D}_{\text{train}}$ of $\mathcal{R}$ such that each attribute encodes a variable in ϕ, (b) datasets $\mathcal{D}_{\text{pred}}$ and $\mathcal{D}^-_{\text{pred}}$ of $\mathcal{R}$, (c) a model $\mathcal{M}$, and (d) an attack function $\mathcal{A}_2$. We show that ϕ is satisfiable iff there exists a set $\Delta\mathcal{D}_T$ such that $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}}) \geq R_{\text{inf}}$ and $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}}) \geq B_{\text{inf}}$. □

3 A DATA ENHANCING FRAMEWORK

This section proposes DE4ML, a framework that integrates ML training and data enhancing. For an ML classifier $\mathcal{M}$, given poisoned datasets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$, DE4ML enhances $\mathcal{D}_{\text{train}}$ with small sets $\Delta\mathcal{D}_V$ and $\Delta\mathcal{D}_T$ of value and tuple perturbations, respectively, such that $\mathcal{M}$ trained with the enhanced dataset is robust against attacks injected in $\mathcal{D}_{\text{train}}$ and unseen attacks at prediction time.

Overview. DE4ML takes as inputs an ML classifier $\mathcal{M}$, a schema $\mathcal{R}$, datasets $\mathcal{D}_{\text{train}}$ of $\mathcal{R}$, possibly with injected attacks, an attribute importance vector $\mathcal{V}$ of $\mathcal{R}$, a data cleaning tool C, an accuracy bound B_{inf}, possible (future) attackers $\mathcal{A}$ on $\mathcal{D}_{\text{pred}}$ and a sampling ratio α of $\mathcal{D}_{\text{train}}$ to train datamodel [44]. As a surrogate model, we use datamodel to identify critical adversarial tuples/values.

DE4ML enhances $\mathcal{D}_{\text{train}}$ and trains $\mathcal{M}$ with the enhanced dataset such that $\mathcal{M}$ is robust against attacks injected in $\mathcal{D}_{\text{train}}$ and those from $\mathcal{A}$, and retains its accuracy above B_{inf}. It works with any classifier $\mathcal{M}$, attacker $\mathcal{A}$ (*e.g.,* [8, 13, 34, 45, 46, 81, 87, 97]), and accurate data cleaning tool C (*e.g.,* Raha [66], Baran [65] and Rock [9]).

As shown in Figure 1, DE4ML has two phases. In phase 1 (for DEAAT), it identifies a set S_{cov} of corrupted values in $\mathcal{D}_{\text{train}}$, defuses attacks in $\mathcal{D}_{\text{train}}$ by using C to fix a subset $\Delta\mathcal{D}_V$ of critical values in S_{cov} that can maximumly improve the robustness and accuracy of $\mathcal{M}$, and trains model $\mathcal{M}$ with the enhanced $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$.

In phase 2 (for DEAAP), DE4ML randomly splits $\mathcal{D}_{\text{train}}$ into two disjoint subsets $\mathcal{D}^*_{\text{train}}$ and $\mathcal{D}_{\text{valid}}$ as the substitute of $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ (unavailable for training), respectively; it generates a set S_{at} of adversarial tuples via $\mathcal{A}$, identifies a subset $\Delta\mathcal{D}_T$ of critical tuples in S_{at} that can maximumly improve the robustness of $\mathcal{M}$ (trained with $\mathcal{D}^*_{\text{train}} \oplus \mathcal{D}_T$) on poisoned/unpoisoned $\mathcal{D}^*_{\text{train}}$, while maintaining its accuracy on uncorrupted $\mathcal{D}^*_{\text{train}}$; it then fine-tunes the $\mathcal{M}$ (trained in phase 1) with the tuples in $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ such that it is immune to the tuples in $\mathcal{D}_{\text{pred}}$ corrupted by $\mathcal{A}$. One may also opt to apply phase 1 (resp. 2) of DE4ML only to defuse attacks in $\mathcal{D}_{\text{train}}$ (resp. $\mathcal{D}_{\text{pred}}$).

Phase 1 (DEAAT). DE4ML integrates data enhancing and model training for DEAAT. It consists of the following steps.

(1) Probing corrupted values. Denote by $C(t, A)$ the function of a cleaning tool C that takes a tuple t and an attribute A as input, and outputs a value v amended for the value/cell $t[A]$. We identifies a maximal set S_{cov} of corrupted attribute values in $\mathcal{D}_{\text{train}}$ that C can detect and fix, *i.e.,* $S_{\text{cov}} = \{t[A] \mid \forall_{t \in \mathcal{D}_{\text{train}}, A \in \mathcal{R}}\ t[A] \neq C(t, A)\}$.

(2) Defusing injected attacks in $\mathcal{D}_{\text{train}}$. Given the set S_{cov}, DE4ML identifies a subset $\Delta\mathcal{D}_V$ of critical corrupted values in S_{cov}, such that the robustness/accuracy of $\mathcal{M}$ can be maximumly improved by fixing values in $\Delta\mathcal{D}_V$ alone. We fix adversarial data instead of removing the tuples in order to (a) enhance the distribution of $\mathcal{D}_{\text{train}}$ and (b) make $\mathcal{M}$ more robust by purposely leaving some non-critical values unfixed. To tackle this intractable problem (Theorem 2), we will develop a method in Section 4 by making use of datamodel.

Example 3: Continuing with Example 1, assume that $S_{\text{cov}} = \{t_2[\text{purpose}], t_4[\text{education}], t_5[\text{education}]\}$ by applying C on $\mathcal{D}_{\text{train}}$ (t_5 not shown in Figure 1). Assume that $\mathcal{V} = [0.6, 1, 0.9, 0.7, 0.1, 0.2, 1]$ for attributes A_1-Y in Figure 1. Given S_{cov} and $\mathcal{V}$, DE4ML generates value perturbations $\Delta\mathcal{D}_V = \{t_2[\text{purpose}], t_4[\text{education}]\}$ (see Example 5, Section 4). It enhances $\mathcal{D}_{\text{train}}$ by defusing the attacks in $\Delta\mathcal{D}_V$, *e.g.,* changing $t_2[\text{purpose}]$ (resp. $t_4[\text{education}]$) from `education` (resp. `Ph.D`) to `car` (resp. `bachelor`). The fixes are identified by association analysis of label Y and attributes A_j [9]. It then trains $\mathcal{M}$ with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_v$. □

Phase 2 (DEAAP). In this phase, DE4ML enhances $\mathcal{D}_{\text{train}}$ with a small set $\Delta\mathcal{D}_T$ of adversarial tuples from $\mathcal{A}$ and fine-tunes $\mathcal{M}$ with the enhanced $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$, such that the $\mathcal{M}$ can defend against attacks in $\mathcal{D}_{\text{pred}}$ unseen in $\mathcal{D}_{\text{train}}$. It consists of the following steps.

(1) Splitting training data. DE4ML first randomly split $\mathcal{D}_{\text{train}}$ into two disjoint subsets $\mathcal{D}^*_{\text{train}}$ and $\mathcal{D}_{\text{valid}}$ in $\eta : 10 - \eta$ ratio, where we set $7 \leq \eta \leq 9$ following the common practice in ML [47].

Input: An ML model $\mathcal{M}$, a schema $\mathcal{R}$, a training dataset $\mathcal{D}_{\mathsf{train}}$ of $\mathcal{R}$,
an attribute importance vector $\mathcal{V}$ of $\mathcal{R}$,
a data cleaning tool C, a possible attacker $\mathcal{A}$ on $\mathcal{D}_{\mathsf{pred}}$,
an accuracy bound B_{inf} and a sampling ratio α.
Output: A robust ML model $\mathcal{M}$ and an enhanced training set $\mathcal{D}_{\mathsf{train}}$.
/* Phase 1: defusing injected attacks in $\mathcal{D}_{\mathsf{train}}$ for DEAAT */
1. $S_{\mathsf{cov}} := \mathsf{ProbeCorruptedValue}(\mathcal{D}_{\mathsf{train}}, C)$;
2. $\Delta\mathcal{D}_V := \mathsf{DET}(\mathcal{D}_{\mathsf{train}}, \mathcal{M}, \mathcal{R}, \mathcal{V}, S_{\mathsf{cov}}, \alpha)$;
3. $\mathcal{D}_{\mathsf{train}} := \mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_V$; train $\mathcal{M}$ with the enhanced $\mathcal{D}_{\mathsf{train}}$;
/* Phase 2: defusing future attacks in $\mathcal{D}_{\mathsf{pred}}$ for DEAAP */
4. $(\mathcal{D}^*_{\mathsf{train}}, \mathcal{D}_{\mathsf{valid}}) := \mathsf{SplitData}(\mathcal{D}_{\mathsf{train}})$;
5. $S_{\mathsf{at}} := \mathsf{GenerateAdversarialTuple}(\mathcal{D}_{\mathsf{train}}, \mathcal{M}, \mathcal{A})$;
6. $\Delta\mathcal{D}_T := \mathsf{DEP}(\mathcal{D}^*_{\mathsf{train}}, \mathcal{D}_{\mathsf{valid}}, S_{\mathsf{at}}, \mathcal{M}, \mathcal{R}, \mathcal{A}, B_{\mathsf{inf}}, \alpha)$;
7. $\mathcal{D}_{\mathsf{train}} := \mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_T$; fine-tune $\mathcal{M}$ with the vaccinated $\mathcal{D}_{\mathsf{train}}$;
8. **return** $(\mathcal{M}, \mathcal{D}_{\mathsf{train}})$;

Figure 2: The workflow of DE4ML

(2) Generating adversarial tuples. Given $\mathcal{D}_{\mathsf{train}}$ and an attacker $\mathcal{A}$, DE4ML invokes $\mathcal{A}$ to attack the features of all tuples in $\mathcal{D}_{\mathsf{train}}$, and obtains a maximal set S_{at} of adversarial tuples, where $|S_{\mathsf{at}}| \leq |\mathcal{D}_{\mathsf{train}}|$ since $\mathcal{A}$ may not attack all tuples in $\mathcal{D}_{\mathsf{train}}$.

(3) Defusing future attacks in $\mathcal{D}_{\mathsf{pred}}$. Given $\mathcal{M}$, S_{at} and $\mathcal{D}_{\mathsf{train}} = (\mathcal{D}^*_{\mathsf{train}}, \mathcal{D}_{\mathsf{valid}})$, DE4ML identifies a subset $\Delta\mathcal{D}_T$ of critical tuples in S_{at}, such that $\mathcal{M}$ trained with $\mathcal{D}^*_{\mathsf{train}} \oplus \Delta\mathcal{D}_T$ can defend against attacks in corrupted $\mathcal{D}_{\mathsf{valid}}$ without degrading its accuracy on uncorrupted $\mathcal{D}_{\mathsf{valid}}$; the objective is to fine-tune $\mathcal{M}$ with $\mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_T$ for its robustness on the corrupted $\mathcal{D}_{\mathsf{pred}}$ (*i.e.,* $\mathcal{D}^-_{\mathsf{pred}}$), and meanwhile retains its accuracy on uncorrupted $\mathcal{D}_{\mathsf{pred}}$. We will address this intractable problem (Theorem 2) in Section 5 by using datamodel.

(4) Fine-tuning with the vaccinated data $\mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_T$. Given the model $\mathcal{M}$ trained in phase 1 and the set $\mathcal{D}_T$ returned from phase 2, DE4ML fine-tunes $\mathcal{M}$ with the enhanced $\mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_T$.

Example 4: Continuing with Example 3, after $\mathcal{M}$ is trained with $\mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_V$, DE4ML randomly splits the enhanced $\mathcal{D}_{\mathsf{train}}$ into disjoint $\mathcal{D}^*_{\mathsf{train}}$ and $\mathcal{D}_{\mathsf{valid}}$, *e.g.,* $\mathcal{D}^*_{\mathsf{train}} = \{t_1, t_3, t_4, \dots\}$ and $\mathcal{D}_{\mathsf{valid}} = \{t_2, t_5, \dots\}$; it generates $S_{\mathsf{at}} = \{t_1', t_2', t_3'\}$ by attacking $\mathcal{D}_{\mathsf{train}}$ with $\mathcal{A}$, where t_1' (resp. t_2' and t_3') has the same attributes and label as t_1 (resp. t_2 and t_3) but wrong education *bachelor* (resp. *Ph.D* and *bachelor*). Given D^*_{train}, $\mathcal{D}_{\mathsf{valid}}$, S_{at} and $\mathcal{A}$, DE4ML generates tuple perturbations $\Delta\mathcal{D}_T = \{t_2'\}$ (see Example 6, Section 5). It adds tuples of $\Delta\mathcal{D}_T$ to $\mathcal{D}_{\mathsf{train}}$, and fine-tunes $\mathcal{M}$ with $\mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_T$; this makes $\mathcal{M}$ robust for attacks in $\mathcal{D}^-_{\mathsf{pred}}$, *e.g.,* t_b and t_d in Table 1. □

Workflow. The entire process is outlined in Figure 2. After phases 1 (lines 1-3) and 2 (lines 4-7), DE4ML returns as output an enhanced training dataset $\mathcal{D}_{\mathsf{train}}$ with value and tuple perturbations, and a robust $\mathcal{M}$ trained and fine-tuned with the enhanced $\mathcal{D}_{\mathsf{train}}$.

Complexity. DE4ML takes $O(c_C + c_{\mathsf{DET}} + c_{\mathsf{train}} + |\mathcal{D}_{\mathsf{train}}| + c_{\mathsf{at}} + c_{\mathsf{DEP}} + c_{\mathsf{ft}})$ time, where c_C is the cost of fixing $\mathcal{D}_{\mathsf{train}}$ via C, c_{DET} is for identifying value perturbations $\Delta\mathcal{D}_V$ in S_{cov}, c_{train} is for training $\mathcal{M}$ with $\mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_V$, $O(|\mathcal{D}_{\mathsf{train}}|)$ is for splitting $\mathcal{D}_{\mathsf{train}}$ into $\mathcal{D}^*_{\mathsf{train}}$ and $\mathcal{D}_{\mathsf{valid}}$, c_{at} is the cost of generating the set S_{at} with $\mathcal{A}$, c_{DEP} is for selecting tuple perturbations $\Delta\mathcal{D}_T$ in S_{at}, and c_{ft} is for fine-tuning $\mathcal{M}$ with the enhanced $\mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_T$. The most costly factor is c_{DET}, followed by c_{DEP}. We will see in Section 6 that DE4ML is efficient when $\mathcal{M}$ and datamodel can be trained efficiently.

4 DEBUGGING TRAINING DATA

This section develops the *data enhanced training* algorithm of DE4ML, denoted by DET, to identify a set of critical value perturbations to $\mathcal{D}_{\mathsf{train}}$ for training a classification model $\mathcal{M}$. It aims to mitigate the impact of poisoned tuples in $\mathcal{D}_{\mathsf{train}}$ on the training of $\mathcal{M}$.

Problem. Given a corrupted dataset $\mathcal{D}_{\mathsf{train}}$ of schema R, a classifier $\mathcal{M}$, and a data cleaning tool C, we want to identify a small subset $\Delta\mathcal{D}_V$ of critical value perturbations such that the robustness and accuracy of $\mathcal{M}$ can be maximumly improved by the perturbations in $\Delta\mathcal{D}_V$. As shown in Theorem 2, this problem is NP-hard.

Approach. It is hard to find a perfect $\Delta\mathcal{D}_V$. Worse yet, it is nontrivial to distinguish adversarial attacks from (innocent) errors. DET approaches the problem by identifying critical values cv that are (a) abnormal, *i.e.,* the tuples with cv have a high training loss; (b) influential, *i.e.,* the tuples with cv highly impact the prediction of $\mathcal{M}$ on other tuples in $\mathcal{D}_{\mathsf{train}}$; and (c) imperceptible, values cv appear in imperceptible attributes of tuples. Note that a tuple with high training loss (*i.e.,* criterion (a)) may not highly impact the prediction of other tuples (*i.e.,* (b)), *e.g.,* an isolated tuple with high loss [50].

DET iteratively identifies such cv based on the three criteria starting from the least important attributes (*i.e.,* those with the lowest importance score v_j) in the ascending order, as an attacker tends to inject more adversarial noise into less important attributes [8, 13]. DET terminates when no more attribute values meet criteria (a)-(c). We iteratively search cv based on $\mathcal{M}$'s error signals since defusing part of $\mathcal{D}_{\mathsf{train}}$ can affect $\mathcal{M}$ on criteria (a)-(b) for the others.

