A Unified Framework for Task-Driven Data Quality Management

Tianhao Wang
Harvard University
tianhaowang@fas.harvard.edu

Yi Zeng
Virginia Tech
yizeng@vt.edu

Ming Jin
Virginia Tech
jinming@vt.edu

Ruoxi Jia
Virginia Tech
ruoxijia@vt.edu

Abstract

High-quality data is critical to train performant *Machine Learning* (ML) models, highlighting the importance of *Data Quality Management* (DQM). Existing DQM schemes often cannot satisfactorily improve ML performance because, by design, they are oblivious to downstream ML tasks. Besides, they cannot handle various data quality issues (especially those caused by adversarial attacks) and have limited applications to only certain types of ML models. Recently, data valuation approaches (e.g., based on the Shapley value) have been leveraged to perform DQM; yet, empirical studies have observed that their performance varies considerably based on the underlying data and training process. In this paper, we propose a *task-driven, multi-purpose, model-agnostic* DQM framework, DATASIFTER, which is optimized towards a given downstream ML task, capable of effectively removing data points with various defects, and applicable to diverse models. Specifically, we formulate DQM as an optimization problem and devise a scalable algorithm to solve it. Furthermore, we propose a theoretical framework for comparing the worst-case performance of different DQM strategies. Remarkably, our results show that the popular strategy based on the Shapley value may end up choosing the worst data subset in certain practical scenarios. Our evaluation shows that DATASIFTER achieves and most often significantly improves the state-of-the-art performance over a wide range of DQM tasks, including backdoor, poison, noisy/mislabel data detection, data summarization, and data debiasing.

1 Introduction

High-quality data is a critical enabler for high-quality *Machine Learning* (ML) applications. However, due to inevitable errors, bias, and adversarial attacks occurring during the data generation and collection processes, real-world datasets often suffer various defects that can adversely impact the learned ML models. Hence, *Data Quality Management* (DQM) has become an essential prerequisite step for building ML applications.

DQM has been extensively studied by the database community in the past. Early works [1, 2, 3] consider DQM as a standalone exercise without considering its connection with downstream ML applications. Studies have shown that such ML-oblivious DQM may not necessarily improve model performance [1]; worse yet, it may even degrade model performance [4]. More recent work started to tailor the DQM strategies to specific ML applications [5, 6, 7]. Still, they apply only to simple models such as convex models, nearest neighbors, and specific data quality issues such as outlier detection. In parallel with these research efforts, the ML community has intensively investigated

techniques focused on addressing a broad variety of data quality issues, such as adversarial [8, 9] and mislabeled [10, 11] data detection, anomaly detection [12], dataset debiasing [13, 14, 15]. However, a DQM scheme that can comprehensively remedy various types of data defects is still lacking.

Our paper aims to address the limitations of prior DQM schemes by developing a unified DQM framework with the following properties: (1) *multi-purpose* – to handle various data quality issues; (2) *task-driven* – to effectively utilize the information from downstream ML tasks; and (3) *model-agnostic* – to incorporate different ML models. The line of existing work closest to achieving these goals is what we will later refer to as data valuation-based approaches. These approaches first adopt some importance quantification metric, e.g., influence functions [10], Shapley values [16, 17] and least cores [18], to quantify each training point according to the contributions toward the training processes, then decide which data to retain or remove based on the valuation rankings. While some existing data valuation-based approaches satisfy the three desiderata, empirical studies have shown that their performance varies considerably based on the underlying data and the learning process. Moreover, there is no clear understanding of such performance variation or formal characterization of the worst-case performance.

In this paper, we start by formulating various DQM tasks into optimal data selection problems. The goal is to find a subset of data points that achieve the highest performance for a given ML task. We propose DATASIFTER, a multi-purpose, task-driven, model-agnostic DQM framework that first learns a data utility model from a small validation set, then selects the subset of data points by optimizing the acquired utility model. With the acquired data utility model, DATASIFTER can go beyond the functionalities offered by existing DQM schemes and further estimate the utility of selected data points. Such information could help data analysts to decide how many data points to choose or whether there is a need to acquire new data. Furthermore, we present a novel theoretical framework based on domination analysis which allows one to rigorously analyze the worst-case performance of data valuation-based DQM approaches and compare them with our approach. Specifically, we show that data valuation-based DQM approaches have unsatisfying worst-case performance guarantees. In particular, the popular Shapley value-based approach will select the worst data in some commonly occurring scenarios. We conduct a thorough empirical study on a range of ML tasks, including adversarially perturbed data detection, noisy label/feature detection, data summarization, and data debiasing. Our experiments demonstrate that DATASIFTER achieves and most often significantly improves the state-of-the-art performance of data valuation-based approaches on various tasks.

2 Related Work

The major differences between this paper and the related works are summarized in Table 1.

Data Cleaning. Classical data cleaning methods are based on simple attributes of a dataset such as completeness [3], consistency [21], and timeliness [2]; however, these attributes may not necessarily correlate with the actual utility of data for training machine learning models. Recent works leverage the information about downstream ML tasks to guide the cleaning process. ActiveClean [5] explored task-driven data cleaning for convex models by selecting data points for human screening. BoostClean [22] sought to automate the manual cleaning by

Method Type	Multi-purpose	Task-Driven	Model-Agnostic	Est. Utility
Traditional	×	×	×	×
Data Cleaning	×	○	○	×
Perm-Shapley [19]	✓	✓	✓	×
TMC-Shapley [16]	✓	✓	✓	×
G-Shapley [16]	✓	✓	×	×
KNN-Shapley [20]	×	×	✓	×
Least Core [18]	✓	✓	✓	×
Leave-one-out [10]	✓	✓	✓	×
Infl. Func. [10]	×	✓	×	×
TracIn [11]	×	✓	×	×
DATASIFTER	✓	✓	✓	✓

Table 1: Summary of the differences between previous works with our methods (DATASIFTER). ○ means only some of the techniques in the type satisfy the property.

determining a predefined cleaning strategy from a library using boosting. AlphaClean [7] also aimed to automate the cleaning process but relied on parallelized tree search. However, those framework's efficacy and generalizability are limited by the cleaning library. Furthermore, the recursive nature of the automatic selection process constrained the use case of those methods to only small models and datasets. CPClean [6] proposed a different strategy for nearest neighbor models based on the concept of Certain Prediction. Still, CPClean is designed explicitly for SQL datasets with greedy

repairing, making it difficult to generalize to larger-scaled cases like image datasets. In summary, the state-of-the-art data cleaning methods are only applicable to certain classes of ML models and datasets. Besides, adapting those cleaning works to other domains requires manual recollection of the cleaning library or human intervention, which can be impractical in many cases.

