Naive Bayes Classifiers over Missing Data: Decision and Poisoning

Song Bian [*1] **Xiating Ouyang** [*1] **Zhiwei Fan** [1] **Paraschos Koutris** [1]

Abstract

We study the certifiable robustness of ML classifiers on dirty datasets that could contain missing values. A test point is *certifiably robust* for an ML classifier if the classifier returns the same prediction for that test point, regardless of which cleaned version (among exponentially many) of the dirty dataset the classifier is trained on. In this paper, we show theoretically that for Naive Bayes Classifiers (NBC) over dirty datasets with missing values: (i) there exists an efficient polynomial time algorithm to decide whether multiple input test points are all certifiably robust over a dirty dataset; and (ii) the data poisoning attack, which aims to make all input test points certifiably non-robust by inserting missing cells to the clean dataset, is in polynomial time for single test points but **NP**-complete for multiple test points. Extensive experiments demonstrate that our algorithms are efficient and outperform existing baselines.

1. Introduction

The reliability of Machine Learning (ML) applications is heavily contingent upon the integrity of the training data. However, real-world datasets are frequently plagued with issues such as missing values, noise, and inconsistencies (Peng et al., 2021; Abedjan et al., 2016; Chu et al., 2016b; Xiong et al., 2021; Picado et al., 2020). Traditional approaches to handling such "dirty" datasets typically involve data cleaning. While effective, data cleaning can be an arduous and resource-intensive process, albeit various efforts to accelerate the data cleaning process (Chu et al., 2013; 2015a; Rekatsinas et al., 2017; Rezig et al., 2021).This raises a pivotal question in the domain of ML:

> *Can we circumvent the exhaustive data cleaning process by understanding the impact of data quality on ML models, particularly focusing on the notion of certifiable robustness*?

We explain it with the following example.

Example 1.1. Consider a data analyst working at an insurance company trying to predict whether an existing customer will leave (churn) using an ML classifier. Figure 1 illustrates three customers with age, residency status (inside or outside the US), and income, which are features used for this classification task. The data for the customer with CustomerID 11013 is incomplete, as the residency information is missing (marked with `NULL`). Therefore there are two possible ways to clean the dataset: we can assign the missing feature "In-US" to either True or False, resulting in two possible worlds generated from the incomplete dataset.

For the customer with CustomerID 11014, the ML classifier may predict True for that customer in both possible worlds. If this information is known before training and making predictions, data cleaning can be skipped since the prediction is *robust* with respect to the missing value. However, the customer with CustomerID 11015 presents a different scenario, where the predictions made by the model trained on two possible worlds disagree: the ML classifier would predict True in the possible world where the feature "In-US" is False, but in the other possible world, the prediction is False. Therefore, data cleaning is necessary to predict the customer with CustomerID 11015 accurately.

We investigate whether ML classifiers produce consistent predictions over incomplete (or dirty) data through the notion of *certifiable robustness*. A possible world (or cleaned dataset) of an incomplete dataset with missing values can be obtained by imputing each missing value marked as `NULL` with a valid value. Formally, a test point is said to be *certifiably robust* for an ML classifier if its prediction for that test point remains the same regardless of which possible world (among exponentially many) generated from the incomplete dataset it was trained on. Otherwise, the test point is said to be *certifiably non-robust*. This notion is helpful because if a test point is certifiably robust, then data cleaning is unnecessary because no matter how we clean the dataset to yield a possible world, the trained classifier will always give the same prediction. Therefore, deciding whether this case holds efficiently would be beneficial to save the cost of data cleaning. We formalize this into the following problem.

[*]Equal contribution [1]Department of Computer Sciences, University of Wisconsin-Madison, Madison WI, USA. Correspondence to: Song Bian <sbian8@wisc.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

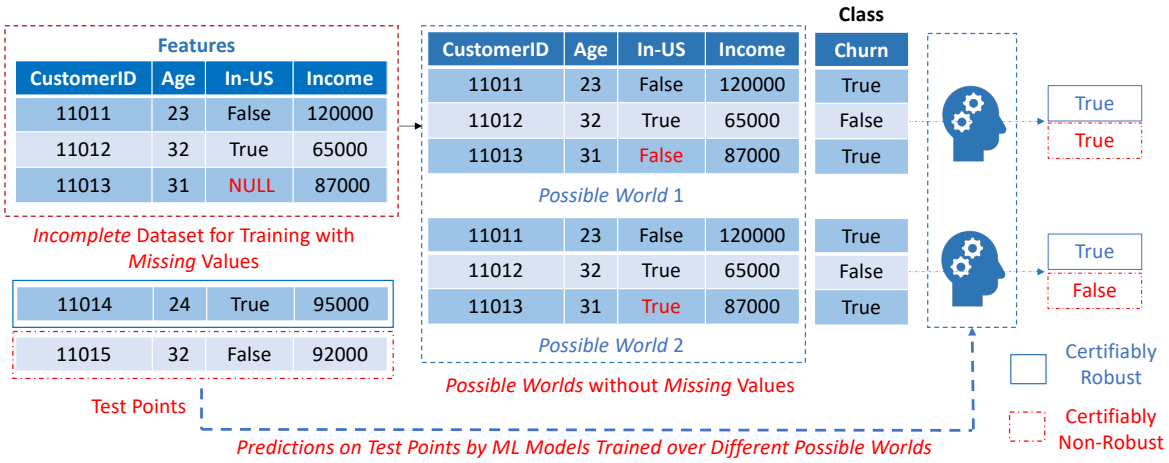


Figure 1: Example of an incomplete dataset, the possible worlds induced by the incomplete dataset, and the certifiably robust/non-robust predictions for data points with missing values over the possible worlds.

Decision Problem: Are given test points certifiably robust for an ML classifier on a given incomplete dataset?

In addition, we are interested in how sensitive the prediction of an ML classifier over a complete dataset is to the presence of missing data. We study such sensitivity through the lens of an attack model, in which the attacker attempt to modify the least amount of original cell values to NULL such that the test point becomes certifiably non-robust. This is formulated into the following problem.

Data Poisoning Problem: Given a number of test points and a clean dataset, what is the minimum number of cells to be modified to NULL such that all test points become certifiably non-robust?

While verifying whether a test point is certifiably robust is in general a computationally expensive task since it may require examing exponentially many possible worlds, it is possible to do so efficiently for some commonly used ML models. In this paper, we study the certifiable robustness of the Naive Bayes Classifier (NBC), a widely-used model in fields such as software defect prediction (Veni & Srinivasan, 2017; Arar & Ayan, 2017), education (Razaque et al., 2017), bioinformatics (Arias-Alcaide et al., 2022; Wan, 2023; Shen et al., 2019; Yu et al., 2019), and cybersecurity (Sahami et al., 1998; Kalutarage et al., 2017). Specifically, we focus on handling dirty datasets that contain missing values marked by NULL.

Contributions. In this paper, we study the two problems (i.e., Decision, and Data Poisoning) related to certifiable robustness for Naive Bayes Classifier (NBC), a simple but powerful supervised learning algorithm for predictive modeling that is widely used in academia and industry. Our main contributions are summarized as follows:

- We show that the Decision Problem for NBC can be solved in time $O(md + nd)$, where n is the number of data points in the dataset, m is the number of labels, and d is the number of features. Since the training and classification for NBC also requires $O(md + nd)$ time, our algorithm exhibits no asymptotic overhead and provides a much stronger guarantee of the classification result compared to the original NBC;

- We show that for a single test point, the Data Poisoning Problem for NBC can be solved in time $O(nmd)$; and for multiple test points, the Data Poisoning Problem is **NP**-complete for datasets containing at least three features. We also provide an efficient heuristic algorithm for multiple points case.

- We conduct extensive experiments using ten real-world datasets and demonstrate that our algorithms are far more efficient than baseline algorithms over Decision Problem and Data Poisoning Problem.[1]

2. Preliminaries

Data Model. We assume a finite set of domain values for each attribute (or feature). All continuous features are properly discretized into bins/buckets. For simplicity, we assume that every feature shares the same domain U, which contains a special element NULL. The set $\mathcal{X} = U^d$ is a feature space of dimension d. A datapoint $\mathbf{x}$ is a vector in $\mathcal{X}$ and we denote $\mathbf{x}_i$ as the attribute value at the i-th position of $\mathbf{x}$. We assume a labeling function $\ell : \mathcal{X} \to \mathcal{Y}$ for a finite set of labels $\mathcal{Y}$. A training dataset $\mathcal{D}$ is a set of data points in $\mathcal{X}$, each associated with a label in $\mathcal{Y}$.

[1]Our implementation is publicly available at https://github.com/Waterpine/NBC-Missing.

Missing Values and Possible Worlds. A data point $\mathbf{x}$ contains missing value if $\mathbf{x}_i = \texttt{NULL}$ for some i. A dataset $\mathcal{D}$ is complete if it does not contain data points with missing values, or is otherwise incomplete. We denote an incomplete dataset as $\mathcal{D}^{\square}$ and assume that all data points have a label. For an incomplete dataset $\mathcal{D}^{\square}$, a possible world can be obtained by replacing each attribute value marked with NULL with a domain value that exists in $\mathcal{D}^{\square}$. This follows the so-called *closed-world semantics* of incomplete data. We denote $\mathcal{P}(\mathcal{D}^{\square})$ as the set of all possible worlds that can be generated from $\mathcal{D}^{\square}$. Given an ML classifier f, we denote $f_{\mathcal{D}} : \mathcal{X} \to \mathcal{Y}$ as the classifier trained on a complete dataset $\mathcal{D}$ that assigns for each complete datapoint $\boldsymbol{t} \in \mathcal{X}$, a label $f_{\mathcal{D}}(\boldsymbol{t}) = l \in \mathcal{Y}$.

Certifiable Robustness. Given an incomplete dataset $\mathcal{D}^{\square}$ and an ML classifier f, a test point $\boldsymbol{t}$ is *certifiably robust* for f over $\mathcal{D}^{\square}$ if there exists a label $l \in \mathcal{Y}$ such that $f_{\mathcal{D}}(\boldsymbol{t}) = l$ for any possible world $\mathcal{D} \in \mathcal{P}(\mathcal{D}^{\square})$. Otherwise, the test point $\boldsymbol{t}$ is said to be *certifiably non-robust*.

Naive Bayes Classifier. Naive Bayes Classifier (NBC) is a simple but widely-used ML algorithm. Given a complete dataset $\mathcal{D}$ and a complete datapoint $\boldsymbol{t} = (x_1, x_2, \ldots, x_d)$ to be classified, the datapoint is assigned to the label $l \in \mathcal{Y}$ such that the probability $\mathbb{P}[l \mid \boldsymbol{t}]$ is maximized. NBC assumes that all features of $\mathcal{X}$ are conditionally independent for each label $l \in \mathcal{Y}$ and estimates, by the Bayes' Theorem, that

$$\begin{aligned}\mathbb{P}[l \mid \boldsymbol{t}] &= \mathbb{P}[l] \cdot \mathbb{P}[\boldsymbol{t}]^{-1} \cdot \mathbb{P}[\boldsymbol{t} \mid l] \\ &= \mathbb{P}[l] \cdot \mathbb{P}[\boldsymbol{t}]^{-1} \cdot \prod_{j=1}^{d} \mathbb{P}[x_j \mid l].\end{aligned}$$

Finally, it assigns for each test point $\boldsymbol{t} = (x_1, x_2, \ldots, x_d)$, the label l that maximizes $\mathbb{P}[l \mid \boldsymbol{t}]$.

To estimate the probabilities $\mathbb{P}[l]$ and $\mathbb{P}[x_j \mid l]$, NBC uses the corresponding relative frequency in the complete dataset $\mathcal{D}$, which we denote as $\Pr(l)_{\mathcal{D}}$ and $\Pr(x_j \mid l)_{\mathcal{D}}$ respectively. When a (complete or incomplete) dataset $\mathcal{D}$ of dimension d, a test point $\boldsymbol{t} = (x_1, x_2, \ldots, x_d)$ and m labels $l_1, l_2, \ldots, l_m$ are understood or clear from context, we denote N_i as the number of data points in $\mathcal{D}$ with label l_i, $E_{i,j}$ as the number of existing data points $\mathbf{x}$ in $\mathcal{D}$ with label l_i and has value x_j as its j-th attribute, and $M_{i,j}$ as the number of data points $\mathbf{x}$ in $\mathcal{D}$ with label l_i and has missing value NULL as its j-th attribute. Hence for a complete dataset $\mathcal{D}$ with n datapoints, NBC would first estimates that

$$\begin{aligned}\mathbb{P}[l_i] &\approx \Pr(l_i)_{\mathcal{D}} = N_i/n \\ \mathbb{P}[x_j \mid l_i] &\approx \Pr(x_j \mid l_i)_{\mathcal{D}} = E_{i,j}/N_i\end{aligned}$$

and then compute a value $S_{\mathcal{D}}(l \mid \boldsymbol{t})$, called a support value of $\boldsymbol{t}$ for label l in $\mathcal{D}$ given by

$$\begin{aligned}S_{\mathcal{D}}(l \mid \boldsymbol{t}) &= \Pr(l)_{\mathcal{D}} \cdot \Pr(\boldsymbol{t} \mid l)_{\mathcal{D}} \\ &= \Pr(l)_{\mathcal{D}} \cdot \prod_{1 \le j \le d} \Pr(x_j \mid l)_{\mathcal{D}}.\end{aligned}$$

Finally, it predicts that

$$f_{\mathcal{D}}(\boldsymbol{t}) = \arg\max_{l \in \mathcal{Y}} \quad S_{\mathcal{D}}(l \mid \boldsymbol{t})$$

Note that given a dataset $\mathcal{D}$ and a test point $\boldsymbol{t}$, the frequencies N_i, $E_{i,j}$ and $M_{i,j}$ can all be computed in time $O(nd)$.