More specifically, DET has the following steps in each round.

*Identifying abnormal imperceptible attributes (*AIT*)*. As observed in [96], abnormal tuples usually have relatively high training loss $l(\cdot)$ in the first few e (*e.g.,* 5) epochs. Hence DET monitors the loss of each tuple $t \in \mathcal{D}_{\mathsf{train}}$ in few epochs based on the loss function of $\mathcal{M}$, and computes the average loss $\hat{l}(t)$ of each tuple $t \in \mathcal{D}_{\mathsf{train}}$, where $\hat{l}(t) = \frac{1}{e} \cdot \sum l(t)$. It ranks the tuples by their average loss in the descending order, and obtains a set AT of abnormal tuples by retrieving the top-50% of tuples in the ranked $\mathcal{D}_{\mathsf{train}}$. We consider top-50% tuples as an attacker typically corrupts at most 50% of the tuples in $\mathcal{D}_{\mathsf{train}}$ (Section 2.1). Denote by $\mathcal{D}_{\mathsf{cot}}$ the set of corrupted tuples in $\mathcal{D}_{\mathsf{train}}$ that have attribute values in S_{cov}. Given $\mathcal{D}_{\mathsf{cot}}$, S_{cov} and the attribute important vector $\mathcal{V}$, DET finds a set $\hat{\mathcal{D}}_{\mathsf{cot}}$ of corrupted tuples by selecting $t \in \mathcal{D}_{\mathsf{cot}}$ with $t[A] \in S_{\mathsf{cov}}$, where A is the (imperceptible) attribute with the smallest $v_j \in \mathcal{V}$, including the label Y, which is also an attribute that may be corrupted.

After this step, DET identifies a set AIT of the abnormal tuples with corrupted imperceptible attributes, *i.e.,* $\mathsf{AIT} = \mathsf{AT} \cap \hat{\mathcal{D}}_{\mathsf{cot}}$.

*Identifying influential tuples (*IT*)*. We adopt datamodel [44] to capture the correlation among tuples in $\mathcal{D}_{\mathsf{tuple}}$ and hence, identify influential tuples. A datamodel $\hat{\mathcal{M}}_t$ is a surrogate of $\mathcal{M}$ that "predicts" $\mathcal{M}(t)$ based on tuples without t; it is a linear function and works better than, *e.g.,* influence function [50] and shapley value [21], for assessing the impact of other tuples on the prediction $\mathcal{M}(t)$ (see [44] for details). In order to train $\hat{\mathcal{M}}_t$ for each $t \in \mathcal{D}_{\mathsf{train}}$, we generate a set $\mathcal{D}_t$ of training examples $(\mathbb{1}_{S_i}, \mathcal{M}_i(t))$, where S_i is a randomly sampled subset of $\mathcal{D}_{\mathsf{train}}$, $\mathbb{1}_{S_i} \in \{0,1\}^{|\mathcal{D}_{\mathsf{train}}|}$ is a $|\mathcal{D}_{\mathsf{train}}|$-dimensional vector that indicates which tuples in $\mathcal{D}_{\mathsf{train}}$ are present in S_i (*i.e.,* 0/1 for the absence/presence of a tuple), and $\mathcal{M}_i(t)$ is the prediction of a model $\mathcal{M}_i$ (trained with S_i) on tuple t.

More specifically, given a sampling ratio α, DET first finds a set S_{dm} of subsets S_i of $\mathcal{D}_{\mathsf{train}}$, where the size $|S_{\mathsf{dm}}|$ of S_{dm} is usually small (*e.g.,* 100 [44]), and each $S_i \in S_{\mathsf{dm}}$ is obtained by randomly sampling α% of tuples in $\mathcal{D}_{\mathsf{train}}$. DET then trains $|S_{\mathsf{dm}}|$ models $\mathcal{M}_i$ ($1 \le i \le |S_{\mathsf{dm}}|$), each is trained with a set $S_i \in S_{\mathsf{dm}}$. For each tuple $t \in \mathcal{D}_{\mathsf{train}}$, DET predicts t with all trained $\mathcal{M}_i$ and obtains the training set $\mathcal{D}_t$ (with examples $(\mathbb{1}_{\mathsf{S_i}}, \mathcal{M}_i(t))$) of the datamodel $\hat{\mathcal{M}}_t$. Finally, DET trains $\hat{\mathcal{M}}_t$ with $\mathcal{D}_t$ for each tuple $t \in \mathcal{D}_{\mathsf{train}}$.

Using the trained datamodels, DE4ML identifies influential tuples as follows. Denote by $\hat{\theta}_t$ the learned parameter vector of a datamodel $\hat{\mathcal{M}}_t$, where $|\hat{\theta}_t| = |\mathcal{D}_{\mathsf{train}}|$. Here $\hat{\theta}_t$ contains $|\mathcal{D}_{\mathsf{train}}|$ values $\hat{\theta}_t[i] \in [-1, 1]$ ($1 \le i \le |\mathcal{D}_{\mathsf{train}}|$), where each $\hat{\theta}_t[i]$ indicates to which extent a tuple (with id i) supports the prediction $\mathcal{M}(t)$; as observed in [44], the tuples with the highest (resp. lowest) $\hat{\theta}_t[i]$ share the most similar attributes and identical (resp. distinct) labels to t. Moreover, an adversary $\mathcal{A}_1$ may inject attacks in attributes or labels following probability $v_j \in \mathcal{V}$ (Section 2.1), and as a result, it may mislead the training of $\mathcal{M}$ to false positive (due to tuples with poisoned attributes) or false negative (due to tuples with poisoned labels) prediction $\mathcal{M}(t)$ on tuple t; this indicates that we need to identify the influential tuples with poisoned attributes (*i.e.,* tuples aligned with the largest $\hat{\theta}_t[i]$) and poisoned labels (*i.e.,* tuples linked with the lowest $\hat{\theta}_t[i]$). Hence, for each $t \in \mathcal{D}_{\mathsf{train}}$, we obtain a set IT_t of influential tuples in $\mathcal{D}_{\mathsf{train}} \setminus \{t\}$ that largely affect the prediction $\mathcal{M}(t)$ by retrieving the tuples in the top-(50-ζ)% of highest (resp. top-ζ% lowest) $\hat{\theta}_t[i]$ of $\hat{\theta}_t$, where $\zeta = \frac{50\cdot(1-v_Y)}{\sum_{v_j\in\mathcal{V}} 1-v_j}$.

Denote by IT the union of IT_t for all tuples t in $\mathcal{D}_{\mathsf{train}}$, *i.e.,* $\mathsf{IT} = \cup_{t\in\mathcal{D}_{\mathsf{train}}}\mathsf{IT}_t$. Intuitively, IT contains all tuples in $\mathcal{D}_{\mathsf{train}}$ that positively (resp. negatively) affect the prediction of $\mathcal{M}$.

Identifying critical values (S_{cv}). Recall that an attacked value is considered "critical" if it is abnormal, influential and imperceptible. Given the set AIT of abnormal tuples with corrupted imperceptible attributes and the set IT of influential tuples obtained as above, we identify a set S_{cv} of critical values in S_{cov} by only considering the values of tuples in a filtered set FS, where $\mathsf{FS} = \mathsf{AIT} \cap \mathsf{IT}$.

Conducting value perturbations. Given the identified S_{cv} of critical values in this round, DET defuses $\mathcal{D}_{\mathsf{train}}$ by fixing the values in S_{cv} via data cleaning tool C. The defused/enhanced $\mathcal{D}_{\mathsf{train}}$ is then forwarded to the next round of DET for further enhancement.

Algorithm. We now develop algorithm DET, as shown in Figure 3. DET takes as input $\mathcal{R}, \mathcal{V}, \mathcal{M}, \mathcal{D}_{\mathsf{train}}, S_{\mathsf{cov}}, C$ and α; it outputs a set $\Delta\mathcal{D}_V$ of critical values in S_{cov} for training classifier $\mathcal{M}$.

Algorithm DET first initializes $\Delta\mathcal{D}_V$ and S_{cv} (line 1). It then iteratively enriches $\Delta\mathcal{D}_V$ with newly identified S_{cv} (lines 2-10). DET identifies AIT based on attribute important vector $\mathcal{V}$ and error signals from $\mathcal{M}$ (line 3), trains the set $S_{\hat{\mathcal{M}}}$ of datamodels $\hat{\mathcal{M}}_t$ with the predictions $\mathcal{M}(t)$ of $\mathcal{M}$ on each $t \in \mathcal{D}_{\mathsf{train}}$ (line 4), identifies IT with the trained datamodels in $S_{\hat{\mathcal{M}}}$ (line 5), and obtains S_{cv} by intersecting AIT and IT (line 6). It then defuses $\mathcal{D}_{\mathsf{train}}$ by fixing poisoned values in S_{cv} with data cleaning tool C (line 9), and updates S_{cov} and $\Delta\mathcal{D}_V$ (line 10). The iteration terminates when no more critical values in S_{cov} can be detected (lines 7-8) or after all attributes in $\mathcal{R}$ are checked (line 2). Finally, DET returns $\Delta\mathcal{D}_V$ (line 10).

Input: $\mathcal{D}_{\mathsf{train}}$, $\mathcal{M}$, $\mathcal{R}$, and $\mathcal{V}$ as in Figure 2, a set S_{cov} of corrupted attribute values, and a sampling ratio α of $\mathcal{D}_{\mathsf{train}}$ to train datamodel.
Output: A set $\Delta\mathcal{D}_V$ of value perturbations to $\mathcal{D}_{\mathsf{train}}$.

1. $\Delta\mathcal{D}_V := \phi$; $S_{\mathsf{cv}} := \phi$;
2. **for** attributes $A \in \mathcal{R}$ sorted by $v_j \in \mathcal{V}$ in an ascending order **do**
3. $\mathsf{AIT} := \mathsf{IdentifyAIT}(\mathcal{D}_{\mathsf{train}}, \mathcal{D}_{\mathsf{dv}}, \mathcal{V})$;
4. $S_{\hat{\mathcal{M}}} := \mathsf{TrainDataModel}(\mathcal{D}_{\mathsf{train}}, \mathcal{M}, \alpha)$;
5. $\mathsf{IT} := \mathsf{IdentifyInfluentialTuples}(S_{\hat{\mathcal{M}}})$;
6. $S_{\mathsf{cv}} := \mathsf{ObtainCriticalValues}(\mathsf{AIT}, \mathsf{IT})$;
7.   **if** $S_{\mathsf{cv}} = \phi$ **then**
8.   **break**;
9. $\mathcal{D}_{\mathsf{train}} := \mathsf{ValuePerturbation}(\mathcal{D}_{\mathsf{train}}, S_{\mathsf{cv}})$;
10. $S_{\mathsf{cov}} := S_{\mathsf{cov}} \setminus S_{\mathsf{cv}}$; $\Delta\mathcal{D}_V := \Delta\mathcal{D}_V \cup S_{\mathsf{cv}}$;
11. **return** $\Delta\mathcal{D}_V$;

Figure 3: Algorithm DET

Remark. (1) Fixing an imperceptible attribute may alter the correlations between tuples in $\mathcal{D}_{\mathsf{train}}$; hence we need to retain datamodels $\hat{\mathcal{M}}_t$ in each round to capture these updates to accurately identify S_{cv} in other attributes. (2) DET needs at most $|\mathcal{R}|$ rounds (one imperceptible attribute per round) when all attributes in $\mathcal{R}$ are imperceptible (*i.e.,* no attribute in $\mathcal{R}$ is manually checked). (3) DET can improve the accuracy of $\mathcal{M}$ trained with $\mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_V$ when the tool C is accurate (*i.e.,* the initially identified S_{cov} is precise), since the values in $\mathcal{D}_V$ are purposely perturbed by attacker $\mathcal{A}_1$ to mislead the training of $\mathcal{M}$ towards a wrong data distribution (*i.e.,* low accuracy); defusing the attacks via C can help align $\mathcal{M}$'s decision boundary with the true data distribution (*i.e.,* high accuracy).

Example 5: Continuing with Example 3, DET takes S_{cov} and $\mathcal{V}$ as input, where $S_{\mathsf{cov}} = \{t_2[\mathsf{purpose}], t_4[\mathsf{education}], t_5[\mathsf{education}]\}$ and $\mathcal{V} = [0.6, 1, 0.9, 0.7, 0.1, 0.2, 1]$. Initially, $\Delta\mathcal{D}_V = \phi$ and $S_{\mathsf{cv}} = \phi$. In the first round, DET picks imperceptible attribute education (*i.e.,* A_5) since its importance score (*i.e.,* 0.1) is the lowest in $\mathcal{V}$. Suppose that we find $\mathsf{AIT} = \{t_4, t_5\}$ and $\mathsf{IT} = \{t_2, t_3, t_4\}$, where each $t' \in \mathsf{IT}$ is an influential tuple that affects the prediction of $\mathcal{M}$ on a different tuple t, *i.e.,* the $\hat{\theta}_t[i]$ aligned with t' is among the top-(50-ζ)% highest ones in the learned vector $\hat{\theta}_t$ of datamodel $\hat{\mathcal{M}}_t$ for t, and $t \ne t'$ (here $\zeta = \frac{50\cdot(1-v_Y)}{\sum_{v_j\in\mathcal{V}} 1-v_j} = 0$). We obtain $S_{\mathsf{cv}} = \{t_4[\mathsf{education}]\}$ by intersecting AIT and IT. We defuse $\mathcal{D}_{\mathsf{train}}$ with value perturbations in S_{cv}, update $S_{\mathsf{cov}} = \{t_2[\mathsf{purpose}], t_5[\mathsf{education}]\}$ and $\Delta\mathcal{D}_V = \{t_4[\mathsf{education}]\}$. DET processes these in the next round. This process proceeds until $S_{\mathsf{cv}} = \phi$. As the result of phase 1, DE4ML returns $\Delta\mathcal{D}_V = \{t_2[\mathsf{purpose}], t_4[\mathsf{education}]\}$ and $\mathcal{M}$ trained with $\mathcal{D}_{\mathsf{train}} \oplus \Delta\mathcal{D}_V$ □

Analyses. DET approaches the intractable DEAAT by distinguishing poisoned data from noise based on datamodel and $\mathcal{M}$'s error signals, and identifying a small set $\Delta\mathcal{D}_V$ of critical adversarial values that are abnormal, influential and imperceptible, such that fixing values in $\Delta\mathcal{D}_V$ can help align $\mathcal{M}$'s decision boundary with the true data distribution, which contributes to a higher accuracy.

DET takes $O(|\mathcal{R}| \cdot (c_{\mathsf{AIT}} + |\mathcal{D}_{\mathsf{train}}| \cdot c_{\hat{\mathcal{M}}_t} + |\mathcal{D}_{\mathsf{train}}|^2 + |\mathcal{D}_{\mathsf{train}}| + |\mathcal{D}_{\mathsf{train}}| \cdot |R| + |S_{\mathsf{cov}}| + 2 \cdot |\mathcal{D}_{\mathsf{train}}|))$ time, where c_{AIT} is the sum of costs of (a) obtaining training loss of tuples in $\mathcal{D}_{\mathsf{train}}$ in few epochs, (b) ranking tuples in $\mathcal{D}_{\mathsf{train}}$ based on their training losses, (c) retrieving $\hat{\mathcal{D}}_{\mathsf{cot}}$ based on $\mathcal{D}_{\mathsf{dv}}$ and $\mathcal{V}$, and (d) computing $\mathsf{AIT} = \mathsf{AT} \cap \hat{\mathcal{D}}_{\mathsf{cot}}$. Here $c_{\hat{\mathcal{M}}_t}$ is the cost of training a linear datamodel $\hat{\mathcal{M}}_t$ for a tuple t,

$O(|\mathcal{D}_{\mathrm{train}}|^2)$ is for computing $\mathsf{IT} = \cup_{\forall t \in \mathcal{D}_{\mathrm{train}}} \mathsf{IT}_t$, $O(|\mathcal{D}_{\mathrm{train}}|)$ is for constructing the $|\mathcal{D}_{\mathrm{train}}|$-dimensional unit vectors of AIT and IT, $O(|\mathcal{D}_{\mathrm{train}}| \cdot |\mathcal{R}|)$ is for conducting value perturbations, $O(|S_{\mathrm{cov}}|)$ is for updating S_{cov} with S_{cv}, and $O(2 \cdot |\mathcal{D}_{\mathrm{train}}|)$ is for updating $\Delta\mathcal{D}_V$ with S_{cv}. The most costly factor is $c_{\hat{\mathcal{M}}_t}$. We will see in Section 6 that DET is efficient for most models; it terminates less than $|\mathcal{R}|$ rounds.

