COVER LETTER OF PAPER 1146

Dear Meta Reviewer and Referees:

We have substantially revised the paper following the Referees' suggestions. For the convenience of Referees, changes for addressing the concerns of R1, R2 and R3 are color-coded in blue, orange and green, respectively. We would like to thank the Referees for insightful comments and for supporting this work!

Below please find our responses to the reviews.

Response to the comments of the Meta Reviewer.

[MetaO1 & Feedback] *The main concern is if the reduction to data cleaning is sufficient for mitigating adversarially corrupted examples.*

Provide strong justifications using empirical studies on real-world adversarial attacks on tabular data to show the tainted data has the same properties as typical data quality issues used in the paper and can be fixed using the tools for data cleaning.

Please pay extra attention to (1) clearly show the connection between cleaning and their influence to the training process, e.g., the loss function optimization; (2) use real-world attacks to tabular data to validate the assumptions.

[A] Thanks! We have addressed the concerns about (a) real-world adversarial attacks in the response to R2[O1,O3,O8] and R3[O1], and (b) data cleaning tools in the response to R1[O6,O7] and R2[O2].

[MetaO2] *Justify rigorously how adding more tuples to the inference data will always prevent an ML model to learn an undesirable model in all settings.*

[A] We have justified this issue in the response to R2[O5].

[MetaO3] *Show that the complexity results in Section 2 hold for input data with a fixed distribution (rather than a fixed dataset).*

[A] We have extended the proof in the response to R2[O6].

[MetaO4] *Show that the approaches proposed in Section 5 for adding new tuples will work for the cases where the tuples are selected from an unknown distribution (as assumed in Section 2.2).*

[A] We have clarified this in the response to R2[O3].

[MetaO5] *Clarify whether data enhancement proposed in Section 5 by adding new tuples to the inference data must be done for every inference for a new data item. If so, show that this can be done efficiently over large datasets.*

[A] We have clarified this issue in the response to R2[O4].

Response to the comments of Referee #1.

[R1O1] *I suggest changing the title to make it more specific, such as data enhancement for binary classification, and perhaps include relational data in the title to avoid confusion with the work for images.*

[A] Thanks! We have changed the title to "Data Enhancement for Binary Classification of Relational Data" as suggested (pp. 1).

[R1O2] *The distinction between data cleaning and data enhancing for ML should be explicitly stated earlier in the introduction, given that the framework does employ data cleaning.*

[A] Thanks! We have clarified the distinction between data cleaning and data enhancing for ML earlier in Section 1, prior to the description of our second contribution (pp. 2).

[R1O3] *Can $\mathcal{A}_1$ and $\mathcal{A}_2$ be the same attacker, is it possible to have several $\mathcal{A}$ in the prediction?*

[A] Thanks! We have clarified this (pp. 4, Section 2.1) as follows.

(1) We use $\mathcal{A}_1$ and $\mathcal{A}_2$ to distinguish adversarial attacks at the training and prediction stages of ML model lifecycle, respectively. Although the two often represent diverse attack patterns, they may be the same attack pattern when an attacker spans both stages.

(2) It is possible to have several $\mathcal{A}$ in the prediction data. We treat the combination of these attackers as a compound attacker $\hat{\mathcal{A}}_2$, and fine-tune $\mathcal{M}$ using tuple perturbations generated by $\hat{\mathcal{A}}_2$.

(3) We have added experiments with compound attackers (Figure 5(m) & pp. 12; please also see the response to R3[O2]).

[R1O4] *Is there a method for quantifying the trade-off between accuracy and robustness, particularly when using both $\Delta\mathcal{D}_V$ and $\Delta\mathcal{D}_T$? Is there any guarantee provided on the gain of robustness?*

[A] Thanks! We have clarified the trade-off as follows.

(1) We have clarified (pp. 4) that it is hard to quantify the trade-off due to three key challenges: (a) the stochastic nature of adversarial attacks [43], which makes attack generation unpredictable; (b) dynamic shifts in data distributions [97], making it hard to find a stable patterns; and (c) the nonlinear and high-dimensional behavior of ML models [75], which further complicates the quantification.

While Theorem 1 establishes an upper bound for the robustness, providing a theoretical lower bound remains an open problem.

(2) It is difficult to precisely characterize the trade-off and robustness gain in terms of $\Delta\mathcal{D}_V$ and $\Delta\mathcal{D}_T$. (a) The trade-off arises only in Phase 2, where $\Delta\mathcal{D}_V$ from Phase 1 debugs adversarial data injected in the training set. (b) As shown in the second ablation study (pp. 12), changes in data distribution introduced by $\Delta\mathcal{D}_V$ of Phase 1 may also influence the robustness gain achieved by $\Delta\mathcal{D}_T$ in Phase 2.

[R1O5] *In what manner do B_{inf} and R_{inf} relate to one another?*

[A] Thanks! We have clarified this in Section 2.2 (pp. 5). More specifically, B_{inf} denotes the accuracy lower bound $1-\eta$ (Theorem 1). When B_{inf} decreases with a larger η, R_{inf} increases, reflecting the accuracy-robustness trade-off outlined in Theorem 1.

[R1O6] *The term "small" was employed to denote $\Delta\mathcal{D}_V$ and $\Delta\mathcal{D}_T$, despite the fact that the size of these sets is not at our control, as they contain all the values or tuples that satisfy the specified conditions (section 5). Could you please clarify this?*

[A] Thanks! We have clarified this (pp. 7-8 and 9, Sections 4 and 5).

(1) The size of $\Delta\mathcal{D}_V$ is relatively small (pp. 7-8). We find that for different datasets, $|\Delta\mathcal{D}_V|$ accounts for 5%-8% of the total data in our experimental study (Section 6). This is because the number of adversarial cells is usually small so that it is hard to detect them [55, 92].

(2) The size of $\Delta\mathcal{D}_T$ is relatively small under the accuracy constraint B_{inf}, since a larger B_{inf} leads to a smaller $|\Delta\mathcal{D}_T|$ (pp. 9).

[R1O7] *In what extent is this framework dependent on the quality of the data cleaning tool C? Moreover, it is stated in example 3 that the fixes are identified through association analysis of label y and attribute A_j. Could you please elaborate on the specificity of this statement if we employ Rock [9]? Or is it a general statement?*

[A] Thanks! We have clarified the following in Sections 3 and 4.

(1) We have clarified that only Phase 1 of DE4ML employs data cleaning tools to fix adversarial errors in $\mathcal{D}_{\text{train}}$ (pp. 6, Section 3), and DET performs better with more accurate C (pp. 8, Section 4).

(2) In Example 3 (pp. 6), we have clarified that the fixes are identified by association analysis of label Y and attributes A_j by data cleaning tools such as Rock [18], which unifies logic reasoning and ML predictions for association analysis and error detection/correction.

(3) We find that tools like Rock [18], Raha [77] and Baran [76] work well. This is because we reduce noise introduced by the tools by selectively fixing the most impactful errors only, outperforming cleaning all errors detected by 8.9% (Figure 5(a)).

[R1O8] *The baselines are not clearly described, so one can interpret the results with more context. Please add more details.*

[A] Thanks! We have added more details of the baselines (pp. 10).

[R1O9] *It would be interesting to see how table representation models would benefit from such a framework.*

[A] As suggested, we have evaluated DE4ML on FT-Transformer, a representative table representation model. As reported in Figures 5(b) and 5(i) (pp. 11), the rob on all tested models is 0.74 (resp. 0.68) on average in defusing attacks in $\mathcal{D}_{\text{train}}$ (resp. $\mathcal{D}_{\text{pred}}$), 6.5% (resp. 54.9%) higher than the baselines, up to 24.6% (resp. 466%). That is, DE4ML performs well on various models.

[R1M1] *Example 1: It would be helpful to make a statement about the amount of such changes, until they could mislead the classifier.*

[A] Thanks! In Example 1 (pp. 1), we have reported that fixing critical corrupted values (about 5%-8% of total data) improves the accuracy/robustness of $\mathcal{M}$ by 6.7% on average in our experiments.

[R1M2-M3] *There is repetition in the sections before the experiments section, which can be reduced. The decision problem of DEAAT, also denoted by DEAAT → remove the redundancy.*

[A] We have removed the repetitions as suggested (pp. 5-8).

Many thanks for your support and constructive suggestions!

Response to the comments of Referee #2.

[R2O1 & Feedback (F)] *The proposed methods in Section 4 make strong assumptions on the properties of corrupted values to detect them in Section 4 should substantiated with real-world studies. Generally, the framework in Section 2 requires detection of tainted data points in the dataset used for inference and prediction. I suggest that the authors clarify whether and in what cases it is possible.*

(F1) *Why is adversarial noise treated as random noise, given its fundamental differences in assumptions and impact (e.g., adversarial Multi-Armed Bandit vs. stochastic ones)?* (F2) *Typical data cleaning methods often fail to detect adversarial noise, such as when an adversary changes a feature value by 0.001, which significantly impacts loss function optimization but is not detected by these methods. The authors need to clarify why they believe noise aimed at influencing training, such as loss function optimization, can always be detected by data cleaning techniques that rely on general dataset characteristics rather than addressing influences on loss functions.* (F3) *The authors should also demonstrate that their claims hold regardless of the model type, loss function optimization method, or hyperparameters. Finally, these claims must be validated with real-world adversarial attack data to support the proposed method.*

[A] Many thanks! We clarify that (F1) we adopt the same setting as Picket [72], the SOTA method on defending against attacks on relational training data; it considers "random" corrupted values without assuming specific attack types. (F2) We focus on adversarial attacks on tabular data, where changes are limited to categorical or discrete attributes within predefined categories [17, 22, 52, 54]. Such attacks often target influential and imperceptible attributes, and incur high training loss (abnormal); these properties are supported by existing studies [17, 22, 41, 55, 56, 92, 107, 108]. Such attributes are often not manually checked (due to limited attention) but may be important to ML models [4]; perturbations to their values are often detectable by tools such as Rock [18] or a combination of Raha [77] and Baran [76], which employ attribute-level logic or association analyses. (F3) We have added experiments to evaluate DE4ML with more models, different loss optimization methods, and real-world attackers on tabular data to validate the assumptions.

More specifically, we have made the following revisions.

(1) We have clarified (pp. 3) that (a) attacks to image data and relational data differ in characteristics and models, and (b) the attacks on table data typically lead to inconsistency among attributes.

(2) We have clarified the following (pp. 5-6): (a) corrupted values in relational data can be detected by analyzing error signals of $\mathcal{M}$ with datamodels, and fixed by data cleaning tools that employ attribute association analyses, since real-life corruptions usually lead to inconsistencies between attributes [8, 52, 54]; (b) when the fraction of corrupted data is large, one may employ data cleaning tools in DE4ML that conduct few iterations with human for a good performance; (c) in Phase 1, DE4ML targets attacks in $\mathcal{D}_{\text{train}}$ that (i) are injected in categorical and discrete attributes and/or (ii) lead to large perturbations in numerical attributes (existing attackers on relational data increase/decrease numerical values by 70.2% on average; see Section 6); and (d) as a topic of future work, we will study the impact of small perturbations in continuous numerical attributes of relational data (especially on deep neural networks).

(3) We have clarified (pp. 7) that existing studies also characterize corrupted values as abnormal [107], influential [41, 55, 56, 92, 108] and imperceptible [17, 22], consistent with our setting.

(4) We have clarified (pp. 10) that the tested attackers (excluding RIA) are real-world attackers originally proposed in image domain and specially adapted to relational data for its characteristics (*e.g.,* feasibility) [52, 54]. Moreover, we have strengthened/added the following experiments (pp. 10-12): ① varying loss function optimizer in Figures 5(g) and 8 (in [6]); ② testing with more real-world attackers $\mathcal{A}_1$ on $\mathcal{D}_{\text{train}}$ in Phase 1 in Figure 5(c); ③ varying the attack ratio in Phase 1 in Figure 5(f); ④ varying $\#\mathcal{A}_2$ of attackers combined in

Phase 2 in Figure 5(m); and ⑤ testing with FT-Transformer, a table representation model, in Phase 1 (resp. 2) in Figures 5(b) (resp. 5(i)).

(5) We have revised our problem statements (pp. 5) and clarified that prediction data $\mathcal{D}_{\text{pred}}$ is *not* used in the training/fine-tuning of $\mathcal{M}$ (pp. 5 & 8; please also see our responses to R2[O4] and R2[O7]).

[R2O2] *It is not clear how the corrupted values are fixed. Data cleaning tools are usually designed for data corruption that happen randomly and not from an attacker. More importantly, they often need reliable information about the data distribution, which should be usually extracted from the dataset itself. It is not clear how to get such reliable information where a large fraction of the dataset is corrupted. I suggest that the authors clarify in details their assumptions on how the data cleaning methods fix corrupted data points*

[A] In addition to changes in response to R2[O1], we have further clarified the following: (a) the fraction of corrupted data is assumed to be less than 50%, consistent with previous studies [72, 93] (pp. 3); hence reliable information about the data distribution is within the reach; (b) there are effective data cleaning tools, and we selectively fix the most impactful (abnormal, influential, and imperceptible) errors only, instead of cleaning all errors detected (pp. 5, see the response to R2[O1]); and (c) even when the fraction of corrupted data were large (*e.g.,* >50%), these data cleaning tools could catch and fix such errors with few interactions with human. Indeed, Rock works well with a few human-corrected rules [18], and Raha [77] and Baran [76] only need 20 human-labeled tuples for a good accuracy.

Moreover, we have added new experiments (pp. 11) to evaluate the effectiveness of DE4ML *w.r.t.* different cell attack ratios. DE4ML performs well, *e.g.,* on average the rob of DE4ML is 0.75, 7.4% higher than the baselines. That is, DE4ML is able to defuse adversarial attacks injected by attackers with different attacker powers.