For a possible world $\mathcal{D}$ generated from an incomplete dataset $\mathcal{D}^{\square}$ and a test point $\boldsymbol{t} = (x_1, x_2, \ldots, x_d)$, we use $\alpha_{i,j}(\mathcal{D})$ to denote the number of data points in $\mathcal{D}^{\square}$ with label l_i and value NULL at its j-th attribute in $\mathcal{D}^{\square}$ and altered to x_j in $\mathcal{D}$. It is easy to see that $0 \le \alpha_{i,j}(\mathcal{D}) \le M_{i,j}$.

Problem Definitions. For the Decision Problem CR-NaiveBayes, we are interested in deciding whether a test point is certifiably robust for NBC:

> Given an incomplete dataset $\mathcal{D}^{\square}$ and a test point $\boldsymbol{t}$, is $\boldsymbol{t}$ certifiably robust for NBC over $\mathcal{D}^{\square}$?

As will become apparent in our technical treatments later, we can easily extend our algorithm for CR-NaiveBayes on a single test point to multiple test points.

Conversely, an adversary may attempt to attack a complete dataset by inserting missing values, or replacing (poisoning) existing values with new ones. The data poisoning problem CR-NaiveBayes† asks for the fewest number of attacks that can make all given test points certifiably non-robust:

> Given a complete dataset $\mathcal{D}$ and test points $\boldsymbol{t}_1$, $\boldsymbol{t}_2$, $\ldots$, $\boldsymbol{t}_k$, find a poisoned instance $\mathcal{D}^{\dagger}$ of $\mathcal{D}$ that has as few missing values as possible such that every $\boldsymbol{t}_i$ is certifiably non-robust for NBC on $\mathcal{D}^{\dagger}$.

A solution to CR-NaiveBayes† provides us with a notion of the robustness of the training dataset against poisoning attacks. For example, if the "smallest" poisoned instance has 100 missing values, this means that an attacker needs to change at least 100 values of the dataset to alter the prediction of the targeted test point(s). In addition, it also implies that as long as fewer than 100 cells are poisoned, the prediction of the test point will not be altered even in the presence of missing data.

3. Decision Algorithms

In this section, we give an efficient algorithm for CR-NaiveBayes. Let $\mathcal{D}^{\square}$ be an incomplete dataset and let $\boldsymbol{t}$

be a test point. For an arbitrary label l, consider the maximum and minimum support value of $\boldsymbol{t}$ for l over all possible world $\mathcal{D}$ of $\mathcal{D}^{\square}$. We define that

$$S^{\uparrow}_{\mathcal{D}^{\square}}(l \mid \boldsymbol{t}) := \max_{\mathcal{D}\in\mathcal{P}(\mathcal{D}^{\square})} S_{\mathcal{D}}(l \mid \boldsymbol{t})$$
$$S^{\downarrow}_{\mathcal{D}^{\square}}(l \mid \boldsymbol{t}) := \min_{\mathcal{D}\in\mathcal{P}(\mathcal{D}^{\square})} S_{\mathcal{D}}(l \mid \boldsymbol{t})$$

Our algorithm relies on the following observation, which was first studied in (Ramoni & Sebastiani, 2001).

Lemma 3.1. *A test point $\boldsymbol{t}$ is certifiably robust for NBC over $\mathcal{D}^{\square}$ if and only if there is a label l such that for any label $l' \neq l$, $S^{\downarrow}_{\mathcal{D}^{\square}}(l \mid \boldsymbol{t}) > S^{\uparrow}_{\mathcal{D}^{\square}}(l' \mid \boldsymbol{t})$.*

The full proof of Lemma 3.1 is given in Appendix A.1.

Algorithm 1: Iterative Algorithm for the Decision Problem

Input: Incomplete dataset $\mathcal{D}^{\square}$, test point $\boldsymbol{t}$
Output: Is $\boldsymbol{t}$ certifiably robust?

1 **foreach** *label* $l_i \in \mathcal{Y}$ **do**
2 $\quad N_i \leftarrow$ the number of data points in $\mathcal{D}^{\square}$ with label l_i
3 $\quad$ **for** $j = 1$ *to* d **do**
4 $\quad\quad E_{i,j} \leftarrow$ the number of existing data points $\boldsymbol{x}$ in $\mathcal{D}^{\square}$ with label l_i and $\boldsymbol{x}_j = \boldsymbol{t}_j$.
5 $\quad\quad M_{i,j} \leftarrow$ the number of missing data points $\boldsymbol{x}$ in $\mathcal{D}^{\square}$ with label l_i and $\boldsymbol{x}_j = \texttt{NULL}$.
6 **foreach** *label* $l_i \in \mathcal{Y}$ **do**
7 $\quad S^{\downarrow}_i \leftarrow N_i^{-(d-1)} \cdot \prod_{1\le j\le d} E_{i,j}$
8 $\quad S^{\uparrow}_i \leftarrow N_i^{-(d-1)} \cdot \prod_{1\le j\le d} (E_{i,j} + M_{i,j})$
9 **if** $\exists l_i \in \mathcal{Y}$ *such that* $S^{\downarrow}_i > \max_{j\neq i} S^{\uparrow}_j$ **then**
10 $\quad$ **return** true
11 **return** false

For each label l_i, both quantities $S^{\downarrow}_{\mathcal{D}^{\square}}(l_i \mid \boldsymbol{t})$ and $S^{\uparrow}_{\mathcal{D}^{\square}}(l_i \mid \boldsymbol{t})$ can be computed efficiently. Intuitively, it suffices to inspect only the "extreme" possible world that is the worst and best for l_i, respectively. We can achieve this by assigning all the missing cells to disagree or agree with the test point on the corresponding attributes.

We now formally prove this fact. Fix a possible world $\mathcal{D}$ of $\mathcal{D}^{\square}$, and let $E_{i,j}$, $\alpha_{i,j}(\mathcal{D})$ and N_i be relative to $\mathcal{D}^{\square}$ and $\boldsymbol{t}$. We have

$$S_{\mathcal{D}}(l_i \mid \boldsymbol{t}) = \Pr(l_i)_{\mathcal{D}} \cdot \prod_{1\le j\le d} \Pr(x_j \mid l_i)_{\mathcal{D}} = \frac{N_i}{n} \cdot \prod_{1\le j\le d} \frac{E_{i,j} + \alpha_{i,j}(\mathcal{D})}{N_i},$$

which is maximized when every $\alpha_{i,j}(\mathcal{D}) = M_{i,j}$ and minimized when every $\alpha_{i,j}(\mathcal{D}) = 0$. Moreover, both extreme values are attainable. Hence

$$S^{\uparrow}_{\mathcal{D}^{\square}}(l_i \mid \boldsymbol{t}) = \frac{N_i}{n} \cdot \prod_{1\le j\le d} \frac{E_{i,j} + M_{i,j}}{N_i} \tag{1}$$

$$S^{\downarrow}_{\mathcal{D}^{\square}}(l_i \mid \boldsymbol{t}) = \frac{N_i}{n} \cdot \prod_{1\le j\le d} \frac{E_{i,j}}{N_i}. \tag{2}$$

Our algorithm is presented in Algorithm 1. We first compute, for each label l, Eq. (1) and (2) in line 1–9, and then check whether there is some label l_i such that for any label $l_j \neq l_i$,

$$S^{\downarrow}_{\mathcal{D}^{\square}}(l_i \mid \boldsymbol{t}) > S^{\uparrow}_{\mathcal{D}^{\square}}(l_j \mid \boldsymbol{t}), \tag{3}$$

where a strict inequality is required to return true.

Running Time. Line 1–5 runs in time $O(nd)$, where n is the number of points in $\mathcal{D}$, and d is the number of dimension of points in the dataset. Line 6–8 takes $O(md)$ time, where m is the number of labels in the dataset. Line 9–11 can be implemented in $O(m)$, by first computing each $\max_{j\neq i} S^{\uparrow}_j$ in $O(m)$ and then iterating through every l_i. In conclusion, the total time complexity of Algorithm 1 is $O(nd)$.

Extension to Multiple Test Points. Given k test points $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_k$, a trivial yet inefficient way is to run Algorithm 1 for every test point, giving a running time of $O(knd)$. We show that Algorithm 1 can be easily adapted to check whether multiple test points are all certifiably robust efficiently with the help of an index in time $O(nd + kmd)$: We can modify line 1–5 to compute an index $E_{i,j}[x_j]$, which represents the number of existing data points $\boldsymbol{x}$ in $\mathcal{D}^{\square}$ with label l_i and value $\mathbf{x}_j$ in $\mathcal{D}^{\square}$, instead of $\boldsymbol{t}_j$ in the original algorithm. This runs in $O(nd)$ time and requires $O(md)$ space. Then we iterate line 6–11 for each of the k test points, where in each iteration, both quantities in line 7–8 can be efficiently computed by checking the index $E_{i,j}[x_j]$ in $O(md)$ time. The overall running time is thus $O(nd + kmd)$.

4. Data Poisoning Algorithms

In this section, we consider the setting where an attacker attempts to poison a complete dataset by injecting some missing cells so that a given set of test points becomes certifiably non-robust. From this setting, we can better comprehend if certifiable robustness is a stringent requirement within specific models. We first present a simple greedy algorithm that solves CR-NaiveBayes† optimally for a single test point in Section 4.1. We then show that CR-NaiveBayes† is **NP**-complete for multiple test points and provide an efficient heuristic algorithm in Section 4.2.

4.1. A Single Test Point

Let us first consider a simpler problem that does not involve missing values and considers only a single test point.

Definition 4.1 (AlterPrediction)**.** Given a complete dataset $\mathcal{D}$, a test point $\boldsymbol{t}$ with $l^* = f_{\mathcal{D}}(\boldsymbol{t})$, a label $l \neq l^*$, the AlterPrediction problem is to find a complete dataset $\mathcal{D}'$, obtained by altering the minimum number of cells in $\mathcal{D}$ so that $S_{\mathcal{D}'}(l \mid \boldsymbol{t}) > S_{\mathcal{D}'}(l^* \mid \boldsymbol{t})$.

We show that an algorithm for AlterPrediction immediately leads to an algorithm for CR-NaiveBayes†.

Lemma 4.2. *Let $\mathcal{D}$ be a complete dataset and $\boldsymbol{t}$ a test point. Let $l^* = f_{\mathcal{D}}(\boldsymbol{t})$. Let k be an integer. Then the following statements are equivalent:*

1. *There exists a solution $\mathcal{D}^\dagger$ for CR-NaiveBayes† for a complete dataset $\mathcal{D}$ and a test point $\boldsymbol{t}$ with k altered cells.*
2. *There exists a label $l \neq l^*$ and a solution $\mathcal{D}'$ for AlterPrediction for a complete dataset $\mathcal{D}$, a test point $\boldsymbol{t}$ and label l^* with k altered cells.*

The full proof of Lemma 4.2 is in Appendix B.1

The outline of our algorithm for CR-NaiveBayes† on a single test point is presented in Algorithm 2. It thus remains to solve AlterPrediction efficiently.