Data Importance Quantification. One simple idea to quantify data importance is to use the leave-one-out error. [10] provides an efficient algorithm to approximate leave-one-out error for each training point. Recent works leverage credit allocation schemes originated from cooperative game theory to quantify data importance. Particularly, Shapley value has been widely used [16, 17, 20, 23, 24], as it uniquely satisfies a set of desirable axiomatic properties. More recently, [18] suggests that the Least core is also a viable alternative to Shapley value for measuring data importance. However, computing the exact Shapley and Least core values are generally NP-hard. Several approximation heuristics, such as TMC-Shapley [16], G-Shapley [16], KNN-Shapley [20], have been proposed for the Shapley value. Despite their computational advantage, they are biased in nature. On the other hand, unbiased estimators such as Permutation Sampling [19] and Group Testing [17] still require retraining models many times for any descent approximation accuracy. TracIn [11] estimates the importance by tracing the test loss change caused by a training example during the training process. The representer point method [25] captures the importance of that training point by decomposing the pre-activation prediction of a neural network into a linear combination of activations of training points. Many of the aforementioned works can only be applied to differentiable models.

3 Formalism and Algorithmic Framework

In general, DQM aims to find a subset of data points with the highest utility. We use the data utility function to characterize the mapping from a set of data points to its utility. Formally, given a dataset $\mathcal{D}$ of size n, a *data utility function* $U : 2^{\mathcal{D}} \to \mathbb{R}$ maps a set of data points $S \subseteq \mathcal{D}$ to a real number indicating the performance of the ML model trained on the set, such as test accuracy and fairness.

With the notion of the data utility function, one can abstract DQM tasks as a *data selection problem*: $\max_{|S|=k} U(S)$, where k indicates the selection budget with $0 < k < n$, which can be predetermined (e.g., based on the prior knowledge about potential data defects or computational requirements). Moreover, the DQM tasks without a specific selection budget can be reduced to a sequence of data selection problems with different values of k.

With the abstraction above, one straightforward way to optimally select data is to exhaustively evaluate $U(S)$ for all possible size-k subsets $S \subseteq \mathcal{D}$ and choose the one that achieves the highest utility. Of course, this naive algorithm requires prohibitively large computational resources because the number of utility evaluations is exponential in k, and worse yet, each evaluation of data utility function requires retraining the model. Fortunately, recent work shows that many common data utility functions can be effectively learned with a relatively small amount of samples [26] because they are "approximately" submodular [27]. The "approximate submodularity" property allows efficient maximization of data utility functions through simple greedy algorithms [28, 29, 30, 31, 32]. Hence, combining data utility function learning and greedy search enables an efficient algorithm for data selection problems.

Specifically, we extend and generalize the data utility learning and optimization technique originally proposed in [26] for active learning to DQM. The proposed DQM framework, termed DATASIFTER, proceeds in two phases: *learning* and *selection* phase.

Learning Phase. Figure 1 depicts the learning phase of the DATASIFTER, which consists of a utility sampling step and a utility model training step. In particular, we assume that we have access to a small validation set representative for potential test samples. Thus, the utility of any given subset can be estimated by feeding the model with this subset then evaluating its performance over the validation set. In the utility sampling step, we randomly sample subsets of the training set, estimate the utility of each sampled subset, and label each using its utility score. We will refer to the scored subset as *utility samples* hereinafter. To accelerate this step for large models such as deep nets, a small proxy model (such as logistic regression) can be used for approximating the utility since data utilities evaluated on deep nets and logistic regression are positively correlated, as shown in [15]. In the utility model training step, we learn a parametric model for the data utility function using the utility samples; particularly, our experiments adopt DeepSets [33] as the utility model. For a large dataset, the utility
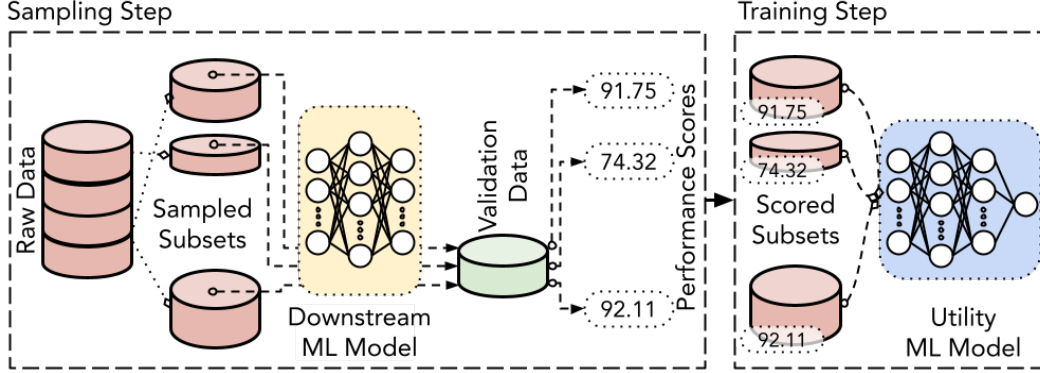
Figure 1: Overview of the learning phase, which consists of two functional steps, the utility sampling step and the training step of the utility ML model. We randomly select subsets from the raw data during the sampling step and assign a utility score on each set based on the downstream ML model's performance evaluated over a validation set. Then, we train the utility ML model that uses those scored pairs to acquire the ability to predict a performance score for a collection of data.

sampling step could be conducted on a small portion of the dataset. Our empirical studies show that the learned utility model can still extrapolate the utility for the unseen part of the dataset.

Selection Phase. We select high-quality data through optimizing the learned model for data utility functions obtained from the previous phase; specifically, we adopt a linear-time stochastic greedy algorithm [34] to perform optimization.

Clearly, DATASIFTER is an optimal solution to the data selection problem if the validation data matches the test data exactly and there are no computational constraints. In practice, despite limited validation dataset and limited computational resources, DATASIFTER is still very effective in selecting high-quality data or filtering bad data as we will show in the evaluation section. In
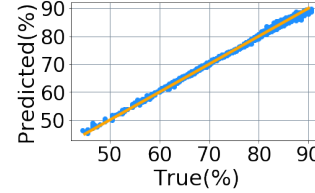

Figure 2: Predicted vs. True Utility for unseen subsets of logistic regression classifier trained on a synthetic dataset. Details are presented in supplementary materials.

addition, with the learned data utility model, DATASIFTER can provide an estimate of the utility for the selected dataset (see example in Figure 2), which will be useful for data analysts to decide the number of data points to select.

4 Worst-Case Analysis

This section presents a theoretical framework for comparing the worst-case performance between DATASIFTER and data valuation-based DQM schemes, such as *leave-one-out* (LOO), Shapley value, and Least core[1], and we assume no computational constraints.

We start by abstracting a general notion from data valuation-based DQM schemes in the literature. We call an algorithm that returns $S \subseteq \mathcal{D}$ of size k a *heuristic* to a (size-k) data selection problem on $\mathcal{D}$. The typical pattern of data valuation-based heuristics is that they first rank the data points according to their corresponding data importance metric and then prioritize the points with the highest importance scores. We will define the heuristics matching this selection pattern as *linear heuristics*.

Definition 1 (Linear heuristic). *We say $\mathcal{M}$ is a linear heuristic for data selection problem if for every instance $\mathcal{I} = (\mathcal{D}, U)$, $\mathcal{M}$ works as follows:*

    1. Assign a score $v = (v_1, \ldots, v_n)$ for every data point $i \in \mathcal{D}$.

---

[1]Least core may not be unique. In this paper, when we talk about the least core, we always refer to the least core vector that has the smallest ℓ_2 norm, following the tie-breaking rule in the original literature [18].