5 ADDING ADVERSARIAL TUPLES

This section develops the algorithm of DE4ML for data enhanced prediction, denoted by DEP, to find tuple perturbations to $\mathcal{D}_{\mathrm{train}}$ for fine-tuning ML classifiers $\mathcal{M}$. It is to make $\mathcal{M}$ robust against attacks in $\mathcal{D}^-_{\mathrm{pred}}$ while retaining its accuracy on unpoisoned $\mathcal{D}_{\mathrm{pred}}$.

Problem. Given an unpoisoned/attack-defused training set $\mathcal{D}^*_{\mathrm{train}}$ of schema $\mathcal{R}$ (provided by DE4ML in Section 3), an unpoisoned testing set $\mathcal{D}_{\mathrm{valid}}$ of $\mathcal{R}$ (provided by DE4ML), an ML classifier $\mathcal{M}$, an attacker $\mathcal{A}$ (*i.e.,* $\mathcal{A}_2$ in Section 2.2), a maximal set S_{at} of adversarial tuples generated via $\mathcal{A}$, and an accuracy bound B_{inf}, we want to identify a small set $\Delta\mathcal{D}_T$ of critical tuples in S_{at}, such that enhancing $\mathcal{D}^*_{\mathrm{train}}$ with perturbations in $\Delta\mathcal{D}_T$ can maximumly improve the robustness of $\mathcal{M}$ on $\mathcal{D}^-_{\mathrm{valid}}$ ($\mathcal{D}_{\mathrm{valid}}$ poisoned by $\mathcal{A}$) while retaining its accuracy on unpoisoned $\mathcal{D}_{\mathrm{valid}}$. This is intractable by Theorem 2.

Naive approach. Given the set S_{at} of candidate tuple perturbations to $\mathcal{D}^*_{\mathrm{train}}$, one may want to select an optimal $\Delta\mathcal{D}_T$ out of $2^{|S_{\mathrm{at}}|}$ candidate sets to boost the performance of $\mathcal{M}$ on both poisoned $\mathcal{D}^-_{\mathrm{valid}}$ and unpoisoned $\mathcal{D}_{\mathrm{valid}}$. However, this approach is too costly to be practical since $\mathcal{M}$ needs to be retrained/fine-tuned $2^{|S_{\mathrm{at}}|}$ times.

Our approach. In contrast, DEP iteratively selects critical tuples ct in S_{at} that are (a) influential, *i.e.,* they improve the robustness of $\mathcal{M}$ on attacked tuples in poisoned $\mathcal{D}^-_{\mathrm{valid}}$; (b) harmless, *i.e.,* they do not largely degrade the accuracy of $\mathcal{M}$ on unpoisoned $\mathcal{D}_{\mathrm{valid}}$; and (c) diversified, they are from different groups (*e.g.,* `bachelor` and `Ph.D`) and enable $\mathcal{M}$ to defend against attacks to different groups of $\mathcal{D}^-_{\mathrm{valid}}$. DEP terminates when either no more tuples can further improve the robustness of $\mathcal{M}$ on $\mathcal{D}^-_{\mathrm{valid}}$, or adding more tuples to $\mathcal{D}^*_{\mathrm{train}}$ may degrade the accuracy of $\mathcal{M}$ on unpoisoned $\mathcal{D}_{\mathrm{valid}}$ (*i.e.,* $< B_{\mathrm{inf}}$). We iteratively search ct based on the distance between tuple embeddings learned via datamodel, since adding a tuple to $\mathcal{D}^*_{\mathrm{train}}$ may alter the criteria (a)-(c) above for other tuples in S_{at}.

Denote by $\mathcal{M}^*$ the $\mathcal{M}$ (trained in phase 1 of DE4ML) fine-tuned with $\mathcal{D}^*_{\mathrm{train}} \oplus$ ct in each round; we use $\mathcal{M}^*$ to find the final $\Delta\mathcal{D}_T$, such that $\mathcal{M}$ fine-tuned with $\mathcal{D}_{\mathrm{train}} \oplus \Delta\mathcal{D}_T$ is robust to attacked tuples in $\mathcal{D}^-_{\mathrm{pred}}$. DEP conducts the following steps in each round.

Attacking testing data with $\mathcal{A}$ (PVD). DEP generates a set PVD of few (*e.g.,* ≤ 5) different poisoned validation datasets $\mathcal{D}^-_{\mathrm{valid}}$, where each $\mathcal{D}^-_{\mathrm{valid}}$ is a poisoned instance of $\mathcal{D}_{\mathrm{valid}}$ attacked with $\mathcal{A}$ for deceiving $\mathcal{M}$ and $|\mathcal{D}^-_{\mathrm{valid}}| = |\mathcal{D}_{\mathrm{valid}}|$. We consider multiple poisoned instances to comprehensively evaluate the effectiveness of $\Delta\mathcal{D}_T$, such that the final $\Delta\mathcal{D}_T$ can make $\mathcal{M}$ robust on the future $\mathcal{D}^-_{\mathrm{pred}}$.

Evaluating $\mathcal{M}^*$ (S_{mt}). Given PVD, $\mathcal{D}_{\mathrm{valid}}$ and the set S_{at} of adversarial tuples, DEP applies $\mathcal{M}^*$ to predict the labels of tuples in $\mathcal{D}_{\mathrm{valid}}$, S_{at} and each $\mathcal{D}^-_{\mathrm{valid}}$ in PVD, respectively, and obtains (a) $\mathrm{acc}_{\mathrm{cur}}$, the accuracy of $\mathcal{M}^*$ on $\mathcal{D}_{\mathrm{valid}}$, (b) $\mathrm{rob}_{\mathrm{cur}}$, the average robustness/accuracy of $\mathcal{M}^*$ on poisoned validation datasets $\mathcal{D}^-_{\mathrm{valid}} \in$ PVD, and (c) S_{mt}, the set of misclassified tuples in S_{at} in this round.

Training datamodels ($\hat{\mathcal{M}}^*_t$). Similar to DET in Section 4, DEP first selects a small set S_{dm} of datasets S_i for training $|S_{\mathrm{dm}}|$ models $\mathcal{M}^*_i$, where each $S_i \in S_{\mathrm{dm}}$ is obtained by randomly sampling α% of tuples in $\mathcal{D}^*_{\mathrm{train}}$ for sampling ratio α. It then predicts the label of each tuple t in $\mathcal{D}^*_{\mathrm{train}} \cup S_{\mathrm{at}}$ with all trained $\mathcal{M}^*_i$, and generates the training set $\mathcal{D}_t$ (with examples $(\mathbb{1}_{S_i}, \mathcal{M}^*_i(t))$) of the datamodel $\hat{\mathcal{M}}^*_t$ for predicting t. DEP next trains $\hat{\mathcal{M}}^*_t$ with $\mathcal{D}_t$ for each t in $\mathcal{D}^*_{\mathrm{train}} \cup S_{\mathrm{at}}$, and projects t into a $|\mathcal{D}^*_{\mathrm{train}}|$-dimensional vector $\hat{\theta}^*_t$. We train datamodels for tuples in both $\mathcal{D}^*_{\mathrm{train}}$ and S_{at} in order to map them into the same embedding space, so as to compute the distance between tuples in $\mathcal{D}^*_{\mathrm{train}} \cup S_{\mathrm{at}}$ and thus select the critical ones in S_{at} that meet the criteria (a)-(c) above (see below).

Selecting critical tuples (S_{ct}). Given the trained datamodel $\hat{\mathcal{M}}^*_t$ for each tuple t in $\mathcal{D}^*_{\mathrm{train}} \cup S_{\mathrm{at}}$, the model parameter $\hat{\theta}^*_t$ can be treated as a $|\mathcal{D}^*_{\mathrm{train}}|$-dimensional embedding of t [44]. DEP adopts k-means [64] to cluster all tuples in $\mathcal{D}^*_{\mathrm{train}} \cup S_{\mathrm{at}}$ into k ($= |S_{\mathrm{mt}}|$) groups g_i ($1 \leq i \leq |S_{\mathrm{mt}}|$), where the initial centers are set as the embeddings $\hat{\theta}_t$ of misclassified tuples t in S_{mt}; here we set $k = |S_{\mathrm{mt}}|$ in case that each misclassified tuple in S_{mt} is an isolated tuple (*i.e.,* tuple pairs in S_{mt} are away from each other). It then sorts groups g_i based on *difference ratio* g_i.dr of g_i in the decreasing order, where $g_i.\mathrm{dr} = \frac{|S_{\mathrm{mt}} \cap g_i| - |(S_{\mathrm{at}} \setminus S_{\mathrm{mt}}) \cap g_i|}{|g_i|}$ is the ratio of difference between misclassified and correctly classified tuples in $S_{\mathrm{at}} \cap g_i$ to all tuples in g_i; intuitively, $\mathcal{M}^*$ performs the worst when g_i has the highest g_i.dr; this indicates that g_i should be prioritized to be enhanced. We assume *w.l.o.g.* that g_i is the top-i ranked group.

Given the sorted g_i, DEP selects a subset S_{ct} of critical tuples from $S_{\mathrm{mt}} \cap g_1$, where tuples $t \in S_{\mathrm{ct}}$ with misclassified labels "y" have the top-$|S_{\mathrm{ct}}|$ minimal distances (calculated based on embedding $\hat{\theta}^*_t$) to the tuples in $\mathcal{D}^*_{\mathrm{train}}$ with correct labels "y". Intuitively, tuples in S_{ct} are the most troublesome ones for $\mathcal{M}^*$ since they have attributes similar to tuples in $\mathcal{D}^*_{\mathrm{train}}$ but are misclassified by $\mathcal{M}^*$ (*i.e.,* $\mathcal{M}^*$ has a low prediction confidence on the tuples in $\mathcal{D}^*_{\mathrm{train}}$ surrounding the tuples in S_{ct}). Thus, tuples in S_{ct} not only (a) enable $\mathcal{M}^*$ to better discriminate tuples in the 'low confidence" area (*i.e.,* influential), but also (b) have (relatively) less impact on tuples away from this area (*i.e.,* harmless), and moreover (c) deescalate the priority of (the most brittle group) g_1 (*i.e.,* diversified). We determine the size of S_{ct} by analyzing the affected ratio g_1 (see [2] for details).

Conducting tuple perturbations. Given the S_{ct} identified in this round, DEP enhances $\mathcal{D}^*_{\mathrm{train}}$ with S_{ct}. It fine-tunes model $\mathcal{M}^*$ with the enhanced $\mathcal{D}^*_{\mathrm{train}}$ and repeats the process.

Algorithm. Algorithm DEP is given in Figure 4. It takes as input $\mathcal{R}$, $\mathcal{M}$, $\mathcal{D}^*_{\mathrm{train}}$, $\mathcal{D}_{\mathrm{valid}}$, $\mathcal{A}$, B_{inf}, S_{at} and α. It returns a set $\Delta\mathcal{D}_T$ of critical tuples to fine-tune classifier $\mathcal{M}$ that is trained in phase 1 of DE4ML.

Algorithm DEP first initializes $\mathcal{M}^*$, $\Delta\mathcal{D}_T$, S_{ct}, $\mathrm{acc}_{\mathrm{pre}}$, $\mathrm{rob}_{\mathrm{pre}}$, $\mathrm{acc}_{\mathrm{cur}}$ and $\mathrm{rob}_{\mathrm{cur}}$ (line 1), where $\mathrm{acc}_{\mathrm{pre}}$ (resp. $\mathrm{rob}_{\mathrm{pre}}$) records the value of $\mathrm{acc}_{\mathrm{cur}}$ (resp. $\mathrm{rob}_{\mathrm{cur}}$) in the last round. It then iteratively enriches $\Delta\mathcal{D}_V$ with newly selected S_{ct} (lines 2-11). DEP generates PVD based on $\mathcal{D}^*_{\mathrm{train}}$ and $\mathcal{A}$ (line 3), obtains S_{mt}, $\mathrm{acc}_{\mathrm{cur}}$ and $\mathrm{rob}_{\mathrm{cur}}$ by evaluating $\mathcal{M}^*$ with S_{at}, $\mathcal{D}_{\mathrm{valid}}$ and PVD, respectively (line 4). It trains a set $S_{\hat{\mathcal{M}}^*}$ of datamodels $\hat{\mathcal{M}}^*_t$ for tuples t in $\mathcal{D}^*_{\mathrm{train}} \cup S_{\mathrm{at}}$ (line 7), selects S_{ct} using the trained datamodels in $S_{\hat{\mathcal{M}}^*}$ (line 8), updates $\mathcal{D}^*_{\mathrm{train}}$, S_{at}, $\Delta\mathcal{D}_T$, $\mathrm{rob}_{\mathrm{pre}}$ and $\mathrm{acc}_{\mathrm{pre}}$ (lines 9-10), and fine-tunes $\mathcal{M}^*$

Input: $\mathcal{R}$, $\mathcal{M}$ and α as in Figure 3, a training dataset $\mathcal{D}^*_{\text{train}}$ of $\mathcal{R}$,
a validation dataset $\mathcal{D}_{\text{valid}}$ of $\mathcal{R}$, an accuracy bound B_{inf},
an attacker $\mathcal{A}$, and a set S_{at} of adversarial tuples generated via $\mathcal{A}$.
Output: A set $\Delta\mathcal{D}_T$ of tuple perturbations to $\mathcal{D}^*_{\text{train}}$.

1. $\mathcal{M}^* := \mathcal{M}$; $\Delta\mathcal{D}_T := \phi$; $S_{\text{ct}} := \phi$; $\text{acc}_{\text{pre}} := \text{rob}_{\text{pre}} := \text{acc}_{\text{cur}} := \text{rob}_{\text{cur}} := 0$;
2. **while** $S_{\text{at}} \neq \phi$ **do**
3.   PVD := AttackValidationData($\mathcal{D}_{\text{valid}}, \mathcal{M}, \mathcal{A}$);
4. $(\text{acc}_{\text{cur}}, \text{rob}_{\text{cur}}, S_{\text{mt}})$:= Evaluate($\mathcal{M}^*, \mathcal{D}_{\text{valid}}$, PVD, S_{at});
5.   **if** $\text{acc}_{\text{cur}} < B_{\text{inf}}$ **or** $\text{rob}_{\text{cur}} < \text{rob}_{\text{pre}}$ **then**
6. $\Delta\mathcal{D}_T := \Delta\mathcal{D}_T \setminus S_{\text{ct}}$; **break**;
7. $S_{\hat{\mathcal{M}}^*}$:= TrainDataModel($\mathcal{D}^*_{\text{train}}, S_{\text{at}}, \mathcal{M}^*, \alpha$);
8. S_{ct} := SelectCriticalTuples($\mathcal{D}^*_{\text{train}}, S_{\text{at}}, S_{\text{mt}}, S_{\hat{\mathcal{M}}^*}$);
9. $\mathcal{D}^*_{\text{train}} := \mathcal{D}^*_{\text{train}} \oplus S_{\text{ct}}$; $S_{\text{at}} := S_{\text{at}} \setminus S_{\text{ct}}$; $\Delta\mathcal{D}_T := \Delta\mathcal{D}_T \cup S_{\text{ct}}$;
10. $\text{rob}_{\text{pre}} := \text{rob}_{\text{cur}}$; $\text{acc}_{\text{pre}} := \text{acc}_{\text{cur}}$;
11. $\mathcal{M}^*$:= FineTune($\mathcal{M}^*, \mathcal{D}^*_{\text{train}}$);
12. **return** $\Delta\mathcal{D}_T$;

Figure 4: Algorithm DEP

with the enhanced $\mathcal{D}^*_{\text{train}}$ (line 11). It stops if (a) all tuples in S_{at} are added to $\mathcal{D}^*_{\text{train}}$ (line 2); (b) the accuracy of $\mathcal{M}^*$ is below B_{inf} (lines 5-6); or (c) the robustness of $\mathcal{M}^*$ is degraded in this round by the tuple perturbations in S_{ct} in the previous round (lines 5-6); if so, we undo the perturbations in S_{ct} (line 6). Finally, DEP returns $\Delta\mathcal{D}_T$ (line 12).