Algorithm 2: CR-NaiveBayes†-Single

Input: A complete dataset $\mathcal{D}$, a test point $\boldsymbol{t}$
Output: An incomplete dataset $\mathcal{D}^\dagger$ obtained by setting the minimum number of cells in $\mathcal{D}$ to NULL such that $\boldsymbol{t}$ is not certifiably-robust for NBC

```
l* = f_D(t)
minAlter ← ∞, D† ← ∅
foreach l in Y \ {l*} do
    D_l ← AlterPrediction(D, t, l)
    alter ← the number of altered cells in D_l w.r.t. D
    if alter < minAlter then
        minAlter ← alter
        D† ← set all altered cells in D_l to NULL
return D†
```

An Optimal Greedy Algorithm for AlterPrediction

Our AlterPrediction algorithm is presented in Algorithm 3. At a high level, it reduces the objective quantity Δ (at line 2), by choosing (at lines 17-20) the better operation of the two, one considered in lines 3-9, and the other in lines 10-16.

Step 1. Let us explain first on why the quantity Δ is crucial to the correctness of this algorithm.

Let $\mathcal{D}$, $\boldsymbol{t}$, $l^* = f_{\mathcal{D}}(\boldsymbol{t})$ and $l \neq l^*$ be input to the AlterPrediction problem, where $\mathcal{D}$ is an incomplete dataset and $\boldsymbol{t}$ is a test point.

We consider an iterative process where at each step, only one single cell is altered: We start from $\mathcal{D}_0 = \mathcal{D}$ and alter one single cell in $\mathcal{D}_0$ and obtain $\mathcal{D}_1$. We then repeat the same procedure on $\mathcal{D}_1$, until we stop at some $\mathcal{D}_k = \mathcal{D}'$. Fix a label $l \neq l^*$. In the original dataset $\mathcal{D}$, the test point $\boldsymbol{t} = (x_1, x_2, \ldots, x_n)$ is predicted l^*, and thus we have necessarily

$$S_{\mathcal{D}}(l^* \mid \boldsymbol{t}) > S_{\mathcal{D}}(l \mid \boldsymbol{t}),$$

or equivalently

$$\Pr(l^*)_{\mathcal{D}_0} \cdot \Pr(\boldsymbol{t} \mid l^*)_{\mathcal{D}_0} - \Pr(\boldsymbol{t} \mid l)_{\mathcal{D}_0} \cdot \Pr(l)_{\mathcal{D}_0} > 0$$

If the ending dataset $\mathcal{D}' = \mathcal{D}_k$ is a solution to AlterPrediction, we have

$$S_{\mathcal{D}'}(l^* \mid \boldsymbol{t}) < S_{\mathcal{D}'}(l \mid \boldsymbol{t})$$

or equivalently,

$$\Pr(l^*)_{\mathcal{D}_k} \cdot \Pr(\boldsymbol{t} \mid l^*)_{\mathcal{D}_k} - \Pr(\boldsymbol{t} \mid l)_{\mathcal{D}_k} \cdot \Pr(l)_{\mathcal{D}_k} < 0.$$

We define the quantity

$$\Delta_i := \Pr(\boldsymbol{t} \mid l^*)_{\mathcal{D}_i} \cdot \Pr(l^*)_{\mathcal{D}_i} - \Pr(\boldsymbol{t} \mid l)_{\mathcal{D}_i} \cdot \Pr(l)_{\mathcal{D}_i}$$

It is easy to see that the AlterPrediction problem is equivalent to finding a smallest k and a sequence $\{\mathcal{D}_i\}_{0 \leq i \leq k}$ such that $\Delta_i > 0$ for $0 \leq i < k$ and $\Delta_k < 0$, which can be solved by finding for a fixed k, the dataset $\mathcal{D}_k$ with the smallest possible Δ_k.

Step 2. We now show that Δ can be reduced to negative using the fewest number of steps by performing only A1 or only A2 on the dataset $\mathcal{D}_i$ iteratively, where

A1: Alter datapoints with label l in $\mathcal{D}_i$ so that $\Pr(\boldsymbol{t} \mid l)_{\mathcal{D}_i}$ increases the most; and

A2: Alter datapoints with label l^* in $\mathcal{D}_i$ so that $\Pr(\boldsymbol{t} \mid l^*)_{\mathcal{D}_i}$ decreases the most.

An example is illustrated in Example B.2.

Note that $\Pr(l)_{\mathcal{D}_i}$ remains the same for any label l. Indeed, altering cells in points with labels not in $\{l, l^*\}$ does not change the value of neither $\Pr(l \mid \boldsymbol{t})_{\mathcal{D}_i}$ nor $\Pr(l^* \mid \boldsymbol{t})_{\mathcal{D}_i}$. Besides, altering cells so that either $\Pr(l^* \mid \boldsymbol{t})_{\mathcal{D}_i}$ increases or $\Pr(l \mid \boldsymbol{t})_{\mathcal{D}_i}$ decreases is not helpful, because we would waste one alternation by increasing Δ_i, not decreasing it.

Note that for any label l and a test point $\boldsymbol{t} = (x_1, x_2, \ldots, x_d)$,

$$\Pr(\boldsymbol{t} \mid l)_{\mathcal{D}_i} = \prod_{1 \leq j \leq d} \Pr(x_j \mid l)_{\mathcal{D}_i} = \prod_{1 \leq j \leq d} \frac{E_j}{N}, \quad (4)$$

where N is the number of data points in $\mathcal{D}_i$ with label l and E_j is the number of data points in N with label l whose

j-th attribute agrees with x_j. Note that N is fixed, hence to make the largest increase or decrease in A1 or A2, we would always alter datapoints in $\mathcal{D}_i$ with the smallest nominator E_j in $\Pr(x_j \mid l)_{\mathcal{D}_i}$ so that the number of data points in $\mathcal{D}_{i+1}$ with label l value x_j as j-th attribute increases or decreases by 1.

The following observations behind the strategies are crucial, which we formally prove in Section B.2 in Appendix B. An example is provided in Example B.3.

O1: Whenever we apply A1 to decrease Δ_i, the reduction in Δ_i is non-decreasing in the step i.

O2: Whenever we apply A2 to decrease Δ_i, the reduction in Δ_i remains the same across every step i.

O3: The reduction in Δ_i obtained by applying A1 (or A2) only depends on the number of times A1 (or A2) has been applied to obtain $\mathcal{D}_i$ previously.

Our key result is summarized by the following Lemma, which we prove in Section B.3 in Appendix B.

Lemma 4.3. *Let δ_i^+ be the decrease in Δ_j whenever $\mathcal{D}_j$ is obtained by applying A1 in the sequence for the i-th time, implied by O1 and O3. Let δ^- be the fixed decrease in Δ_i whenever we apply A2 to $\mathcal{D}_i$, implied by O2 and O3. Then*

$$\Delta_k \geq \Delta_0 - \max\{k \cdot \delta^-, \sum_{1 \leq j \leq k} \delta_j^+\}, \tag{5}$$

and equality is attainable.

Lemma 4.3 means that among all possible datasets $\mathcal{D}_k$ obtained by k alternations to $\mathcal{D}$, the dataset with the smallest Δ_k can be obtained by either applying only A1 or only A2.

Algorithm 3 implements this idea, in which line 3–9 computes the poisoned dataset $\mathcal{D}^+$ with minimum number k^+ of alternations to $\mathcal{D}$ by applying A1. Similarly, line 10–16 computes $\mathcal{D}^-$ and k^- respectively for repeatedly applying A2 to $\mathcal{D}$.

Running Time. For Algorithm 3, each `while` loop will be executed at most n times. Lines 5–9 and 12–16 can be completed in constant time by preprocessing the table while executing line 1. Hence Algorithm 3 runs in $O(nd + md)$ time. Algorithm 2 then takes $O(nmd)$ time.

Algorithm 3: AlterPrediction

Input: A complete dataset $\mathcal{D}$, test point $\boldsymbol{t}$, and a label $l \neq f_{\mathcal{D}}(\boldsymbol{t})$

Output: A complete dataset $\mathcal{D}'$ with minimum number of altered cells in $\mathcal{D}$ such that $S(l^* \mid \boldsymbol{t})_{\mathcal{D}'} < S(l \mid \boldsymbol{t})_{\mathcal{D}'}$

1 $l^* \leftarrow f_{\mathcal{D}}(\boldsymbol{t})$
2 $\Delta = \Pr(\boldsymbol{t} \mid l^*)_{\mathcal{D}} \cdot \Pr(l^*)_{\mathcal{D}} - \Pr(\boldsymbol{t} \mid l)_{\mathcal{D}} \cdot \Pr(l)_{\mathcal{D}}$
3 $\Delta^+ = \Delta$, $k^+ = 0$, $\mathcal{D}^+ = \mathcal{D}$
4 **while** $\Delta^+ > 0$ **do**
5 $\quad j \leftarrow$ an attribute index with the smallest nominator in $\Pr(\boldsymbol{t}_j \mid l)_{\mathcal{D}^+}$
6 $\quad p^+ \leftarrow$ a point in $\mathcal{D}^+$ with label l and $p_j^+ \neq x_j$
7 $\quad \mathcal{D}^+ \leftarrow$ set p_j^+ as x_j
8 $\quad \Delta^+ = \Pr(\boldsymbol{t} \mid l^*)_{\mathcal{D}^+} \cdot \Pr(l^*)_{\mathcal{D}^+} - \Pr(\boldsymbol{t} \mid l)_{\mathcal{D}^+} \cdot \Pr(l)_{\mathcal{D}^+}$
9 $\quad k^+ \leftarrow k^+ + 1$
10 $\Delta^- = \Delta$, $k^- = 0$, $\mathcal{D}^- = \mathcal{D}$
11 **while** $\Delta^- > 0$ **do**
12 $\quad j \leftarrow$ an attribute index with the smallest nominator in $\Pr(\boldsymbol{t}_j \mid l^*)_{\mathcal{D}^-}$
13 $\quad p^- \leftarrow$ a point in $\mathcal{D}^-$ with label l^* and $p_j^- = x_j$
14 $\quad \mathcal{D}^- \leftarrow$ set p_j^- as c where $c \neq x_j$
15 $\quad \Delta^- = \Pr(\boldsymbol{t} \mid l^*)_{\mathcal{D}^-} \cdot \Pr(l^*)_{\mathcal{D}^-} - \Pr(\boldsymbol{t} \mid l)_{\mathcal{D}^-} \cdot \Pr(l)_{\mathcal{D}^-}$
16 $\quad k^- \leftarrow k^- + 1$
17 **if** $k^- < k^+$ **then**
18 $\quad$ **return** $\mathcal{D}^-$
19 **else**
20 $\quad$ **return** $\mathcal{D}^+$

4.2. Multiple Test Points

We now consider the case in which an attacker wishes to attack a complete dataset $\mathcal{D}$ with minimum number of cells such that all k test points $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_k$ are certifiably non-robust for NBC.

A simple heuristic algorithm simply iteratively runs Algorithm 2 over every test point $\boldsymbol{t}_i$ and then take the union of all the missing cells found, which we present in Algorithm 4 in Appendix B.4 in detail. While it can be easily verified that all test points are not certifiably robust for NBC over the incomplete dataset produced by Algorithm 4, the number of poisoned cells is not necessarily minimal. Indeed, compared with the single test point setting, the challenge of poisoning a dataset for multiple test points is that altering one single cell may affect the prediction of all test points that agree on that cell.

This observation allows us to show that the data poisoning problem for multiple test points is **NP**-complete for datasets over at least 3 dimensions.

Theorem 4.4. *For every $d \geq 3$, CR-NaiveBayes† is* **NP**-*complete on datasets with d dimensions and multiple test points.*

The full proof of Theorem 4.4 is deferred to Appendix B.5, and an example reduction is provided in Example B.4.

5. Experiments

In this section, we present the results of our experimental evaluation. we perform experiments on ten real-world datasets from Kaggle (web, 2022a), and compare the performance of our efficient algorithms (as presented in Sections 3 and 5) against other straightforward solutions as baselines.

5.1. Experimental Setup

We briefly describe here the setup for our experiments.

System Configuration. Our experiments were performed on a bare-metal server provided by Cloudlab (CloudLab). The server is equipped with two 10-core Intel Xeon E5-2660 CPUs running at 2.60 GHz.

Datasets. We use ten real-world datasets from Kaggle: heart (HE), fitness-club (FC), fetal-health (FH), employee (EM), winequalityN (WQ), company-bankruptcy (CB), Mushroom (MR), bodyPerformance (BP), star-classification (SC), creditcard (CC). The details and the metadata of our datasets are summarized in Appendix C.1. We first preprocess every dataset so that it contains only categorical features by partitioning each numerical feature into 5 segments (or bins) of equal size using sklearn's KBinsDiscretizer (web, 2022b). The datasets are originally complete and do not have missing values.