2. *Sort $\mathcal{D}$ in the descending order according to v and obtain sorted data sequence $\mathcal{D}'$. Certain rules are applied to break tie.*

3. *For any query of selecting k high-quality data points, return the first k data points in $\mathcal{D}'$.*

Our theoretical framework for studying the worst-case performance of data selection heuristics extends the domination analysis initially proposed in [35]. Our worst-case performance metric is *domination number*, which measures how many subsets achieve lower utility than the selected set in the worst-case scenario.

Definition 2 (Domination number). *The domination number of a heuristic $\mathcal{M}$ for the data selection problem is the maximum integer $d(n, k)$ s.t., for every problem instance $\mathcal{I} = (\mathcal{D}, U)$ on a dataset $\mathcal{D}$ of size n and utility function U, $\mathcal{M}(\mathcal{I}, k)$ produces a size-k subset $S \subseteq \mathcal{D}$ which has utility $U(S)$ no worse than at least $d(n, k)$ size-k subsets.*

The domination number is well defined for every data selection heuristic. A heuristic with a higher domination number may be a better choice than a heuristic with a smaller domination number due to the better worst-case guarantee. The best heuristic for data selection has domination number $d(n, k) = \binom{n}{k}$ for every $k \leq n$, which means that it will select the size-k data subset with the highest utility for every possible data utility function.

Clearly, assuming no computational constraints, DATASIFTER is among the best heuristics which achieve the largest possible domination number. In contrast, the following result shows that no linear heuristic is the best whenever $n \geq 3$. We will defer all proofs to Appendix.

Theorem 1. *For $n \geq 3$, there exists no linear heuristic $\mathcal{M}$ s.t. $d(n, k) = \binom{n}{k}$ for every $k \in \{1, \ldots, n\}$.*

Furthermore, we can tighten the upper bound of the domination number for data valuation-based heuristics by noticing another common property: two data points will receive the same importance score if they contribute equally to all possible subsets of the training data. This property is often referred to as *symmetry axiom* in the literature.

Definition 3 (Symmetry axiom). *We say a linear heuristic $\mathcal{M}$ satisfies symmetry axiom if its scoring mechanism satisfies: $[(\forall S \in \mathcal{D} \setminus \{i, j\})U(S \cup \{i\}) = U(S \cup \{j\})] \implies v_i = v_j$.*

The symmetry axiom may be desired for application scenarios requiring fairness, e.g., data importance scores are used to assign monetary rewards for data sharing or responsibility for ML decisions. However, for data selection, symmetry axiom may be undesirable because simply gathering high-value data points may lead to a set of redundant points. Based on this intuition, the following theorem gives an upper bound of domination number for non-trivial linear heuristics that with symmetry property.

Theorem 2. *If a linear heuristic $\mathcal{M}$ assigns different scores to different data points and satisfies symmetry axiom, then the domination number of $\mathcal{M}$ is $d(n, k) \leq c\binom{\lceil n/c \rceil}{k}$ where $c = \lfloor \frac{n}{k} \rfloor$.*
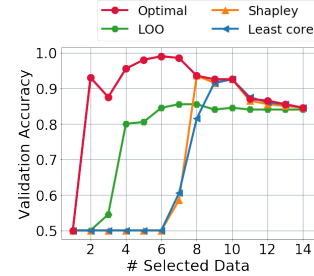

Figure 3: Results of data selection with different heuristics on a tiny dataset with natural redundancy. Dataset and implementation are detailed in the Appendix.

To better illustrate the issue raised by symmetry axiom, we evaluate the LOO, Shapley, and least core heuristic on a synthetic dataset with 15 training data points (so that we can compute the exact Shapley and least core values, as well as obtain the optimal solution for data selection problem). The utility metric is the test accuracy of a *Support Vector Machine* (SVM) classifier trained on the dataset. We simulate the natural redundancy in a dataset by replicating 5 data points three times and adding slight Gaussian noise to differentiate. Figure 3 shows that with small selection budgets, the subsets selected by all the heuristics have low utility as the heuristics fail to promote diversity during selection.

Notably, we show that the Shapley value heuristic would select the data subset with the lowest utility for certain data utility functions, including submodular ones. The Shapley value of a training point is calculated by taking a weighted average of the contribution of the point to all possible subsets of the

training set, and the weights are independent of the selection budget k. Moreover, the Shapley value of training data weights higher for its marginal contributions on small datasets. Thus, data points that make a larger contribution on tiny datasets may be assigned with higher Shapley value, even if they make little or negative contributions in every dataset of desired selection size k.

Theorem 3. *For any $n \geq 4$ and $k \in \{1, \dots, n\}$, the domination number of Shapley value is $d(n,k) = 1$, even if the utility function U is submodular.*

5 Evaluation

We evaluate DATASIFTER on six DQM tasks, as listed in Table 2. We consider various benchmark models and datasets used in past literature for each DQM task. Since we can observe similar results on different datasets, this section will only describe the result on *one* representative dataset for each task and leave the other dataset in the Appendix. Finally, we discuss the scalability of the DATASIFTER on larger datasets. The implementation details and the additional results are presented in the Appendix.

Task	Datasets	
	Main Text	Appendix
I. Backdoor Detection	CIFAR-10 [36]	MNIST[37]
II. Poisoned Data Detection	CIFAR-10 [36]	Dog vs. Cat [38]
III. Noisy Feature Detection	CIFAR-10 [36]	MNIST [37]
IV. Mislabeling Detection	SPAM [39]	CIFAR-10 [36]
V. Data Summarization	PubFig83 [40]	COVID-CT [41]
VI. Data Debiasing	Adult [42]	COMPAS [43]

Table 2: Summary of DQM tasks and datasets. We discuss one dataset for each task and defer the results over the other dataset to the Appendix.

5.1 Baselines

We focus on comparing data valuation-based approaches as they are closest to achieving the properties of multi-purpose, task-driven, and model-agnostic. We omit the data cleaning methods from the comparison as their applicability is limited to specific DQM tasks and specific models. Specifically, we consider the following eight state-of-art data valuation-based approaches: (1) *Shapley Permutation Sampling* (**Perm-SV**) [19], a Monte Carlo-based algorithm for Shapley value estimation. (2) *TMC-Shapley* (**TMC-SV**) [16], a refined version of the Perm-SV, where the computation is focused on the subsets whose utility changes significantly when an extra point is added. (3) *G-Shapley* (**G-SV**) [16], which approximates the Shapley value by anticipating the utility change caused by an extra point with its gradient. (4) *KNN-Shapley* (**KNN-SV**) [20], which approximates the Shapley value by using the K-Nearest-Neighbor as a proxy model. (5) *Least Core* (**LC**) [18], another data value notion in cooperative game theory. (6) *Leave-one-out* (**LOO**) [16] evaluates the change of model performance when a data point is removed. (7) *Influence Function* (**INF**) [10], which approximates the LOO error with influence functions. (8) **TracIn** [11], which traces the test loss change during the training process whenever the training point of interest is utilized. (9) **Random** is a setting where we randomly select a subset from the target dataset.