Example 6: Continuing with Example 4, suppose that initially, $\Delta\mathcal{D}_T = \phi$, $S_{\text{ct}} = \phi$, $\text{acc}_{\text{pre}} = 0$, $\text{rob}_{\text{pre}} = 0$, $\text{acc}_{\text{cur}} = 0$ and $\text{rob}_{\text{cur}} = 0$, where $S_{\text{at}} = \{t_1^{'}, t_2^{'}, t_3^{'}\}$. Consider $B_{\text{inf}} = 0.85$. In the first round, suppose that we generate $S_{\text{mt}} = \{t_1^{'}, t_2^{'}\}$ with $\text{acc}_{\text{cur}} = 0.9$ and $\text{rob}_{\text{cur}} = 0.7$, divide tuples in $\mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$ into two groups g_1 and g_2, and generate $S_{\text{ct}} = \{t_2^{'}\}$, where $g_1 = \{t_1, t_2^{'}, t_2, t_5, \dots\}$ with $g_1.\text{dr} = 0.6$ and $g_2 = \{t_1^{'}, t_3, t_3^{'}, t_4, \dots\}$ with $g_1.\text{dr} = 0.5$. We enhance $\mathcal{D}^*_{\text{train}}$ with S_{ct}, update $S_{\text{at}} = \{t_1^{'}, t_3^{'}\}$, $\Delta\mathcal{D}_T = \{t_2^{'}\}$, $\text{acc}_{\text{pre}} = 0.9$ and $\text{rob}_{\text{pre}} = 0.7$. This process proceeds until either $\text{acc}_{\text{cur}} < 0.85$ or $S_{\text{at}} = \phi$. After phase 2, DE4ML returns $\Delta\mathcal{D}_T = \{t_2^{'}\}$ and $\mathcal{M}$ fine-tuned with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$, since adding more tuples (*e.g.*, $t_1^{'}$) to $\mathcal{D}^*_{\text{train}}$ leads to $\text{acc}_{\text{cur}} < 0.85$ in the next round. □

Analyses. DEP approaches the intractable DEAAP by projecting tuples in $\mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$ into a same embedding space based on datamodel, and identifying a small set $\Delta\mathcal{D}_T$ of critical adversarial tuples that are influential, harmless and diversified, such that fine-tuning $\mathcal{M}$ (trained in phase 1 of DE4ML) with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ can make $\mathcal{M}$ robust against attacked tuples in poisoned $\mathcal{D}^-_{\text{pred}}$ while retaining its accuracy on unpoisoned $\mathcal{D}_{\text{pred}}$ (*i.e.*, $\geq B_{\text{inf}}$).

DEP takes $O(|S_{\text{at}}| \cdot (c_{\text{pvd}} + c_{\text{eva}} + (|\mathcal{D}^*_{\text{train}}| + |S_{\text{at}}|) \cdot c_{\hat{\mathcal{M}}^*_t} + c_{\text{ct}} + c_{\text{up}} + c_{\text{ft}}))$ time, where c_{pvd} is the cost of generating PVD with $\mathcal{D}_{\text{valid}}$, $\mathcal{M}$ and $\mathcal{A}$, c_{eva} is the sum of costs of evaluating $\mathcal{M}^*$ with $\mathcal{D}_{\text{valid}}$, the poisoned $\mathcal{D}^-_{\text{valid}}$ in PVD, and S_{at}; $c_{\hat{\mathcal{M}}^*_t}$ is for training a linear datamodel $\hat{\mathcal{M}}^*_t$ for a tuple $t \in \mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$, c_{ct} is for selecting S_{ct} in S_{at} using datamodels in $S_{\hat{\mathcal{M}}^*}$, and c_{up} is the sum of costs of updating $\mathcal{D}^*_{\text{train}}$ (*i.e.*, $O(|D^*_{\text{train}}| + |S_{\text{at}}|)$), S_{at} ($O(|S_{\text{at}}|)$), $\Delta\mathcal{D}_T$ ($O(|S_{\text{at}}|)$), rob_{pre} ($O(1)$) and acc_{pre} ($O(1)$), and c_{ft} is for fine-tuning $\mathcal{M}^*$ with $\mathcal{D}^*_{\text{train}}$. The most costly factor is $c_{\hat{\mathcal{M}}^*_t}$. We will see in Section 6 that DEP is efficient for most models; it terminates far less than $|S_{\text{at}}|$ rounds.

6 EXPERIMENTAL STUDY

Using real-life datasets, this section experimentally evaluates the effectiveness and efficiency of DE4ML for mitigating the impact of adversarial attacks and improving the robustness of ML classifiers.

Datasets	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{R}\rvert$	Cell Attack Ratio (%) in Ph.1	Cell Attack Ratio (%) in Ph.2	B_{inf} in Ph.2
German [89]	1,000	20	20.9	15.3	0.7
Mushroom [3]	8,124	22	24.9	NA	NA
Adult [4]	48,842	15	24.9	26.3	0.7
Marketing [62]	8,993	13	26.3	NA	NA
Bank [89]	49,732	16	19.9	24.7	0.7

Table 3: The tested datasets

Experimental setting. We start with our experimental settings.

<u>*Datasets*.</u> We tested 5 datasets as shown in Table 3. (1) German, a dataset of bank accounts; the ML classification is to determine the credit risk of a holder. (2) Mushroom, a dataset of mushrooms in the Agaricus and Lepiota Family; it is to determine whether a mushroom is poisonous. (3) Adult, demographic data of individuals; it is to determine whether a person's income exceeds 50K per year. (4) Marketing, demographic data of households; it is to predict whether the annual income of a household is below 25K. (5) Bank, a dataset from marketing campaigns of a Portuguese banking institution; it is to predict whether a client will subscribe to term deposits. Each dataset $\mathcal{D}$ was randomly spit into disjoint $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ in $8:2$ ratio. Following [62], we undersampled the tuples with labels "y" in unbalanced $\mathcal{D}$ when they account for more than 70% of tuples in $\mathcal{D}$, to reduce the impact of class imbalance on $\mathcal{M}$.

<u>*ML classifiers*.</u> We tested logistic regression (LR) and MLPClassifier (MLP) as in [62]; and XGBoost [17] and CatBoost [24].

<u>*Attackers*.</u> We considered the following attackers $\mathcal{A}$ with cell attack power (rate) in Table 3. (1) Random attack (RA) and DeepFool [43, 70] that poison $\mathcal{D}_{\text{train}}$; *e.g.*, given an attribute importance vector $\mathcal{V}$ for each dataset obtained via the feature importance function in skleran [73], RA randomly injects attacks in attributes A_j with probability $(1 - v_j)$ (Section 2.1). (2) BIM [43], DeepFool, Carlini [43] and fast gradient sign method (FGSM) [36, 43] that poison $\mathcal{D}_{\text{pred}}$.

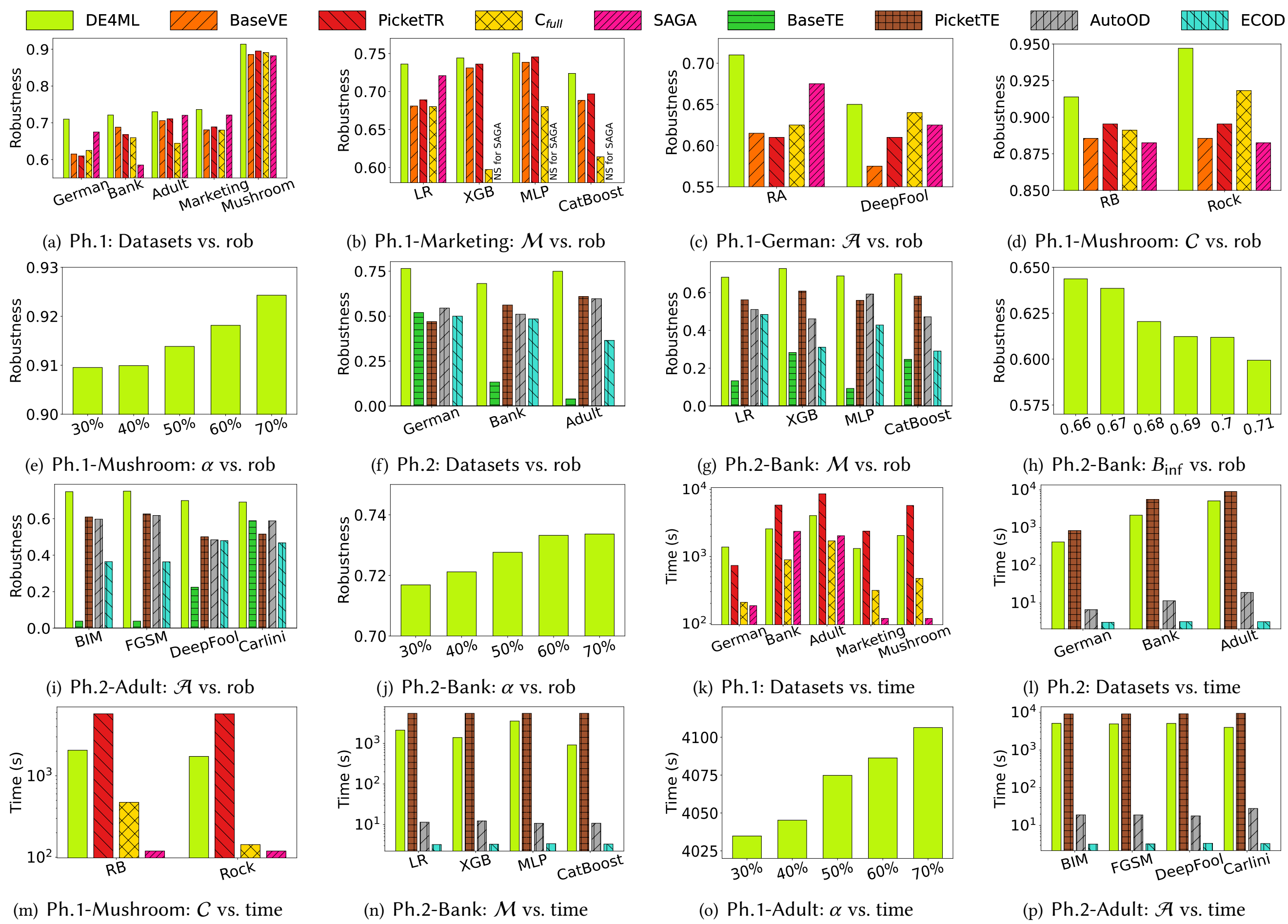
<u>*Data cleaning tools*.</u> We used (1) RB, an integrated approach that detects errors with Raha [66] and corrects errors with Baran [65]; and (2) Rock, a system [9] for error detection and correction.

<u>*Baselines*.</u> We implemented DE4ML in python. We used the following to defuse attacks in $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$, with the released codes.

(1) Defusing attacks in $\mathcal{D}_{\text{train}}$. (a) BaseVE, a "lower-bound" baseline without value perturbations. (b) PicketTR [62], which defuses attacks in $\mathcal{D}_{\text{train}}$ by removing poisoned tuples. (c) C_{full}, which cleans all errors in $\mathcal{D}_{\text{train}}$ with tool C. (d) SAGA [79], a state-of-the-art approach that automatically generates the top-K most effective data cleaning pipelines for $\mathcal{M}$; we set K as 1 in the tests.

(2) Defusing attacks in $\mathcal{D}_{\text{pred}}$. No prior methods allow $\mathcal{M}$ to directly correct poisoned tuple at prediction time; instead, they flag suspicious tuples in $\mathcal{D}_{\text{pred}}$ that $\mathcal{M}$ may mispredict. To compare DE4ML with existing methods, we flipped the prediction of tuples flagged as suspicious. We tested the following. (a) BaseTE, a "lower-bound" baseline with value perturbations but without tuple perturbations. (b) AllTE, a variant with (i) value perturbations as DE4ML and (ii) tuple perturbations including all attacked tuples (*i.e.*, $\Delta\mathcal{D}_T = S_{\text{at}}$). (c) PicketTE [62], which trains a detector to flag suspicious data in $\mathcal{D}_{\text{pred}}$ that may be mislabeled by $\mathcal{M}$. (d) AutoOD [11], which automatically detects outliers by integrating prior detection techniques. (e) ECOD [60], an unsupervised approach that detects outliers by employing empirical cumulative distribution functions.

<u>*Configurable parameters*.</u> By default, we set (1) $\alpha = 50\%$ for the sampling ratio of data for datamodel; (2) for DEAAP, the accuracy



(a) Ph.1: Datasets vs. rob
(b) Ph.1-Marketing: $\mathcal{M}$ vs. rob
(c) Ph.1-German: $\mathcal{A}$ vs. rob
(d) Ph.1-Mushroom: C vs. rob
(e) Ph.1-Mushroom: α vs. rob
(f) Ph.2: Datasets vs. rob
(g) Ph.2-Bank: $\mathcal{M}$ vs. rob
(h) Ph.2-Bank: B_{inf} vs. rob
(i) Ph.2-Adult: $\mathcal{A}$ vs. rob
(j) Ph.2-Bank: α vs. rob
(k) Ph.1: Datasets vs. time
(l) Ph.2: Datasets vs. time
(m) Ph.1-Mushroom: C vs. time
(n) Ph.2-Bank: $\mathcal{M}$ vs. time
(o) Ph.1-Adult: α vs. time
(p) Ph.2-Adult: $\mathcal{A}$ vs. time

Figure 5: Performance evaluation

bound $\mathcal{B}_{\text{inf}}$ as in Table 3, (3) we used LR as $\mathcal{M}$, RB as data cleaning tool C, and RA (resp. BIM) as attacker $\mathcal{A}$ for DEAAT (resp. DEAAP).

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

Experimental findings. We next report our findings.

Exp-1 Effectiveness in defusing attacks in $\mathcal{D}_{\text{train}}$. We first evaluated the robustness of DE4ML versus baselines in phase 1 (Ph.1) for DEAAT. We adopt the rob measure in Equation 2. As remarked in Section 2.1, here a higher rob also indicates a higher accuracy.

Robustness vs. baselines As shown in Figure 5(a), DE4ML has the highest robustness score for DEAAT. Besides, we find the following.

(1) DE4ML outperforms PicketTR, *e.g.,* its robustness is 0.76 on average, as opposed to 0.71 of PicketTR, 7% higher. This is because DE4ML (a) fixes corrupted values in $\mathcal{D}_{\text{train}}$ rather than discarding the attacked tuples, such that the distribution of $\mathcal{D}_{\text{train}}$ is better preserved, and (b) identifies poisoned imperceptible attributes that are overlooked by prior outlier detection strategy. This justifies the need to rectify corrupted values in data, instead of removing tuples.

(2) On average, DE4ML beats BaseVE, C_{full} and SAGA by 6.7%, 8.9% and 6.3%, respectively. This validates the strategy of partial fixing: (a) DE4ML catches critical poisoned cells whose fixing can improve $\mathcal{M}$ the most, and (b) enables $\mathcal{M}$ for better generality on unseen data by leaving the non-critical errors unfixed. Moreover, as "data cleaning for AI" may ignore adversarial noises in imperceptible attributes, it does not suffice to defuse attacks and cannot replace DE4ML.

We next evaluated the impact of various parameters.

Varying $\mathcal{M}$. We tested the effectiveness of DEAAT on different models $\mathcal{M}$. As shown in Figure 5(b), DE4ML performs well, *e.g.,* the rob on LR, MLP, XGBoost and CatBoost is 0.74 on average, 7.1% higher than the baselines, up to 24.6%. This is because with datamodel, DE4ML can well predict the prediction of various $\mathcal{M}$.

Varying $\mathcal{A}$. We tested the impact of attackers $\mathcal{A}$ on DEAAT. As shown in Figure 5(c), DE4ML adapts the training of $\mathcal{M}$ to different $\mathcal{A}$. Its rob on German is 0.71 (resp. 0.65) under RA (resp. DeepFool); Hence, DE4ML is able to identify/fix adversarial attacks injected by different attackers; with datamodel, it can identify tuples with poisoned attributes that are abnormal, influential and imperceptible.

Varying C. We tested the impact of data cleaning tool C. As shown in Figure 5(d), DE4ML with more accurate C can lead to a better $\mathcal{M}$, *e.g.,* its rob with Rock is 0.95 on Mushroom, 3.64% higher than with RB. Since a more accurate C can fix poisoned cells and introduce less errors, $\mathcal{M}$ can be better trained with an enhanced $\mathcal{D}_{\mathsf{train}}$. PicketTR and SAGA do not leverage C when C evolves to be more accurate.