To obtain perturbed incomplete datasets with missing values from each of the ten datasets, we sample data cells to be marked as NULL as missing values uniformly across all cells from all features. The number of cells to be sampled is determined by a *missing rate*, which is defined as the ratio between number of cells with missing values and the number of total cells considered in the dataset. We generate the incomplete datasets by varying the missing rate from 20% to 80% with an increment of 20%. To fill cells where values are missing, we consider the *active domain* (i.e., values observed in the dataset) of each categorical feature. Given the complete dataset, we randomly divide the dataset into two subsets: 80% for training and 20% for testing. Next, the training set is used for model training and we randomly select 1 to 16 data points from the testing set as test points.

5.2. Algorithms Evaluated

5.2.1. Decision Problem.

For CR-NaiveBayes, we compare the following algorithms.

Approximate Decision (AD). This algorithm simply samples 100 possible worlds uniformly at random, and returns *certifiably robust* if the NBC prediction for the test point agrees for every sampled possible world that NBC is trained on, or otherwise returns *certifiably non-robust*. Note that this algorithm might return false positive results.

Iterative Algorithm (Iterate). This is the algorithm mentioned in (Ramoni & Sebastiani, 2001). When we are given k test points, the algorithm mentioned (Ramoni & Sebastiani, 2001) will run Algorithm 1 for k times without using the indexing techniques proposed in Section 3.

Iterative Algorithm with Index (Iterate+Index). For multiple test points, we run Algorithm 1 combined with the indexing techniques described in Section 3.

5.2.2. Data Poisoning Problem.

For CR-NaiveBayes†, we evaluate the following algorithms.

Random Poisoning (RP). This algorithm randomly selects a cell to be marked as NULL iteratively from the original complete dataset until it produces an incomplete dataset for which $\boldsymbol{t}$ is certifiably non-robust, which is checked by running **Iterate**. There is no limit on the number of cells to be selected.

Smarter Random Poisoning (SR). In this algorithm, we first obtain the prediction l of the input test point over the input complete dataset, and randomly fix another label $l' \neq l$. Then, we randomly perform operation A1 or A2, as introduced in Section 4, until $\boldsymbol{t}$ becomes certifiable non-robust, which is checked using Algorithm 1.

Greedy Search Poisoning (GS). This algorithm is described in the Algorithm 2 mentioned in Section 4. This algorithm is able to achieve optimal solution when we are given single test point.

When we are given more than one test point, we use the above algorithms in the Line 3 of Algorithm 4 in Appendix B.4.

5.3. Evaluation Results

For both CR-NaiveBayes and CR-NaiveBayes†, we report the average running time among the 5 executions for fixed missing rate and the number of points. For CR-NaiveBayes†, we additionally report the poisoning rate (for poisoning algorithms only), defined as the ratio between the number of cells marked as NULL by the data poisoning algorithm and the total number of cells in the training set. We defer some additional results to Appendix C.

5.3.1. Decision Problems.

We first evaluate the running time of different decision algorithms (**AD**, **Iterate** and **Iterate+Index**) for 16 test points on different datasets over varying missing rates from 20% to 80%, with an increment of 20%. From Figure 2, we observe that **Iterate+Index** is almost $20\times$ faster than **Iterate** and much faster than the straightforward solution **AD**, which nonetheless does not have correctness guarantees. We can also see that the efficiency of **Iterate+Index** is robust

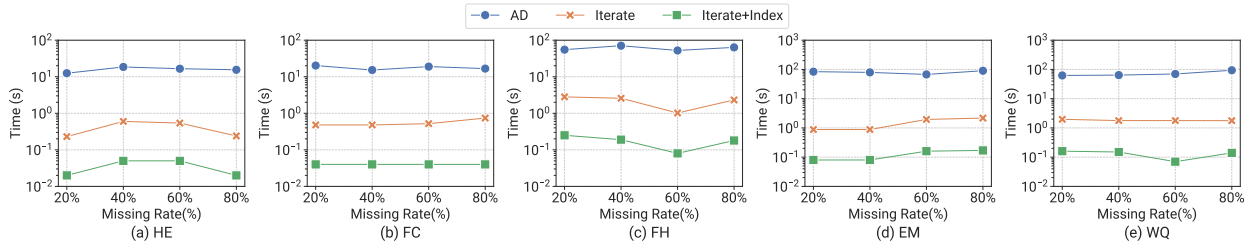


Figure 2: Decision - Running Time vs Missing Rates on Different Datasets

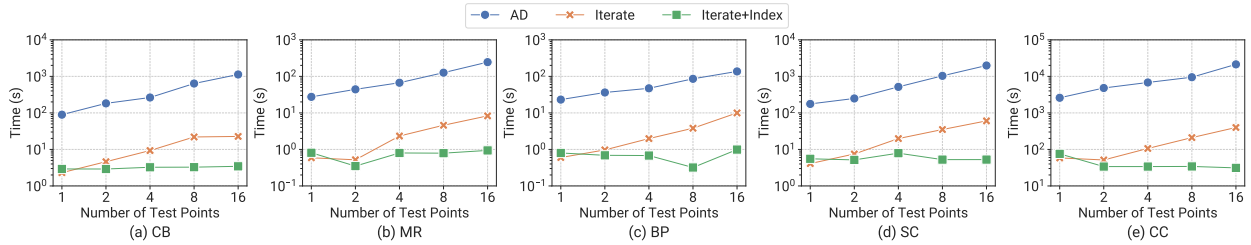


Figure 3: Decision - Running Time vs Number of Test Points on Different Datasets

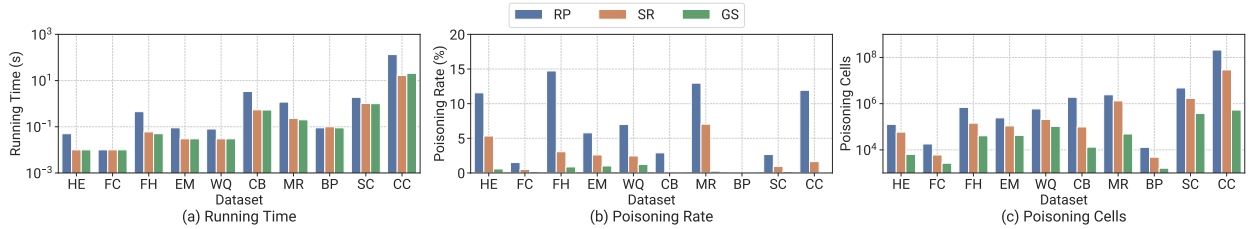


Figure 4: Single Point Data Poisoning—Running Time and Poisoning Rate on Different Datasets

against the missing rate in the datasets as the running time of our algorithms is almost uniform. This shows that it is viable to efficiently apply certifiable robustness in practice to obtain much stronger guarantees in addition to the traditional ML pipeline.

In addition, we run **AD**, **Iterate** and **Iterate+Index** on varying number of test points for each dataset. Figure 3 shows the running time of each algorithm over datasets obtained by varying from 1 test point to 16 test points. We observe that the running time of **AD** and **Iterate** grows almost linearly with the number of test points, whereas the running time of **Iterate+Index** remains almost the same regardless of the number of test points. The initial offline index introduced in the end of Section 3 can be slow to build at the beginning (see the sub-Figures when the number of test points is 1), but its benefits are more apparent when the number of test points increases. For 16 test points, we see that **Iterate+Index** outperforms **Iterate** by $10\times$.

In summary, **Iterate+Index** is a practical algorithm that can efficiently certify robustness for NBC for multiple test points and large datasets.

5.3.2. Data Poisoning Problem.

We now present the evaluation results of different data poisoning algorithms for a single test point. We defer the evaluation of multiple test points to the Appendix C.3 due to space constraints.

In Figure 4, we see that the running time of **GS** is significantly faster than **RP** and marginally faster than **SR** over most datasets. This is because **SR** uses the optimal strategies A1 and A2 as in **GS**, but not in the optimal sequence.

Moreover, Figure 4 presents the ratio between the number of poisoned cells and the total number of cells, called the poisoning rate. Note that **RP** and **SR** always have a higher poisoning rate compared to **GS** over Single Point Data Poisoning Problem. Since **GS** is provably optimal when we are given a single test point, it has the smallest poisoning rate across all algorithms for the same dataset.

Furthermore, Figure 4 shows the exact number of poisoned cells. Note that the number of poisoning cells is not small in Figure 4, which indicates that the answer to the Decision Problem can help us to accelerate the data cleaning process.

Finally, note that **GS** achieves different poisoning rates for different datasets and the poisoning rates are often low. For example, the poisoning rate for FC is much lower than that for WQ, showing that the dataset FC is more prone to a data poisoning attack than WQ. In addition, the experimental results over ten real datasets indicate that NBC is sensitive to the data quality.

In conclusion, **GS** not only provides guarantees for finding the minimum number of poisoned cells when we are given only one test point, but also achieves the best performance in terms of running time and over multiple points case. It also provides insights on the applicability of certifiable robustness.

6. Related Work

Data Poisoning. Data poisoning attacks can be divided into triggered and triggerless attacks. As for triggered attacks, they modify the training dataset by adding poison examples, which leads to the models incorrectly categorizing test examples (Chen et al., 2017; Gu et al., 2017; Liu et al., 2018; Turner et al., 2019; Nguyen & Tran, 2020; Carlini & Terzis, 2021; Souri et al., 2022). As for triggerless attacks, they introduce minor adversarial disturbances to the existing training dataset, causing the models to incorrectly classify test examples (Shafahi et al., 2018; Zhu et al., 2019; Geiping et al., 2020; Aghakhani et al., 2021; Yang et al., 2022).

Robustness in ML. Robust learning algorithms have received much attention recently. Robustness of decision tree under adversarial attack has been studied in (Chen et al., 2019; Vos & Verwer, 2021) and interests in certifying robust training methods have been observed (Shi et al., 2021). Most recently, efficient algorithms have been developed for certifying the robustness of k-Nearest Neighbors (Karlaš et al., 2020), SVM and linear regression (Steinhardt et al., 2017; Zhen et al., 2023).

Querying over Inconsistent Databases. Inconsistent database has been studied for several decades. *Consistent query answering* is a principled approach to answer consistent answer from an inconsistent database violating from (Arenas et al., 1999; Koutris & Wijsen, 2018a;b). A long line of research has been dedicated to studying this question from theoretical perspective (Libkin, 2011; Koutris & Suciu, 2012; Koutris & Wijsen, 2015; Koutris et al., 2021; 2024) and system implementations (Dixit & Kolaitis, 2019; Marileo & Bertossi, 2010; Manna et al., 2015; Khalfioui et al., 2020; Fan et al., 2023).

Data Cleaning. Cleaning the dirty data is the most natural resolution in the presence of dirty data. There has been a long line of research on data cleaning, such as error detection (Abedjan et al., 2016; Mahdavi et al., 2019; Heidari et al., 2019; Ilyas & Rekatsinas, 2022), missing value imputation (Trushkowsky et al., 2013; Wu et al., 2020), and data deduplication (Chu et al., 2016a; Zhou et al., 2022; Feng & Deng, 2021; Wang et al., 2022). Multiple data cleaning frameworks have been developed (Wang et al., 2014; Krishnan et al., 2016; 2017). For example, SampleClean (Wang et al., 2014) introduces the Sample-and-Clean framework, which answers aggregate query by cleaning a small subset of the data. ActiveClean (Krishnan et al., 2016) is used to clean data for convex models which are trained by using gradient descent. BoostClean (Krishnan et al., 2017) selects data cleaning algorithms from pre-defined space. Moreover, there also exists some data cleaning frameworks by using knowledge bases and crowdsourcing (Bergman et al., 2015; Chu et al., 2015b; Li et al., 2016). For instance, KATARA (Chu et al., 2015b) resolves ambiguity by using knowledge bases and crowdsourcing marketplaces. Furthermore, there have been efforts to accelerate the data cleaning process (Rekatsinas et al., 2017; Chu et al., 2013; 2015a; Rezig et al., 2021; Yan et al., 2020). Recently, CP-Clean (Karlaš et al., 2020) provides a data cleaning framework based on certain prediction.

7. Conclusion, Limitations, and Future Work

In this paper, we study the certifiable robustness of Naive Bayes classifiers over incomplete datasets by solving the Decision Problem and the Data Poisoning Problem algorithmically. The experimental results show that all our algorithms exhibit a significant speed-up against the baseline algorithm.

While our results only hold for Naive Bayes Classifiers, an illuminating future direction to extend the study of certifiable robustness to other popularly used models with more complex structures, such as random forest and gradient boosted trees. It is also intriguing to design a general framework to study certifiable robustness for general ML classifiers.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback and comments. We also thank Austen Z. Fan for his constructive discussions during the early development of this project. This work is supported by the National Science Foundation under grant IIS-1910014 and the Anthony C. Klug NCR Fellowship.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Kaggle, 2022a. URL https://www.kaggle.com/.