For fair comparison between DATASIFTER and baselines, we fix the number of utility sampling as 4000 for DATASIFTER and baseline algorithms that require utility sampling. The implementations of DATASIFTER and baseline algorithms will be detailed in the Appendix. We repeat model training ten times for each selected set of data points to obtain the error bars.

5.2 Results

5.2.1 Filtering out Harmful Data

Training data could be contaminated by various harmful examples, e.g., backdoor triggers, poison information, noisy/mislabeled samples. Our goal here is to identify data points that are most likely to be harmful. These points can either be discarded or presented with high priorities to human experts for manual cleaning. To evaluate the performance of different DQM techniques, we examine the training instances according to the quality ranks outputted by each method and plot the change of the fraction of detected corrupted data with the fraction of the checked training data. Additionally, for poisoned/backdoor data detection, we plot the change of *Attack Success Rate* (ASR), and for noisy feature/label detection, we plot the change of model accuracies after filtering out the low-quality data points selected by each technique. The validation data in utility sampling are 300 clean data points sampled from the test data of the corresponding datasets.
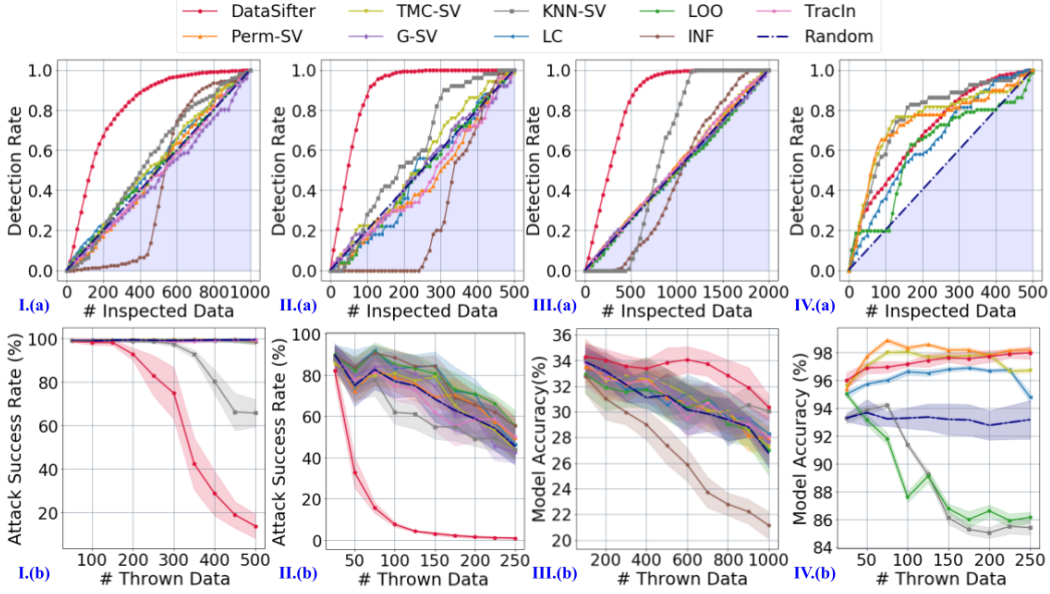
6

Figure 4: The experimental results and comparisons of the DATASIFTER under the case of filtering out harmful data (application I-IV). The light blue region in each (a) graph represents the area that a method is no better than a random selection. For I.(b) and II.(b), we depict the Attack Success Rate (ASR), where a lower ASR indicates a more effective detection. For III.(b) and IV.(b), we show the model test accuracy, where a higher accuracy means a better selection.

I. Backdoor Detection. Backdoor attacks [44, 45] embed an exploit at training time that is subsequently invoked by the presence of a "trigger" at test time. They are considered particularly dangerous since they make models predict a target output on inputs with predefined triggers while still retain state-of-the-art performance on the clean data. Since data points with the backdoor triggers contribute little to the learning of clean validation samples, we could expect to identify them by minimizing the data utility model. This experiment studies the effectiveness of DATASIFTER for removing backdoored examples. We evaluate BadNets [45] and Trojan attack [46], the two most famous backdoor attacks in the literature. We adopted a three-layer CNN as the target model, a poison rate of 0.2, and a target label 'Airplane.' Figure 4 I.(a) and I.(b) elaborate the Trojan attack detection results for a 1,000-size randomly selected subset of the CIFAR-10 dataset. As we can see, DATASIFTER significantly outperforms other DQM approaches; for instance, it achieves a detection rate of 90% with **51.17%** fewer inspected data points than the others.

II. Poisoned Data Detection. Adversaries make slight modifications to some training samples in data poisoning attacks to cause malicious behaviors in the test phase (e.g., misclassifying target test examples). We evaluate different DQM techniques on two popular attacks, namely, feature collision attack [47] and influence function-based attack [10]. These two are clean-label poisoning attacks where the attacker does not need to control the labeling of training data. We left the detailed descriptions of the attacks in the Appendix. Figure 4 II.(a) and II.(b) show the results for feature collision attack [47] on a 500-size randomly selected CIFAR-10 subset, where 50 data points of class 'cat' are perturbed with features extracted from a 'frog' sample in the test set. We see that DATASIFTER significantly outperforms all other DQM methods in the poisoned data detection task; for instance, it attains a 90% detection rate with **75.41%** fewer examined data points.

III. Noisy Feature Detection. Noise in features originated from sampling or transmitting (e.g., Gaussian noise) may decrease classification accuracy. Following the settings in [26], we add white noise to clean samples, and we evaluate the performance of each DQM technique on detecting those samples. For the CIFAR-10 dataset, we corrupt 25% of the train data images by adding white noise. Based on Figure 4 III.(a) and III.(b), we can conclude that DATASIFTER significantly outperforms all other methods on this task; for example, it achieves a 90% of detection rate by examining **67.25%** fewer data points. Meanwhile, the KNN-SV approach exhibits a distinctive trend – it only starts finding the noisy data points until filtering out a certain amount of clean data. This is mainly because

all noisy data points are out-of-distribution (OOD). The mechanism of KNN-SV tends to assign 0 values to OOD data points, while it will also assign negative values to some clean data points. We provide a more detailed explanation in the Appendix.

IV. Mislabeling Detection. Labels in the real world are often noisy due to automatic or non-expert labeling. Following [16, 23], we perform experiments on two datasets and present the results of SVM trained on Enron1 SPAM dataset [39] and the CIFAR-10 dataset. We adopt a bag-of-words representation of the Enron1 for training. The noise flipping ratio is 15%. Under this setting, Influence-based techniques and G-SV are not applicable since they require the model trained with gradient-based approaches. Figure 4 IV.(a) and IV.(b) show that although DATASIFTER does not attain the highest detection rate, the accuracies of the model trained on the selected data are competitive with the most effective approaches. For the Enron SPAM dataset, a small amount of mislabeled data points do not significantly affect the model performance; thus, those mislabeled samples could evade our detection based on the validation performance. By comparing Figure 4 IV.(a) and IV.(b), we can tell such evasion is acceptable as the model trained over the data points selected by DATASIFTER still achieves a competitive accuracy. On the other hand, we find KNN-SV and LOO can accomplish a decent detection rate but end up with a lower validation accuracy. This is because they select very unbalanced data points, as both of them satisfy the symmetry axiom discussed in Section 4.