Varying α. We varied the sampling ratio α of data for training datamodel from 30% to 70%. Figure 5(e) shows that with larger α, DE4ML does better, *e.g.,* its rob on Mushroom increases by 1.7% when α varies from 40% to 70%. This is because with a larger α, datamodel can better capture the association of tuples in a finer granularity, such that poisoned tuples critical to $\mathcal{M}$ can be better identified. Good performance is observed at $\alpha \geq 50\%$.

Exp-2 Effectiveness in defusing attacks in $\mathcal{D}_{\mathsf{pred}}$. We then evaluated the effectiveness of DE4ML and baselines in phase 2 (Ph.2) for DEAAP. As stated in Theorem 1, there is a tradeoff between the accuracy and robustness. The accuracy of DE4ML and baselines is above the accuracy bound B_{inf} as shown in Table 3.

Robustness vs. baselines. As shown in Figure 5(f), DE4ML beats all baselines for DEAAP, *e.g.,* its rob is 0.73 on average, 217%, 33.8%, 32.9% and 62.8% higher than BaseTE, PicketTE, AutoOD and ECOD, respectively. This verifies that flagging suspicious tuples does not suffice to make $\mathcal{M}$ robust to corrupted tuples unseen in $\mathcal{D}_{\mathsf{train}}$.

We next tested the impact of various parameters on the quality.

Varying $\mathcal{M}$. We tested the effectiveness of DEAAP on various ML models. As shown in Figure 5(g), when defending against unseen attacks in prediction data, *e.g.,* the rob of DE4ML is 0.7 on average, 47.7% higher than the baselines, up to 84.6%. This is because DE4ML enhances the training data of various $\mathcal{M}$ by carefully selecting adversarial tuples that are influential, harmless and diversified.

Varying B_{inf}. We varied the accuracy bound B_{inf} for DEAAP from 0.66 to 0.71. As shown in Figure 5(h), when B_{inf} increases from 0.66 to 0.71, the rob of DE4ML decreases from 0.64 to 0.6. This is because of the inherent tradeoff between the accuracy and robustness of $\mathcal{M}$ (Theorem 1); hence a low (resp. high) accuracy may lead to a high (resp. low) robustness. We observe that $B_{\mathsf{inf}} = 0.7$ works well.

Varying $\mathcal{A}$. We evaluated the impact of different attackers $\mathcal{A}$. As shown in Figure 5(i), the rob of DE4ML is sensitive to $\mathcal{A}$, *e.g.,* it is 0.75, 0.7, 0.69 and 0.75 on Adult under BIM, DeepFool, Carlini and FGSM, respectively, since with a fixed B_{inf}, it is more challenging to retain a high robustness when $\mathcal{A}$ (*e.g.,* DeepFool) is more powerful. Nonetheless, DE4ML consistently beats baselines. This is because we fine-tune $\mathcal{M}$ with adversarial examples generated by $\mathcal{A}$, such that $\mathcal{M}$ is vaccinated to be immune to those tuples in $\mathcal{D}^{-}_{\mathsf{pred}}$ corrupted by $\mathcal{A}$, and correctly classify them.

Varying α. Varying α, Figure 5(j) shows that DE4ML does better with a larger α for DEAAP, *e.g.,* its rob on Bank is 0.734 with α = 70%, 2.3% larger than with α = 30%. This is because a larger α enables DE4ML to represent tuples in a higher-dimensional vector space, and makes it easier to identify critical tuples for boosting the robustness of $\mathcal{M}$ on poisoned $\mathcal{D}^{-}_{\mathsf{pred}}$ with more referenced features.

Ablation study. (1) We also studied the impact of tuple perturbations on accuracy and robustness (not shown). On average its acc (resp. rob) is 25.8% (resp. 20%) higher (resp. smaller) than AllTE. This is because of the tradeoff between accuracy and robustness; thus enhancing $\mathcal{D}_{\mathsf{train}}$ with more adversarial tuples can negatively affect the accuracy of $\mathcal{M}$ on unpoisoned $\mathcal{D}_{\mathsf{pred}}$. This said, DE4ML enables $\mathcal{M}$ to defend against unseen attacks with a small accuracy drop, *e.g.,* when rob = 0.73 on Adult, its acc is 0.7, only 4.1% lower than its value without fine-tuning. (2) We also studied the contribution of DEAAT and DEAAP to DE4ML by disabling one of them (not shown). Denote by BaseNone the "lower bound" baseline with neither value perturbations nor tuple perturbations. DEAAT, DEAAP and DE4ML beat BaseNone by 25.5%, 398% and 541%, respectively, on Bank. This indicates that DEAAT or DEAAP alone does not suffice to work the best. We suggest to use DE4ML with both DEAAT and DEAAP.

Exp-3 Efficiency. We compared the time cost of DE4ML and baselines in the default setting for DEAAT and DEAAP. We also evaluated the impact of different parameters on the efficiency of DE4ML.

Comparison vs. baselines. As shown in Figures 5(k) and 5(l), (a) on average, DE4ML is 2.04X and 2.02X faster than Picket (the work closest to ours) for DEAAT and DEAAP, respectively. This is because DE4ML can quickly defuse attacks in $\mathcal{D}_{\mathsf{train}}$ and $\mathcal{D}_{\mathsf{pred}}$ with a set of light-weight datamodels, rather than relying on the tremendous feedback from downstream models $\mathcal{M}$ as in Picket. (b) DE4ML is slower than C_{full} and SAGA, since it needs to identify critical corrupted cells that are abnormal, influential and imperceptible, rather than only cleaning noises as in C_{full} and SAGA; in exchange, DE4ML beats C_{full} and SAGA by 8.9% and 6.3% in robustness, respectively. (c) DE4ML is efficient when C, $\mathcal{A}$ and the training of datamodel are efficient, since they take up most of the time of DE4ML (see below).

Varying C. We tested the impact of data cleaning tool C on DEAAT. As shown in Figure 5(m), it needs less time when C is more efficient, *e.g.,* it takes 1727s on Mushroom with Rock, 1.19X faster than with RB. That is, DE4ML is more efficient for DEAAT when C is faster. In contrast, PicketTR and SAGA do not take advantage of evolving C.

Varying $\mathcal{M}$. As shown in Figure 5(n), the cost of DE4ML is sensitive to $\mathcal{M}$, *e.g.,* it takes 3,565s on Bank with MLP, 2.9X longer than with CatBoost. We find that with datamodel, DE4ML can better fit the prediction of tree-based models (*e.g.,* CatBoost and XGBoost) for DEAAP (with fewer iterations) than other models (*e.g.,* MLP).

Varying α. Varying the sampling ratio α from 30% to 70%, DE4ML takes slightly longer with larger α for DEAAT. As shown in Figure 5(o), it takes 4,106s when α = 70%, 1.8% longer than when α = 30%. This is because (a) what dominates the cost of DE4ML is the the total time of cleaning $\mathcal{D}_{\mathsf{train}}$ via C, generating adversarial data via $\mathcal{A}$ and training datamodels, which accounts for $\geq$ 85%; and (b) a larger α yields more training data for $\mathcal{M}$, and DE4ML takes longer to generate training set $\mathcal{D}_t$ for each tuple in $\mathcal{D}_{\mathsf{train}}$.

Varying $\mathcal{A}$. We tested the impact of different attackers $\mathcal{A}$ on the efficiency of DEAAP. As shown in Figure 5(p), DE4ML is more efficient when $\mathcal{A}$ needs less time to generate adverarial tuples, *e.g.,* it takes 5,090s on Adult with BIM, 1,119s less than with Carlini.

Moreover, we tested the impact of accuracy bound B_{inf} on DEAAP (not shown). It needs more time with a lower B_{inf}, since we trade accuracy for higher robustness with more iterations.

Parameter settings. We find that α = 0.5 suffices for a balance between efficiency and effectiveness. As data cleaning tools, Rock [9] and RB (Raha [66] and Baran [65]) work well. One may use DE4ML to defend against any attacker $\mathcal{A}$ *w.r.t.* any accuracy bound B_{inf}.

Summary. We find the following. (1) The robustness of DE4ML for DEAAT is 0.76 on average, as opposed to 0.71 of PicketTR. This verifies that DE4ML can accurately defuse attacks in $\mathcal{D}_{\text{train}}$. (2) It is 8.9% more accurate than cleaning the entire $\mathcal{D}_{\text{train}}$, and validates its strategy of fixing critical poisoned cells only. (3) DE4ML beats Picket (the SOTA) by 7% and 33.8% for defusing attacks in the training data and defending against unseen attacks at prediction time, respectively, 20.4% on average in each phase. This shows that DE4ML defuses not only attacks injected in $\mathcal{D}_{\text{train}}$, but also unseen attacks at prediction time. (4) For various ML models, the robustness of DE4ML is 0.74 and 0.7 on average for DEAAT and DEAAP, respectively. In the two phases together, it is 27.4% better than the baselines. (5) DE4ML enables $\mathcal{M}$ to defend against various types of attacks; the robustness of DE4ML for DEAAP is 0.7 under DeepFool, only 6.7% lower than the highest one when $\mathcal{A}$ is BIM. (6) DE4ML is efficient for defusing attacks, *e.g.,* on average it is 2.04X (resp. 2.02X) faster than Picket for DEAAT (resp. DEAAP).

7 RELATED WORK

Adversarial attacks. There has been a host of work on generating adversarial attacks. The notion of imperceptible attacks on relations was formalized in [8], which also showed how to generate adversarial examples with numeric values. A surrogate model of [67] crafts adversarial examples for relations with heterogeneous attributes. Subpopulation attacks of [45] degrade the accuracy of ML models on a specific group/subpopulation of tuples by adding adversarial tuples. Gradient-based attacks were investigated in [13], to impact the fairness of ML models. Two types of such attacks were proposed in [68]. Triggers on time series were studied in [22] to hack deep neural networks (DNNs) with backdoor attacks. Attacks on learned indices were studied in [52]. Adversaries on relations are surveyed in [43], on texts in [88], and in [31, 83] on graph models.

For mitigating the impact of adversarial attacks, the prior work has mostly focused on image models, *e.g.,* robust training for norm-bounded adversarial attack [90], worst case perturbations [80], acceleration of model verification [85, 91], specification of a verifiable robust model [23] and the trade-off between accuracy and robustness [86] (see [20] for a survey). In contrast, not much has been done on how to improve the robustness of $\mathcal{M}$ on relations. An interactive game-theoretic model was proposed in [32] to detect attacks. ARDA [19] adopts schema enrichment to improve the resistance of $\mathcal{M}$ to noises in tabular data. TargAD [63] extends outlier detection to identify specific types of poisoned tuples. MWOC [38] employs kernel-based test and label enrichment (*i.e.,* adjusting the output of $\mathcal{M}$ from binary to ternary) to detect adversarial tuples. TOAO [75] applies statistical test *w.r.t.* log-odds to discern adversarial data. Robust training was also studied for graph models [46, 58].

Closer to this work is Picket [62], which safeguards tabular data against corruptions at both training and prediction phases of ML model; it (a) detects and removes corrupted tuples from training data, and (b) flags corrupted tuples at prediction time.

There has also been a large body of work on fairness and biased data [7, 10, 15, 16, 26, 34, 37, 41, 48, 59, 61, 71, 74, 84, 92–94]. In particular, the unavoidable trade-off between the accuracy and fairness of ML models is studied in [18, 29, 49, 51, 69, 76, 95].

This work differs from the prior work in the following. (1) We identify when there exists unavoidable tradeoff between robustness and accuracy on relations, beyond image models [86]. (2) We formulate two data enhancing problems to defend against adversarial attacks, and prove their intractability. (3) As opposed to Picket [62], we (a) fix poisoned attributes in $\mathcal{D}_{\text{train}}$ to retain the distribution of $\mathcal{D}_{\text{train}}$ and the accuracy of ML models $\mathcal{M}$, instead of removing tuples, (b) we add supplement tuples to make $\mathcal{M}$ robust against adversarial attacks unseen in the training data, and (c) train a single ML model to defend against attacks at both training time and inference time, instead of training two models separately [62]. (4) We enhance $\mathcal{D}_{\text{train}}$ with value and tuple perturbations, rather than schema enrichment as in ARDA [19]. (5) We defuse attacks in $\mathcal{D}_{\text{pred}}$ by adding adversarial tuples that are influential, harmless and diversified, instead of (a) relying on the distance among tuples alone as in [32], (b) extending prior outlier detection strategies as in TargAD [63], or (c) using statistical tests as in MWOC [38] and TOAO [75].

Data cleaning for AI. There has also been work on data cleaning for ML/AutoML, mainly to improve ML accuracy. ActiveClean [54] employs data cleaning via stochastic gradient descent. Coco [27, 28] detects unfriendly tuples for ML models using arithmetic constraints on numerical attributes. Amalur [40] improves the accuracy via data integration. BoostClean [53] boosts the accuracy by selecting suitable data cleaning tools from a given pool. GoodCore [14] selects coresets of incomplete data through gradient approximation. CategDedup [78] empirically shows that categorical duplicates can have negative effect on the accuracy. AutoSklearn [30] improves AutoML approaches by referencing previous performance on similar datasets. AlphaClean [55] tunes parameters for data cleaning pipelines to improve the accuracy. AutoCure [5] cleans training data for ML pipelines via error detection and data augmentation. AutoClean [72] extends AutoSklearn with advanced data imputation and outlier detection. SAGA [79] identifies the top-k promising data cleaning pipelines for ML models. Moreover, REIN [4] provides a benchmark to evaluate data cleaning approaches in ML pipelines.

As remarked in Section 1, data enhancing for ML and data cleaning for ML differ in both objectives (for improving the robustness and the accuracy, respectively) and methods. For example, we selectively defuse poisoned (imperceptible) cells to reserve data distribution, instead of fixing all errors as in data cleaning.

8 CONCLUSION

The novelty of the work consists of the following. (1) We identify when inherent tradeoff exists between the accuracy and robustness of ML classifiers $\mathcal{M}$ on relations. We show that both accuracy and robustness can be improved when defusing attacks in training data $\mathcal{D}_{\text{train}}$, but the tradeoff is inevitable when adding tuples to training data to defend against attacks unseen in $\mathcal{D}_{\text{train}}$. (2) We propose DE4ML for data enhancing to improve the robustness of $\mathcal{M}$ against both types of attacks while retaining the accuracy. (3) We show that it is intractable to defuse any of the two types of attacks, but provide an effective algorithm for each. Our experimental study has verified that data enhancing with DE4ML is promising in practice.

One topic for future work is to extend DE4ML and improve the fairness of ML classifiers against bias. Another topic is to integrate data enhancing and data cleaning in a uniform process, and strike a balance among the accuracy, fairness and robustness of ML models.

REFERENCES

[1] 2019. What banks look for when reviewing a loan application. https://www.wolterskluwer.com/en/expert-insights/what-banks-look-for-when-reviewing-a-loan-application.

[2] 2024. Code, datasets and full version. https://anonymous.4open.science/r/DE4ML-SIGMOD25-ECFE.

[3] 2024. Mushroom. https://archive.ics.uci.edu/dataset/73/mushroom.

[4] Mohamed Abdelaal, Christian Hammacher, and Harald Schöning. 2023. REIN: A Comprehensive Benchmark Framework for Data Cleaning Methods in ML Pipelines. In *EDBT*. OpenProceedings.org, 499–511.

[5] Mohamed Abdelaal, Rashmi Koparde, and Harald Schöning. 2023. AutoCure: Automated Tabular Data Curation Technique for ML Pipelines. In *aiDM@SIGMOD*. ACM, 1:1–1:11.

[6] Akshay Agarwal and Nalini K Ratha. 2021. Black-Box Adversarial Entry in Finance through Credit Card Fraud Detection. In *CIKM*.

[7] Agathe Balayn, Christoph Lofi, and Geert-Jan Houben. 2021. Managing bias and unfairness in data for decision support: A survey of machine learning and data engineering approaches to identify and mitigate bias and unfairness within data management and analytics systems. *VLDBJ* 30, 5 (2021), 739–768.

[8] Vincent Ballet, Xavier Renard, Jonathan Aigrain, Thibault Laugel, Pascal Frossard, and Marcin Detyniecki. 2019. Imperceptible Adversarial Attacks on Tabular Data. *CoRR* abs/1911.03274 (2019).