Sklearn, 2022b. URL https://scikit-learn.org/stable/index.html.

bodyPerformance Dataset, 2023a. URL https://www.kaggle.com/kukuroo3/body-performance-data.

Company Bankruptcy Dataset, 2023b. URL https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction.

Employee Dataset, 2023c. URL https://www.kaggle.com/datasets/tawfikelmetwally/employee-dataset.

fetal_health Dataset, 2023d. URL https://www.kaggle.com/andrewmvd/fetal-health-classification.

Heart Dataset, 2023e. URL https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction.

mushroom Dataset, 2023f. URL https://www.kaggle.com/uciml/mushroom-classification.

Wine Quality Dataset, 2023g. URL https://www.kaggle.com/datasets/rajyellow46/wine-quality.

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

Aghakhani, H., Meng, D., Wang, Y.-X., Kruegel, C., and Vigna, G. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In *2021 IEEE European symposium on security and privacy (EuroS&P)*, pp. 159–178. IEEE, 2021.

Arar, Ö. F. and Ayan, K. A feature dependent naive bayes approach and its application to the software defect prediction problem. *Applied Soft Computing*, 59:197–209, 2017.

Arenas, M., Bertossi, L., and Chomicki, J. Consistent query answers in inconsistent databases. In *PODS*, 1999.

Arias-Alcaide, C., Soguero-Ruiz, C., Santos-Alvarez, P., Arche, J. F. V., and Mora-Jiménez, I. Local naïve bayes for predicting evolution of covid-19 patients on self organizing maps. In *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1443–1450. IEEE, 2022.

Bergman, M., Milo, T., Novgorodov, S., and Tan, W.-C. Query-oriented data cleaning with oracles. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1199–1214, 2015.

Brooks, R. L. On colouring the nodes of a network. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 37, pp. 194–197. Cambridge University Press, 1941.

Carlini, N. and Terzis, A. Poisoning and backdooring contrastive learning. *arXiv preprint arXiv:2106.09667*, 2021.

Chen, H., Zhang, H., Si, S., Li, Y., Boning, D., and Hsieh, C.-J. Robustness verification of tree-based models. *arXiv:1906.03849*, 2019.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Chu, X., Ilyas, I. F., and Papotti, P. Holistic data cleaning: Putting violations into context. In *ICDE*, pp. 458–469. IEEE Computer Society, 2013.

Chu, X., Morcos, J., Ilyas, I. F., Ouzzani, M., Papotti, P., Tang, N., and Ye, Y. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD Conference*, pp. 1247–1261. ACM, 2015a.

Chu, X., Morcos, J., Ilyas, I. F., Ouzzani, M., Papotti, P., Tang, N., and Ye, Y. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp. 1247–1261, 2015b.

Chu, X., Ilyas, I. F., and Koutris, P. Distributed data deduplication. *Proceedings of the VLDB Endowment*, 9(11): 864–875, 2016a.

Chu, X., Ilyas, I. F., Krishnan, S., and Wang, J. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 international conference on management of data*, pp. 2201–2206, 2016b.

CloudLab. https://www.cloudlab.us/, 2018.

dee dee. Fitness Club Dataset, 2023. URL https://www.kaggle.com.

Dixit, A. A. and Kolaitis, P. G. A sat-based system for consistent query answering. In *SAT*, volume 11628 of *Lecture Notes in Computer Science*, pp. 117–135. Springer, 2019.

Elgiriyewithana, N. Credit Card Dataset, 2023. URL https://www.kaggle.com.

Fan, Z., Koutris, P., Ouyang, X., and Wijsen, J. LinCQA: Faster consistent query answering with linear time guarantees. *Proc. ACM Manag. Data*, 1(1):38:1–38:25, 2023.

fedesoriano. star_classification Dataset, 2023. URL https://www.kaggle.com.

Feng, W. and Deng, D. Allign: Aligning all-pair near-duplicate passages in long texts. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 541–553, 2021.

Geiping, J., Fowl, L., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Heidari, A., McGrath, J., Ilyas, I. F., and Rekatsinas, T. Holodetect: Few-shot learning for error detection. In *Proceedings of the 2019 International Conference on Management of Data*, pp. 829–846, 2019.

Ilyas, I. F. and Rekatsinas, T. Machine learning and data cleaning: Which serves the other? *ACM Journal of Data and Information Quality (JDIQ)*, 14(3):1–11, 2022.

Kalutarage, H. K., Nguyen, H. N., and Shaikh, S. A. Towards a threat assessment framework for apps collusion. *Telecommunication systems*, 66(3):417–430, 2017.

Karlaš, B., Li, P., Wu, R., Gürel, N. M., Chu, X., Wu, W., and Zhang, C. Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions. *arXiv:2005.05117*, 2020.

Khalfioui, A. A. E., Joertz, J., Labeeuw, D., Staquet, G., and Wijsen, J. Optimization of answer set programs for consistent query answering by means of first-order rewriting. In *CIKM*, pp. 25–34. ACM, 2020.

Koutris, P. and Suciu, D. A dichotomy on the complexity of consistent query answering for atoms with simple keys. *arXiv:1212.6636*, 2012.

Koutris, P. and Wijsen, J. The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints. In *PODS*, 2015.

Koutris, P. and Wijsen, J. Consistent query answering for primary keys and conjunctive queries with negated atoms. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 209–224, 2018a.

Koutris, P. and Wijsen, J. Consistent query answering for primary keys in logspace. *arXiv preprint arXiv:1810.03386*, 2018b.

Koutris, P., Ouyang, X., and Wijsen, J. Consistent query answering for primary keys on path queries. In Libkin, L., Pichler, R., and Guagliardo, P. (eds.), *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pp. 215–232. ACM, 2021. doi: 10.1145/3452021.3458334. URL https://doi.org/10.1145/3452021.3458334.

Koutris, P., Ouyang, X., and Wijsen, J. Consistent query answering for primary keys on rooted tree queries. *Proc. ACM Manag. Data*, 2(2), may 2024. doi: 10.1145/3651139. URL https://doi.org/10.1145/3651139.

Krishnan, S., Wang, J., Wu, E., Franklin, M. J., and Goldberg, K. Activeclean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, 2016.

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

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, G., Wang, J., Zheng, Y., and Franklin, M. J. Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2296–2319, 2016.

Libkin, L. Incomplete information and certain answers in general data models. In *PODS*, 2011.

Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.

Mahdavi, M., Abedjan, Z., Castro Fernandez, R., Madden, S., Ouzzani, M., Stonebraker, M., and Tang, N. Raha: A configuration-free error detection system. In *Proceedings of the 2019 International Conference on Management of Data*, pp. 865–882, 2019.

Manna, M., Ricca, F., and Terracina, G. Taming primary key violations to query large inconsistent data via ASP. *Theory Pract. Log. Program.*, 15(4-5):696–710, 2015.

Marileo, M. C. and Bertossi, L. E. The consistency extractor system: Answer set programs for consistent query

answering in databases. *Data Knowl. Eng.*, 69(6):545–572, 2010.

Nguyen, T. A. and Tran, A. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

Peng, J., Wu, W., Lockhart, B., Bian, S., Yan, J. N., Xu, L., Chi, Z., Rzeszotarski, J. M., and Wang, J. Dataprep. eda: task-centric exploratory data analysis for statistical modeling in python. In *SIGMOD*, pp. 2271–2280, 2021.

Picado, J., Davis, J., Termehchy, A., and Lee, G. Y. Learning over dirty data without cleaning. In Maier, D., Pottinger, R., Doan, A., Tan, W., Alawini, A., and Ngo, H. Q. (eds.), *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pp. 1301–1316. ACM, 2020. doi: 10.1145/3318464.3389708. URL https://doi.org/10.1145/3318464.3389708.

Ramoni, M. and Sebastiani, P. Robust bayes classifiers. *Artificial Intelligence*, 125(1-2):209–226, 2001.

Razaque, F., Soomro, N., Shaikh, S. A., Soomro, S., Samo, J. A., Kumar, N., and Dharejo, H. Using naïve bayes algorithm to students' bachelor academic performances analysis. In *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pp. 1–5. IEEE, 2017.

Rekatsinas, T., Chu, X., Ilyas, I. F., and Ré, C. Holoclean: Holistic data repairs with probabilistic inference. *Proc. VLDB Endow.*, 10(11):1190–1201, 2017.

Rezig, E. K., Ouzzani, M., Aref, W. G., Elmagarmid, A. K., Mahmood, A. R., and Stonebraker, M. Horizon: Scalable dependency-driven data cleaning. *Proc. VLDB Endow.*, 14(11):2546–2554, 2021.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pp. 98–105. Citeseer, 1998.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018.

Shen, Y., Li, Y., Zheng, H.-T., Tang, B., and Yang, M. Enhancing ontology-driven diagnostic reasoning with a symptom-dependency-aware naïve bayes classifier. *BMC bioinformatics*, 20:1–14, 2019.

Shi, Z., Wang, Y., Zhang, H., Yi, J., and Hsieh, C.-J. Fast certified robust training with short warmup. *NeurIPS*, 2021.

Souri, H., Fowl, L., Chellappa, R., Goldblum, M., and Goldstein, T. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *Advances in Neural Information Processing Systems*, 35: 19165–19178, 2022.

Steinhardt, J., Koh, P. W. W., and Liang, P. S. Certified defenses for data poisoning attacks. *NIPS*, 2017.

Trushkowsky, B., Kraska, T., Franklin, M. J., and Sarkar, P. Crowdsourced enumeration queries. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 673–684. IEEE, 2013.

Turner, A., Tsipras, D., and Madry, A. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

Veni, S. and Srinivasan, A. Defect classification using naïve bayes classification. *Interbational Journal of Applied Engineering Research*, 12(22):12693–12700, 2017.

Vos, D. and Verwer, S. Efficient training of robust decision trees against adversarial examples. In *ICML*, 2021.

Wan, C. Predicting the effect of genes on longevity with novel hierarchical dependency-constrained tree augmented naïve bayes classifiers. In *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 926–929. IEEE, 2023.

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

Wang, Z., Zuo, C., and Deng, D. Txtalign: efficient near-duplicate text alignment search via bottom-k sketches for plagiarism detection. In *Proceedings of the 2022 International Conference on Management of Data*, pp. 1146–1159, 2022.

Wu, R., Zhang, A., Ilyas, I., and Rekatsinas, T. Attention-based learning for missing data imputation in holoclean. *Proceedings of Machine Learning and Systems*, 2:307–325, 2020.

Xiong, T., Bao, Y., Zhao, P., and Wang, Y. Covariance estimation and its application in large-scale online controlled experiments. *arXiv preprint arXiv:2108.02668*, 2021.

Yan, J. N., Schulte, O., Zhang, M., Wang, J., and Cheng, R. Scoded: Statistical constraint oriented data error detection. In *SIGMOD*, pp. 845–860, 2020.

Yang, Y., Liu, T. Y., and Mirzasoleiman, B. Not all poisons are created equal: Robust training against data poisoning. In *International Conference on Machine Learning*, pp. 25154–25165. PMLR, 2022.

Yu, J., Xuan, Z., Feng, X., Zou, Q., and Wang, L. A novel collaborative filtering model for lncrna-disease association prediction based on the naïve bayesian classifier. *BMC bioinformatics*, 20:1–13, 2019.

Zhen, C., Chabada, A. S., and Termehchy, A. When can we ignore missing data in model training? In *Proceedings of the Seventh Workshop on Data Management for End-to-End Machine Learning, DEEM 2023, Seattle, WA, USA, 18 June 2023*, pp. 4:1–4:4. ACM, 2023. doi: 10.1145/3595360.3595854. URL https://doi.org/10.1145/3595360.3595854.

Zhou, L., Chen, J., Das, A., Min, H., Yu, L., Zhao, M., and Zou, J. Serving deep learning models with deduplication from relational databases. *arXiv preprint arXiv:2201.10442*, 2022.

Zhu, C., Huang, W. R., Li, H., Taylor, G., Studer, C., and Goldstein, T. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*, pp. 7614–7623. PMLR, 2019.

X	Y	
a	b	
c	b	l^*
a	d	
a	$\square$	

X	Y	
a	b	
b	c	
$\square$	$\perp_4$	l
$\square$	$\perp_5$	
$\perp_6$	$\perp_7$	

(a) An incomplete dataset $\mathcal{D}^{\square}$

X	Y	
a	b	
c	b	l^*
a	d	
a	$\boxed{b}$	

X	Y	
a	b	
b	c	
$\boxed{a}$	$\perp_1$	l
$\boxed{b}$	$\perp_2$	
$\perp_3$	$\perp_4$	

(b) A possible world $\mathcal{D}$ of $\mathcal{D}^{\square}$.