5.2.2 Selecting High-quality Data

The DQM tasks considered in this section aim to select a subset that is most likely to help improve model test accuracy and fairness.

V. Data Summarization. Data summarization aims to select a small, representative subset from a massive dataset, which can retain a comparable utility to that of the whole dataset. We use a convolutional neural network trained on the PubFig83 dataset in this experiment. Figure 5 V shows that DATASIFTER and KNN-SV significantly outperform all the other DQM techniques, which have similar performance as the random selection.
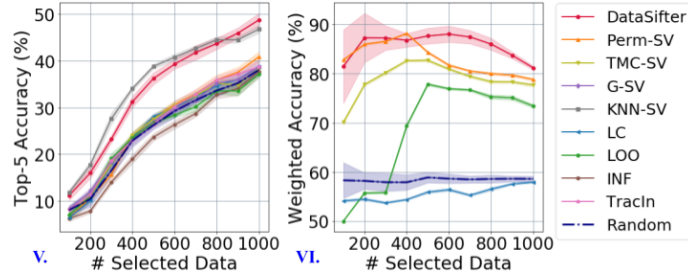

Figure 5: The experimental results and comparison of the DATASIFTER under the case of selecting high-quality data (application V and VI). We depict the validation accuracy for both cases. A higher accuracy indicates a better performance.

VI. Data Debiasing. We explore whether DQM techniques can help select a subset of training data that improves both fairness and performance for the ML task. We use logistic regression trained on the UCI Adult Census dataset as the task model. We measure the fairness by weighted accuracy – the average of model classification accuracy over females and that over males. G-SV, KNN-SV, and Influence-based techniques are not applicable for this application since they either require the model trained using the SGD, or are designed for computing data importance when the metric is test accuracy or loss. Therefore, we only compare with the remaining six baselines. Figure 5 VI shows that DATASIFTER achieves the top-tire performance along with the Perm-SV.

5.3 Comparisons on Larger Datasets

We compare the scalability between DATASIFTER and other baselines on large datasets. We show the results for backdoor detection on a 10,000-size Trojan square poisoned CIFAR-10 subset here. For DATASIFTER, we only sample data subset utilities from 1000 data points as we did in Section 5.2.1 I, but use the learned utility model to select data points on the entire 10,000 data points. When executed on NVIDIA Tesla K80 GPU, the clock time for the utility sampling step is within 5 hours for 4000 utility samples with a small CNN model, as the data size is fairly small. The LOO, the Least core, and all the Shapley value-based approaches except KNN-SV did not terminate in 24 hours, so we remove them from comparison. As we can see from Figure 6, DATASIFTER once again outperforms all the remaining approaches. The results show that the learned utility model can also provide utility

estimations for a set of unseen data points, which largely improves the scalability of DATASIFTER. On the contrary, the existing valuation-based approaches cannot predict the importance of unseen data points. Thus their utility sampling has to be conducted over the entire dataset.

6 Limitations of the DATASIFTER

We now discuss two limitations of the DATASIFTER.

Scalability. While Section 5.3 shows that DATASIFTER is often much more efficient than most of the other DQM schemes with similar design goals, its scalability to large dataset still needs further investigation. DATASIFTER could be slow as the utility sampling step requires retraining models for thousands of times. Although we already have several solutions for improving the scalability, such

Figure 6: The experimental results and comparison of the DATASIFTER and baseline algorithms for detecting backdoored data on larger datasets.

as using a smaller proxy model and/or only conduct utility sampling on a small portion of the dataset, it might still require hours of training. Further improving the scalability of DATASIFTER through some efficient approximation heuristics of data utility functions would be interesting future works.

Utility Learning Model. In this work, we use the popular set function learning model–DeepSet–as our utility learning model for all of the experiments. However, as shown in several previous works [48, 49, 26], many data utility functions using commonly used learning algorithms are close to submodular functions. While DeepSet-based utility learning models have already shown promising results in our experiment, DeepSets does not provide a mechanism to incorporate such prior knowledge. As another interesting line of future work, we would like to exploit the approximate submodularity of these kinds of data utility functions and use more fine-grained architectures or training algorithms for utility learning, e.g., submodular regularizations [50].

7 Conclusion

This paper presents DATASIFTER as a unified framework for realizing task-driven, multi-purpose, model-agnostic data quality management. We theoretically analyzed the worst-case performance of existing data valuation-based DQM schemes and show that these approaches suffer unsatisfying performance guarantees. This sheds light on the empirical observations that existing data valuation-based DQM schemes exhibit significant performance variation over different datasets and tasks. Based on an extensive evaluation of the DATASIFTER over six types of DQM tasks and eight different datasets, we showed that DATASIFTER is more comprehensive and robust than the state-of-the-art DQM approaches with similar design goals. For future work, we would like to further improve the scalability of the DATASIFTER as well as design utility learning models that are better aligned with the properties of data utility functions.
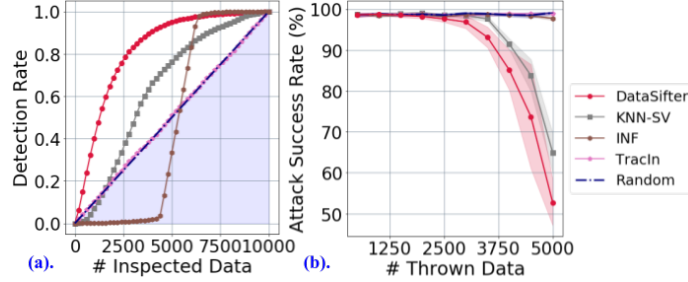
References

[1] Felix Neutatz, Binger Chen, Ziawasch Abedjan, and Eugene Wu. From cleaning before ml to cleaning for ml. *Data Engineering*, page 24, 2021.

[2] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1247–1261, 2015.

[3] Sebastian Schelter, Felix Biessmann, Dustin Lange, Tammo Rukat, Phillipp Schmidt, Stephan Seufert, Pierre Brunelle, and Andrey Taptunov. Unit testing data with deequ. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1993–1996, 2019.

[4] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE, 2019.

[5] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, 9(12):948–959, 2016.

[6] Bojan Karlaš, Peng Li, Renzhi Wu, Nezihe Merve Gürel, Xu Chu, Wentao Wu, and Ce Zhang. Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions. *arXiv preprint arXiv:2005.05117*, 2020.

[7] Sanjay Krishnan and Eugene Wu. Alphaclean: Automatic generation of data cleaning pipelines. *arXiv preprint arXiv:1904.11827*, 2019.

[8] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[9] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *IJCAI*, pages 4658–4664, 2019.

[10] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

[11] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33, 2020.

[12] Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. *arXiv preprint arXiv:1911.07116*, 2019.

[13] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International conference on machine learning*, pages 325–333. PMLR, 2013.

[14] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations. In *International Conference on Machine Learning*, pages 3384–3393. PMLR, 2018.