[9] Xianchun Bao, Zian Bao, Qingsong Duan, Wenfei Fan, Hui Lei, Daji Li, Wei Lin, Peng Liu, Zhicong Lv, Mingliang Ouyang, Jiale Peng, Jing Zhang, Runxiao Zhao, Shuai Tang, Shuping Zhou, Yaoshu Wang, Qiyuan Wei, Min Xie, Jing Zhang, Xin Zhang, Runxiao Zhao, and Shuping Zhou. 2024. Rock: Cleaning Data by Embedding ML in Logic Rules. In *SIGMOD (industrial track)*.

[10] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. 2009. Building Classifiers with Independency Constraints. In *ICDM Workshops*. IEEE Computer Society, 13–18.

[11] Lei Cao, Yizhou Yan, Yu Wang, Samuel Madden, and Elke A. Rundensteiner. 2023. AutoOD: Automatic Outlier Detection. *Proc. ACM Manag. Data* 1, 1 (2023), 20:1–20:27.

[12] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *SP*. IEEE.

[13] Francesco Cartella, Orlando Anunciação, Yuki Funabiki, Daisuke Yamaguchi, Toru Akishita, and Olivier Elshocht. 2021. Adversarial Attacks for Tabular Data: Application to Fraud Detection and Imbalanced Data. In *SafeAI (CEUR Workshop Proceedings, Vol. 2808)*. CEUR-WS.org.

[14] Chengliang Chai, Jiabin Liu, Nan Tang, Ju Fan, Dongjing Miao, Jiayi Wang, Yuyu Luo, and Guoliang Li. 2023. GoodCore: Data-effective and Data-efficient Machine Learning through Coreset Selection over Incomplete Data. *Proc. ACM Manag. Data* (2023).

[15] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. 2021. Bias in Machine Learning Software: Why? How? What to do? *CoRR* abs/2105.12195 (2021). arXiv:2105.12195 https://arxiv.org/abs/2105.12195

[16] Joymallya Chakraborty, Suvodeep Majumder, Zhe Yu, and Tim Menzies. 2020. Fairway: A way to build fair ML software. In *ESEC/FSE*. ACM, 654–665.

[17] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *SIGKDD*. 785–794.

[18] Zhenpeng Chen, Jie M Zhang, Federica Sarro, and Mark Harman. 2022. MAAT: A Novel Ensemble Approach to Addressing Fairness and Performance Bugs for Machine Learning Software. In *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1122–1134.

[19] Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. 2020. ARDA: Automatic relational data augmentation for machine learning. *arXiv preprint arXiv:2003.09758* (2020).

[20] Joana C Costa, Tiago Roxo, Hugo Proença, and Pedro RM Inácio. 2024. How deep learning sees the world: A survey on adversarial attacks & defenses. *IEEE Access* (2024).

[21] Daniel Deutch, Nave Frost, Amir Gilad, and Oren Sheffer. 2021. Explanations for Data Repair Through Shapley Values. In *CIKM*. ACM.

[22] Daizong Ding, Mi Zhang, Yuanmin Huang, Xudong Pan, Fuli Feng, Erling Jiang, and Min Yang. 2022. Towards backdoor attack on deep learning based time series classification. In *ICDE*. IEEE, 1274–1287.

[23] Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. 2018. Training verified learners with learned verifiers. *CoRR* abs/1805.10265 (2018).

[24] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv preprint arXiv:2003.06505* (2020).

[25] Wenqi Fan, Han Xu, Wei Jin, Xiaorui Liu, Xianfeng Tang, Suhang Wang, Qing Li, Jiliang Tang, Jianping Wang, and Charu C. Aggarwal. 2023. Jointly Attacking Graph Neural Network and its Explanations. In *ICDE*. IEEE, 654–667.

[26] Wenqi Fan, Xiangyu Zhao, Xiao Chen, Jingran Su, Jingtong Gao, Lin Wang, Qidong Liu, Yiqi Wang, Han Xu, Lei Chen, and Qing Li. 2022. A Comprehensive Survey on Trustworthy Recommender Systems. *CoRR* abs/2209.10117 (2022). https://doi.org/10.48550/ARXIV.2209.10117 arXiv:2209.10117

[27] Anna Fariha, Ashish Tiwari, Alexandra Meliou, Arjun Radhakrishna, and Sumit Gulwani. 2021. CoCo: Interactive Exploration of Conformance Constraints for Data Understanding and Data Cleaning. In *SIGMOD*. ACM, 2706–2710.

[28] Anna Fariha, Ashish Tiwari, Arjun Radhakrishna, Sumit Gulwani, and Alexandra Meliou. 2021. Conformance Constraint Discovery: Measuring Trust in Data-Driven Systems. In *SIGMOD*. ACM, 499–512.

[29] Tom Farrand, Fatemehsadat Mireshghallah, Sahib Singh, and Andrew Trask. 2020. Neither Private Nor Fair: Impact of Data Imbalance on Utility and Fairness in Differential Privacy. In *PPMLP*. ACM, 15–19.

[30] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *NIPS*. 2962–2970.

[31] Scott Freitas, Diyi Yang, Srijan Kumar, Hanghang Tong, and Duen Horng Chau. 2022. Graph vulnerability and robustness: A survey. *TKDE* 35, 6 (2022), 5915–5934.

[32] Yue Fu, Qingqing Ye, Rong Du, and Haibo Hu. 2024. Interactive Trimming against Evasive Online Data Manipulation Attacks: A Game-Theoretic Approach. *arXiv preprint arXiv:2403.10313* (2024).

[33] Michael Garey and David Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

[34] Naman Goel, Alfonso Amayuelas, Amit Deshpande, and Amit Sharma. 2021. The Importance of Modeling Data Missingness in Algorithmic Fairness: A Causal Perspective. In *AAAI*. AAAI Press, 7564–7573.

[35] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. 2672–2680.

[36] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[37] Stefan Grafberger, Paul Groth, Julia Stoyanovich, and Sebastian Schelter. 2022. Data Distribution Debugging in Machine Learning Pipelines. *VLDBJ* 31, 5 (2022), 1103–1126.

[38] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[39] Shubha Guha, Falaah Arif Khan, Julia Stoyanovich, and Sebastian Schelter. 2023. Automated Data Cleaning Can Hurt Fairness in Machine Learning-based Decision Making. In *ICDE*. IEEE, 3747–3754.

[40] Rihan Hai, Christos Koutras, Andra Ionescu, Ziyu Li, Wenbo Sun, Jessie van Schijndel, Yan Kang, and Asterios Katsifodimos. 2023. Amalur: Data Integration Meets Machine Learning. In *ICDE*. IEEE, 3729–3739.

[41] Moritz Hardt, Eric Price, and Nathan Srebro. 2016. Equality of Opportunity in Supervised Learning. *CoRR* abs/1610.02413 (2016). arXiv:1610.02413 http://arxiv.org/abs/1610.02413

[42] Masoud Hashemi and Ali Fathi. 2020. Permuteattack: Counterfactual explanation of machine learning credit scorecards. *arXiv preprint arXiv:2008.10138* (2020).

[43] Zhipeng He, Chun Ouyang, Laith Alzubaidi, Alistair Barros, and Catarina Moreira. 2024. Investigating Imperceptibility of Adversarial Attacks on Tabular Data: An Empirical Analysis. *arXiv preprint arXiv:2407.11463* (2024).

[44] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. 2022. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622* (2022).

[45] Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. 2020. Subpopulation Data Poisoning Attacks. *CoRR* abs/2006.14026 (2020). https://arxiv.org/abs/2006.14026

[46] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph Structure Learning for Robust Graph Neural Networks. *CoRR* abs/2005.10203 (2020).

[47] V Roshan Joseph and Akhil Vakayil. 2022. SPlit: An optimal method for data splitting. *Technometrics* (2022).

[48] Faisal Kamiran and Toon Calders. 2011. Data preprocessing techniques for classification without discrimination. *Knowl. Inf. Syst.* 33, 1 (2011), 1–33.

[49] Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2016. Inherent Trade-Offs in the Fair Determination of Risk Scores. *CoRR* abs/1609.05807 (2016). arXiv:1609.05807 http://arxiv.org/abs/1609.05807

[50] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*. PMLR, 1885–1894.

[51] Nikola Konstantinov and Christoph H Lampert. 2022. Fairness-Aware PAC Learning from Corrupted Data. *The Journal of Machine Learning Research* 23, 1 (2022), 7173–7232.

[52] Evgenios M Kornaropoulos, Silei Ren, and Roberto Tamassia. 2022. The price of tailoring the index to your data: Poisoning attacks on learned index structures. In *SIGMOD*. 1331–1344.

[53] Sanjay Krishnan, Michael J. Franklin, Ken Goldberg, and Eugene Wu. 2017. BoostClean: Automated Error Detection and Repair for Machine Learning. *CoRR* abs/1711.01299 (2017).

[54] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Gold-

berg. 2016. ActiveClean: Interactive Data Cleaning For Statistical Modeling. *PVLDB* 9, 12 (2016), 948–959.
[55] Sanjay Krishnan and Eugene Wu. 2019. AlphaClean: Automatic Generation of Data Cleaning Pipelines. *CoRR* abs/1904.11827 (2019). http://arxiv.org/abs/1904.11827
[56] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. 2020. Adversarial machine learning-industry perspectives. In *IEEE security and privacy workshops (SPW)*. IEEE, 69–75.
[57] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).
[58] Haoyang Li, Shimin Di, Calvin Hong Yi Li, Lei Chen, and Xiaofang Zhou. 2024. Fight Fire with Fire: Towards Robust Graph Neural Networks on Dynamic Graphs via Actively Defense. *PVLDB* (2024).
[59] Yanhui Li, Linghan Meng, Lin Chen, Li Yu, Di Wu, Yuming Zhou, and Baowen Xu. 2022. Training Data Debugging for the Fairness of Machine Learning Software. In *International Conference on Software Engineering*. 2215–2227.
[60] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George H Chen. 2022. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *TKDE* 35, 12 (2022), 12181–12193.
[61] Yin Lin, Samika Gupta, and HV Jagadish. 2024. Mitigating subgroup unfairness in machine learning classifiers: A data-driven approach. In *ICDE*. IEEE, 2151–2163.
[62] Zifan Liu, Zhechun Zhou, and Theodoros Rekatsinas. 2020. Picket: Self-supervised Data Diagnostics for ML Pipelines. *CoRR* abs/2006.04730 (2020). https://arxiv.org/abs/2006.04730
[63] Guanyu Lu, Fang Zhou, Martin Pavlovski, Chenyi Zhou, and Cheqing Jin. 2024. A Robust Prioritized Anomaly Detection When Not All Anomalies are of Primary Interest. In *ICDE*. IEEE, 775–788.
[64] J Macqueen. 1967. Some methods for classification and analysis of multivariate observations. In *University of California Press*.
[65] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *PVLDB* 13, 12 (2020), 1948–1961.
[66] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A Configuration-Free Error Detection System. In *SIGMOD*. 865–882.
[67] Yael Mathov, Eden Levy, Ziv Katzir, Asaf Shabtai, and Yuval Elovici. 2020. Not all datasets are born equal: On heterogeneous data and adversarial examples. *arXiv preprint arXiv:2010.03180* (2020).
[68] Ninareh Mehrabi, Muhammad Naveed, Fred Morstatter, and Aram Galstyan. 2021. Exacerbating Algorithmic Bias through Fairness Attacks. In *AAAI*. AAAI Press, 8930–8938.
[69] Aditya Krishna Menon and Robert C. Williamson. 2018. The cost of fairness in binary classification. In *Conference on Fairness, Accountability and Transparency (FAT) (Proceedings of Machine Learning Research, Vol. 81)*. PMLR, 107–118.
[70] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *ICCV*.
[71] Vedant Nanda, Samuel Dooley, Sahil Singla, Soheil Feizi, and John P Dickerson. 2021. Fairness through robustness: Investigating robustness disparity in deep learning. In *Conference on Fairness, Accountability, and Transparency*. 466–477.
[72] Felix Neutatz, Binger Chen, Yazan Alkhatib, Jingwen Ye, and Ziawasch Abedjan. 2022. Data cleaning and AutoML: Would an optimizer choose to clean? *Datenbank-Spektrum* (2022).
[73] 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. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[74] Romila Pradhan, Jiongli Zhu, Boris Glavic, and Babak Salimi. 2022. Interpretable Data-Based Explanations for Fairness Debugging. In *SIGMOD*. 247–261.
[75] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. 2019. The odds are odd: A statistical test for detecting adversarial examples. In *ICML*. PMLR, 5498–5507.
[76] Ricardo Salazar, Felix Neutatz, and Ziawasch Abedjan. 2021. Automated Feature Engineering for Algorithmic Fairness. *PVLDB* 14, 9 (2021), 1694–1702.
[77] Suproteem K Sarkar, Kojin Oshiba, Daniel Giebisch, and Yaron Singer. 2018. Robust classification of financial risk. *arXiv preprint arXiv:1811.11079* (2018).
[78] Vraj Shah, Thomas Parashos, and Arun Kumar. 2024. How do Categorical Duplicates Affect ML? A New Benchmark and Empirical Analyses. *PVLDB* (2024).
[79] Shafaq Siddiqi, Roman Kern, and Matthias Boehm. 2024. SAGA: A Scalable Framework for Optimizing Data Cleaning Pipelines for Machine Learning Applications. *Proc. ACM Manag. Data* (2024).
[80] Aman Sinha, Hongseok Namkoong, and John C. Duchi. 2018. Certifying Some Distributional Robustness with Principled Adversarial Training. In *International Conference on Learning Representations (ICLR)*. OpenReview.net.
[81] David Solans, Battista Biggio, and Carlos Castillo. 2020. Poisoning Attacks on Algorithmic Fairness. In *ECML PKDD (Lecture Notes in Computer Science, Vol. 12457)*. Springer, 162–177.
[82] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2020. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks* 129 (2020), 249–260.
[83] Lichao Sun, Yingtong Dou, Carl Yang, Kai Zhang, Ji Wang, S Yu Philip, Lifang He, and Bo Li. 2022. Adversarial attack and defense on graph data: A survey. *TKDE* 35, 8 (2022), 7693–7711.
[84] Ki Hyun Tae and Steven Euijong Whang. 2021. Slice tuner: A selective data acquisition framework for accurate and fair machine learning models. In *SIGMOD*. 1771–1783.
[85] Vincent Tjeng, Kai Yuanqing Xiao, and Russ Tedrake. 2019. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In *International Conference on Learning Representations (ICLR)*. OpenReview.net.
[86] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness May Be at Odds with Accuracy. In *International Conference on Learning Representations (ICLR)*. OpenReview.net.
[87] Xingchen Wan, Henry Kenlay, Robin Ru, Arno Blaas, Michael A. Osborne, and Xiaowen Dong. 2021. Adversarial Attacks on Graph Classifiers via Bayesian Optimisation. In *NIPS*. 6983–6996.
[88] Wenqi Wang, Run Wang, Lina Wang, Zhibo Wang, and Aoshuang Ye. 2021. Towards a robust deep neural network against adversarial texts: A survey. *TKDE* 35, 3 (2021), 3159–3179.
[89] Zichong Wang, Yang Zhou, Meikang Qiu, Israat Haque, Laura Brown, Yi He, Jianwu Wang, David Lo, and Wenbin Zhang. 2023. Towards Fair Machine Learning Software: Understanding and Addressing Model Bias Through Counterfactual Thinking. *arXiv preprint arXiv:2302.08018* (2023).
[90] Eric Wong and J. Zico Kolter. 2018. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *ICML (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 5283–5292.
[91] Kai Yuanqing Xiao, Vincent Tjeng, Nur Muhammad (Mahi) Shafiullah, and Aleksander Madry. 2019. Training for Faster Adversarial Robustness Verification via Inducing ReLU Stability. In *International Conference on Learning Representations (ICLR)*. OpenReview.net.
[92] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating Unwanted Biases with Adversarial Learning. *CoRR* abs/1801.07593 (2018). arXiv:1801.07593 http://arxiv.org/abs/1801.07593
[93] Hantian Zhang, Xu Chu, Abolfazl Asudeh, and Shamkant B Navathe. 2021. Omnifair: A declarative system for model-agnostic group fairness in machine learning. In *SIGMOD*. 2076–2088.
[94] Hantian Zhang, Ki Hyun Tae, Jaeyoung Park, Xu Chu, and Steven Euijong Whang. 2023. iFlipper: Label Flipping for Individual Fairness. *Proc. ACM Manag. Data* 1, 1 (2023), 8:1–8:26.
[95] Jie M. Zhang and Mark Harman. 2021. "Ignorance and Prejudice" in Software Fairness. In *ICSE*. IEEE, 1436–1447.
[96] Tianyi Zhou, Shengjie Wang, and Jeff A. Bilmes. 2021. Robust Curriculum Learning: From Clean Label Detection to Noisy Label Self-correction. In *International Conference on Learning Representations (ICLR)*.
[97] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations (ICLR)*. OpenReview.net.