[R2O3] *The problem definition for data enhancement by adding more tuples in Section 2.2 considers the poisoned tuples to be from an unknown distribution. The solution in Section 5 makes some assumptions about properties of the poisoned tuple's properties that are not justified. I suggest that the authors explain why they believe those properties will defend against the attacks on prediction data that follow an unknown distribution.*

[A] Thanks! We have clarified (pp. 3, Section 2) that $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ are sampled from a fixed but unknown distribution $\mathcal{P}$, and $\mathcal{D}^{-}_{\text{pred}}$ poisoned by $\mathcal{A}_2$ follows a poisoned distribution $\mathcal{P}^{-}$.

We have also clarified (pp. 6, Section 3) that (a) we select a subset $\Delta\mathcal{D}_T$ from the set S_{at} of adversarial tuples from $\mathcal{A}_2$ to minimize their negative impact on the model's accuracy while maximizing the improvement in robustness, since a classifier $\mathcal{M}$ fine-tuned with $\mathcal{D}_{\text{train}} \oplus S_{\text{at}}$ can have bad accuracy on clean $\mathcal{D}_{\text{pred}}$ ($< B_{\text{inf}}$); intuitively, if S_{at} is sufficiently large, then model $\mathcal{M}$ will have enough samples from the $\mathcal{A}_2$ distribution, allowing it to achieve high accuracy on poisoned prediction data $\mathcal{D}^{-}_{\text{pred}}$ and thereby maximize robustness; and (b) we enforce the accuracy lower bound B_{inf} to prevent the ML algorithm from learning an undesired model.

[R2O4] *According to the problem definition for data enhancement by adding more tuples, the inference data is part of the input. Section 5 proposes some re-training methods to improve robustness of prediction given a fixed dataset used for prediction, therefore, it must be repeated for each inference. This does not seem to be scalable and significantly increases the time for inference and prediction. Fine-tuning of a model is usually done once for a new domain or task and not for every inference. If the inference data is not part of the input for this problem, I suggest that the authors clarify it from the start.*

[A] We have clarified the problem statement of DEAAP (pp. 5) by replacing poisoned (resp. clean) prediction data $\mathcal{D}^{-}_{\text{pred}}$ (resp. $\mathcal{D}_{\text{pred}}$) with distribution $\mathcal{P}^{-}$ (resp. $\mathcal{P}$), and clarified that $\mathcal{P}^{-}$ (resp. $\mathcal{P}$) is only used to generate the $\mathcal{D}^{-}_{\text{pred}}$ (resp. $\mathcal{D}_{\text{pred}}$) for evaluation. It is not used in the training/fine-tuning of $\mathcal{M}$ (pp. 8, Section 5).

We have also clarified (pp. 6) that we only need to fine-tune $\mathcal{M}$ once, *i.e.,* once $\mathcal{M}$ is fine-tuned on $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$, $\mathcal{M}$ can be applied to any subsequent prediction datasets from the same distribution.

We have tested the scalability of DE4ML on a new larger dataset CoverType (with 495,141 tuples) in Figure 5(t). We have verified that DE4ML with proposed optimization strategy for large datasets is efficient, *e.g.,* it takes less than 4 hours on CoverType for LR (pp. 12).

[R2O5] *I am not fully convinced that adding variations of the untainted data will always prevent the ML algorithm from learning an undesired model in all settings. For example, modifying a couple of support vectors might result in a different learned SVM. This might also depend on the skewness of the input data distribution. I suggest that the authors clarify the limitation of their assumptions.*

[A] As clarified in our responses to R2[O3], we enforce the accuracy lower bound B_{inf} to prevent $\mathcal{M}$ from learning an undesired model.

[R2O6] *In ML literature, formal definitions and complexity results regarding accuracy are defined and derived with the respect to some (test/prediction) sample distribution. The complexity results , e.g., in Section 2, seem be with respect to a fixed testing data set. This approach does not guarantee and hurt generalizability of the results. The same seems to be true for some of the algorithms. I suggest that the authors state and prove their formal results for an arbitrary distribution.*

[A] We have stated and proved the complexity results *w.r.t.* the sample distribution as suggested (pp. 4-5; see [6] for a full proof).

[R2O7] *It is not clear why the authors have defined robustness over the enhanced data. Data enhancement seems to be a solution to achieve robustness where robustness is the property of the underlying model (with or without enhancement). If the authors prefer to define robustness over the enhanced data, i.e., robustness is the property of ML over enhanced data, then it is strange that accuracy is defined over the original data (Equation 1). These definitions make them hard to compare as they are defined over different datasets. This confusion extends to the discussion of trade-off between robustness and accuracy where it seems that both are essentially the same thing over enhanced data. If this is the case, one can simply use only accuracy (before or after data enhancement) to analyze the proposed approach.*

[A] Thanks! We have clarified these in Section 2.1 (pp. 3-4).

(1) Accuracy (Equation 1) indicates the model $\mathcal{M}$'s performance on clean prediction data $\mathcal{D}_{\text{pred}}$, sampled from an unknown but fixed distribution $\mathcal{P}$, and the training data can either be the original or the enhanced version. Robustness is defined in two stages: (a)

Equation 2 measures $\mathcal{M}$'s robustness based on clean prediction data $\mathcal{D}_{\text{pred}}$ but with adversarial attacks in the training data $\mathcal{D}_{\text{train}}$; and (b) Equation 3 measures it on poisoned prediction data $\mathcal{D}^{-}_{\text{pred}}$, which follows the poisoned distribution $\mathcal{P}^{-}$, where $\mathcal{D}_{\text{train}}$ remains unattacked and enhanced by tuple perturbations $\Delta\mathcal{D}_T$.

We have clarified the roles of Equations 1, 2 and 3 (pp. 3-4), which are consistent with the previous work [67, 81, 96, 97, 101, 102].

(2) The notion of robustness is needed to measure how well the model maintains performance on prediction data subject to adversarial attacks. We have elaborated on how improving robustness through data enhancement (Equation 2) (Phase 2 in DE4ML) can lead to a decrease in accuracy on clean prediction data (Equation 1, pp. 4). We have also expanded on the implications of this trade-off with a simple example (pp. 4), *i.e.,* when accuracy approaches 1.

[R2O8] *It will help the readers to understand the context of the problem if the authors provide real-world examples of the attacks on tabular data in training and inference time. For instance, it is not quite clear how an attacker might access to the tabular data used during inference to modify the result of inference.*

[A] We have added real-world examples and statistics to illustrate the attacks on tabular data at training and prediction time (pp. 1; please also see our responses to R3[O1] for additional details).

[R2O9] *Generally, the formal framework at places seems quite crowded. It can be simplified, e.g., removing importance vector.*

[A] Thanks! We have simplified the framework in Figure 2 (pp. 6); however, the attribute importance vector is essential, since it relates to the imperceptible attribute and is used in the DET Algorithm in Figure 3 (pp. 8). We have also revised the paper by removing unnecessary parts (please see our responses to R3[O4]).

[R2M1] *in page one "bad prediction" => "inaccurate prediction".*

[A] Thanks! We have revised the sentence as suggested (pp. 1).

We thank the Referee for observing the novelty of this work and for providing constructive suggestions for us to improve the paper!

Response to the comments of Referee #3.

[R3O1 & Feedback (F)] *Artificial problem scenario. The paper motivates the problem with a credit risk prediction application and states that the used model might be vulnerable to changes in specific features. However, this incorrect input data is simply a problem of the surrounding application (case of credit fraud) not really the ML model. I would recommend to motivate the paper with a realistic scenario where readers intuitively understand the existence of a real problem.*

(F1) *Instead of making the credit risk model "robust" against wrong input data, there are dedicated fraud detection models deployed (as a prefilter) that are purposefully heavily-biased for high recall. Please add additional arguments to make a case for this point.*

[A] Thanks! We have improved Example 1 (pp. 1) by (a) clarifying that load/credit risk is a common case of adversarial attacks on relational data; some banks (*e.g.,* ING [5]) have deployed ML models for automatic loan processing without human involvement, (b) providing real-world examples/statistics to illustrate the threat/loss of attacks in loans (also see our responses to R2[O8]), (c) clarifying that the German dataset in Table 1 is sampled from a real-world benchmark of loan/credit risk, and (d) clarifying that existing filters does not suffice to make model robust for automatic business processing by flagging suspicious tuples. Moreover, we have revised Table 1 with the original schema/tuples sampled from German [1] and updated all running examples accordingly (pp. 1, 4, 6, 8-9).

(2) Regarding F1, we would like to clarify that DE4ML has two objectives (pp. 2): (a) Phase 1 aims to debug adversarial data during model training, which fraud detection models such as prefilters are not designed to handle; and (b) Phase 2 targets attacks at prediction time. Prefilter is an alternative approach. For example, Picket [72], the SOTA approach for defending against attacks on relational data, flags suspicious tuples for further checking (by human + ML). We find that DE4ML is 33.8% more accurate than Picket (pp. 12).

[R3O2] *Need for known attackers. The introduced enhancement algorithm need the specific attacker A as an input. Such a setting is infeasible in real world deployments and it remains unclear how quickly model performance deteriorates as we scale the number of known attacks we want to make the model robust against.*

[A] Thanks! we have added new tests on the effectiveness of DE4ML *w.r.t.* different number of attackers (Figure 5(m) & pp. 12). We varied the # of attackers (combined into one) from 1 to 4, where the total attack ratio remains the same; the rob of DE4ML is still as high as 0.75 against 4 attackers. This shows that DE4ML is able to defend attacks from multiple attackers in the prediction.

As surveyed in the latest study [52], there are a limited number (< 10) of effective attackers on relational data. Moreover, DE4ML can defend against new types of attackers as soon as they emerge.

[R3O3] *Partially incomplete experiments. Unfortunately, the experiments only cover binary classification on tabular data, and remain rather shallow. First, please add the baseline accuracy/robustness for different datasets. For example, the Adult results are more than 10% worse than without robustness. Second, it would be interesting to understand the impact of different attacks and their ratios in much more detail. Third, all the used datasets are tiny. Such characteristics raise concerns that the datasets where cherry-picked to make them easy to attack with a reasonable number of tuples.*

[A] Thanks! We have added new tests (pp. 10-12): (a) the accuracy in Phase 2 before/after tuple perturbations $\Delta\mathcal{D}_T$ for different datasets, attackers and models in Table 4, (b) the scalability of DE4ML on a new larger dataset CoverType in Figure 5(t), (c) the effectiveness of DE4ML *w.r.t.* (i) different cell attack ratios for DEAAT in Figure 5(f) (please see the response to R2[O2]), and (ii) different number of attackers for DEAAP in Figure 5(m) (the response to R3[O2]).

[R3O4] *Presentation: structures, bullet lists, heavy notations.*
[A] Thanks! We have improved the presentation as follows.

(1) We have revised the draft with a better structure, *e.g.,* subsubsection and itemization with indentation.

(2) We have removed notations: filtered set FS (Section 4), datamodels $\hat{\mathcal{M}}^{*}_{t}$ (Section 5) and the model parameter $\hat{\theta}^{*}_{t}$ (Section 5).

Many thanks for your support and constructive suggestions!

Data Enhancement for Binary Classification of Relational Data

Paper ID: 1146 → 1565

ABSTRACT

This paper studies enhancement of training data $\mathcal{D}$ to improve the robustness of machine learning (ML) classifiers $\mathcal{M}$ against adversarial attacks on relational data. 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 [89]. 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 [17, 42, 97]. The poisoned data attempts to get unwarranted advantageous decisions, and may even come from malicious attacks of fraudsters [22]. As shown in a recent survey [66], (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 [22] were mis-predicated by a real-world production system. In 2018, adversarial credit fraud incurred loss of $24.26 billion [15].

For example, load/credit risk is a common case of adversarial attacks on relational data [17, 22, 41, 51]; some banks [5] and insurance companies [11] have deployed ML models for automatic business processing without human involvement. Adversarial attacks can take place in both the training phase and prediction/deployment phase of ML models. (a) Training data may not be trusted since they may contain fraud records committed by internal employees [7], or be illegally accessed by external individual (*e.g.,* JPMorgan Chase [2] and Capital One [3]); ML models trained with poisoned training data may "discriminate against certain demographics during loan processing" [12]. (b) Users may falsify their information to fool the system and get unjustified loans, *e.g.,* "in Q2 2022, an estimated 0.76% of mortgage applications contained fraud" [8], and worse still, "over $200 billion of funding from the Small Business Administration's (SBA) pandemic assistance loan programs is estimated to have gone to fraudsters" [9].

Example 1: Consider possible attacks on German dataset in Table 1 sampled from a real-world benchmark of loan/credit risk [1]. German has attributes age, savingaccount, housing, creditamount, job, status and purpose; its classification label is risk. It is partitioned into training data (tuples $t_1, t_2, \ldots$) and data for prediction (tuples $t_a, t_b, \ldots$), where the numbers in parentheses are the real IDs (starting from 0) in German [1]. Each tuple in the training table represents a person who is assigned a credit by a bank, and is classified as good or bad, where values in creditamount (resp. status) are mapped to {small, medium, large} (resp. {short, long}) based on whether the credit amount is beyond 0, 1,551 and 3,334, respectively (resp. the tenure in current job exceeds 7 years). Here attributes colored in green (resp. gray) are very likely (resp. unlikely) checked by the bank worker; 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 [17]. Values colored in red are corrupted by attackers.

tid	age (A_1)	savingaccount (A_2)	housing (A_3)	creditamount (A_4)	job (A_5)	status (A_6)	purpose (A_7)	risk (Y)
t_1 (4)	53	little	free	large	skilled	long	car	bad
t_2 (49)	28	moderate	own	medium	skilled	short	TV	good
t_3 (63)	25	little	own	large	skilled	short	business	bad
t_4 (520)	44	little	free	large	unskilled	long	repairs	good
...	...	...	...	...	...	...	...	...
the training dataset								
t_a (11)	24	little	rent	large	skilled	short	TV	?
t_b (293)	56	little	free	large	skilled	long	car	?
t_c (645)	27	NaN	rent	large	skilled	short	TV	?
t_d (968)	29	NaN	rent	large	skilled	short	TV	?
...	...	...	...	...	...	...	...	...
the prediction dataset								

Table 1: The German Dataset

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

(1) ***Attacks in training data.*** In the training data, tuple t_1 (resp. t_4) is corrupted at attribute status and job *e.g.,* t_1[status] (resp. t_4[job]) is modified from short (resp. skilled) to long (resp. unskilled). Such perturbations may mislead ML models $\mathcal{M}$ to learn that a loan for experienced (resp. unskilled) worker has a bad (resp. good) risk, and make inaccurate prediction, *e.g.,* it may classify tuple t_d in the prediction table as bad risk. As will be seen in Section 6, fixing such corruptions (about 5%-8% of the data) improves the accuracy/robustness of classifiers by 6.7% on average.