[15] Tianhao Wang, Zana Buçinca, and Zilin Ma. Learning interpretable fair representations. 2021.

[16] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

[17] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2019.

[18] Tom Yan and Ariel D Procaccia. If you like shapley then you'll love the core, 2020.

[19] Sasan Maleki. *Addressing the computational issues of the Shapley value with applications in the smart grid*. PhD thesis, University of Southampton, 2015.

[20] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *arXiv preprint arXiv:1908.08619*, 2019.

[21] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372, 2011.

[22] Sanjay Krishnan, Michael J Franklin, Ken Goldberg, and Eugene Wu. Boostclean: Automated error detection and repair for machine learning. *arXiv preprint arXiv:1711.01299*, 2017.

[23] Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? *arXiv preprint arXiv:1911.07128*, 2019.

[24] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. A principled approach to data valuation for federated learning. In *Federated Learning*, pages 153–167. Springer, 2020.

[25] Chih-Kuan Yeh, Joon Sik Kim, Ian EH Yen, and Pradeep Ravikumar. Representer point selection for explaining deep neural networks. *arXiv preprint arXiv:1811.09720*, 2018.

[26] Tianhao Wang, Si Chen, and Ruoxi Jia. One-round active learning. *arXiv preprint arXiv:2104.11843*, 2021.

[27] Maria-Florina Balcan and Nicholas JA Harvey. Learning submodular functions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 793–802, 2011.

[28] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer, 1978.

[29] Thibaut Horel and Yaron Singer. Maximization of approximately submodular functions. In *NIPS*, volume 16, pages 3045–3053, 2016.

[30] Avinatan Hassidim and Yaron Singer. Optimization for approximate submodularity. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 394–405, 2018.

[31] Abhimanyu Das and David Kempe. Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. *The Journal of Machine Learning Research*, 19(1):74–107, 2018.

[32] Flavio Chierichetti, Anirban Dasgupta, and Ravi Kumar. On Additive Approximate Submodularity. *arXiv e-prints*, art. arXiv:2010.02912, October 2020.

[33] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

[34] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[35] Fred Glover and Abraham P Punnen. The travelling salesman problem: new solvable cases and linkages with the development of approximation algorithms. *Journal of the Operational Research Society*, 48(5):502–510, 1997.

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[37] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[38] Kaggle. https://www.kaggle.com/c/dogs-vs-cats/overview.

[39] Rushdi Shams and Robert E Mercer. Classifying spam emails using text and readability features. In *2013 IEEE 13th international conference on data mining*, pages 657–666. IEEE, 2013.

[40] Nicolas Pinto, Zak Stone, Todd Zickler, and David Cox. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *CVPR 2011 WORKSHOPS*, pages 35–42. IEEE, 2011.

[41] Jinyu Zhao, Yichen Zhang, Xuehai He, and Pengtao Xie. Covid-ct-dataset: a ct scan dataset about covid-19. *arXiv preprint arXiv:2003.13865*, 2020.

[42] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[43] Hongjun Yoon. A machine learning evaluation of the compas dataset. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5474–5474. IEEE, 2018.

[44] Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia. Rethinking the backdoor attacks' triggers: A frequency perspective. *arXiv preprint arXiv:2104.03413*, 2021.

[45] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[46] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

[47] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv preprint arXiv:1804.00792*, 2018.

[48] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963. PMLR, 2015.

[49] Dongge Han, Michael Wooldridge, Alex Rogers, Shruti Tople, Olga Ohrimenko, and Sebastian Tschiatschek. Replication-robust payoff-allocation for machine learning data markets. *arXiv preprint arXiv:2006.14583*, 2020.

[50] Ayya Alieva, Aiden Aceves, Jialin Song, Stephen Mayo, Yisong Yue, and Yuxin Chen. Learning to make decisions via submodular regularization.

[51] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226. Springer, 2004.

[52] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[53] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

A   Proof of Theorem 1

Theorem 1. *For $n \geq 3$, there exists no linear heuristic $\mathcal{M}$ s.t. $d(n,k) = \binom{n}{k}$ for every $k \in \{1,\ldots,n\}$.*

Proof. Suppose, for contradiction, that there exists a linear heuristic $\mathcal{M}$ s.t. $d(n,k) = \binom{n}{k}$ for all k. For a dataset $\mathcal{D}$ and utility function U, WLOG assume that the ranks (in non-ascending order) output by $\mathcal{M}$ in the Step 2 of Definition 1 is $(1,\ldots,n)$. Then it means

$$U(\{1\}) \geq U(S) \text{ for all } S \text{ s.t. } |S| = 1,$$
$$U(\{1,2\}) \geq U(S) \text{ for all } S \text{ s.t. } |S| = 2,$$
$$\ldots$$
$$U(\{1,\ldots,n-1\}) \geq U(S) \text{ for all } S \text{ s.t. } |S| = n-1.$$

We construct a simple counter example of U to demonstrate such a $\mathcal{M}$ does not exist: let $n = 3$, we define U as follows:

$$U(\emptyset) = 0,$$
$$U(\{1\}) = 7, U(\{2\}) = U(\{3\}) = 5,$$
$$U(\{1,2\}) = 9, U(\{1,3\}) = 9, U(\{2,3\}) = 10,$$
$$U(\{1,2,3\}) = 10.$$

To make $d(3,1) = 3$, $\mathcal{M}$ must choose 1 for $k = 1$. However, for size-2 subsets, $\mathcal{M}$ can only choose between $\{1,2\}$ and $\{1,3\}$, whose utilities are both $9 < U(\{2,3\})$. Therefore, $d(3,2) = 2 < \binom{3}{2} = 3$. $\qquad \square$

B   Proof of Theorem 2

To formally state and prove Theorem 2, we introduce the formal definition of data type here.

Definition 4. *Given a dataset $\mathcal{D}$ and utility function U, if for all subset $S \subseteq \mathcal{D} \setminus \{i,j\}$, we have*

$$U(S \cup \{i\}) = U(S \cup \{j\}),$$

we say two data points i and j are of the same type.

In other words, two data points are of the same type if they will be scored equally by every linear heuristic that satisfies Symmetry Axiom. Theorem 2 essentially says that for all linear heuristic that will assign different scores to different types of data points, their domination numbers can be further upper bounded. We stress that this is a very mild assumption, especially when the space of the scores are continuous, which are the case for most of the existing data importance scoring mechanisms.

To simplify the notations for set operations, we use $k \times \{D\}$ to denote a dataset that contains k replicates of data point D, and we also denote the union of two data sets $S_1 \cup S_2 = S_1 + S_2$. The proof idea of Theorem 2 is to construct a *balanced* dataset that contains same amount of data points from the same types. If a linear heuristic $\mathcal{M}$ satisfies symmetry axiom, then $\mathcal{M}$ has to select data points of the same type when the target selection number is small, as all data points of the same type will receive the same scores. Of course, a dataset of only one type of data points will have nearly no utility.