A PROOF OF THEOREM 1

Theorem 1: *There exists an inherent tradeoff between the accuracy and robustness of any ML classifier $\mathcal{M}$, when $\mathcal{M}$ is fine-tuned with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ and evaluated with the tuples in unpoisoned $\mathcal{D}_{\text{pred}}$ (resp. poisoned $\mathcal{D}^-_{\text{pred}}$) for accuracy (resp. robustness).*

Specifically, under any attacker with power $\lambda \in (0, 0.5)$, any ML classifier $\mathcal{M}$ with accuracy $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}}) = 1 - \eta$ for $\eta \in [0, 1]$, its robustness $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}})$ is at most $(1-\lambda)\cdot(1-\eta) + \frac{\lambda\cdot(p+v_d-2\cdot v_d\cdot p)\cdot\eta}{(1-p)}$, where $p \in [0.5, 1]$ quantifies the correlation between decisive attributes and labels in $\mathcal{D}_{\text{pred}}$, and v_d is the probability for an attacker to poison decisive attributes. □

The robustness is bounded by three factors: (a) the correlation p between the decisive attributes and the labels in $\mathcal{D}_{\text{pred}}$; the higher p is, the most robust $\mathcal{M}$ is. (b) The probability v_d for the decisive attributes being attacked; the higher v_d, the less robust. (c) The attacker's power λ; the higher λ is, the lower the robustness is.

Proof: We start with basic constructs and notations.

Datasets: Our datasets consist of tuples with attribute-label pairs (x, y) of a database schema $\mathcal{R}$, where $y \in \{0, 1\}$. Assume *w.l.o.g.* that $\mathcal{R}$ includes a decisive attribute A_1 that has a significantly stronger correlation with the label y of tuples than the other $k-1$ non-decisive attributes $A_2, \ldots, A_k$ in $\mathcal{R}$. Note that A_1 can represent a combination of multiple attributes whose joint distribution collectively influences y. Moreover, we assume *w.l.o.g.* that the (sampled) $\mathcal{D}_{\text{pred}}$ is balanced, *i.e.,* it contains the same number of tuples *w.r.t.* different labels ($\Pr[y = 0] = \Pr[y = 1] = 0.5$).

Assume *w.l.o.g.* that in unpoisoned prediction datasets $\mathcal{D}_{\text{pred}}$, the values x_1 of attribute A_1 in the tuples follow the distribution:

$$x_1 = \begin{cases} y & \text{with the probability } p, \\ \neg y & \text{with the probability } 1-p, \end{cases}$$

Assume *w.l.o.g.* $p \geq 0.5$. This distribution simplifies the event "x_1 *implies* y" (resp. " x_1 implies $\neg y$") into $x_1 = y$ (resp. $x_1 = \neg y$), abstracting the complexity of A_1's role into a binary relationship for analytical clarity, *i.e.,* for tuples with $x_1 = y$ (resp. $x_1 = \neg y$), A_1 is positively (resp. negatively) correlated with y. Denote by $\mathcal{P}_0$ (resp. $\mathcal{P}_1$) the distribution of the values $x_2, \ldots, x_k$ of attributes $A_2, \ldots, A_k$ in the tuples when the tuple label $y = 0$ (resp. $y = 1$).

Expected loss of $\mathcal{M}$: As shown in [86], ML classifiers learn decisive attributes for stability but often rely on non-decisive attributes to boost their accuracy, making them more vulnerable to attacks. Observe that the expected loss of $\mathcal{M}$ is given by the probability $\Pr[M(x) = y]$ that the predictions match the true labels. To see the impact of both attribute types on performance, consider four events where $\mathcal{M}$ makes the prediction $y = 1$ based on the decisive attribute x_1 and the non-decisive attributes $x_2, \ldots, x_k$. For prediction tuples with non-decisive attributes following $\mathcal{P}_1$, we have:

- e_1: $\mathcal{M}$ predicts $y = 1$ when $x_1 = 1$, *i.e.,* $\{M(x) = 1 \mid x_1 = 1\}$.
- e_2: $\mathcal{M}$ predicts $y = 1$ when $x_1 = 0$, *i.e.,* $\{M(x) = 1 \mid x_1 = 0\}$.

Similarly, when these attributes follow $\mathcal{P}_0$, we define:

- e_3: $\mathcal{M}$ predicts $y = 1$ when $x_1 = 1$, *i.e.,* $\{M(x) = 1 \mid x_1 = 1\}$.
- e_4: $\mathcal{M}$ predicts $y = 1$ when $x_1 = 0$, *i.e.,* $\{M(x) = 1 \mid x_1 = 0\}$.

Denote by $\Pr[e_i] = p_i$ for $i \in [1, 4]$ the probabilities of these events.

Note that although e_1 and e_3 (resp. e_2 and e_4) share the same condition $x_1 = 1$, they represent different prediction data with distinct non-decisive attribute distributions, and moreover, the complement of each event e_i ($i \in [1, 4]$) represents the probability that classifier $\mathcal{M}$ predicts $y = 0$ under the same conditions.

Attacker $\mathcal{A}$: An attacker $\mathcal{A}$ with power λ selects λ% of tuples in $\mathcal{D}_{\text{train}}$ and corrupts their attributes $A_j \in \mathcal{R}$ with the probability $1 - v_j$ ($v_j \in \mathcal{V}$), where $\mathcal{V}$ is the importance vector defined in Section 2.1. Observe that $\mathcal{D}^-_{\text{pred}}$ contains λ% of poisoned tuples, denoted as $\mathcal{D}_{\text{poi}}$, and $(1-\lambda)$% of unpoisoned tuples from $\mathcal{D}_{\text{pred}}$, denoted as $\mathcal{D}_{\text{upoi}}$, *i.e.,* $\mathcal{D}^-_{\text{pred}} = \mathcal{D}_{\text{poi}} \cup \mathcal{D}_{\text{upoi}}$ and $\mathcal{D}_{\text{poi}} \cap \mathcal{D}_{\text{upoi}} = \emptyset$. We assume *w.l.o.g.* that all attacks on the selected tuples are "effective", *i.e.,* they invert the attribute value distributions from y to $\neg y$. For instance, given a selected tuple t, when $\mathcal{A}$ chooses to poison x_1, it flips x_1; when non-decisive attributes $x_2, \ldots, x_k$ are chosen, if $t[y] = 0$ (resp. $t[y] = 1$), $\mathcal{A}$ changes the values of non-decisive attributes to follow the distribution $\mathcal{P}_1$ (resp. $\mathcal{P}_0$). Moreover, as discussed in Section 2.1, we focus on attackers that tend to inject noise into imperceptible (more likely to be non-decisive) attributes, and thus, we assume that attackers select and poison non-decisive (resp. decisive) attributes with probability 1 (resp. v_d). Consequently, for the attacked x_1 in $\mathcal{D}_{\text{poi}}$, denoted by x'_1, we have:

$$x'_1 = \begin{cases} y & \text{with the probability } p\cdot(1-v_d) + (1-p)\cdot v_d, \\ \neg y & \text{with the probability } p\cdot v_d + (1-p)\cdot(1-v_d). \end{cases}$$

Denote $p\cdot(1-v_d)+(1-p)\cdot v_d$ as p' and $p\cdot v_d+(1-p)\cdot(1-v_d) = 1-p'$.

In addition, we consider the impact of adversarial attacks on non-decisive attributes $x_2, \ldots, x_k$ where the adjusted probabilities p'_i for events e_i ($i \in [1, 4]$) are calculated as follows:

$$p'_1 = p_3, \quad p'_2 = p_4, \quad p'_3 = p_1, \quad p'_4 = p_2,$$

We next study the accuracy and robustness of the given ML classifier $\mathcal{M}$ trained with the enhanced dataset $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$.

Accuracy of $\mathcal{M}$: To simplify notations, we denote $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}})$ by acc. Then we can deduce the following:

$$\begin{aligned} \text{acc} &= \Pr_{(x,y)\in\mathcal{D}_{\text{pred}}}[\mathcal{M}(x) = y] = \Pr[y = 1]\cdot[p\cdot p_1 + (1-p)\cdot p_2] \\ &\quad + \Pr[y = 0]\cdot[p\cdot(1-p_4) + (1-p)\cdot(1-p_3)] \\ &= \frac{1}{2}\cdot[p\cdot(1+p_1-p_4) + (1-p)\cdot(1+p_2-p_3)], \end{aligned} \tag{4}$$

i.e., the conditional probability that $\mathcal{M}$ correctly predicts the label of the given prediction tuples based on the conditions on x_1 and the corresponding distributions of $x_2, \ldots, x_k$.

Robustness of $\mathcal{M}$: Denote $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}})$ by rob, where $\text{rob} = \Pr_{(x,y)\in\mathcal{D}^-_{\text{pred}}}[\mathcal{M}(x) = y]$. Since $\mathcal{D}^-_{\text{pred}}=\mathcal{D}_{\text{poi}} \bigcup \mathcal{D}_{\text{upoi}}$ and $\mathcal{D}_{\text{poi}} \bigcap \mathcal{D}_{\text{upoi}} = \emptyset$, one can get the following:

$$\begin{aligned} \text{rob} &= (1-\lambda)\cdot \Pr_{(x,y)\in\mathcal{D}_{\text{upoi}}}[\mathcal{M}(x) = y] + \lambda\cdot \Pr_{(x,y)\in\mathcal{D}_{\text{poi}}}[\mathcal{M}(x) = y] \\ &= (1-\lambda)\cdot\text{acc} + \lambda\cdot\text{acc}_{\text{poi}}, \end{aligned} \tag{5}$$

where acc_{poi} is the accuracy of $\mathcal{M}$ on the poisoned data $\mathcal{D}_{\text{poi}}$.

Recall that we assume $\mathcal{A}$ flips x_1 to $\neg x_1$ with the probability v_d and steers $\mathcal{P}_0$ (resp. $\mathcal{P}_1$) to $\mathcal{P}_1$ (resp. $\mathcal{P}_0$) for $x_2, \ldots, x_k$, changing the

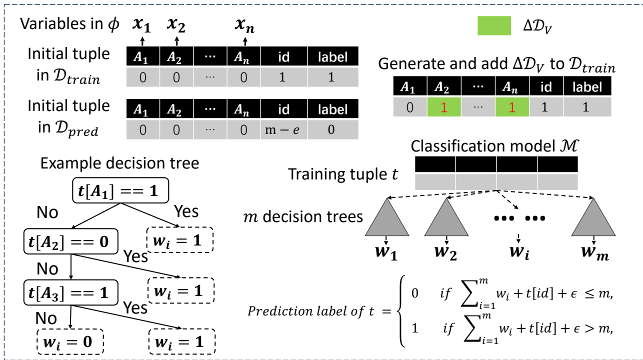


Figure 6: Reduction example for DEAAT

distributions for both decisive and non-decisive attributes. Putting these together, we can calculate acc_{poi} as follows:

$$\begin{aligned}
\text{acc}_{\text{poi}} &= \Pr[y=1] \cdot \left[p' \cdot p'_1 + (1-p') \cdot p'_2\right] \\
&\quad + \Pr[y=0] \cdot \left[p' \cdot (1-p'_4) + (1-p') \cdot (1-p'_3)\right] \\
&= \frac{1}{2}\left[p' \cdot (1+p_3-p_2) + (1-p') \cdot (1+p_4-p_1)\right]. \qquad (6)
\end{aligned}$$

Tradeoff between accuracy and robustness. We show the tradeoff by studying the upper bound of the robustness of $\mathcal{M}$ (trained on $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$) when its accuracy is $1-\eta$, where $\eta \in [0,1]$.

Denote by θ_1 and θ_2 the terms $(1+p_4-p_1)$ and $(1+p_3-p_2)$ in Equations 6, respectively. When $\text{acc} = 1-\eta$, we have

$$\begin{aligned}
\text{acc} &= \frac{1}{2} \cdot \left[p \cdot (2-\theta_1) + (1-p) \cdot (2-\theta_2)\right] \\
&= 1 - \frac{1}{2} \cdot \left[p \cdot \theta_1 + (1-p) \cdot \theta_2\right] = 1-\eta \\
\Rightarrow &\qquad p \cdot \theta_1 + (1-p) \cdot \theta_2 = 2\eta \qquad (7)
\end{aligned}$$

We amplify Equation 6 by a coefficient $\frac{p \cdot p'}{(1-p)\cdot(1-p')}$ to the term $(1-p') \cdot (1+p_4-p_1)$. Since $p \geq 0.5$ and $v_d \leq 1$, it follows that $\frac{p \cdot p'}{(1-p)\cdot(1-p')} \geq 1$. Combined with Equation 6 and 7 we have:

$$\begin{aligned}
\text{acc}_{\text{poi}} &= \frac{1}{2} \cdot \left[(1-p') \cdot \theta_1 + p' \cdot \theta_2\right] \\
&\leq \frac{1}{2} \cdot \left[\frac{p \cdot p'}{(1-p)\cdot(1-p')} \cdot (1-p') \cdot \theta_1 + p' \cdot \theta_2\right] \\
&= \frac{1}{2} \cdot \frac{p'}{(1-p)} \cdot \left[p \cdot \theta_1 + (1-p) \cdot \theta_2\right] \\
&= \frac{(p + v_d - 2 \cdot v_d \cdot p) \cdot \eta}{(1-p)}. \qquad (8)
\end{aligned}$$

Putting this together with Equation 5, we can calculate the upper bound of the robustness of model $\mathcal{M}$ as follows:

$$\begin{aligned}
\text{rob} &= (1-\lambda) \cdot \text{acc} + \lambda \cdot \text{acc}_{\text{poi}} \\
&\leq (1-\lambda) \cdot (1-\eta) + \frac{\lambda \cdot (p + v_d - 2 \cdot v_d \cdot p) \cdot \eta}{(1-p)} \qquad (9)
\end{aligned}$$

That is, when $\mathcal{M}$ trained with enhanced $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ achieves the accuracy $1-\eta$ for $\eta \in [0,1]$, the robustness is at most $(1-\lambda) \cdot (1-\eta) + \frac{\lambda \cdot (p+v_d-2\cdot v_d \cdot p)\cdot\eta}{(1-p)}$. This completes the proof. □

B PROOF OF THEOREM 2

Theorem 2: *The* DEAAT *and* DEAAP *problems are NP-hard.* □

Proof: We show that DEAAT and DEAAP are NP-hard by reduction from 3SAT (cf. [33]). Given a Boolean formula in conjunctive normal form (CNF) with n variables and m clauses $\phi = (c_1 \wedge c_2 \wedge \ldots \wedge c_m)$, where each clause c_i is a disjunction of literals $c_i = (l_{i1} \vee l_{i2} \vee l_{i3})$ and each literal l_{ij} is a variable x_k or its negation $\neg x_k$, 3SAT is to determine whether there exists an assignment of truth values to the variables such that the entire formula ϕ is true.

DEAAT. The reduction from 3SAT to DEAAT is given as follows.

(1) Schema $\mathcal{R}$: We use a database schema $(A_1, \ldots, A_n, \text{id}, \text{label})$, where for all $i \in [1,n]$, A_i is a Boolean attribute, id is a positive integer, and label is a binary $\{0,1\}$.

(2) Datasets: We construct relations $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ of schema $\mathcal{R}$, each consisting of one tuple. For the tuple $t \in \mathcal{D}_{\text{train}}$, $t[\text{id}] = 1$, $t[A_i] = 0$ $(i \in [1,n])$, and $t[\text{label}] = 1$. Intuitively, we use t's attributes $A_1 \sim A_n$ to encode the n variables of the 3SAT instance ϕ.

For the tuple $s \in \mathcal{D}_{\text{pred}}$, we set $s[\text{id}] = m-e$, $s[A_i] = 0$ $(i \in [1,n])$, and $s[\text{label}] = 0$, where e is the number of clauses satisfied in the 3SAT instance ϕ when all variables are set to false. We assume *w.l.o.g.* $e \leq m-1$; otherwise, a satisfied truth assignment is found.

(3) Data cleaning tool C: When a tuple's Boolean attribute is deemed poisoned, C switches it from true to false, and vice versa.