(2) ***Attacks at prediction time.*** After $\mathcal{M}$ is trained, when applied to corrupted t_a and t_c in which attribute purpose is changed from business to TV, $\mathcal{M}$ may mis-classify them as good risk, as skilled young workers are well-paid and can 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 [41, 55, 56, 92, 108] or prediction time [17, 22, 32, 51, 78, 98]; 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 [21, 30, 43, 67, 81, 91, 96, 101, 102]. Against attacks in relational data, a method proposed in [72] suggests to (a) directly remove poisoned tuples from $\mathcal{D}_{\text{train}}$ to avoid misleading ML models $\mathcal{M}$ at training time, and (b) train a prefilter to flag suspicious tuples in $\mathcal{D}_{\text{pred}}$ that may misguide $\mathcal{M}$ at prediction time for further checking (by human + ML).

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 [97]. 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 a purpose. 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 status in tuple t_1 and job 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 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 (*i.e.,* prefilter) 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. To answer the questions, we propose a framework DE4ML (Data Enhancing for ML). When training $\mathcal{M}$ with dataset $\mathcal{D}$, DE4ML enhances $\mathcal{D}$ 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 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.

Data enhancing for ML is quite different from data cleaning for AI [13, 14, 34, 37, 49, 63–65, 72]. 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 $\mathcal{M}$ [48]. Since there is also inherent tradeoff between the robustness and accuracy, data enhancing cannot be replaced by data cleaning.

(2) ***A framework*** (Section 3). We propose a two-phase DE4ML that integrates ML training and data enhancing. In phase 1, DE4ML takes as input a classifier $\mathcal{M}$, a training dataset $\mathcal{D}_{\text{train}}$, and a data cleaning tool C. It selects (imperceptible) attributes to fix using C, and trains $\mathcal{M}$ with the enhanced $\mathcal{D}_{\text{train}}$. In phase 2, given the trained $\mathcal{M}$, the enhanced $\mathcal{D}_{\text{train}}$ and possible attackers $\mathcal{A}_2$, DE4ML generates adversarial tuples via $\mathcal{A}_2$, enhances $\mathcal{D}_{\text{train}}$ with the tuples, and fine-tunes $\mathcal{M}$. One may use DE4ML to enhance an arbitrary classifier $\mathcal{M}$ for defending against (a) injected attacks in $\mathcal{D}_{\text{train}}$, (b) attacks on data in $\mathcal{D}_{\text{pred}}$ 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 [53] 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.,* [17, 22, 41, 55, 56, 92, 98, 108], 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.,* skilled and unskilled) 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 [72] 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 0.71 on average, 30.7% 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 [6].

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 start with basic notations, followed by the definition of accuracy and robustness of classification models.

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. Compared to image data that is high-dimensional, homogeneous and interchangeable (for pixel), relational data is usually low-dimensional, heterogeneous (*e.g.,* categorical, discrete and numerical) and non-interchangeable (for attributes); moreover, there are complex inter-dependencies among attributes in relational data [17, 22, 52, 54].

ML classifiers. Following [61, 69], we consider *binary classifier* $\mathcal{M}(x) : \mathbb{R}^k \to \{0, 1\}$, *i.e.,* a function that maps an attribute (feature) vector $x = [A_1, \ldots, A_k] \in \mathbb{R}^k$ to a class label $y \in \{0, 1\}$. Assume a fixed but unknown distribution $\mathcal{P}$; we denote by $\mathcal{D}_{\text{train}}$ a dataset of schema $\mathcal{R}$ sampled from $\mathcal{P}$ for training $\mathcal{M}$, and by $\mathcal{D}_{\text{pred}}$ a dataset of $\mathcal{R}$ sampled from $\mathcal{P}$ for prediction by $\mathcal{M}$ after $\mathcal{M}$ is trained.

Attribute importance vector [17]. 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; different sectors adhere to different $\mathcal{V}$, *e.g.,* Wolters Kluwer [4] (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 schema $\mathcal{R}$ with the vector $\mathcal{V}$, an attribute $A_j \in \mathcal{R}$ is *imperceptible* if its importance score v_j is below a user-defined threshold $\mu \in (0, 1]$; one may combine multiple A_j with low v_j and treat the combo as a single attribute. Note that an imperceptible attribute may have a strong correlation with the label of its tuple since $\mathcal{V}$ is subjectively summed up by human.

Attacker model. Unlike attacks on image data via the l_p norm, adversarial attacks on relational data have unique characteristics, *e.g.,* (a) feasibility: attributes may have different semantics, types, distribution and predefined categories, and (b) immutability: some attributes are manually checked or even non-editable [17, 22, 52, 54].

Following [17, 22], we assume that an attacker tends to inject more noises in imperceptible attributes that are less likely to be inspected by experts but may be important to classifiers [17]. To corrupt a tuple $t \in \mathcal{D}$, an attacker interpolates attributes A_j following probability $1 - v_j$, *e.g.,* corrupting a selected cell $t[A_j]$ by replacing its original value c with value c' ($\neq c$). 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% [72, 93]. This is the attacker model commonly adopted on relations, *e.g.,* [17, 22, 92]. In particular, the attacks can lead to inconsistency among attributes [8, 52, 54].

We consider two types of attackers $\mathcal{A}_1$ and $\mathcal{A}_2$, where (a) $\mathcal{A}_1$ attacks the attributes and labels of training data $\mathcal{D}_{\text{train}}$ [41, 55, 56, 92, 108], and (b) $\mathcal{A}_2$ attacks the attributes of prediction data $\mathcal{D}_{\text{pred}}$ [17, 22, 32, 51, 78, 98]. We denote prediction data $\mathcal{D}_{\text{pred}}$ poisoned by $\mathcal{A}_2$ as $\mathcal{D}^-_{\text{pred}}$ that follows a poisoned distribution $\mathcal{P}^-$.

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 $\mathcal{M}$. We also study the tradeoff between the accuracy and robustness of model $\mathcal{M}$ when $\mathcal{D}_{\text{train}}$ is enhanced.

Corrupted $\mathcal{D}_{\text{train}}$ by $\mathcal{A}_1$. Attacker $\mathcal{A}_1$ aims to corrupt $\mathcal{D}_{\text{train}}$ to mislead the training of $\mathcal{M}$. To defuse attacks injected by $\mathcal{A}_1$, we enhance the training data $\mathcal{D}_{\text{train}}$ with *value perturbations* $\Delta\mathcal{D}_V$.

Accuracy. Denote by $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{P})$ *the accuracy of* $\mathcal{M}$ that is trained with (possibly enhanced) $\mathcal{D}_{\text{train}}$ and evaluated with unpoisoned dataset $\mathcal{D}_{\text{pred}}$ sampled from the distribution $\mathcal{P}$. Following [97], we measure $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{P})$ in terms of the *expected loss*:

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

where (x, y) is a tuple sampled from $\mathcal{P}$, $\mathcal{M}(x)$ 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. We use robustness to measure to which extent training of $\mathcal{M}$ is insensitive to the attacks in $\mathcal{D}_{\text{train}}$ injected by $\mathcal{A}_1$ after $\Delta\mathcal{D}_V$. Denote by $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P})$ *the robustness of* $\mathcal{M}$ *trained with enhanced* $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$ and evaluated on unattacked dataset $\mathcal{D}_{\text{pred}}$ sampled from $\mathcal{P}$. Following [97], we define

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

where (x, y), $\mathcal{M}(\cdot)$ and $L(\cdot, \cdot)$ are the same as in Equation 1. Here 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 dataset $\mathcal{D}_{\text{pred}}$ sampled from $\mathcal{P}$; a model $\mathcal{M}$ with the higher $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P})$ indicates that the training of $\mathcal{M}$ is more robust.

Accuracy vs. robustness. 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 $\Delta\mathcal{D}_V$, and train $\mathcal{M}$ with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$. 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 $\mathcal{P}$. 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 $\mathcal{P}$, improving $\mathcal{M}$'s accuracy 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.,* $\mathcal{D}^-_{\text{pred}}$) by $\mathcal{A}_2$. Assume that a model $\mathcal{M}$ is trained with $\mathcal{D}_{\text{train}}$, where $\mathcal{D}_{\text{train}}$ is either (1) originally clean, or (2) enhanced with $\Delta\mathcal{D}_V$ to neutralize $\mathcal{A}_1$ (*i.e.,* $\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$). Attacker $\mathcal{A}_2$ aims to poison the prediction dataset $\mathcal{D}_{\text{pred}}$, generate a poisoned version $\mathcal{D}^-_{\text{pred}}$ of $\mathcal{D}_{\text{pred}}$ following distribution $\mathcal{P}^-$, and make $\mathcal{M}$ misjudge on tuples in $\mathcal{D}^-_{\text{pred}}$. To defend against $\mathcal{A}_2$, we fine-tune $\mathcal{M}$ with $\mathcal{D}_{\text{train}}$ enhanced by *tuple perturbations* $\Delta\mathcal{D}_T$, such that $\mathcal{M}$ is able to resist unseen attacks.

Given possibly corrupted prediction data, following [67, 81, 96, 97, 101, 102], we use *accuracy* (resp. *robustness*) to measure how well

the $\mathcal{M}$ can correctly classify unpoisoned (resp. poisoned) tuples in $\mathcal{D}_{\text{pred}}$ (resp. $\mathcal{D}^-_{\text{pred}}$), which follows the distribution $\mathcal{P}$ (resp. $\mathcal{P}^-$).

Accuracy. By Equation 1, denote by $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P})$ the accuracy (measured via Equation 1) of $\mathcal{M}$ that is fine-tuned with $\mathcal{D}_{\text{train}}$ enhanced by tuple perturbations $\Delta\mathcal{D}_T$, and is evaluated on unpoisoned prediction data $\mathcal{D}_{\text{pred}}$ sampled from $\mathcal{P}$.

Robustness. Denote by $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-)$ *the robustness of* $\mathcal{M}$ *that is fine-tuned with enhanced* $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ and evaluated on attacked $\mathcal{D}^-_{\text{pred}}$ that follows the distribution $\mathcal{P}^-$. We define

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

where (x, y), $\mathcal{M}(\cdot)$ and $L(\cdot,\cdot)$ are the same as in Equation 1 but $\mathcal{P}^-$ is a poisoned version of $\mathcal{P}$ by $\mathcal{A}_2$. Intuitively, a large (resp. small) robustness indicates that a trained $\mathcal{M}$ is (resp. is not) able to resist the attacks in the $\mathcal{D}^-_{\text{pred}}$ sampled from the poisoned distribution $\mathcal{P}^-$.

Accuracy vs. robustness. To defend against attacks in $\mathcal{D}^-_{\text{pred}}$ unseen in $\mathcal{D}_{\text{train}}$, we find 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 an inherent trade-off between the accuracy and robustness of $\mathcal{M}$, since the former reflects $\mathcal{M}$'s performance on unpoisoned $\mathcal{D}_{\text{pred}}$ of $\mathcal{P}$, while the latter pertains to poisoned $\mathcal{D}_{\text{pred}}$ of $\mathcal{P}^-$. Fine-tuning $\mathcal{M}$ with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ may change the distribution of $\mathcal{M}$ learned towards the poisoned $\mathcal{P}^-$ (*i.e.,* improving robustness), and steer the distribution away from the unpoisoned $\mathcal{P}$ (*i.e.,* degrading accuracy), unless $\mathcal{P} = \mathcal{P}^-$.

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 sampled from unpoisoned distribution* $\mathcal{P}$ *(resp. poisoned* $\mathcal{P}^-$*) for accuracy (resp. robustness).*

That is, under any attacker with power $\lambda \in (0, 0.5)$*, any classifier* $\mathcal{M}$ *with accuracy* $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}) = 1 - \eta$ *for* $\eta \in [0, 1]$*, its robustness* $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-)$ *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 of the data from* $\mathcal{P}$*, 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 bounded. (3) The tradeoff is indicated by η, *e.g.,* with $\eta \to 0$, 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 data from unpoisoned distribution $\mathcal{P}$. For any ML classifier $\mathcal{M}$, its expected loss relies on A_1 but $\mathcal{M}$ also uses non-decisive attributes for higher accuracy [97]. An attacker with power λ corrupts the prediction data $\mathcal{D}_{\text{pred}}$ sampled from $\mathcal{P}$; it flips decisive attribute A_1 with probability v_d, and alters non-decisive attributes that impact the label. The attack leads to a poisoned data $\mathcal{D}^-_{\text{pred}}$ of distribution $\mathcal{P}^-$, making it hard for $\mathcal{M}$ to distinguish between data from $\mathcal{P}$ and $\mathcal{P}^-$. 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 the prediction data from $\mathcal{P}$, the robustness of $\mathcal{M}$ on the prediction data from $\mathcal{P}^-$ is necessarily upper bounded; hence the inherent tradeoff. □

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
$\mathcal{P}$, $\mathcal{P}^-$	a fixed but unknown distribution, a poisoned distribution
$\mathcal{D}_{\text{train}}$ (resp. $\mathcal{D}_{\text{pred}}$)	training (resp. prediction) dataset of $\mathcal{M}$ sampled from $\mathcal{P}$
α, $\hat{\theta}_t$	sampling ratio for training $\hat{\mathcal{M}}_t$, parameter vector of $\hat{\mathcal{M}}_t$
$\mathcal{A}$, $\mathcal{D}^-_{\text{pred}}$	attacker, poisoned $\mathcal{D}_{\text{pred}}$ sampled from $\mathcal{P}^-$
$\Delta\mathcal{D}_V$ (resp. $\Delta\mathcal{D}_T$), $\mathcal{C}$	value (resp. tuple) perturbations, data cleaning tool
$\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}$	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

Remark. (1) It is possible that $\mathcal{A}_1$ and $\mathcal{A}_2$ are the same attacker. We use $\mathcal{A}_1$ and $\mathcal{A}_2$ to distinguish attacks at model training and prediction stages, respectively, often with diverse patterns. (2) Both $\mathcal{A}_1$ and $\mathcal{A}_2$ may involve multiple attackers, which we model collectively as a compound attacker $\hat{\mathcal{A}}$, with each attacker characterized by a distinct power λ. (3) Theorem 1 establishes an upper bound on the trade-off between $\mathcal{M}$'s accuracy and robustness in the prediction stage. However, quantifying a lower bound remains open due to (a) the stochastic nature of adversarial attacks [43], making attacks unpredictable; (b) dynamic shifts in data distributions [97], without a stable pattern; and (c) the nonlinear and high-dimensional behavior of ML models [75], complicating quantification.