Theorem 2 (Restated). *If a linear heuristic $\mathcal{M}$ satisfies symmetry axiom and will always assign different scores for different types of data points, then the domination number $d(n,k)$ of $\mathcal{M}$ is upper bounded by $\lfloor n/k \rfloor \binom{\lceil \frac{n}{\lfloor n/k \rfloor} \rceil}{k}$ for each $k \in \{1,\ldots,n\}$.*

Proof. Suppose there are c *types* of data points: $D_1,\ldots,D_c$. Let $r = n \mod c$. We construct the dataset $\mathcal{D}$ that contains $\lfloor n/c \rfloor$ data points for each of $D_1,\ldots,D_{n-r}$, and contains $\lceil n/c \rceil$ data points for each of $D_{n-r+1},\ldots,D_n$. We construct utility function U as follows:

$$U(\emptyset) = 0;$$
$$U(i_1 \times \{D_1\} \ldots + i_c \times \{D_c\}) = 1,$$

for every tuple of non-negative integers $(i_1, \ldots, i_c)$ s.t. $1 \leq \sum_{j=1}^{c} i_j \leq n$, except that

$$U(k \times \{D_1\}) = \ldots = U(k \times \{D_c\}) = 0$$

for all $k \leq \lfloor \frac{n}{c} \rfloor$. This construction reflects the rationale that a dataset that only contains one type of data points (e.g. all of the same label) provide little information for the ML task.

Since $\mathcal{M}$ satisfies symmetry axiom, we know that all data points of the same type will receive the same scores. Besides, we know that data points of different types will receive different scores. Therefore, when the target selection size $k \leq \lfloor \frac{n}{c} \rfloor$, $\mathcal{M}$ will return $k \times \{D_j\}$, which has the worst utilities for subset at size k and there are $(c - r)\binom{\lfloor n/c \rfloor}{k} + r\binom{\lceil n/c \rceil}{k}$ such subsets that only contains single types of data points. For each k, by taking the largest possible c such that $k \leq \lfloor \frac{n}{c} \rfloor$, we obtain the desired bound. □

We note that the upper bound is non-trivial for every $k \leq n/2$. We also note the assumption that $\mathcal{M}$ always assigns different scores for different data types can be further relaxed as long as *there exists* such a balanced dataset described in the proof that $\mathcal{M}$ assigns different scores for different data types.

C Proof of Theorem 3

Given a dataset $\mathcal{D} = \{1, \ldots, n\}$ and a submodular utility function U, the Shapley value is computed as

$$v_{\text{shap}}(i) = \frac{1}{n} \sum_{S \subseteq \mathcal{D} \backslash \{i\}} \frac{1}{\binom{n-1}{|S|}} \left[U(S \cup \{i\}) - U(S) \right] \tag{1}$$

Theorem 3 (Restated). *The domination number $d(n, k)$ of Shapley value is 1 for every $n \geq 4$ and any $k \in \{1, \ldots, n\}$, even if we restrict the utility function U to be submodular.*

Proof. We first consider the case when $k \geq 3$.

We construct an instance of a dataset $\mathcal{D} = \{1, \ldots, n\}$ and a submodular utility function U as follows:

$U(\emptyset) = 0;$
$U(\{1\}) = U(\{2\}) = \ldots = U(\{k\}) = 7, U(\{i\}) = 5$ for $i \geq k + 1;$
$U(S) = 2|S| + 5$ for all S s.t. $2 \leq |S| \leq k - 1;$
$U(\{1, \ldots, k\}) = 2k + 4, \quad U(S) = 2k + 5$ for all other S s.t. $|S| = k;$
$U(S) = 2k + 5$ for all S s.t. $|S| \geq k + 1.$

We can compute Shapley value according to its definition in (1):

$$v_{shap}(1) = \ldots = v_{shap}(k) = \frac{1}{n} \left[7 + \frac{2(k-1) + 4(n-k)}{n-1} + 2(k-3) + \frac{2\binom{n-1}{k-1} - 1}{\binom{n-1}{k-1}} \right]$$

$$= \frac{1}{n} \left[2k + 3 + \frac{4n - 2k - 2}{n-1} - \frac{1}{\binom{n-1}{k-1}} \right]$$

$$v_{shap}(k+1) = \ldots = v_{shap}(n) = \frac{1}{n} \left[5 + \frac{2k + 4(n-k-1)}{n-1} + 2(k-3) + \frac{1}{\binom{n-1}{k-1}} \right]$$

$$= \frac{1}{n} \left[2k + 1 + \frac{4n - 2k - 4}{n-1} - \frac{1}{\binom{n-1}{k-1}} \right]$$

Since

$$v_{shap}(1) - v_{shap}(k+1) = \frac{1}{n} \left[2 + \frac{2}{n-1} - \frac{1}{\binom{n-1}{k-1}} - \frac{1}{\binom{n-1}{k}} \right] \geq \frac{2}{n(n-1)} > 0,$$

14

we know that $\mathcal{M}$ will always output $\{1, \ldots, k\}$, which achieves the lowest utility among all data subsets of size k. Therefore, Shapley value's domination number $d(n, k) = 1$ for all $3 \leq k \leq n - 1$.

We then consider the case when $k = 2$. The submodular data utility functions for the case of $k \geq 3$ can be easily adapted as follows:

$$U(\emptyset) = 0;$$
$$U(\{1\}) = U(\{2\}) = 7, U(\{i\}) = 5 \text{ for } i \geq 3;$$
$$U(\{1, 2\}) = 8, \ \ U(S) = 9 \text{ for all other } S \text{ s.t. } |S| = 2;$$
$$U(S) = 9 \text{ for all } S \text{ s.t. } |S| \geq 3.$$

The Shapley value is computed as follows:

$$v_{shap}(1) = v_{shap}(2) = \frac{1}{n} \left[7 + \frac{1 + 4(n - 2)}{n - 1} \right]$$
$$= \frac{1}{n} \left[11 - \frac{3}{n - 1} \right]$$

$$v_{shap}(3) = \ldots = v_{shap}(n) = \frac{1}{n} \left[5 + \frac{4 + 4(n - 3)}{n - 1} + \frac{2}{(n - 1)(n - 2)} \right]$$
$$= \frac{1}{n} \left[9 - \frac{4}{n - 1} + \frac{2}{(n - 1)(n - 2)} \right]$$

Since

$$v_{shap}(1) - v_{shap}(3) = \frac{1}{n} \left[2 + \frac{2}{n - 1} - \frac{1}{\binom{n-1}{k-1}} - \frac{1}{\binom{n-1}{k}} \right] \geq \frac{2}{n(n - 1)} > 0,$$

we know that $\mathcal{M}$ will always output $\{1, \ldots, 2\}$, which achieves the lowest utility among all data subsets of size 2. Therefore, for Shapley value, $d(n, 2) = 1$.