Intuitively, we use the data cleaning tool C to perform value perturbations on a subset of values $t[A_i]$ $(i \in [1,n])$ for the tuple $\tau \in \mathcal{D}_{\text{train}}$, *i.e.,* we flip $t[A_i]$ for attributes of t, to simulate the truth assignment of ϕ. We will use ML model $\mathcal{M}$ to verify clause satisfaction in ϕ, and moreover, we will utilize the pairwise correspondence between attributes of the tuple t (resp. s) in $\mathcal{D}_{\text{train}}$ (resp. $\mathcal{D}_{\text{pred}}$) to identify a set $\Delta\mathcal{D}_V$ of value perturbations.

(4) Model $\mathcal{M}$: We develop a tree-based (regression) classification model $\mathcal{M}$ that leverages decision trees, where each tree assesses the satisfaction of a specific 3SAT clause by evaluating the A_i values.

The model's regression function takes as input n attribute values of a tuple. The decision trees output $w_i = 1$ when a clause C_i is satisfied, and 0 otherwise. Figure 6 illustrates the function with an example clause $C_i = (x_1 \vee \neg x_2 \vee x_3)$. The regression function $f(t) = \sum_{i=1}^{m} w_i + t[\text{id}] + \epsilon$ combines the outputs of the trees, and $\mathcal{M}$ predicts the label for the input tuple t as 0 if $f(t) \leq m$, and 1 otherwise, where ϵ is a trainable parameter.

Observe that before perturbing values on $\mathcal{D}_{\text{train}}$, for the tuple $t \in \mathcal{D}_{\text{train}}$ with $t[\text{id}] = 1$ and $t[\text{label}] = 1$, the function returns $f(t) = e+1+\epsilon$. Based on the mean squared error, the training adjusts ϵ such that $\epsilon > m-e-1$. When applying $\mathcal{M}$ to predict the label of tuple $s \in \mathcal{D}_{\text{pred}}$, we have $f(s) = e + s[\text{id}] + \epsilon > 2m-e-1 > m$, such that the predicted label is 1 which is not equal to $s[\text{label}]$ (*i.e.,* 0); as a result, $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{pred}}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{pred}}) = 0$.

(5) Parameter B_{train}: Define $B_{\text{train}} = 1$, which is the maximum possible improvement with the initial $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{pred}}) = 0$.

The reduction can be obviously constructed in PTIME. We next show that there exists a set $\Delta\mathcal{D}_V$ of value perturbations that makes $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) \geq B_{\text{train}}$ if and only if the 3SAT instance ϕ is satisfiable.

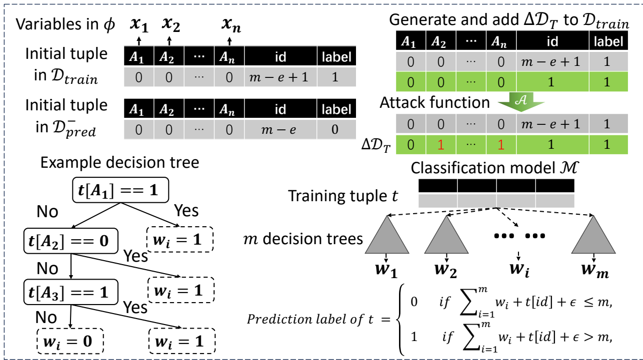


Figure 7: Problem DEAAP reduction example

$\Rightarrow$ First, assume that there exists a set $\Delta\mathcal{D}_V$ such that after the value perturbations on $\mathcal{D}_{\text{train}}$, $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) = 1$. More specifically, the perturbations ensure that for any tuple t, if $t[A_i] \in \Delta\mathcal{D}_V$, $t[A_i]$=1; otherwise, $t[A_i]$ remains 0. In fact, by definition, $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) = 1$ implies that the expected loss of the trained $\mathcal{M}$ on $\mathcal{D}_{\text{pred}}$ is 0, *i.e.,* the model $\mathcal{M}$ reaches its maximum possible accuracy on $\mathcal{D}_{\text{pred}}$. Note that this can be achieved when the assigned attribute values of t satisfy all clauses. Because given $s[\text{label}] = 0$ and $f(s) = \sum_{i=1}^{m} w_i + s[\text{id}] + \epsilon = m + \epsilon$, we must have $\epsilon = \frac{\sum_{t \in (\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V)} (m+1-t[\text{id}] - \sum_{i=1}^{m} w_i)}{|\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V|}$. If $\epsilon = 0$, 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 $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$.

We give the truth assignment in ϕ as follows. For the tuple t in $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}$, if $t[A_i] = 1$, then variable x_i is assigned true. Otherwise, if $t[A_i] = 0$, then x_i is set to be false.

One can verify that this truth assignment satisfies ϕ. Observe that the truth assignment is determined by $\mathcal{M}$'s decision trees, each aligned with a specific clause C_i from ϕ. A tree yields $w_i = 1$ iff for the $\mathcal{M}$'s input tuple $t \in \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}$ and ordered by $t[\text{id}]$, at least one $t[A_i]$ value in t meets the decision criterion, which is equivalent to satisfying clause C_i by ensuring the truth of a literal. This leads to $\sum_{i=1}^{m} w_i = m$, with $w_i = 1$ for all trees, which verifies ϕ's satisfiability by the truth assignment derived from $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$.

$\Leftarrow$ Conversely, we show that the existence of a satisfying truth assignment for ϕ guarantees the existence of a set $\Delta\mathcal{D}_V$ such that flipping the values in $\mathcal{D}_{\text{train}}$ that appear in $\Delta\mathcal{D}_V$ ensures $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) = 1$. Given a satisfying truth assignment for ϕ, the set $\Delta\mathcal{D}_V$ is identified as follows. For each variable x_i in ϕ, if x_i is true in the truth assignment, then for the tuple t in $\mathcal{D}_{\text{train}}$, we include the attribute $t[A_i]$ in the value perturbation set $\Delta\mathcal{D}_V$. Then we perform the perturbations to obtain $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$, where for each $t[A_i] \in \Delta\mathcal{D}_V$, $t[A_i] = 1$, while $t[A_j] = 0$ for $A_j \notin \Delta\mathcal{D}_V$. Suppose by contradiction that $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{D}_{\text{pred}}) < 1$; then it implies that the tuple s in $\mathcal{D}_{\text{pred}}$ is incorrectly classified. Given $f(s) = e + s[\text{id}] + \epsilon = m + \epsilon$, where $s[\text{label}] = 0$, ϵ must be greater than 0 in order to produce incorrect predictions. A positive ϵ contradicts the premise that ϕ is satisfiable. Indeed, during the training, for the tuple $t \in \mathcal{D}_{\text{train}}$ with $t[\text{label}] = 1$, $\mathcal{M}$ outputs $f(t) = m + t[\text{id}] + \epsilon > m$; this indicates that $\epsilon = 0$ to train an accurate classifier $\mathcal{M}$ with the minimum training loss.

<u>DEAAP</u>. We prove its NP-hardness by reducing 3SAT to a special case of DEAAP where $B_{\text{inf}} = 0$, *i.e.,* it is intractable even without considering the accuracy of $\mathcal{M}$ on the original test data $\mathcal{D}_{\text{pred}}$.

(1) Schema $\mathcal{R}$: We use the same schema $(A_1, \ldots, A_n, \text{id}, \text{label})$ as in DEAAT, where A_i $(i \in [1, n])$ is a Boolean attribute, id is a positive integer and label is a binary value in $\{0, 1\}$.

(2) Datasets: We construct relations $\mathcal{D}_{\text{train}}$, $\mathcal{D}^-_{\text{pred}}$ and $\mathcal{D}_{\text{pred}}$ of schema $\mathcal{R}$, each having one tuple. For the tuple $t \in \mathcal{D}_{\text{train}}$, $t[\text{id}] = m - e + 1$, $t[A_i] = 0$ $(i \in [1, n])$, and $t[\text{label}] = 1$, where e is the number of satisfied clauses in ϕ when all variables are set false. Similarly, we use t's attributes $A_1 \sim A_n$ to encode n variables of ϕ.

For datasets $\mathcal{D}^-_{\text{pred}}$ and $\mathcal{D}_{\text{pred}}$, for the tuple $s \in \mathcal{D}^-_{\text{pred}}$, we let $s[\text{id}] = m - e$, $s[A_i] = 0$ $(i \in [1, n])$, and $s[\text{label}] = 0$. For the tuple $c \in \mathcal{D}_{\text{pred}}$, its id, A_i and label can be any valid value.

(3) Attack function $\mathcal{A}$: We develop a function $\mathcal{A}$ that flips the Boolean value of $t[A_i]$ given the input tuple t and attributes A_i.

Intuitively, to enhance $\mathcal{D}_{\text{train}}$ with tuples $u \in \Delta\mathcal{D}_T$ of the schema $\mathcal{R}$, we *w.l.o.g.* construct one initial tuple with $u[\text{id}] = 1$, $u[A_i] = 0$ for $i \in [1, n]$ and $u[\text{label}] = 1$. Then, attacker $\mathcal{A}$ adjusts the tuple u so that $\mathcal{M}$ learns the attack pattern, where $\mathcal{A}$ simulates the truth assignment of ϕ. The decision trees in model $\mathcal{M}$ then check clause satisfaction in ϕ, and the prediction result on the dataset $\mathcal{D}^-_{\text{pred}}$ identifies the suitable tuple perturbations $\Delta\mathcal{D}_T$.

(4) Model $\mathcal{M}$: We develop a tree-based classification model $\mathcal{M}$ that uses m decision trees, where each tree evaluates the satisfaction of a specific 3SAT clause by assessing $t[A_i]$ for each tuple t. The i-th tree yields $w_i = 1$ if clause C_i is satisfied, otherwise $w_i = 0$. For example, Figure 7 shows a decision tree for the clause $C_i = (x_1 \vee \neg x_2 \vee x_3)$.

More specifically, the model $\mathcal{M}$ predicts labels for each tuple t using the same regression function as in DEAAT, *i.e.,* $f(t) = \sum_{i=1}^{m} w_i + t[\text{id}] + \epsilon$ with the prediction result as 0 if $f(t) \leq m$ and 1 otherwise, where ϵ is a trainable parameter.

(5) Parameters B_{inf} and R_{inf}: Let $B_{\text{inf}} = 0$, making $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}}) \geq B_{\text{inf}}$ hold for any $\Delta\mathcal{D}_T$ and $\mathcal{D}_{\text{pred}}$; and $R_{\text{inf}} = 1$, *i.e.,* the maximum possible value of $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}})$.

The reduction can be constructed in PTIME. We next show that there exists a tuple perturbation set $\Delta\mathcal{D}_T$ that makes $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}}) \geq R_{\text{inf}}$ and $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}}) \geq B_{\text{inf}}$ iff the 3SAT instance ϕ is satisfiable.

$\Rightarrow$ Assume that there exists a set $\Delta\mathcal{D}_T$ such that $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}}) \geq 1$. More specifically, $\Delta\mathcal{D}_T$ contains tuples generated by $\mathcal{A}$ perturbing on the initial tuple u with $u[\text{id}]$=1, $u[A_i]$=0 for $i \in [1, n]$ and $u[\text{label}] = 1$. Moreover, $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}}) \geq 1$ implies that $\mathcal{M}$ classifies the tuple s in $\mathcal{D}^-_{\text{pred}}$ correctly. Given $f(s) = \sum_{i=1}^{m} w_i + s[\text{id}] + \epsilon = m + \epsilon$ and $s[\text{label}] = 0$, it necessitates $\epsilon = 0$. Recall that during training on tuples t in $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ that contain 2 tuples t_1 and t_2 with $t_1[\text{id}] = m - e + 1$, $t_2[\text{id}] = m - e$, $t_1[\text{label}] = t_2[\text{label}] = 1$, $f(t_1) = m + 1 + \epsilon$ and $f(t_2) = \sum_{i=1}^{m} w_i + 1 + \epsilon$. In order to classify t_1 (resp. t_2) correctly, it requires $\epsilon > -1$ (resp. $\epsilon > m - \sum_{i=1}^{m} w_i - 1$). The lower bound of the optimal ϵ is determined by $\max\{-1, m-\sum_{i=1}^{m} w_i - 1\}$; if the optimal ϵ is found to be 0, it implies that $m-\sum_{i=1}^{m} w_i - 1 < 0 \Rightarrow \sum_{i=1}^{m} w_i = m$, indicating that each of the m clauses has been satisfied by the attribute settings of t_2, where $\{t_2\} = \Delta\mathcal{D}_T$.

We give the truth assignment in ϕ as follows. For the tuple t_2 in $\Delta\mathcal{D}_T$ and $i \in [1, n]$, if $t_2[A_i] = 1$, then variable x_i is assigned true. Otherwise, if $t_2[A_i] = 0$, then x_i is set to be false.

One can verify that this truth assignment satisfies ϕ. Observe that the truth assignment is checked by $\mathcal{M}$'s decision trees, each aligned with a clause C_i from ϕ. A tree yields $w_i = 1$ iff for the tuple t_2 and any $i \in [1, n]$, at least one $t_2[A_i]$ 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 $\Delta\mathcal{D}_T$.

$\Leftarrow$ Conversely, we show that the existence of a satisfying truth assignment for ϕ guarantees the existence of a set $\Delta\mathcal{D}_T$ of tuple perturbations such that $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}}) \geq 1$ subject to $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}_{\text{pred}}) \geq 0$. Given a satisfying truth assignment for ϕ, the set $\Delta\mathcal{D}_T$ is identified as follows. We first initialize a tuple u in $\Delta\mathcal{D}_T$ with $u[\text{id}]$=1 and $u[\text{label}]$=1. For each variable x_i in ϕ, if x_i is true in the truth assignment, then we use the attack function $\mathcal{A}$ to set $u[A_i]$=1; otherwise $u[A_i]$=0. Suppose by contradiction that $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{D}^-_{\text{pred}}) < 1$, *i.e.,* the tuple s in $\mathcal{D}^-_{\text{pred}}$ is incorrectly classified. Given $f(s) = e + s[\text{id}] + \epsilon$, where $s[\text{id}] = m - e$ and $s[\text{label}] = 0$, ϵ must be greater than 0 in order to produce incorrect predictions. A non-zero ϵ contradicts the premise that ϕ is satisfiable. Indeed, during the training, for tuples t_1 and t_2 in $\mathcal{D} \oplus \Delta\mathcal{D}_T$ with $t_1[\text{label}] = t_2[\text{label}] = 1$, the regression function outputs $f(t_1) = f(t_2) = m + 1 + \epsilon$; this indicates that $\epsilon = 0$ can produce an accurate classifier $\mathcal{M}$ that correctly predicts the label of both t_1 and t_2 during the training process. □

C DETAILS

Determining the size of S_{ct}. We separate the following cases. (a) If $g_1.\text{dr} \geq 0$ (*i.e.,* there exist more misclassified tuples than correctly predicted ones in g_1), we enhance $\mathcal{D}_{\text{train}}$ with $|S_{\text{ct}}|$ tuples in g_1 such that $\mathcal{M}$ can correctly predict more than 50% tuples in $g_1 \cap S_{\text{at}}$, *i.e.,* $\frac{|S_{\text{mt}} \cap g_1| - |(S_{\text{at}} \setminus S_{\text{mt}}) \cap g_1| - |S_{\text{ct}}|}{|g_1|} < 0 < \frac{|S_{\text{mt}} \cap g_1| - |(S_{\text{at}} \setminus S_{\text{mt}}) \cap g_1| - (|S_{\text{ct}}| - 1)}{|g_1|}$. (b) If $g_1.\text{dr} < 0$ (*i.e.,* $\mathcal{M}$ can correctly classify more than 50% tuples in $S_{\text{at}} \cap g_i$ for all g_i), we enhance $\mathcal{D}_{\text{train}}$ with $|S_{\text{ct}}|$ tuples in g_1 such that the priority level of g_1 is likely to be reduced from #1 to #2, *i.e.,* $\frac{|S_{\text{mt}} \cap g_1| - |(S_{\text{at}} \setminus S_{\text{mt}}) \cap g_1| - |S_{\text{ct}}|}{|g_1|} < \frac{|S_{\text{mt}} \cap g_2| - |(S_{\text{at}} \setminus S_{\text{mt}}) \cap g_2|}{|g_2|} \leq \frac{|S_{\text{mt}} \cap g_1| - |(S_{\text{at}} \setminus S_{\text{mt}}) \cap g_1| - (|S_{\text{ct}}| - 1)}{|g_1|}$. We hereby deduce the size $|S_{\text{ct}}|$.