Example 2: Continuing with Example 1, consider training data $\mathcal{D}_{\text{train}}$ in Table 1 enhanced by value perturbations, *e.g.,* the status (resp. job) of t_1 (resp. t_4) is fixed from `long` (resp. `unskilled`) to `short` (resp. `skilled`). 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_a and t_c. 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 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_a' and t_c' with the same attributes as t_a and t_c, respectively, but with label bad, 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_c) on poisoned $\mathcal{D}^-_{\text{pred}}$. However, its accuracy on unpoisoned $\mathcal{D}_{\text{pred}}$ is degraded, *e.g.,* it misclassifies t_d after adding adversarial tuple t_a' and t_c' 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.

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 a classifier $\mathcal{M}$ with $\mathcal{D}_{\text{train}}$ poisoned by adversary $\mathcal{A}_1$, to maximize the robustness and accuracy of $\mathcal{M}$. We sample prediction data $\mathcal{D}_{\text{pred}}$ from the unpoisoned data distribution $\mathcal{P}$, where $\mathcal{D}_{\text{pred}}$ is disjoint from $\mathcal{D}_{\text{train}}$, and enhance $\mathcal{D}_{\text{train}}$ with value perturbations $\Delta\mathcal{D}_V$. The robustness and accuracy of $\mathcal{M}$ can be simultaneously improved in this case, *i.e.,* maximizing $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P})$ is equivalent

to maximizing $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P})$ (see Section 2.1).

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

- *Input*: A database schema $\mathcal{R}$, the training dataset $\mathcal{D}_{\text{train}}$ 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*: Simultaneously maximize $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P})$ and $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P})$, where $\mathcal{P}$ is the distribution *only* used to generate the prediction data for evaluating $\mathcal{M}$.

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 DEAAT decision problem decides, given $\mathcal{R}, \mathcal{D}_{\text{train}}, C, \mathcal{M}$ and a bound B_{train}, whether there exists a set $\Delta\mathcal{D}_V$ such that $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) \geq B_{\text{train}}$ and $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) \geq B_{\text{train}}$.

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 $\mathcal{D}_{\text{train}}$. An adversary $\mathcal{A}_2$ attacks the attributes of tuples in $\mathcal{D}_{\text{pred}}$ (which originally follows the distribution $\mathcal{P}$), and generates poisoned prediction data $\mathcal{D}^-_{\text{pred}}$ following a fixed but unknown distribution $\mathcal{P}^-$. 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 $\mathcal{M}$ also guards against attacks in $\mathcal{D}^-_{\text{pred}}$.

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

- *Input*: $\mathcal{R}$, $\mathcal{M}$, $\mathcal{D}_{\text{train}}$ as above, an adversarial perturbation function $\mathcal{A}_2$, and a bound B_{inf}.
- *Output*: A set $\Delta\mathcal{D}_T$ of tuple perturbations to $\mathcal{D}_{\text{train}}$.
- *Objective*: Maximize $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-)$, subject to $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}) \geq B_{\text{inf}}$, where $\mathcal{P}^-$ (resp. $\mathcal{P}$) is the distribution *only* used to generate the $\mathcal{A}_2$ attacked (resp. clean) prediction data for evaluating $\mathcal{M}$'s robustness (resp. accuracy).

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

Its decision version is to decide, given $\mathcal{R}, \mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{A}_2, B_{\text{inf}}$ and a bound R_{inf}, whether there exists a set $\Delta\mathcal{D}_T$ such that $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}) \geq B_{\text{inf}}$ and $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-) \geq R_{\text{inf}}$. Here B_{inf} denotes the accuracy lower bound, *i.e.,* $1-\eta$ in Theorem 1. When B_{inf} decreases with a larger η, R_{inf} increases, reflecting the accuracy-robustness trade-off outlined in Theorem 1.

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 [40]. Given a 3SAT instance ϕ, we construct a DEAAT instance with (a) schema $\mathcal{R}$, (b) unpoisoned (resp. $\mathcal{A}_1$ attacked) data distribution $\mathcal{P}$ (resp. $\mathcal{P}^-$), which produces prediction (resp. training) tuples $\mathcal{D}_{\text{pred}}$ (resp. $\mathcal{D}_{\text{train}}$) of $\mathcal{R}$ such that each of the 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 $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) \geq B_{\text{train}}$ iff ϕ is satisfiable.

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

3 A DATA ENHANCING FRAMEWORK

This section proposes DE4ML, a framework that integrates ML training and data enhancing. It enhances training set with small sets $\Delta\mathcal{D}_V$ and $\Delta\mathcal{D}_T$ of value and tuple perturbations, respectively, such that a classifier 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 a 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 [53]. As a surrogate model, we use datamodel to identify critical adversarial tuples/values.

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 $\mathcal{D}_{\text{valid}}$ while maintaining its accuracy on uncorrupted $\mathcal{D}_{\text{valid}}$; it then fine-tunes the $\mathcal{M}$ (trained in phase 1) with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ 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 for amending 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.,* $S_{\text{cov}} = \{t[A] \mid \forall_{t \in \mathcal{D}_{\text{train}}, A \in \mathcal{R}}\ t[A] \neq C(t, A)\}$. Note that corrupted values can be fixed by data cleaning tools (*e.g.,* Rock [18] or Raha [77]+Baran [76]) that employ attribute association analyses, since real-life corruptions usually lead to inconsistency between attributes [8, 52, 54]. Moreover, when the fraction of corrupted data is large (*e.g.,* > 50%), these data cleaning tools can still catch and fix such errors with interactions with human (*e.g.,* Raha and Baran with a few human-labeled tuples [76, 77]) for a good performance.

(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 with datamodel.

Example 3: Continuing with Example 1, assume $S_{\text{cov}} = \{t_1[\text{status}],$

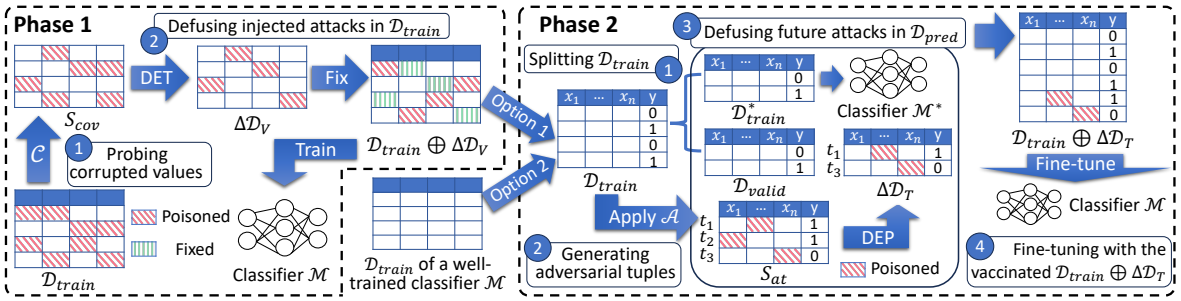


Figure 1: Overview of DE4ML

t_4[job], t_5[job]} by applying C on $\mathcal{D}_{\text{train}}$ (t_5 not shown in Figure 1). Assume $\mathcal{V}$ = [0.6, 0.8, 0.7, 0.9, 0.3, 0.1, 0.2, 1] for attributes A_1-Y in Figure 1. Given S_{cov} and $\mathcal{V}$, DE4ML generates value perturbations $\Delta\mathcal{D}_V = \{t_1[\text{status}], t_4[\text{job}]\}$ (see Example 5, Section 4). It enhances $\mathcal{D}_{\text{train}}$ by defusing the attacks in $\Delta\mathcal{D}_V$, *e.g.*, changing t_1[status] (resp. t_4[job]) from long (resp. unskilled) to short (resp. skilled). The fixes are identified by association analysis of label Y and attributes A_j with data cleaning tools, *e.g.*, Rock [18] unifies logic reasoning and ML predictions for association analysis and error detection/correction. 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 $7 \le \eta \le 9$ following the common practice in ML [57].

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

(3) ***Defusing future attacks in*** $\mathcal{D}_{\text{pred}}$. Then we identify a subset $\Delta\mathcal{D}_T$ of critical tuples in S_{at}, such that $\mathcal{M}$ trained with $\mathcal{D}^*_{\text{train}} \oplus \Delta\mathcal{D}_T$ can defend against attacks in $\mathcal{D}_{\text{valid}}$ without learning an undesired model by enforcing its accuracy on $\mathcal{D}_{\text{valid}}$ at least B_{inf}; it fine-tunes $\mathcal{M}$ with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$ for its robustness on corrupted $\mathcal{D}^-_{\text{pred}}$, and retains its accuracy on uncorrupted $\mathcal{D}_{\text{pred}}$. We will address this intractable problem (Theorem 2) in Section 5.

(4) ***Fine-tuning with the vaccinated data*** $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$. 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 $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$. Note that we only need to fine-tune $\mathcal{M}$ once, *i.e.*, once model $\mathcal{M}$ is fine-tuned on $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$, $\mathcal{M}$ can be applied to any subsequent prediction datasets from the same distribution.

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

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

Figure 2: The workflow of DE4ML

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 $\mathcal{D}_{\text{train}}$ with value and tuple perturbations, and a robust $\mathcal{M}$ trained and fine-tuned with the enhanced $\mathcal{D}_{\text{train}}$.

Remark. (1) Only phase 1 of DE4ML employs data cleaning tools C to fix adversarial errors in $\mathcal{D}_{\text{train}}$. (2) In phase 1, DE4ML targets attacks in $\mathcal{D}_{\text{train}}$ that (a) are injected into categorical and discrete attributes and/or (b) lead to large perturbations in numerical attributes, *e.g.*, existing attackers increase/decrease numerical values by 70.2% on average (see Section 6); we leave the study of impact/challenges introduced by small perturbations in continuous numerical attributes (especially on deep neural networks) as a topic of future work. (3) In phase 2 of DE4ML, when S_{at} is sufficiently large, $\mathcal{M}$ fine-tuned with $\mathcal{D}_{\text{train}} \oplus S_{\text{at}}$ is robust on poisoned $\mathcal{D}^-_{\text{pred}}$; however, this often reduces the accuracy on clean $\mathcal{D}_{\text{pred}}$. Thus, we select a subset $\Delta\mathcal{D}_T$ from S_{at} to minimize their impact on the model's accuracy while maximizing the improvement in robustness, *i.e.*, we enforce the bound B_{inf} to ensure $\mathcal{M}$ remain a desired model.

Complexity. DE4ML takes $O(c_C + c_{\text{DET}} + c_{\text{train}} + |\mathcal{D}_{\text{train}}| + c_{\text{at}} + c_{\text{DEP}} + c_{\text{ft}})$ time, where c_C is the cost of fixing $\mathcal{D}_{\text{train}}$ via C, c_{DET} is for identifying value perturbations $\Delta\mathcal{D}_V$ in S_{cov}, c_{train} is for training $\mathcal{M}$ with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$, $O(|\mathcal{D}_{\text{train}}|)$ is for splitting $\mathcal{D}_{\text{train}}$ into $\mathcal{D}^*_{\text{train}}$ and $\mathcal{D}_{\text{valid}}$, 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 $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T$. 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; given a corrupted dataset $\mathcal{D}_{\text{train}}$ of schema R, a classifier $\mathcal{M}$, a data cleaning tool C, and a set S_{cov} of

corrupted attribute values in $\mathcal{D}_{\text{train}}$ detected by C, DET identifies a set $\Delta\mathcal{D}_V$ of critical value perturbations in S_{cov} to $\mathcal{D}_{\text{train}}$ for training the $\mathcal{M}$. It aims to mitigate the impact of poisoned tuples in $\mathcal{D}_{\text{train}}$ on the training of $\mathcal{M}$. As shown in Theorem 2, the problem is NP-hard.

Our 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 in the first few epochs [107]; (b) influential, *i.e.,* the tuples with cv highly impact the prediction of $\mathcal{M}$ on other tuples at prediction time [41, 55, 56, 92, 108]; and (c) imperceptible, values cv appear in imperceptible attributes of tuples preventing from being detected by human [17, 22]. 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 [60].

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 [17, 22]. 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 $\mathcal{D}_{\text{train}}$ can affect $\mathcal{M}$ on criteria (a)-(b) for the others.

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

(1) ***Identifying abnormal imperceptible attributes*** (AIT). DET monitors the loss of each tuple $t \in \mathcal{D}_{\text{train}}$ in the first few (*e.g.,* 5) epochs based on the loss function of $\mathcal{M}$, and computes the average loss $\hat{l}(t)$ of each tuple $t \in \mathcal{D}_{\text{train}}$, 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-ranked (*e.g.,* 50%) tuples in the ranked $\mathcal{D}_{\text{train}}$. Denote by $\mathcal{D}_{\text{cot}}$ the set of corrupted tuples in $\mathcal{D}_{\text{train}}$ that have attribute values in S_{cov}. Given $\mathcal{D}_{\text{cot}}$, S_{cov} and the attribute important vector $\mathcal{V}$, DET finds a set $\hat{\mathcal{D}}_{\text{cot}}$ of corrupted tuples by selecting $t \in \mathcal{D}_{\text{cot}}$ with $t[A] \in S_{\text{cov}}$, 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.,* $\text{AIT} = \text{AT} \cap \hat{\mathcal{D}}_{\text{cot}}$.