X	Y	
a	b	
c	b	l^*
a	d	
a	$\boxed{d}$	

X	Y	
a	b	
b	c	
$\boxed{a}$	$\perp_1$	l
$\boxed{a}$	$\perp_2$	
$\perp_3$	$\perp_4$	

(c) An extreme possible world $\mathcal{D}^*$ of $\mathcal{D}^{\square}$.

Figure 5: The incomplete data $\mathcal{D}^{\square}$ has (among others) two possible worlds $\mathcal{D}$ and $\mathcal{D}^*$ shown in (b) and (c) respectively.

A. Missing Details in Section 3

A.1. Proof of Lemma 3.1

Proof. It is easy to see that $\boldsymbol{t}$ is certifiably robust if there exists some label l such that for any label $l' \neq l$, $S^{\downarrow}_{\mathcal{D}^{\square}}(l \mid \boldsymbol{t}) > S^{\uparrow}_{\mathcal{D}^{\square}}(l' \mid \boldsymbol{t})$. In this case, the label l will always be predicted for $\boldsymbol{t}$. Indeed, for any possible world $\mathcal{D}$ of $\mathcal{D}^{\square}$, we have

$$S_{\mathcal{D}}(l \mid \boldsymbol{t}) \geq S^{\downarrow}_{\mathcal{D}^{\square}}(l \mid \boldsymbol{t}) > S^{\uparrow}_{\mathcal{D}^{\square}}(l' \mid \boldsymbol{t}) \geq S_{\mathcal{D}}(l' \mid \boldsymbol{t}),$$

and thus $f_{\mathcal{D}}(\boldsymbol{t}) = l$. To help the readers understand the intuition, we give an example in Appendix A.2.

The other direction holds somewhat nontrivially: if $\boldsymbol{t}$ is certifiably robust, then we claim that there must be a label l such that for any label $l' \neq l$, $S^{\downarrow}_{\mathcal{D}^{\square}}(l \mid \boldsymbol{t}) > S^{\uparrow}_{\mathcal{D}^{\square}}(l' \mid \boldsymbol{t})$. Assume that $\boldsymbol{t}$ is certifiably robust. Let $\mathcal{D}$ be a possible world of $\mathcal{D}^{\square}$, and let $l = f_{\mathcal{D}}(\boldsymbol{t})$. Suppose for contradiction that there is a label $l' \neq l$ such that

$$S^{\downarrow}_{\mathcal{D}^{\square}}(l \mid \boldsymbol{t}) \leq S^{\uparrow}_{\mathcal{D}^{\square}}(l' \mid \boldsymbol{t}).$$

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be possible worlds of $\mathcal{D}^{\square}$ such that

$$S_{\mathcal{D}_1}(l \mid \boldsymbol{t}) = S^{\downarrow}_{\mathcal{D}^{\square}}(l \mid \boldsymbol{t})$$

and

$$S_{\mathcal{D}_2}(l' \mid \boldsymbol{t}) = S^{\uparrow}_{\mathcal{D}^{\square}}(l' \mid \boldsymbol{t}).$$

Consider an arbitrary possible world $\mathcal{D}^*$ of $\mathcal{D}^{\square}$ that contains all data points in $\mathcal{D}_1$ with label l, all data points in $\mathcal{D}_2$ with label l'. It is easy to verify that $S_{\mathcal{D}^*}(l \mid \boldsymbol{t}) = S_{\mathcal{D}_1}(l \mid \boldsymbol{t})$ and $S_{\mathcal{D}^*}(l' \mid \boldsymbol{t}) = S_{\mathcal{D}_2}(l' \mid \boldsymbol{t})$. We then have

$$\begin{aligned} S_{\mathcal{D}^*}(l \mid \boldsymbol{t}) &= S_{\mathcal{D}_1}(l \mid \boldsymbol{t}) = S^{\downarrow}_{\mathcal{D}^{\square}}(l \mid \boldsymbol{t}) \\ &\leq S^{\uparrow}_{\mathcal{D}^{\square}}(l' \mid \boldsymbol{t}) = S_{\mathcal{D}_2}(l' \mid \boldsymbol{t}) = S_{\mathcal{D}^*}(l' \mid \boldsymbol{t}) \end{aligned}$$

and therefore $f_{\mathcal{D}^*}(\boldsymbol{t}) \neq l$, a contradiction to that $\boldsymbol{t}$ is certifiably robust for NBC over $\mathcal{D}^{\square}$. $\square$

A.2. Additional Examples

Example A.1. Consider the incomplete dataset $\mathcal{D}^{\square}$ in Figure 5(a) and a test point $\boldsymbol{t} = (a, b)$. For a possible world $\mathcal{D}$ of $\mathcal{D}^{\square}$, we have that

$$S_{\mathcal{D}}(l^* \mid \boldsymbol{t}) = \frac{4}{9} \cdot \frac{3}{4} \cdot \frac{3}{4} \quad \text{and} \quad S_{\mathcal{D}}(l \mid \boldsymbol{t}) = \frac{5}{9} \cdot \frac{2}{5} \cdot \frac{1}{5}.$$

For another possible world $\mathcal{D}^*$ of $\mathcal{D}^{\square}$, we have

$$S_{\mathcal{D}^*}(l^* \mid \boldsymbol{t}) = \frac{4}{9} \cdot \frac{3}{4} \cdot \frac{2}{4} \quad \text{and} \quad S_{\mathcal{D}^*}(l \mid \boldsymbol{t}) = \frac{5}{9} \cdot \frac{3}{5} \cdot \frac{1}{5}.$$

Among all possible worlds of $\mathcal{D}^{\square}$, $\mathcal{D}^*$ has the lowest possible support value $S_{\mathcal{D}}(l^* \mid \boldsymbol{t})$ and the highest support value $S_{\mathcal{D}}(l \mid \boldsymbol{t})$ for l. But still,

$$S_{\mathcal{D}^*}(l^* \mid \boldsymbol{t}) > S_{\mathcal{D}^*}(l \mid \boldsymbol{t}),$$

so l^* will always be predicted regardless of which possible world of $\mathcal{D}^{\square}$ NBC is trained on.

B. Missing Details in Section 4

B.1. Proof of Lemma 4.2

Proof. We first establish the following claim.

Lemma B.1. *Let D be a complete dataset and t a test point. Let $l^* = f_D(t)$. Let k be an integer. Then the following statements are equivalent:*

1. *there exists an incomplete dataset D^+ with k missing cells for which t is certifiably non-robust; and*
2. *there exists a label $l \neq l^*$ and a complete dataset D' obtained by altering k cells in D such that $f_{D'}(t) = l$.*

Proof. If (1) holds, let $\mathcal{D}^\dagger$ be a solution $\mathcal{D}^\dagger$ for CR-NaiveBayes† on input $(\mathcal{D}, \boldsymbol{t})$ with k altered cells. Since $\mathcal{D}$ is a possible world of $\mathcal{D}^\dagger$, then there exists a possible world $\mathcal{D}'$ of $\mathcal{D}^\dagger$ such that $l = f_{\mathcal{D}'}(\boldsymbol{t}) \neq l^*$, and therefore

$$S_{\mathcal{D}'}(l \mid \boldsymbol{t}) > S_{\mathcal{D}'}(l^* \mid \boldsymbol{t}).$$

If (2) holds, there exists some label $l \neq l^*$, and a complete dataset $\mathcal{D}'$ obtained by altering k cells in $\mathcal{D}$ so that

$$S_{\mathcal{D}'}(l \mid \boldsymbol{t}) > S_{\mathcal{D}'}(l^* \mid \boldsymbol{t}).$$

Hence $f_{\mathcal{D}'}(\boldsymbol{t}) \neq l^*$. Consider the incomplete dataset $\mathcal{D}^\dagger$ obtained by setting all the altered cells in $\mathcal{D}'$ to `NULL`. Then $\boldsymbol{t}$ is certifiably non-robust, since for the two possible worlds $\mathcal{D}$ and $\mathcal{D}'$, $f_{\mathcal{D}'}(\boldsymbol{t}) \neq l^*$ and $f_{\mathcal{D}}(\boldsymbol{t}) = l^*$. □

To show that Lemma 4.2 holds, note that the two items in Claim B.1 share the same integer parameter k, and thus the number of missing cells k for CR-NaiveBayes† is minimized simultaneously as we minimize the number of altered cells for AlterPrediction. □

B.2. Proof of Observations O1, O2, and O3

All three observations can also be formally explained from Equation (4). Suppose that $\Pr(x_j \mid l)_{\mathcal{D}_i}$ is the smallest among all $1 \leq j \leq d$. If we apply A2 to decrease $\Pr(x_j \mid l^*)_{\mathcal{D}_i}$ in $\mathcal{D}_i$ and obtain $\mathcal{D}_{i+1}$, then $\Pr(x_j \mid l^*)_{\mathcal{D}_{i+1}}$ is still the smallest among all $1 \leq j \leq d$, and thus the reduction in $\Pr(\boldsymbol{t} \mid l^*)_{\mathcal{D}_i}$ remains the same since we always decrease the same value at the same attribute. However, if we apply A1 to increase $\Pr(x_j \mid l)_{\mathcal{D}_i}$ in $\mathcal{D}_i$ and obtain $\mathcal{D}_{i+1}$, then $\Pr(x_j \mid l)_{\mathcal{D}_{i+1}}$ may not be the smallest among all $1 \leq j \leq d$. If it still is, then the next increase in $\Pr(\boldsymbol{t} \mid l)_{\mathcal{D}_{i+1}}$ would be the same, or otherwise the increase in $\Pr(\boldsymbol{t} \mid l)_{\mathcal{D}_i}$ would grow. Since $l \neq l^*$, the increase in $\Pr(\boldsymbol{t} \mid l)_{\mathcal{D}_i}$ by applying A1 does not change the value of $\Pr(\boldsymbol{t} \mid l^*)_{\mathcal{D}_i}$, and thus the increase only depends on the number of times that A1 has already been applied to obtain $\mathcal{D}_i$. A similar argument also holds for A2, and thus O3 holds.

B.3. Proof of Lemma 4.3

Proof. By definition, we have $\delta_1^+ \leq \delta_2^+ \leq \dots$.

We are now ready to prove our key result: Δ_k can *always* be minimized by applying only A1 or A2 to the orignal dataset $\mathcal{D}$, or equivalently,

To show Equation (5), assume that $\mathcal{D}_k$ is obtained by applying A1 i^+ times and applying A2 i^- times, where $k = i^+ + i^-$. Then,

$$\Delta_k = \Delta_0 - i^- \cdot \delta^- - \sum_{j \leq i^+} \delta_j^+.$$

Let $k^* \in \{0, 1, 2, \dots\}$ be the largest such that

$$k^* \cdot \delta^- \geq \sum_{j \leq k^*} \delta_j^+.$$

Then it must be that $\delta_{k^*+1}^+ > \delta^-$.

If $i^+ \leq k^*$, then we have

$$\begin{aligned}\Delta_k &= \Delta_0 - i^- \cdot \delta^- - \sum_{j \leq i^+} \delta_j^+ \\ &\geq \Delta_0 - (i^- + i^+) \cdot \delta^- = \Delta_0 - k \cdot \delta^-\end{aligned}$$

and if $i^+ > k^*$, we have for any $j \geq i^+$, $\delta_j^+ \geq \delta_{k^*+1}^+ > \delta^-$, and thus

$$\begin{aligned}\Delta_k &= \Delta_0 - i^- \cdot \delta^- - \sum_{j=1}^{i^+} \delta_j^+ \\ &\geq \Delta_0 - \sum_{j=i^++1}^{i^++i^-} \delta_j^+ - \sum_{j=1}^{i^+} \delta_j^+ = \Delta_0 - \sum_{j \leq k} \delta_j^+\end{aligned}$$

as desired. □

B.4. A Heuristic Algorithm for Poisoning Multiple Test Points

Algorithm 4: CR-NaiveBayes†-Multiple

Input: A complete dataset $\mathcal{D}$, k test points $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_k$
Output: An incomplete dataset $\mathcal{D}^\dagger$ obtained by setting some cells in $\mathcal{D}$ to `NULL` such that every $\boldsymbol{t}_i$ is not certifiably-robust for NBC

```
S ← ∅
foreach i = 1, 2, ... k do
    D_i^† ← CR-NaiveBayes†-Single(D, t_i)
    add the missing cells in D_i^† to S
D^† ← set every cells in S to NULL in D
return D^†
```

B.5. Proof of Theorem 4.4

Proof. To show membership in **NP**, note that we may guess an incomplete dataset $\mathcal{D}^\square$ from the input complete dataset $\mathcal{D}$, and verify in polynomial time that all test points are certifiably non-robust over $\mathcal{D}^\square$, using Algorithm 1.