(2) ***Identifying influential tuples*** (IT). We adopt datamodel [53] to capture the correlation among tuples in $\mathcal{D}_{\text{train}}$ 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 [60] and shapley value [28], for assessing the impact of other tuples on the prediction $\mathcal{M}(t)$ (see [53] for details). In order to train $\hat{\mathcal{M}}_t$ for each $t \in \mathcal{D}_{\text{train}}$, 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 $\mathcal{D}_{\text{train}}$, $\mathbb{1}_{S_i} \in \{0,1\}^{|\mathcal{D}_{\text{train}}|}$ is a $|\mathcal{D}_{\text{train}}|$-dimensional vector that indicates which tuples in $\mathcal{D}_{\text{train}}$ 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 $\mathcal{D}_{\text{train}}$, where the size $|S_{\text{dm}}|$ of S_{dm} is usually small (*e.g.,* 100 [53]), and each $S_i \in S_{\text{dm}}$ is obtained by randomly sampling α% of tuples in $\mathcal{D}_{\text{train}}$. DET then trains $|S_{\text{dm}}|$ models $\mathcal{M}_i$ $(1 \le i \le |S_{\text{dm}}|)$, each is trained with a set $S_i \in S_{\text{dm}}$. For each tuple $t \in \mathcal{D}_{\text{train}}$, DET predicts t with all trained $\mathcal{M}_i$ and obtains the training set $\mathcal{D}_t$ (with examples $(\mathbb{1}_{S_i}, \mathcal{M}_i(t))$) of the datamodel $\hat{\mathcal{M}}_t$. Finally, DET trains $\hat{\mathcal{M}}_t$ with $\mathcal{D}_t$ for each tuple $t \in \mathcal{D}_{\text{train}}$.

Using the trained datamodels, DET identifies influential tuples as follows. Denote by $\hat{\theta}_t$ the learned parameter vector of a datamodel $\hat{\mathcal{M}}_t$, where $|\hat{\theta}_t| = |\mathcal{D}_{\text{train}}|$. Here $\hat{\theta}_t$ contains $|\mathcal{D}_{\text{train}}|$ values $\hat{\theta}_t[i] \in [-1, 1]$ $(1 \le i \le |\mathcal{D}_{\text{train}}|)$, where each $\hat{\theta}_t[i]$ indicates to which extent a tuple (with id i) supports the prediction $\mathcal{M}(t)$; as observed in [53], 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 $t \in \mathcal{D}_{\text{train}}$, we obtain a set IT_t of influential tuples in $\mathcal{D}_{\text{train}} \setminus \{t\}$ 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 $\mathcal{D}_{\text{train}}$, *i.e.,* $\text{IT} = \cup_{t \in \mathcal{D}_{\text{train}}} \text{IT}_t$. Intuitively, IT contains all tuples in $\mathcal{D}_{\text{train}}$ that positively (resp. negatively) affect the prediction of $\mathcal{M}$.

(3) ***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 both AIT and IT.

(4) ***Conducting value perturbations***. Given the identified S_{cv} of critical values in this round, DET defuses $\mathcal{D}_{\text{train}}$ by fixing the values in S_{cv} via data cleaning tool C. The defused/enhanced $\mathcal{D}_{\text{train}}$ 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 $\mathcal{R}$, $\mathcal{V}$, $\mathcal{M}$, $\mathcal{D}_{\text{train}}$, S_{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 $t \in \mathcal{D}_{\text{train}}$ (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 $\mathcal{D}_{\text{train}}$ 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 11).

Remark. (1) Fixing an imperceptible attribute may alter the correlations between tuples in $\mathcal{D}_{\text{train}}$; 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) The size of $\Delta\mathcal{D}_V$ is relatively small, *e.g.,* for different datasets, $|\Delta\mathcal{D}_V|$ accounts for 5%-

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

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

Figure 3: Algorithm DET

8% of the total data (Section 6). This is because attackers inject few adversarial cells [55, 92] so that it is hard to detect them. (4) DET can improve the accuracy of $\mathcal{M}$ trained with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$ 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 helps align $\mathcal{M}$'s decision boundary with the true data distribution (*i.e.,* high accuracy); DET works better with more accurate C (see Section 6). (5) When $\mathcal{D}_{\text{train}}$ is too large for datamodels to fit in GPU memory, one can employ DET to obtain the set IT with following optimizations: (a) sample a subset S_{train} of $\mathcal{D}_{\text{train}}$ and train datamodels with S_{train} to obtain an initial IT of influential tuples in S_{train}; and (b) enlarge IT with tuples in $\mathcal{D}_{\text{train}} \setminus S_{\text{train}}$ if their nearest neighbors in S_{train} lie in the initial IT.

Example 5: Continuing with Example 3, DET takes the same S_{cov} and $\mathcal{V}$ as input as in Example 3. Initially, $\Delta\mathcal{D}_V{=}\phi$ and $S_{\text{cv}}{=}\phi$. In the first round, DET picks imperceptible attribute status (*i.e.,* A_6) since its importance score (*i.e.,* 0.1) is the lowest in $\mathcal{V}$. Suppose that we find AIT = $\{t_1\}$ and IT = $\{t_1, t_4, t_5\}$, where each $t' \in$ 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 $t \neq t'$ (here $\zeta = \frac{50\cdot(1-v_Y)}{\sum_{v_j\in\mathcal{V}} 1-v_j} = 0$). We obtain $S_{\text{cv}} = \{t_1[\text{status}]\}$ by intersecting AIT and IT. We defuse $\mathcal{D}_{\text{train}}$ with value perturbations in S_{cv}, update $S_{\text{cov}} = \{t_4[\text{job}], t_5[\text{job}]\}$ and $\Delta\mathcal{D}_V = \{t_1[\text{status}]\}$. DET processes these in the next round. This process proceeds until $S_{\text{cv}} = \phi$. As the result of phase 1, DE4ML returns $\Delta\mathcal{D}_V = \{t_1[\text{status}], t_4[\text{job}]\}$ and $\mathcal{M}$ trained with $\mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V$. □

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

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

$O(|\mathcal{D}_{\text{train}}| \cdot |\mathcal{R}|)$ is for conducting value perturbations, $O(|S_{\text{cov}}|)$ is for updating S_{cov} with S_{cv}, and $O(2 \cdot |\mathcal{D}_{\text{train}}|)$ 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 fast for most models; it terminates in less than $|\mathcal{R}|$ rounds.

5 ADDING ADVERSARIAL TUPLES

This section develops the algorithm of DE4ML for *data enhanced prediction*, denoted by DEP; given an unpoisoned/attack-defused training set $\mathcal{D}^*_{\text{train}}$ of schema $\mathcal{R}$ (provided by DE4ML in Section 3), an unpoisoned validation set $\mathcal{D}_{\text{valid}}$ 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}, DEP identifies a small set $\Delta\mathcal{D}_T$ of critical tuples in S_{at}, such that enhancing $\mathcal{D}^*_{\text{train}}$ with perturbations in $\Delta\mathcal{D}_T$ can maximumly improve the robustness of $\mathcal{M}$ on $\mathcal{D}^-_{\text{valid}}$ ($\mathcal{D}_{\text{valid}}$ poisoned by $\mathcal{A}$) while retaining its accuracy on unpoisoned $\mathcal{D}_{\text{valid}}$ above B_{inf}.

The identified $\Delta\mathcal{D}_T$ is finally used to fine-tune the model $\mathcal{M}$ trained in phase 1, such that it is robust against attacks in $\mathcal{D}^-_{\text{pred}}$ while retaining its accuracy on unpoisoned $\mathcal{D}_{\text{pred}}$. Here $\mathcal{D}^-_{\text{pred}}$ and $\mathcal{D}_{\text{pred}}$ are *only* used for evaluating $\mathcal{M}$, *not* in training or fine-tuning.

Naive approach. Given the set S_{at} of candidate tuple perturbations to $\mathcal{D}^*_{\text{train}}$, one may want to select an optimal $\Delta\mathcal{D}_T$ out of $2^{|S_{\text{at}}|}$ candidate sets to boost the performance of $\mathcal{M}$ on both poisoned $\mathcal{D}^-_{\text{valid}}$ and unpoisoned $\mathcal{D}_{\text{valid}}$. However, this approach is too costly to be practical since $\mathcal{M}$ needs to be retrained/fine-tuned $2^{|S_{\text{at}}|}$ 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 $\mathcal{D}^-_{\text{valid}}$; (b) harmless, *i.e.,* they do not largely degrade the accuracy of $\mathcal{M}$ on unpoisoned $\mathcal{D}_{\text{valid}}$; and (c) diversified, they are from different groups (*e.g.,* skilled and unskilled) and enable $\mathcal{M}$ to defend against attacks to different groups of $\mathcal{D}^-_{\text{valid}}$. DEP terminates when either no more tuples can further improve the robustness of $\mathcal{M}$ on $\mathcal{D}^-_{\text{valid}}$, or adding more tuples to $\mathcal{D}^*_{\text{train}}$ may degrade the accuracy of $\mathcal{M}$ on unpoisoned $\mathcal{D}_{\text{valid}}$ (*i.e.,* $< B_{\text{inf}}$). We iteratively search ct based on the distance between tuple embeddings learned via datamodel, since adding a tuple to $\mathcal{D}^*_{\text{train}}$ may alter the criteria (a)-(c) above for other tuples in S_{at}.

Denote by $\mathcal{M}^*$ the $\mathcal{M}$ (trained in phase 1) fine-tuned with $\mathcal{D}^*_{\text{train}} \oplus$ ct in each round. DEP iteratively conducts following steps.

(1) ***Attacking testing data with*** $\mathcal{A}$ (PVD). DEP generates a set PVD of few different poisoned validation datasets $\mathcal{D}^-_{\text{valid}}$, where

each $\mathcal{D}^-_{\text{valid}}$ is a poisoned instance of $\mathcal{D}_{\text{valid}}$ attacked with $\mathcal{A}$ for deceiving $\mathcal{M}$ and $|\mathcal{D}^-_{\text{valid}}| = |\mathcal{D}_{\text{valid}}|$. 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 $\mathcal{D}^-_{\text{pred}}$.

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

(3) ***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 $|S_{\text{dm}}|$ models $\mathcal{M}^*_i$, where each $S_i \in S_{\text{dm}}$ is obtained by randomly sampling α% of tuples in $\mathcal{D}^*_{\text{train}}$ for sampling ratio α. It then predicts the label of each tuple t in $\mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$ 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 $\mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$, and projects t into a $|\mathcal{D}^*_{\text{train}}|$-dimensional vector $\hat{\theta}_t$. We train datamodels for tuples in both $\mathcal{D}^*_{\text{train}}$ and S_{at} in order to map them into the same embedding space, so as to compute the distance between tuples in $\mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$ and thus select the critical ones in S_{at} that meet the criteria (a)-(c) above (see below).

(4) ***Selecting critical tuples*** (S_{ct}). Given the trained datamodel $\hat{\mathcal{M}}_t$ for each tuple t in $\mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$, the model parameter $\hat{\theta}_t$ can be treated as a $|\mathcal{D}^*_{\text{train}}|$-dimensional embedding of t [53]. DEP adopts k-means [74] to cluster all tuples in $\mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$ into k ($= |S_{\text{mt}}|$) groups g_i ($1 \le i \le |S_{\text{mt}}|$), where the initial centers are set as the embeddings $\hat{\theta}_t$ of misclassified tuples t in S_{mt}; here we set $k = |S_{\text{mt}}|$ 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 $g_i.\text{dr} = \frac{|S_{\text{mt}} \cap g_i| - |(S_{\text{at}} \setminus S_{\text{mt}}) \cap g_i|}{|g_i|}$ is the ratio of difference between misclassified and correctly classified tuples in $S_{\text{at}} \cap g_i$ 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 $S_{\text{mt}} \cap g_1$, where tuples $t \in S_{\text{ct}}$ with misclassified labels "y" have the top-$|S_{\text{ct}}|$ minimal distances (calculated based on embedding $\hat{\theta}_t$) to the tuples in $\mathcal{D}^*_{\text{train}}$ 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 $\mathcal{D}^*_{\text{train}}$ but are misclassified by $\mathcal{M}^*$ (*i.e.,* $\mathcal{M}^*$ has a low prediction confidence on the tuples in $\mathcal{D}^*_{\text{train}}$ 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 [6] for details).

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

Algorithm. Algorithm DEP is given in Figure 4. It takes as input $\mathcal{R}$, $\mathcal{M}$, $\mathcal{D}^*_{\text{train}}$, $\mathcal{D}_{\text{valid}}$, $\mathcal{A}$, 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.

DEP first initializes $\mathcal{M}^*$, $\Delta\mathcal{D}_T$, S_{ct}, acc_{pre}, rob_{pre}, acc_{cur} and rob_{cur} (line 1), where acc_{pre} (resp. rob_{pre}) records the value of acc_{cur} (resp. rob_{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 $\mathcal{D}^*_{\text{train}}$ and $\mathcal{A}$ (line 3), and gets S_{mt}, acc_{cur} and rob_{cur} by evaluating $\mathcal{M}^*$ with S_{at}, $\mathcal{D}_{\text{valid}}$ and PVD, respectively (line 4). It trains a set $S_{\hat{\mathcal{M}}^*}$ of datamodels $\hat{\mathcal{M}}_t$ for tuples t in $\mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$ (line 7), selects S_{ct} using the trained $\hat{\mathcal{M}}_t$ in $S_{\hat{\mathcal{M}}^*}$ (line 8), updates $\mathcal{D}^*_{\text{train}}$, S_{at}, $\Delta\mathcal{D}_T$, rob_{pre} and acc_{pre} (lines 9-10), and fine-tunes $\mathcal{M}^*$ 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).

Remark. (1) The size of $\Delta\mathcal{D}_T$ is relatively small under the accuracy constraint B_{inf}, since a larger B_{inf} leads to a smaller $|\Delta\mathcal{D}_T|$. (2) We adopt similar optimization for datamodel on large datasets as in DET (Section 4), to obtain the set S_{ct} in each round.

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 $S_{\text{at}} = \{t_1', t_3', t_4'\}$. Consider $B_{\text{inf}} = 0.7$. In the first round, suppose that we generate $S_{\text{mt}} = \{t_1', t_3'\}$ with $\text{acc}_{\text{cur}} = 0.71$ and $\text{rob}_{\text{cur}} = 0.52$, divide tuples in $\mathcal{D}^*_{\text{train}} \cup S_{\text{at}}$ into two groups g_1 and g_2, and generate $S_{\text{ct}} = \{t_3'\}$, where $g_1 = \{t_3, t_3', t_4', \dots\}$ with $g_1.\text{dr} = 0.7$ and $g_2 = \{t_1, t_1', t_4, \dots\}$ with $g_2.\text{dr} = 0.5$. We enhance $\mathcal{D}^*_{\text{train}}$ with S_{ct}, update $S_{\text{at}} = \{t_1', t_4'\}$, $\Delta\mathcal{D}_T = \{t_3'\}$, $\text{acc}_{\text{pre}} = 0.71$ and $\text{rob}_{\text{pre}} = 0.52$. This process proceeds until either $\text{acc}_{\text{cur}} < 0.7$ or $S_{\text{at}} = \phi$. After phase 2, DE4ML returns $\Delta\mathcal{D}_T = \{t_3'\}$ 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 $\text{acc}_{\text{cur}} < 0.7$ 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.,* $\ge B_{\text{inf}}$).

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	$\|\mathcal{D}\|$	$\|\mathcal{R}\|$	Cell Attack Ratio (%)		B_{inf} in Ph.2
			in Ph.1	in Ph.2	
German [100]	1,000	20	20.9	15.3	0.7
Mushroom [10]	8,124	22	24.9	17.6	0.85
Adult [13]	48,842	15	24.9	35.0	0.7
Marketing [72]	8,993	13	26.3	8.7	0.7
Bank [100]	49,732	16	19.9	24.7	0.61
CoverType [46]	495,141	55	25.1	36.7	0.7

Table 3: The tested datasets

varying datasets with LR	German	Bank	Adult		
	0.71/0.725	0.72/0.64	0.73/0.703		
varying attackers on Adult with LR	BIM	FGSM	DeepFool	Carlini	
	0.73/0.71	0.73/0.71	0.73/0.7	0.73/0.695	
varying ML models on Bank	LR	XGB	MLP	CatBoost	FTT
	0.72/0.64	0.67/0.66	0.65/0.631	0.657/0.661	0.614/0.614

Table 4: The accuracy (Ph.2) before/after perturbations $\Delta\mathcal{D}_T$

6.1 Experimental setting

Datasets. We tested six datasets as shown in Table 3 (see [6] for details). Each dataset $\mathcal{D}$ was randomly spit into disjoint $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ in 8 : 2 ratio. Following [72], 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 class imbalance on $\mathcal{M}$.

Classifiers. We tested logistic regression (LR) [85], MLPClassifier (MLP) [72]; random forest XGBoost [25] and CatBoost [31]; and a representative table representation model FT-Transformer [44].

Attackers. We considered the following attackers $\mathcal{A}$ with cell attack power in Table 3. (1) Random imperceptible attack (RIA), *e.g.,* given an attribute importance vector $\mathcal{V}$ for each dataset obtained via the feature importance function in skleran [85], RIA randomly injects attacks in attributes A_j with probability $(1-v_j)$ (Section 2.1). (2) BIM [52], DeepFool [52, 81], Carlini [52] and fast gradient sign method (FGSM) [43, 52]. These (excluding RIA) are real-world attackers originally proposed in image domain and specially adapted to relational data for its characteristics (*e.g.,* feasibility) [52, 54].

Data cleaning tools. We used (1) RB, an integrated approach that detects (resp. corrects) errors with Raha [77] (resp. Baran [76]); and (2) Rock, a system [18] for error detection and correction.

Loss optimizers. We tested the following optimizers in [84]: SGD, ASGD, Adam, AdamW, NAdam, Adagrad, RMSprop and Rprop.

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

(1) ***Defusing attacks in*** $\mathcal{D}_{\text{train}}$. (a) BaseVE, a "lower-bound" baseline with neither value perturbations nor tuple perturbations (*i.e.,* training directly on poisoned $\mathcal{D}_{\text{train}}$). (b) PicketTR [72], which defuses attacks in $\mathcal{D}_{\text{train}}$ by removing poisoned tuples (rather than fixing them). (c) $\mathcal{C}_{\text{full}}$, which cleans all errors in $\mathcal{D}_{\text{train}}$ with tool $\mathcal{C}$. (d) SAGA [90], a state-of-the-art approach that automatically generates the top-K most effective data cleaning pipelines for $\mathcal{M}$; we used the top-1 generated pipeline to fix $\mathcal{D}_{\text{train}}$ in the test.

(2) ***Defusing attacks in*** $\mathcal{D}_{\text{pred}}$. Prior methods flag suspicious tuples in $\mathcal{D}_{\text{pred}}$ that $\mathcal{M}$ may mispredict for further checking (by human + ML). 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 of DE4ML 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 [72], which trains a detector with self-generated samples to flag suspicious data in $\mathcal{D}_{\text{pred}}$ that may be mislabeled by $\mathcal{M}$. (d) AutoOD [20], which automatically detects outliers by integrating some base detectors. (e) ECOD [70], an unsupervised approach that detects outliers by employing empirical cumulative distribution functions.

Configurable parameters. By default, we set (1) α = 50% for the sampling ratio of data for datamodel; (2) for DEAAP, the accuracy bound $\mathcal{B}_{\text{inf}}$ as in Table 3, (3) we used LR as $\mathcal{M}$, SGD as loss function optimizer, RB as data cleaning tool $\mathcal{C}$, and RIA (resp. BIM) as attacker $\mathcal{A}$ for DEAAT (resp. DEAAP).

Configuration. We ran 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.

6.2 Experimental findings

6.2.1 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 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.

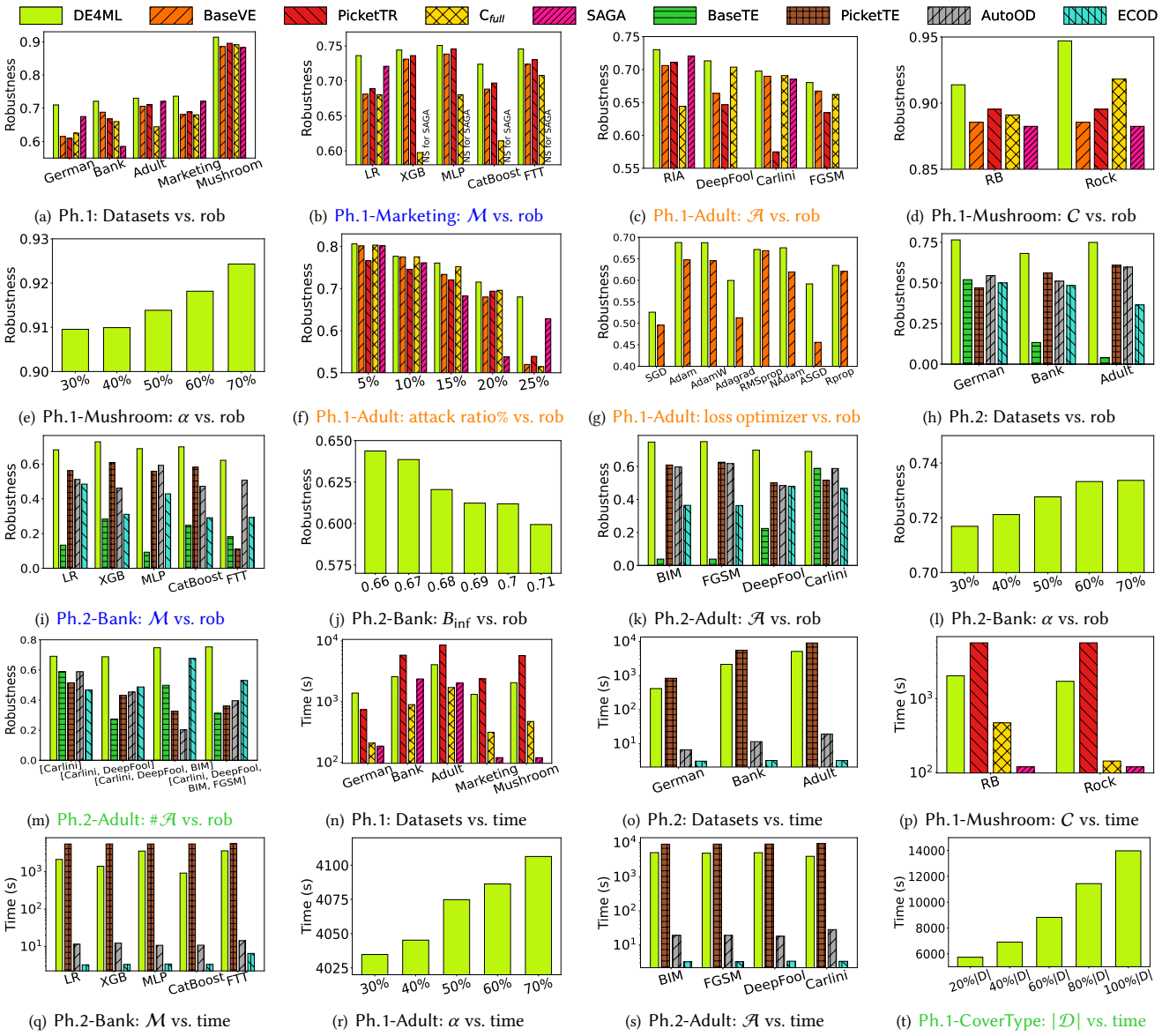
(2) On average, DE4ML beats BaseVE, $\mathcal{C}_{\text{full}}$ and SAGA by 6.7%, 8.9% and 6.3%, respectively. This validates the strategy of partial fixing: (a) DE4ML catches critical (5%$|\mathcal{D}_{\text{train}}|$-8%$|\mathcal{D}_{\text{train}}|$) poisoned cells whose fixing can improve $\mathcal{M}$ 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, CatBoost and FT-Transformer is 0.74 on average, 6.5% higher than baselines, up to 24.6%. This is because with datamodel, DE4ML well predicts the prediction of various $\mathcal{M}$.

Varying $\mathcal{A}$. We tested the impact of attackers $\mathcal{A}$ on DEAAT; they poison categorical and discrete attributes, and perturb numerical attributes by altering values by 70.2% on average. As shown in Figure 5(c), DE4ML adapts the training of $\mathcal{M}$ to different $\mathcal{A}$. Its rob is 0.73 (resp. 0.71, 0.7 and 0.68) under RIA (resp. DeepFool, Carlini and FGSM). Hence, DE4ML can identify/fix adversarial attacks injected by different $\mathcal{A}$; with data cleaning tools and datamodel, it can identify tuples with poisoned attributes that are abnormal, influential and imperceptible, *e.g.,* it improves the rob by 3.5% on average, by fixing adversarial values (about 7.37% of the data) alone.

Varying $\mathcal{C}$. We tested the impact of data cleaning tool $\mathcal{C}$. As shown in Figure 5(d), DE4ML with more accurate $\mathcal{C}$ can lead to a better $\mathcal{M}$, *e.g.,* its rob with Rock is 0.95 on Mushroom, 3.64% higher than with RB. That is, (1) DE4ML with accurate $\mathcal{C}$ can identify and fix corrupted values by employing attribute-level logic or association



(a) Ph.1: Datasets vs. rob
(b) Ph.1-Marketing: $\mathcal{M}$ vs. rob
(c) Ph.1-Adult: $\mathcal{A}$ vs. rob
(d) Ph.1-Mushroom: C vs. rob
(e) Ph.1-Mushroom: α vs. rob
(f) Ph.1-Adult: attack ratio% vs. rob
(g) Ph.1-Adult: loss optimizer vs. rob
(h) Ph.2: Datasets vs. rob
(i) Ph.2-Bank: $\mathcal{M}$ vs. rob
(j) Ph.2-Bank: B_{inf} vs. rob
(k) Ph.2-Adult: $\mathcal{A}$ vs. rob
(l) Ph.2-Bank: α vs. rob
(m) Ph.2-Adult: #$\mathcal{A}$ vs. rob
(n) Ph.1: Datasets vs. time
(o) Ph.2: Datasets vs. time
(p) Ph.1-Mushroom: C vs. time
(q) Ph.2-Bank: $\mathcal{M}$ vs. time
(r) Ph.1-Adult: α vs. time
(s) Ph.2-Adult: $\mathcal{A}$ vs. time
(t) Ph.1-CoverType: $|\mathcal{D}|$ vs. time

Figure 5: Performance evaluation

analyses; and (2) since a more accurate C introduces less errors, $\mathcal{M}$ can be better trained with an enhanced $\mathcal{D}_{\text{train}}$. 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 $\alpha \geq 50\%$.

Varying cell attack ratio%. We tested the effectiveness of DEAAT under different cell attack ratios by varying the ratio from 5% to 25%. As shown in Figure 5(f), DE4ML is sensitive to attack ratio, *e.g.,* the rob of DE4ML degrades from 0.81 to 0.68 when attack ratio increases from 5% to 25%. Nevertheless, DE4ML still performs the best compared with all baselines, *e.g.,* on average the rob of DE4ML is 0.75, 7.4% higher than the baselines. That is, DE4ML can defuse adversarial attacks injected by attackers with different attack power.