For **NP**-hardness, we present a reduction from the VertexCover problem on d-regular graphs: Given a d-regular graph G with vertex set V and edge set E, in which all vertices have degree d, find a set $S \subseteq V$ of minimum size such that every edge in E is adjacent to some vertex in S.

Let $V = \{1, 2, \ldots, n\}$ and $m = |E|$. Without loss of generality, we assume that G is not a clique. By the Brook's theorem (Brooks, 1941), the d-regular graph G is d-colorable. Let $\chi : V \to \{A_1, A_2, \ldots, A_d\}$ be a d-coloring of G, which can be found in linear time.

Constructing m test points. We first construct m test points on attributes $A_1, A_2, \ldots, A_d$ as follows: For each edge $\{u, v\} \in E$, introduce a test point $\boldsymbol{t}_{u,v}$ with values u and v at attributes $\chi(u)$ and $\chi(v)$, and a fresh integer value for all other attribute A_k where $k \in \{1, 2, \ldots, d\} \setminus \{\chi(u), \chi(v)\}$.

It is easy to see that for two distinct edges $e = \{u, v\}$, $e' = \{u', v'\} \in E$, if e and e' do not share vertices, then $\boldsymbol{t}_{u,v}$ and $\boldsymbol{t}_{u',v'}$ do not agree on any attribute; and if they share a vertex w, then $\boldsymbol{t}_{u,v}$ and $\boldsymbol{t}_{u',v'}$ would agree on the attribute $\chi(w)$ with value w.

Note that to construct the test points, we used $n + (d-2)m$ domain values, n for each vertex V and $(d-2)m$ for all the fresh constants.

Constructing a clean dataset. For an attribute A_k, a domain value u and a label l, we denote $p(A_k, u, l)$ as a datapoint

$$p(A_k, u, l) := (\square, \cdots, u, \cdots, \square),$$

with label l, in which the attribute A_k has domain value u, and each $\square$ denotes a fresh domain value.

Let $M = n + (1 + n/d)^3 + 1$ and $N = n + (d-2)m \cdot M$. We construct a dataset $\mathcal{D}$ of $2N$ points, in which N points have label l_1, and N points have label l_2 as follows.

- Datapoints with label l_1: for each domain value u at attribute A_k, if $u \in V$, we introduce $p(A_k, u, l_1)$ once (and in this case, $A_k = \chi(u)$); or otherwise, we introduce $p(A_k, u, l_1)$ M times. It is easy to see that there are $N = n+(d-2)m \cdot M$ datapoints with label l_1.
- Datapoints with label l_2: for each domain value u at attribute A_k, we introduce $p(A_k, u, l_2)$ once, and we introduce $N - M$ fresh datapoints of the form
$$(\square, \square, \cdots, \square).$$

This construction can be done in polynomial time of m and n. By construction, for each testpoint $\boldsymbol{t}_{u,v} = (c_1, c_2, \ldots, c_d)$, if $c_k \in V$, then the domain value c_k occurs exactly once as a domain value of A_k, or otherwise M times among all datapoints with label l_1; and c_k occurs exactly once as a domain value of A_k among all datapoints with label l_2.

Hence for each test point $\boldsymbol{t}_{u,v}$, $f_{\mathcal{D}}(\boldsymbol{t}_{u,v}) = l_1$, by noting that

$$\begin{aligned}
\Pr(l_1 \mid \boldsymbol{t}_{u,v})_{\mathcal{D}} &= \frac{1}{\Pr(\boldsymbol{t}_{u,v})_{\mathcal{D}}} \cdot \frac{1 \cdot 1 \cdot M^{d-2}}{N^d} \cdot \frac{N}{2N} \\
&> \frac{1}{\Pr(\boldsymbol{t}_{u,v})_{\mathcal{D}}} \cdot \frac{1 \cdot 1 \cdot 1^{d-2}}{N^d} \cdot \frac{N}{2N} \\
&= \Pr(l_2 \mid \boldsymbol{t}_{u,v})_{\mathcal{D}}.
\end{aligned}$$

Now we argue that there is a vertex cover S of G with size at most k if and only if we may alter at most k missing values in $\mathcal{D}$ to obtain $\mathcal{D}'$ such that for each point $\boldsymbol{t}_{u,v}$,

$$\Pr(l_1 \mid \boldsymbol{t}_{u,v})_{\mathcal{D}'} < \Pr(l_2 \mid \boldsymbol{t}_{u,v})_{\mathcal{D}'},$$

which yields $f_{\mathcal{D}}(\boldsymbol{t}_{u,v}) \neq f_{\mathcal{D}'}(\boldsymbol{t}_{u,v})$.

$\boxed{\Longrightarrow}$ Let $S \subseteq V$ be a vertex cover of G. Consider the dataset $\mathcal{D}'$ obtained by altering for each $u \in S$ with $A_k = \chi(u)$, the datapoint

$$p(A_k, u, l_1) = (\square, \cdots, u, \cdots, \square)$$

into

$$(\square, \cdots, u', \cdots, \square),$$

where $u' \notin V$. For every test point $\boldsymbol{t}_{u,v}$, we argue that either u does not occur at all at attribute $\chi(u)$, or v does not occur at all at attribute $\chi(v)$ in $\mathcal{D}'$. Indeed, since S is a vertex cover, either $u \in S$ or $v \in S$, and thus either $p(\chi(u), u, l_1)$ or $p(\chi(v), v, l_1)$ is altered so that either u or v does not occur at all at attributes $\chi(u)$ or $\chi(v)$ in $\mathcal{D}'$. Hence NBC would estimate that

$$\Pr(l_1 \mid \boldsymbol{t}_{u,v})_{\mathcal{D}'} = 0 < \frac{1}{\Pr(\boldsymbol{t}_{u,v})_{\mathcal{D}'}} \cdot \frac{1}{N^d} \cdot \frac{N}{2N} = \Pr(l_2 \mid \boldsymbol{t}_{u,v})_{\mathcal{D}'},$$

as desired.

$\boxed{\Longleftarrow}$ Assume that there is a dataset $\mathcal{D}'$ with at most k alternations to $\mathcal{D}$ such that $f_{\mathcal{D}}(\boldsymbol{t}_{u,v}) = l_1$ but $f_{\mathcal{D}'}(\boldsymbol{t}_{u,v}) = l_2$ for every test point $\boldsymbol{t}_{u,v}$. It suffices to show that G has a vertex cover of size at most k. Since G must have a vertex cover of size $k \geq n$, we assume that $k < n$.

Consider an arbitrary test point $\boldsymbol{t}_{u,v} = (c_1, c_2, \ldots, c_n)$. We have that

$$\Pr(l_1 \mid \boldsymbol{t}_{u,v})_{\mathcal{D}'} < \Pr(l_2 \mid \boldsymbol{t}_{1,2})_{\mathcal{D}'}.$$

Let x_i and y_i be the absolute value of the change of frequency from $\mathcal{D}$ to $\mathcal{D}'$ of value c_i for attribute A_i in test points with labels l_1 and l_2 respectively. Then we have

$$0 \leq (1-x_i)(1-x_j) \cdot \prod_{k \in \{1,2,\ldots,d\} \setminus \{i,j\}} (M - x_k) < \prod_{1 \leq k \leq d} (1 + y_k). \tag{6}$$

We argue that without loss of generality, we can assume all x_i and y_i are nonnegative: if y_i is negative, then right-hand-side of (Eq. 6) is 0 and cannot hold; and if x_i is negative, i.e., the alternations increase the frequency of certain attributes among points with label l_1, then not performing such alterations would also preserve the inequality (Eq. 6) with less than k alterations.

Note that $x_1 + x_2 + \cdots + x_d + y_1 + y_2 + \cdots + y_d \leq k$.

For each test point $\boldsymbol{t}_{u,v}$, let $A_i = \chi(u)$ and $A_j = \chi(v)$, and we argue that we must have $x_j = 1$ or $x_j = 1$ (or both). Suppose for contradiction that $x_i = x_j = 0$. Then, we have

$$\begin{aligned}
&\prod_{k \in \{1,2,\ldots,d\} \setminus \{i,j\}} (M - x_k) \\
&= (1-x_i)(1-x_j) \cdot \prod_{k \in \{1,2,\ldots,d\} \setminus \{i,j\}} (M - x_k) \\
&< \prod_{1 \leq k \leq d} (1 + y_k) \\
&\leq (1 + \frac{y_1 + y_2 + \cdots + y_d}{d})^d \\
&\leq (1 + \frac{n}{d})^d.
\end{aligned}$$

On the other hand, we have for each $k \in \{1, 2, \ldots, d\} \setminus \{i, j\}$, $x_k \leq n$, and thus

$$M - x_k \geq M - n > (1 + \frac{n}{d})^3 \geq (1 + \frac{n}{d})^{\frac{d}{d-2}},$$

and it follows that

$$\prod_{k \in \{1,2,\ldots,d\} \setminus \{i,j\}} (M - x_k) \geq (1 + \frac{n}{d})^d,$$

which is a contradiction.

Now consider the set

$$S = \{u \mid A_k = \chi(u), x_k = 1\}.$$

The set S is a vertex cover, since for every edge $\{u, v\} \in E$, if the test point $\boldsymbol{t}_{u,v}$ has $A_i = \chi(u)$ and $A_j = \chi(v)$, we must have $x_i = 1$ or $x_j = 1$, and thus either $u \in S$ or $v \in S$. Since each $x_k = 1$ corresponds to an alteration, the size of S is at most k.

The proof concludes by noting that both $\mathcal{D}$ and $\mathcal{D}'$ are possible worlds of the incomplete dataset $\mathcal{D}^\dagger$ obtained by setting the altered cells in $\mathcal{D}'$ from $\mathcal{D}$ as missing cells. □

B.6. Additional Examples

Example B.2. Let us consider the dataset $\mathcal{D}$ in Figure 6(a) and a test point $\boldsymbol{t} = (a, b)$. We have that.

$$S_{\mathcal{D}}(l^* \mid \boldsymbol{t}) = \frac{4}{9} \cdot \frac{2}{4} \cdot \frac{3}{4} \quad \text{and} \quad S_{\mathcal{D}}(l \mid \boldsymbol{t}) = \frac{5}{9} \cdot \frac{1}{5} \cdot \frac{2}{5},$$

and thus $S_{\mathcal{D}}(l^* \mid \boldsymbol{t}) > S_{\mathcal{D}}(l \mid \boldsymbol{t})$.

- To increase $S_{\mathcal{D}}(l \mid \boldsymbol{t})$, we can (1) alter the value $\perp_2$ in attribute X into a to obtain a dataset $\mathcal{D}_+$, or (2) alter the value $\perp_3$ in attribute Y into b to obtain a dataset $\mathcal{D}'_+$ shown in Figure 6(b). Note that

$$S_{\mathcal{D}_+}(l \mid \boldsymbol{t}) = \frac{5}{9} \cdot \frac{2}{5} \cdot \frac{2}{5} \quad \text{and} \quad S_{\mathcal{D}'_+}(l \mid \boldsymbol{t}) = \frac{5}{9} \cdot \frac{1}{5} \cdot \frac{3}{5},$$

X	Y	
a	b	
a	$\perp_1$	l^*
$\perp_2$	b	
$\perp_3$	b	

X	Y	
a	b	
$\perp_1$	b	
$\perp_2$	$\perp_3$	l
$\perp_4$	$\perp_5$	
$\perp_6$	$\perp_7$	

(a) A possible world $\mathcal{D}$

X	Y	
a	b	
a	$\perp_1$	l^*
$\perp_2$	b	
$\perp_3$	b	

X	Y	
a	b	
$\cancel{\perp_1}\ a$	b	
$\perp_2$	$\cancel{\perp_3}\ b$	l
$\perp_4$	$\perp_5$	
$\perp_6$	$\perp_7$	

(b) $\mathcal{D}_+$ and $\mathcal{D}'_+$, obtained by altering $\perp_1$ and $\perp_3$ to a and b in $\mathcal{D}$ respectively.

X	Y	
a	b	
$\cancel{a}\ c_1$	$\perp_1$	l^*
$\perp_2$	b	
$\perp_3$	$\cancel{b}\ c'_1$	

X	Y	
a	b	
$\perp_1$	b	
$\perp_2$	$\perp_3$	l
$\perp_4$	$\perp_5$	
$\perp_6$	$\perp_7$	

(c) $\mathcal{D}_-$ and $\mathcal{D}'_-$, obtained by altering a and b to c_1 and c'_1 in $\mathcal{D}$ respectively.