Varying loss function optimizer. As shown in Figure 5(g), DE4ML is effective with different optimizers, *e.g.,* the rob of DE4ML is 0.63 on average, 8.7% higher than the baselines, up to 29.8%. That is, DE4ML with data cleaning tools effectively defuses attacks in $\mathcal{D}_{\text{train}}$ by fixing the critical poisoned values of imperceptible attributes that impair the model performance on unseen data $\mathcal{D}_{\text{pred}}$.

We also studied the training loss optimization with different optimizers for $\mathcal{M}$ on relational data (see [6] for details). We find that there is no consistent pattern regarding whether (a) defusing attacks in $\mathcal{D}_{\text{train}}$ impacts the convergence rate of $\mathcal{M}$, and (b) the convergence rate of $\mathcal{M}$ is related to its effectiveness on clean $\mathcal{D}_{\text{pred}}$.

6.2.2 Exp-2 Effectiveness in defusing attacks in $\mathcal{D}_{\text{pred}}$. 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. Moreover, we provide

the accuracy before/after tuple perturbations $\Delta\mathcal{D}_T$ in Table 4.

Robustness vs. baselines. As shown in Figure 5(h), 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 $\mathcal{D}_{\text{train}}$.

We next tested the impact of various parameters on robustness.

Varying $\mathcal{M}$. We tested the effectiveness of DEAAP on various ML models. As shown in Figure 5(i), when defending against unseen attacks in prediction data, the rob of DE4ML is 0.68 on average, 54.9% higher than the baselines, up to 466%. 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(j), 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 $B_{\text{inf}} = 0.7$ works well.

Varying $\mathcal{A}$. We evaluated the impact of different attackers $\mathcal{A}$. As shown in Figure 5(k), 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 $\mathcal{D}^-_{\text{pred}}$ corrupted by $\mathcal{A}$, and correctly classify them.

Varying α. Varying α, Figure 5(l) 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 $\mathcal{D}^-_{\text{pred}}$ with more referenced features.

Varying #$\mathcal{A}$ combined. We varied the #$\mathcal{A}$ of attackers combined into one for DEAAP from 1 to 4, keeping the total attack ratio unchanged. As shown in Figure 5(m), when more attackers are combined, it may not lead to a low robustness, *e.g.,* the rob of DE4ML is 0.69 and 0.75 when #$\mathcal{A}$ is 1 and 4, respectively. This is because different attackers interfere with each other, such that the effectiveness of the attack may be affected. This shows that DE4ML is able to defend attacks from multiple attackers in the prediction.

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 $\mathcal{D}_{\text{train}}$ with more adversarial tuples can negatively affect the accuracy of $\mathcal{M}$ on unpoisoned $\mathcal{D}_{\text{pred}}$. 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. The three 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. (3) We also validated the effectiveness of DE4ML with the proposed sampling strategy for training datamodels with large $\mathcal{D}_{\text{train}}$ (not shown). On average, its rob for DEAAT is 0.757, only 0.7% smaller than without the sampling strategy, but faster. That is, DE4ML with the proposed optimization strategy also works nicely.

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

Comparison vs. baselines. As shown in Figures 5(n) and 5(o), (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 $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ 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 $\mathcal{C}_{\text{full}}$ and SAGA, since it needs to identify critical corrupted cells that are abnormal, influential and imperceptible, rather than only cleaning noises as in $\mathcal{C}_{\text{full}}$ and SAGA; in exchange, DE4ML beats $\mathcal{C}_{\text{full}}$ and SAGA by 8.9% and 6.3% in robustness, respectively. (c) DE4ML is efficient when $\mathcal{C}$, $\mathcal{A}$ and the training of datamodel are efficient, since they take up most of the time of DE4ML (see below).

Varying $\mathcal{C}$. We tested the impact of data cleaning tool $\mathcal{C}$ on DEAAT. As shown in Figure 5(p), it needs less time when $\mathcal{C}$ 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 $\mathcal{C}$ is faster. In contrast, PicketTR and SAGA do not take advantage of evolving $\mathcal{C}$.

Varying $\mathcal{M}$. As shown in Figure 5(q), 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(r), 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 $\mathcal{D}_{\text{train}}$ via $\mathcal{C}$, 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 $\mathcal{D}_{\text{train}}$.

Varying $\mathcal{A}$. We tested the impact of different attackers $\mathcal{A}$ on the efficiency of DEAAP. As shown in Figure 5(s), 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.

Varying $|\mathcal{D}|$. We tested the size of $\mathcal{D}$ on the efficiency of DEAAT, with Rock as the $\mathcal{C}$ for its efficiency. As shown in Figure 5(t), DE4ML runs slower when $|\mathcal{D}|$ is larger, since more tuples need to be processed. This said, DE4ML with the proposed optimization strategy for large datasets is efficient, *e.g.,* it takes less than 4 hours when $\mathcal{D}$ has 495,141 tuples for DEAAT; similarly for DEAAP.

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.

6.2.4 Parameter settings. We find $\alpha = 0.5$ suffices for a balance between efficiency and effectiveness. As data cleaning tools, Rock [18] and RB (Raha [77] and Baran [76]) work well. One may use DE4ML to defend against any attacker *w.r.t.* any accuracy bound B_{inf}.

6.2.5 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 0.68 on average for DEAAT and DEAAP, respectively. In the two phases together, it is 30.7% 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

The related work is categorized as follows.

Adversarial attacks. There has been a host of work on generating adversarial attacks. The notion of imperceptible attacks on relations was formalized in [17]. A surrogate model of [78] crafts adversarial examples for relations with heterogeneous attributes. Subpopulation attacks of [55] degrade the accuracy of ML models on a specific group/subpopulation of tuples by adding adversarial tuples. Gradient-based attacks were investigated in [22], to impact the fairness of ML models. Two types of such attacks were proposed in [79]. Triggers on time series were studied in [29] to hack deep neural networks (DNNs) with backdoor attacks. Attacks on learned indices were studied in [62]. Adversaries on relations are surveyed in [52], on texts in [99], and in [38, 94] 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 [101], worst case perturbations [91], acceleration of model verification [96, 102], specification of a verifiable robust model [30] and the trade-off between accuracy and robustness [97] (see [27] 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 [39]. ARDA [26] adopts schema enrichment to improve the resistance of $\mathcal{M}$ to noises in tabular data. TargAD [73] extends outlier detection to identify specific types of poisoned tuples. MWOC [47] employs kernel-based test and label enrichment to detect adversarial tuples. TOAO [87] applies statistical test *w.r.t.* log-odds to discern adversarial data. Robust training was also studied for graph models [56, 68].

Closer to this work is Picket [72], 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 (prefilter).

There has also been a large body of work on fairness and biased data [16, 19, 23, 24, 33, 41, 45, 50, 58, 69, 71, 82, 86, 95, 103–105]. In particular, the unavoidable trade-off between the accuracy and fairness of ML models is studied in [36, 59, 61, 80, 88, 106].

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 [97]. (2) We formulate two data enhancing problems to defend against adversarial attacks, and prove their intractability. (3) As opposed to Picket [72], 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 [72]. (4) We enhance $\mathcal{D}_{\text{train}}$ with value and tuple perturbations, rather than schema enrichment as in ARDA [26]. (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 [39], (b) extending prior outlier detection strategies as in TargAD [73], or (c) using statistical tests as in MWOC [47] and TOAO [87].

Data cleaning for AI. There has also been work on data cleaning for ML/AutoML, mainly to improve accuracy. ActiveClean [64] employs data cleaning via stochastic gradient descent. Coco [34, 35] detects unfriendly tuples for ML models using arithmetic constraints on numerical attributes. Amalur [49] improves the accuracy via data integration. BoostClean [63] boosts the accuracy by selecting suitable data cleaning tools from a given pool. AutoSklearn [37] improves AutoML approaches by referencing previous performance on similar datasets. AlphaClean [65] tunes parameters for data cleaning pipelines to improve the accuracy. AutoCure [14] cleans training data for ML pipelines via error detection and data augmentation. AutoClean [83] extends AutoSklearn with advanced data imputation and outlier detection. SAGA [90] identifies the top-k promising cleaning pipelines for ML models. Moreover, REIN [13] 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] 1994. German Credit Risk. https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data.

[2] 2014. JPMorgan Chase data breach. https://en.wikipedia.org/wiki/2014_JPMorgan_Chase_data_breach.

[3] 2019. Information on the Capital One cyber incident. https://www.capitalone.com/digital/facts2019/.

[4] 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.

[5] 2023. How artificial intelligence is influencing the banking of the future. https://www.ingwb.com/en/insights/innovation/orange-blog/how-artificial-intelligence-is-influencing-the-banking-of-the-future.

[6] 2024. Code, datasets and full version. https://anonymous.4open.science/r/DE4ML-SIGMOD25-ECFE.

[7] 2024. Fraude Trend report Benelux 2024. https://www.allianz-trade.com/en_BE/news/fraud/fraud-survey.html?.

[8] 2024. Guard Your Business Against Loan Fraud: Types, Impact, and Prevention. https://www.getfocal.ai/knowledgebase/loan-fraud.

[9] 2024. Loan fraud: Statistics, types, and warning signs. https://lifelock.norton.com/learn/fraud/loan-fraud?.

[10] 2024. Mushroom. https://archive.ics.uci.edu/dataset/73/mushroom.

[11] 2024. Root Insurance. https://www.joinroot.com/.

[12] 2024. What is Data Poisoning? Types and Best Practices. https://www.sentinelone.com/cybersecurity-101/cybersecurity/data-poisoning/.

[13] 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.

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

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

[16] 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.

[17] 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).

[18] 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)*.

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

[20] 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.

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

[22] 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.

[23] 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

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

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

[26] 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).

[27] 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).

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

[29] 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.

[30] 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).

[31] 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).

[32] 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.

[33] 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

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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).

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

[41] 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.

[42] 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.

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

[44] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *NeurIPS* 34 (2021), 18932–18943.

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

[46] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data?. In *Advances in Neural Information Processing Systems*.

[47] 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).

[48] 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.

[49] 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.

[50] 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

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

[52] 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).

[53] 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).

[54] Yael Itzhakev, Amit Giloni, Yuval Elovici, and Asaf Shabtai. 2024. Addressing Key Challenges of Adversarial Attacks and Defenses in the Tabular Domain: A Methodological Framework for Coherence and Consistency. *arXiv preprint arXiv:2412.07326* (2024).

[55] 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

[56] 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).

[57] V Roshan Joseph and Akhil Vakayil. 2022. SPlit: An optimal method for data

splitting. *Technometrics* (2022).
[58] Faisal Kamiran and Toon Calders. 2011. Data preprocessing techniques for classification without discrimination. *Knowl. Inf. Syst.* 33, 1 (2011), 1–33.
[59] 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
[60] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*. PMLR, 1885–1894.
[61] Nikola Konstantinov and Christoph H Lampert. 2022. Fairness-Aware PAC Learning from Corrupted Data. *The Journal of Machine Learning Research* (2022).
[62] 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.
[63] 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).
[64] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Goldberg. 2016. ActiveClean: Interactive Data Cleaning For Statistical Modeling. *PVLDB* 9, 12 (2016), 948–959.
[65] Sanjay Krishnan and Eugene Wu. 2019. AlphaClean: Automatic Generation of Data Cleaning Pipelines. *CoRR* abs/1904.11827 (2019). http://arxiv.org/abs/1904.11827
[66] 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.
[67] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).
[68] 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).
[69] 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.
[70] 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.
[71] Yin Lin, Samika Gupta, and HV Jagadish. 2024. Mitigating subgroup unfairness in machine learning classifiers: A data-driven approach. In *ICDE*.
[72] 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
[73] 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.
[74] J Macqueen. 1967. Some methods for classification and analysis of multivariate observations. In *University of California Press*.
[75] Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *stat* 1050, 9 (2017).
[76] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *PVLDB* 13, 12 (2020), 1948–1961.
[77] 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.
[78] 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).
[79] Ninareh Mehrabi, Muhammad Naveed, Fred Morstatter, and Aram Galstyan. 2021. Exacerbating Algorithmic Bias through Fairness Attacks. In *AAAI*. AAAI Press, 8930–8938.
[80] 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.
[81] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *ICCV*.
[82] 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.
[83] 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).
[84] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *NIPS* (2019).
[85] 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* (2011).
[86] Romila Pradhan, Jiongli Zhu, Boris Glavic, and Babak Salimi. 2022. Interpretable Data-Based Explanations for Fairness Debugging. In *SIGMOD*. 247–261.
[87] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. 2019. The odds are odd: A statistical test for detecting adversarial examples. In *ICML*. PMLR, 5498–5507.
[88] Ricardo Salazar, Felix Neutatz, and Ziawasch Abedjan. 2021. Automated Feature Engineering for Algorithmic Fairness. *PVLDB* 14, 9 (2021), 1694–1702.
[89] Suproteem K Sarkar, Kojin Oshiba, Daniel Giebisch, and Yaron Singer. 2018. Robust classification of financial risk. *arXiv preprint arXiv:1811.11079* (2018).
[90] 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).
[91] 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.
[92] 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.
[93] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2020. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks* 129 (2020), 249–260.
[94] 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.
[95] Ki Hyun Tae and Steven Euijong Whang. 2021. Slice tuner: A selective data acquisition framework for accurate and fair machine learning models. In *SIGMOD*.
[96] 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.
[97] 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.
[98] 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.
[99] 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.
[100] 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).
[101] 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.
[102] 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.
[103] 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
[104] 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.
[105] 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.
[106] Jie M. Zhang and Mark Harman. 2021. "Ignorance and Prejudice" in Software Fairness. In *ICSE*. IEEE, 1436–1447.
[107] 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)*.
[108] 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 sampled from unpoisoned distribution $\mathcal{P}$ (resp. poisoned $\mathcal{P}^-$) for accuracy (resp. robustness).*

That is, under any attacker with power $\lambda \in (0, 0.5)$, any classifier $\mathcal{M}$ with accuracy $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}) = 1 - \eta$ for $\eta \in [0, 1]$, its robustness $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-)$ 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 of the data from $\mathcal{P}$, 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 more 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 are sampled from a distribution $\mathcal{P}$, where each tuple (x, y) of a database schema $\mathcal{R}$ is drawn randomly, and $y \in \{0, 1\}$. Assume *w.l.o.g.* that $\mathcal{P}$ includes a decisive attribute A_1 with a stronger correlation with y than attributes $A_2, \ldots, A_k$. 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 the unpoisoned prediction data distribution $\mathcal{P}$, the values x_1 of A_1 follow the probability:

$$\Pr_{(x_1,y)\sim\mathcal{P}}[x_1 = y \mid y] = p, \qquad \Pr_{(x_1,y)\sim\mathcal{P}}[x_1 \neq y \mid y] = 1 - p,$$

where $p \geq 0.5$ and $x \in \{0, 1\}$.