Figure 6: For the test point $\boldsymbol{t} = (a, b)$, altering $\perp_1$ to a in $\mathcal{D}_+$ (among other possible ways) achieves the most increase in $S_{\mathcal{D}}(l \mid \boldsymbol{t})$, and altering a to c_1 in $\mathcal{D}_-$ (among other possible ways) achieves the most decrease in $S_{\mathcal{D}}(l^* \mid \boldsymbol{t})$.

performing (1) achieves a bigger reduction in $S_{\mathcal{D}}(l^* \mid \boldsymbol{t})$ than (2), since for any test point $x = (x_1, x_2)$ in $\mathcal{D}$, $\Pr(x_1 = a \mid l)_{\mathcal{D}} < \Pr(x_2 = b \mid l)_{\mathcal{D}}$.

- To decrease $S_{\mathcal{D}}(l^* \mid \boldsymbol{t})$, we can (1) alter the value a in attribute X into some c_1 to obtain a dataset $\mathcal{D}_-$, or (2) alter the value b in attribute Y into some c'_1 to obtain a dataset $\mathcal{D}'_-$ shown in Figure 6(c). Note that

$$S_{\mathcal{D}_-}(l^* \mid \boldsymbol{t}) = \frac{4}{9} \cdot \frac{1}{4} \cdot \frac{3}{4} \quad \text{and} \quad S_{\mathcal{D}'_-}(l^* \mid \boldsymbol{t}) = \frac{4}{9} \cdot \frac{2}{4} \cdot \frac{2}{4},$$

performing (1) achieves a bigger reduction in $S_{\mathcal{D}}(l^* \mid \boldsymbol{t})$ than (2), since for any test point $x = (x_1, x_2)$ in $\mathcal{D}$, $\Pr(x_1 = a \mid l^*)_{\mathcal{D}} < \Pr(x_2 = b \mid l^*)_{\mathcal{D}}$.

Example B.3. For $\mathcal{D}_+$ in Example B.2, we have

$$S_{\mathcal{D}_+}(l \mid \boldsymbol{t}) = \frac{5}{9} \cdot \frac{2}{5} \cdot \frac{2}{5}.$$

In this case, altering $\perp_2$ to a or $\perp_5$ to b achieves the largest increase in $S_{\mathcal{D}_+}(l \mid \boldsymbol{t})$, since both values have the smallest frequency. Suppose that we alter $\perp_2$ to a and obtain dataset $\mathcal{D}_{++}$. We have

$$S_{\mathcal{D}_{++}}(l \mid \boldsymbol{t}) = \frac{5}{9} \cdot \frac{3}{5} \cdot \frac{2}{5}.$$

Then, we would alter $\perp_5$ to b, and achieve

$$S_{\mathcal{D}_{+++}}(l \mid \boldsymbol{t}) = \frac{5}{9} \cdot \frac{3}{5} \cdot \frac{3}{5}.$$

Hence O1 holds in this case, since

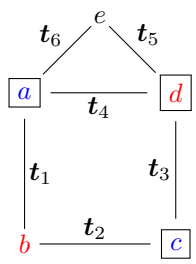
$$\begin{aligned} S_{\mathcal{D}_+}(l \mid \boldsymbol{t}) - S_{\mathcal{D}}(l \mid \boldsymbol{t}) &= S_{\mathcal{D}_{++}}(l \mid \boldsymbol{t}) - S_{\mathcal{D}_+}(l \mid \boldsymbol{t}) \\ &< S_{\mathcal{D}_{+++}}(l \mid \boldsymbol{t}) - S_{\mathcal{D}_{++}}(l \mid \boldsymbol{t}) \end{aligned}$$

For $\mathcal{D}_-$ in Example B.2, we have

$$S_{\mathcal{D}_-}(l^* \mid \boldsymbol{t}) = \frac{4}{9} \cdot \frac{1}{4} \cdot \frac{3}{4}.$$

	X	Y	Z
$\boldsymbol{t}_1$	a	b	$\perp_1$
$\boldsymbol{t}_2$	c	b	$\perp_2$
$\boldsymbol{t}_3$	c	d	$\perp_3$
$\boldsymbol{t}_4$	a	d	$\perp_4$
$\boldsymbol{t}_5$	$\perp_5$	d	e
$\boldsymbol{t}_6$	a	$\perp_6$	e

(a)



(b)

X	Y	Z	
$\boxed{a:1}$	$b:1$	$e:1$	
$\boxed{c:1}$	$\boxed{d:1}$	$\perp_1:100$	
$\perp_5:100$	$\perp_6:100$	$\perp_2:100$	l_1
$\dots$	$\dots$	$\perp_3:100$	
		$\perp_4:100$	
$a:1$	$b:1$	$e:1$	
$c:1$	$d:1$	$\perp_1:1$	
$\perp_5:1$	$\perp_6:1$	$\perp_2:1$	l_2
$\dots$	$\dots$	$\perp_3:1$	
		$\perp_4:1$	

(c)

Figure 7: The test points are shown in (a). The graph in (b) captures the correlations among the test points in (a). A dataset is shown in (c). Poisoning the minimum number of cells in (c) such that all test points in (a) are not certifiably robust can be simulated by finding a minimum vertex cover, marked by boxes, in the graph in (b).

Different from $\mathcal{D}_+$, further altering the only a in attribute X to some other value to obtain $\mathcal{D}_{--}$ still achieves the largest decrease in $S_{\mathcal{D}_-}(l^* \mid \boldsymbol{t})$, since the frequency of a is still the smallest. We have

$$S_{\mathcal{D}_{--}}(l^* \mid \boldsymbol{t}) = \frac{4}{9} \cdot \frac{0}{4} \cdot \frac{3}{4}.$$

Hence O2 holds in this case, since

$$S_{\mathcal{D}_-}(l \mid \boldsymbol{t}) - S_{\mathcal{D}}(l \mid \boldsymbol{t}) = S_{\mathcal{D}_{--}}(l \mid \boldsymbol{t}) - S_{\mathcal{D}_-}(l \mid \boldsymbol{t})$$

Example B.4. Consider the 6 test points shown in Figure 7(a). Note that since the test points $\boldsymbol{t}_1$, $\boldsymbol{t}_4$ and $\boldsymbol{t}_6$ agree on the attribute value a for attribute X, the change in the relative frequency of the attribute value a in the datasets affects the support values (and thus the predictions) to all 3 test points simultaneously. The graph in Figure 7(b) captures precisely this correlation among the test points, in which each vertex represents a value from an attribute, and each edge represents a test point containing those two values at their corresponding attributes. The 3 attributes X, Y and Z are distinguished by a 3-coloring of the graph.

Consider a complete dataset $\mathcal{D}$ containing $2N$ datapoints shown in Figure 7(c), in which N datapoints have label l_1 and N datapoints have label l_2 for some sufficiently large N. For each attribute $A \in \{X, Y, Z\}$, the entry $v : n$ with label $l \in \{l_1, l_2\}$ indicates that there are n datapoints with value v for attribute A in $\mathcal{D}$.

It is easy to see that for each test point $\boldsymbol{t}_i$ in Figure 7(a),

$$f_{\mathcal{D}}(\boldsymbol{t}_i) = l_1,$$

since the relative frequencies of each $\perp_i$ among datapoints with label l_1 is much higher than those with label l_2.

Note that if we alter $\mathcal{D}$ to decrease the frequency of value a in attribute X in label l_1, this single alteration causes the support values of l_1 for test points $\boldsymbol{t}_1$, $\boldsymbol{t}_4$ and $\boldsymbol{t}_6$ to drop to 0, and flips the predictions of all three test points to l_2. Therefore, the minimum number of alterations in the datasets corresponds to a minimum vertex cover in the graph in Figure 7(b).

C. Additional Experimental Results in Section 5

C.1. Datasets

We use ten real-world datasets from Kaggle (web, 2022a): heart (HE) (dat, 2023e), fitness-club (FC) (dee dee, 2023), fetal-health (FH) (dat, 2023d), employee (EM) (dat, 2023c), winequalityN (WQ) (dat, 2023g), company-bankruptcy (CB) (dat, 2023b), Mushroom (MR) (dat, 2023f), bodyPerformance (BP) (dat, 2023a), star-classification (SC) (fedesoriano, 2023), creditcard (CC) (Elgiriyewithana, 2023). The metadata of our datasets are summarized in Table 1.

Dataset	# Rows	# Features	# Labels
heart (HE)	918	12	2
fitness-club (FC)	1,500	8	2
fetal-health (FH)	2,126	22	3
employee (EM)	4,653	9	2
winequalityN (WQ)	6,497	13	2
Company-Bankruptcy (CB)	6,819	96	2
Mushroom (MR)	8,124	23	2
bodyPerformance (BP)	13,393	12	4
star-classification (SC)	100,000	18	3
creditcard (CC)	568,630	31	2

Table 1: Metadata of the datasets.

C.2. Additional Results for Decision Problem

In this section, we provide additional results for the decision problem in Figure 8 and Figure 9.

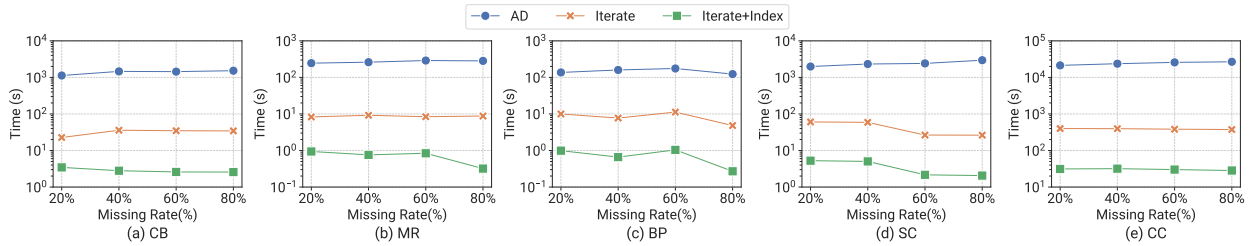


Figure 8: Decision - Running Time vs Missing Rates on Different Datasets

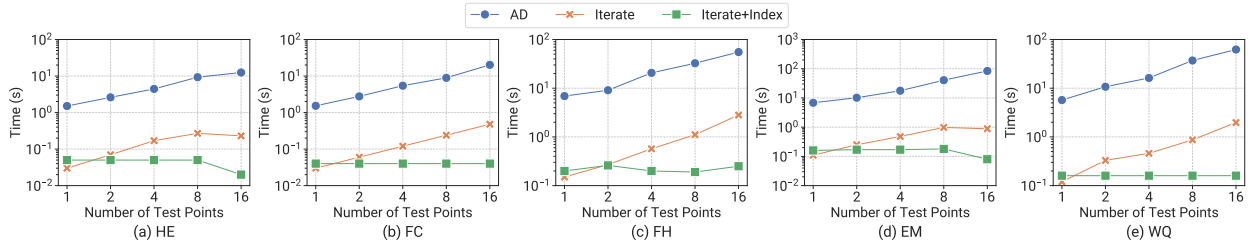


Figure 9: Decision - Running Time vs Number of Test Points on Different Datasets

C.3. Additional Results for Poisoning Problem

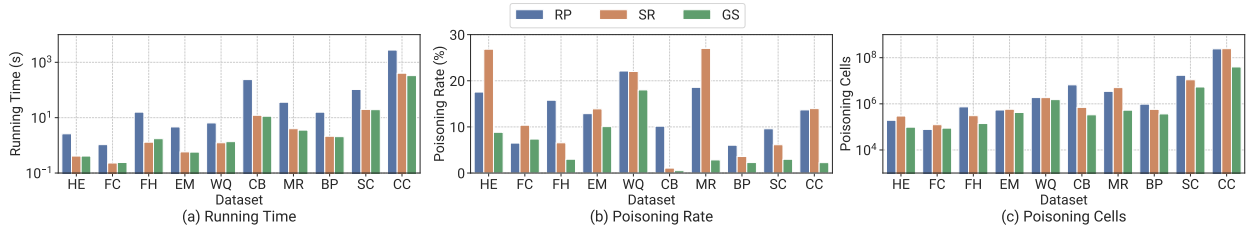


Figure 10: Multiple Points Data Poisoning—Running Time and Poisoning Rate on Different Datasets

We show the additional results for the poisoning problem in Figure 10. As for the Multiple Points Data Poisoning Problem, **GS** is more effective than **RP** and **SR** over most datasets. Note that despite **SR** and **GS** have similar running times but their

poisoning rate varies: This implies that the optimal strategies A1 and A2 mentioned in Section 4 are efficient to execute and the optimal sequence of applying A1 and A2 can minimize the number of poisoned cells drastically. Furthermore, Figure 10 shows the exact number of poisoned cells. Note that the number of poisoning cells is not small in Figure 10, which indicates that the answer to the Decision Problem can help us to accelerate the data cleaning process.