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 conditional 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 [97], 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 defined based on the data distribution $\mathcal{P}$ as the probability $\Pr_{(x,y)\sim\mathcal{P}}[M(x) = y]$, *i.e.,* the likelihood 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 perturbs the data distribution $\mathcal{P}$, by corrupting 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, resulting in a poisoned distribution $\mathcal{P}^-$. The prediction dataset $\mathcal{D}^-_{\text{pred}}$, which follows the distribution $\mathcal{P}^-$, contains $\lambda|\mathcal{D}^-_{\text{pred}}|$ of poisoned tuples, denoted as $\mathcal{D}_{\text{poi}}$, and $(1 - \lambda)|\mathcal{D}^-_{\text{pred}}|$ 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. $t[y] = 1$), attacker $\mathcal{A}$ 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 attack 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, the values x'_1 of A_1 follow the probability:

$$\Pr_{(x'_1,y)\sim\mathcal{P}^-}[x'_1 = y \mid y] = p \cdot (1 - v_d) + (1 - p) \cdot v_d,$$
$$\Pr_{(x'_1,y)\sim\mathcal{P}^-}[x'_1 \neq y \mid y] = p \cdot v_d + (1 - p) \cdot (1 - v_d).$$

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, denote $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P})$ by acc. Then we can deduce the following:

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

i.e., the conditional probability that $\mathcal{M}$ correctly predicts the label of the given prediction tuples, sampled from the unpoisoned distribution $\mathcal{P}$, based on the conditions on the values x_1 (resp. $x_2, \ldots, x_k$) of deceive (resp. non-deceive) attribute(s).

Robustness of $\mathcal{M}$: Denote $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-)$ by rob, where $\text{rob} = \Pr_{(x,y)\sim\mathcal{P}^-}[\mathcal{M}(x) = y]$. Since $\mathcal{D}^-_{\text{pred}}$ denotes sampled prediction tuples from the poisoned distribution $\mathcal{P}^-$, and $\mathcal{D}^-_{\text{pred}} = \mathcal{D}_{\text{poi}} \bigcup \mathcal{D}_{\text{upoi}}$ 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 distributions for both decisive and non-decisive attributes. Putting these together, we can calculate acc_{poi} as follows:

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

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

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

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:

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

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

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

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. [40]). Given a Boolean formula in the 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 $t \in \mathcal{D}_{\text{train}}$ with $t[\text{id}] = 1$, its attribute values $t[A_i]$ ($i \in [1,n]$), and label $t[\text{label}]$ are generated from the attacked distribution $\mathcal{P}^-$, defined as:

$$\Pr_{\mathcal{P}^-}[t[A_i]=0] = 1, \qquad \Pr_{\mathcal{P}^-}[t[\text{label}]=1] = 1,$$

where $\mathcal{P}^-$ is a degenerate distribution that always generates tuples with fixed attribute values. 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 $s[\text{id}] = m - e$, where e is the number of clauses satisfied in ϕ when all variables are set to false. Its attribute values $s[A_i]$ ($i \in [1,n]$) and label $s[\text{label}]$ are generated from the unpoisoned distribution $\mathcal{P}$, defined as:

$$\Pr_{\mathcal{P}}[s[A_i]=0] = 1, \qquad \Pr_{\mathcal{P}}[s[\text{label}]=0] = 1,$$

i.e., the attacker $\mathcal{A}$ leads to a poisoned distribution $\mathcal{P}^-$ by flipping the label value in $\mathcal{P}$. We assume that *w.l.o.g.* $e \leq m-1$ since 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, initially, $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{P}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}}, \mathcal{P}) = 0$.

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

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 $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) \geq B_{\text{train}}$ if and only if the 3SAT instance ϕ is satisfiable.

$\Rightarrow$ First, assume that there exists a set $\Delta\mathcal{D}_V$ such that after the value perturbations on $\mathcal{D}_{\text{train}}$, $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) = 1$. More specifically, the perturbations

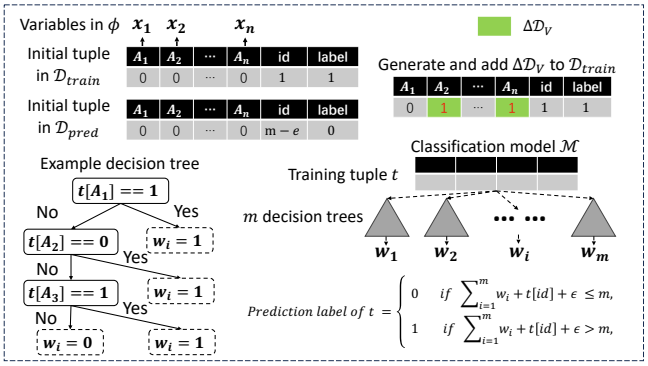


Figure 6: Reduction example for DEAAT

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, $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) = 1$ implies that the expected loss of the trained $\mathcal{M}$ on $\mathcal{D}_{\text{pred}}$ sampled from $\mathcal{P}$ is 0, *i.e.,* the model $\mathcal{M}$ reaches its maximum possible accuracy on $\mathcal{P}$. 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 $\mathcal{D}_{\text{train}}$, sampled from $\mathcal{P}^-$, that appear in $\Delta\mathcal{D}_V$ ensures $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) = 1$. 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 $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) = \text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_V, \mathcal{P}) < 1$; then it implies that the tuple s in $\mathcal{D}_{\text{pred}}$ sampled from $\mathcal{P}$ 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.

DEAAP. We prove its NP-hardness by reducing 3SAT to a special case of DEAAP where $B_{\text{inf}} = 0$, *i.e.,* the problem is already intractable even without considering the accuracy of $\mathcal{M}$ on the unpoisoned prediction data $\mathcal{D}_{\text{pred}}$ sampled from $\mathcal{P}$.

(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 $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{pred}}$ of schema $\mathcal{R}$ above, each consisting of one tuple. For the tuple $t \in \mathcal{D}_{\text{train}}$, we set $t[\text{id}] = m - e + 1$, where e is the number of satisfied clauses in the 3SAT instance ϕ when all variables are set to false. Its attribute values $t[A_i]$ ($i \in [1, n]$) and label $t[\text{label}]$ are generated from the original distribution $\mathcal{P}$. The distribution $\mathcal{P}$ is defined as:

$$\Pr_{\mathcal{P}}[t[A_i] = 0] = 1, \qquad \Pr_{\mathcal{P}}[t[\text{label}] = 1] = 1,$$

Similarly, $\mathcal{P}$ is a degenerate distribution that always generates tuples with fixed attribute values, and we use t's attributes $A_1 \sim A_n$ to encode n variables of ϕ. For the tuple $c \in \mathcal{D}_{\text{pred}}$, we set $c[\text{id}] = m - e$. Its attributes and label are also generated from $\mathcal{P}$, and thus, $c[A_i] = 0$ ($i \in [1, n]$) and $c[\text{label}] = 1$.

(3) Attacker $\mathcal{A}$ and $\mathcal{D}^-_{\text{pred}}$: We develop an attacker $\mathcal{A}$ which perturbs the dataset $\mathcal{D}_{\text{pred}}$ and generates the attacked dataset $\mathcal{D}^-_{\text{pred}}$. For each tuple $c \in \mathcal{D}_{\text{pred}}$, (a) if $c[\text{id}] = m - e$, $\mathcal{A}$ flips the tuple label, and (b) if $c[\text{id}] \neq m - e$, $\mathcal{A}$ flips each attribute $c[A_i]$ with a probability of 0.5, while keeping the label unchanged. Consequently, $\mathcal{D}^-_{\text{pred}}$ follows the attacked distribution $\mathcal{P}^-$, defined as:

$$\mathcal{P}^- = \begin{cases} \Pr[t[A_i] = 0] = 1, \Pr[t[\text{label}] = 0] = 1, \text{ if } t[\text{id}] = m - e, \\ \Pr[t[A_i] = 0] = \frac{1}{2}, \Pr[t[\text{label}] = 1] = 1, \text{ if } t[\text{id}] \neq m - e. \end{cases}$$

Thus, we construct the attacked $\mathcal{D}_{\text{pred}}$ (*i.e.,* $\mathcal{D}^-_{\text{pred}}$) with one tuple s by setting $s[\text{id}] = m - e$, $s[A_i] = 0$ ($i \in [1, n]$), and $s[\text{label}] = 0$.

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 $f(t) \le m$ and 1 otherwise, where ϵ is a trainable parameter.

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

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

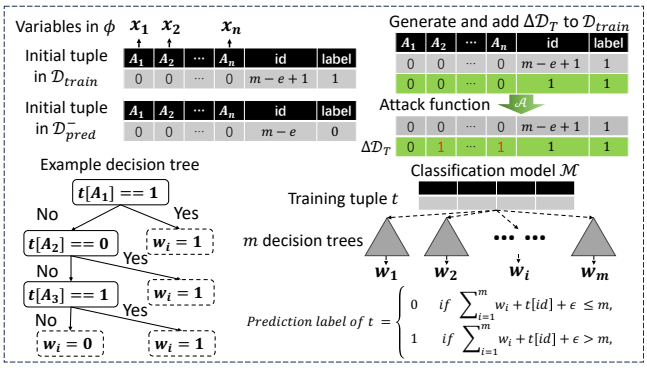
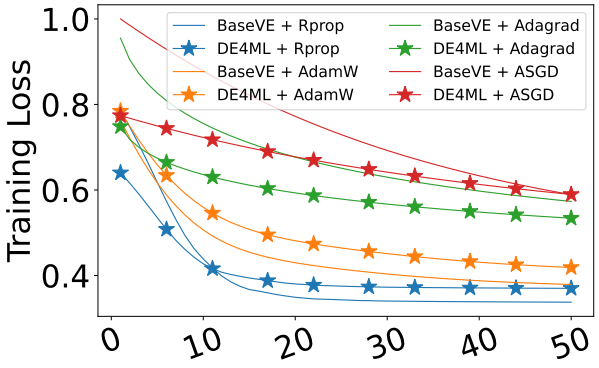
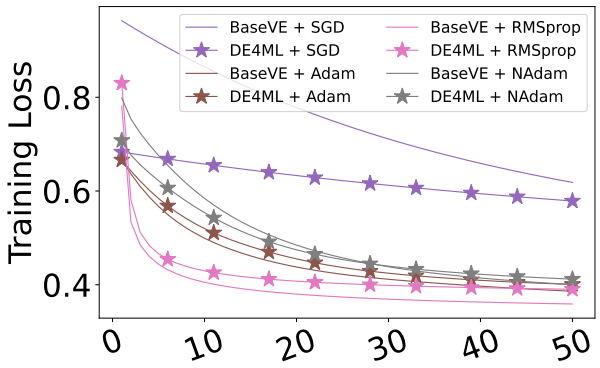


Figure 7: Problem DEAAP **reduction example**



(a) training epoch



(b) training epoch

Figure 8: Ph.1-Adult**: loss optimizer vs. training loss**

$\Rightarrow$ Assume that there exists a set $\Delta\mathcal{D}_T$ such that $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-) \geq 1$. 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, $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-) \geq 1$ 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 $t_1[\text{id}] = m - e + 1$, $t_2[\text{id}] = 1$, $t_1[\text{label}] = t_2[\text{label}] = 1$, $f(t_1) = m + 1 + \epsilon$ 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 $\{t_2\} = \Delta\mathcal{D}_T$.

More specifically, 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 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 $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-) \geq 1$ subject to $\text{acc}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}) \geq 0$. 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 $\text{rob}(\mathcal{M}, \mathcal{D}_{\text{train}} \oplus \Delta\mathcal{D}_T, \mathcal{P}^-) < 1$, *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 $|S_{\text{ct}}|$.

D DATASETS

Datasets. We tested six datasets. (1) German, a dataset of bank accounts; the ML classification is to determine the credit risk of a holder. (2) Mushroom, 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 mar-

keting campaigns; is to predict whether a client will subscribe to term deposits. (6) CoverType [46], a dataset of trees from Roosevelt National Forest in Colorado; it is to predict the types of trees.

E MORE RESULTS

Training loss optimization. We studied the training loss optimization with different loss optimizers for DEAAT, by observing the change of training loss in 50 epoches. As shown in Figure 8, when the training epoch increases, the training loss of $\mathcal{M}$ with $\mathcal{D}_{\text{train}}$ decreases for both poisoned $\mathcal{D}_{\text{train}}$ and defused $\mathcal{D}_{\text{train}}$. However, there is no unified trend regarding whether defusing attacks in $\mathcal{D}_{\text{train}}$ helps $\mathcal{M}$ converge faster, *e.g.*, fixing poisoned $\mathcal{D}_{\text{train}}$ via DE4ML makes LR converge faster (resp. slower) with SGD (resp. Adam). Nevertheless, DE4ML enables $\mathcal{M}$ to perform better on unseen prediction data $\mathcal{D}_{\text{pred}}$ with all tested optimizers (see Figure 5(g) in Section 6). This indicates that there may not exist a definite connection between the convergence rate of $\mathcal{M}$ on poisoned/defused $\mathcal{D}_{\text{train}}$ and its effectiveness on clean $\mathcal{D}_{\text{pred}}$.