Finally, we consider the case when $k = 1$. Similarly, we construct a submodular utility function as follows:

$$U(\emptyset) = 0;$$
$$U(\{1\}) = 6, U(\{i\}) = 7 \text{ for } i \geq 2;$$
$$U(\{1, i\}) = 11 \text{ for } i \geq 2, U(\{i, j\}) = 9 \text{ for } i, j \geq 2;$$
$$U(S) = 11 \text{ for all } S \text{ s.t. } |S| \geq 3.$$

The Shapley value is computed as follows:

$$v_{shap}(1) = \frac{1}{n}[6 + 4 + 2] = \frac{12}{n}$$

$$v_{shap}(i) = \frac{1}{n} \left[7 + \frac{5 + 2(n - 2)}{n - 1} + \frac{2(n - 2)(n - 3)}{(n - 1)(n - 2)} \right]$$
$$= \frac{1}{n} \left[11 - \frac{1}{n - 1} \right]$$
$$< \frac{12}{n} = v_{shap}(1).$$

Therefore, Shapley value's domination number $d(n, k) = 1$ for $k = 1$. $\qquad \square$

D   Experiment Details and Results on More Datasets

D.1   Details of Figure 2

In Figure 2 of the maintext, we showed the predicted vs true data utility values (test accuracy) for a synthetic dataset with logistic regression. For the synthetic data generation, we sample 200 data

points from 50-dimensional standard Gaussian distribution. All of the 50-dimensional parameters are independently and uniformly drawn from $[-1, 1]$. Each data point is labeled by the sign of its vector's sum. The data utility model we use is a three-layer MLP (note that a set function can be represented by a function on $\{0, 1\}^n$ in a natural way).

D.2 Details of Figure 3

In Figure 3 of the maintext, the tiny synthetic dataset is generated by sample data points from a 2-dimensional standard Gaussian distribution, where the mean vector of the Gaussian distribution is $(0.1, -0.1)$. Each data point is labeled by the sign of its vector's sum. We first sample 9 data points with positive label and 2 data points with negative label. We then replicate each of the two negatively labeled data points for two times. To simulate natural noise, we add Gaussian noise to the copied data vector with scale 10^{-5}. By sampling and copying, we obtained 15 data points with natural redundancy. Since there are only 6 data points with negative label, they tend to be assigned with larger (and similar) importance scores by linear heuristics like Shapley value. Both Shapley and Least core thus rank negative points with higher importance. This means that when the target selection size is less than 6, the selected dataset will have only single kind of labels and no information about the other label class at all. As shown in Figure 3, both Shapley and Least core achieves trivial utility for the first 6 selected data points.

D.3 Baseline Implementation

For fair comparisons between DATASIFTER and baselines, we fix the total number of utility sampling as 4000 for DATASIFTER and baseline algorithms that require utility sampling, including Perm-SV, TMC-SV, G-SV, and LC. Following the settings in [16], we set the performance tolerance in TMC-Shapley as 10^{-3}. Following the settings in [20], we set $K = 5$ for KNN-Shapley. We use CVXOPT[2] library to solve the constrained minimization problem in the least core calculation. For influence function technique, we rank training data points according to their influences on the model loss over the validation data. The code is adapted from the PyTorch implementation of influence function on GitHub[3]. For TracIn technique, we only use the parameters in the last layer, following the settings in [11]. We sample checkpoints for every 15 epochs. The implementation is adapted from the official GitHub repository[4].

D.4 Details of Datasets Used in Section 5

CIFAR-10 [36].   CIFAR-10 consists of 60,000 3-channel images in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck). Each image is of size 32×32.

MNIST [37].   MNIST consists of 70,000 handwritten digits. The images are 28×28 grayscale pixels.

Dog vs. Cat [38].   Dog vs. Cat dataset consists of 2000 images (1000 for 'dog' and 1000 for 'cat') extracted from CIFAR-10 dataset. Each image is of size 32×32.

Enron SPAM [39].   Enron SPAM dataset consists of 2000 emails extracted from Enron corpus [51]. The bag-of-words representation has 10714 dimensions.

PubFig83 [40].   PubFig83 is a real-life dataset of 13,837 facial images for 83 individuals. Each image is resized to 32×32.

Covid-CT [41].   The COVID-CT-Dataset has 746 CT images in total, containing 349 images from 216 COVID-19 patients and the rest of them are from healthy people. The dataset is separated into 543 training images and 203 test images. We resized each image to 32×32.

---

[2]https://cvxopt.org/
[3]https://github.com/nimarb/pytorch_influence_functions
[4]https://github.com/frederick0329/TracIn

UCI Adult Census [42]. The Adult dataset contains 48,842 records from the 1994 Census database. Each record has 14 attributes, including gender and race information. The task is to predict whether one's income exceeds $50K/yr based on census data.

COMPAS [43]. We use a subset of the COMPAS dataset that contains 6172 data records used by the COMPAS algorithm in scoring defendants, along with their outcomes within two years of the decision, for criminal defendants in Broward County, Florida. Each data record has features including the number of priors, age, race, etc.

D.5 Implementation Details

For the experiment of backdoor detection, data poisoning detection, noisy detection, and mislabel detection on the CIFAR-10 dataset, the CNN model we use has two convolutional layers. A max-pooling layer follows each with the ReLU as the activation function. For the experiment of Backdoor detection and noisy feature detection on the MNIST dataset, we use LeNet adapted from [52], which has two convolutional layers, two max-pooling layers, and one fully-connected layer. For the experiment of data summarization, a large CNN model is adopted to train on the PubFig83 dataset, which has six convolutional layers, and each of them is followed by a batch normalization layer and a ReLU activation function. For the experiment on poisoning detection over the Dog vs. Cat dataset as well as the data summarization over the COVID-CT dataset, we use a small CNN model adapted from PyTorch tutorial[5], which contains two convolutional layers, two max-pooling layers, and followed by three fully-connected layers. We use Adam optimizer with learning rate 10^{-3}, mini-batch size 32 to train all of the models mentioned above for 30 epochs, except that we train LeNet for five epochs on MNIST. For the experiment of data biasing on the Adult dataset, we implement logistic regression in scikit-learn [53] and use the LibLinear solver. For the experiment of mislabeling detection on SPAM and data debiasing on COMPAS, we adopt SVM implementation from scikit-learn library [53] with RBF kernel.

A DeepSets model is a set function $f(S) = \rho\left(\sum_{x \in S} \phi(x)\right)$ where both ρ and ϕ are neural networks. In our experiment, both ϕ and ρ networks have three fully-connected layers. For the COMPAS dataset, we set the number of neurons in every hidden layer and the dimension of set features (i.e., the output of ϕ network) to be 64. For all other datasets, we set the number of neurons and set dimension to be 128 . We use the Adam optimizer with learning rate 10^{-4}, mini-batch size of 32, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ to train all of the DeepSets utility models, for up to 20 epochs.

D.6 Additional Results

In this section, we present experiment details and results on more datasets corresponding to the applications introduced in the main body (see Section 5).

D.6.1 Backdoor Attack

We consider the two most popular types of backdoor attacks, namely the BadNets [45] and the Trojan square trigger [46]. Those two attacks' major difference is the trigger itself, where BadNets adopts a white block trigger at the right corner, and Trojan attack adopts a square trigger.

Here, we show the results of DATASIFTER and baseline techniques over detecting BadNets triggers on MNIST dataset. The poisoning rate is 0.25, and the target label is '0'. The performance of different DQM techniques is illustrated in Figure 7 I.(a) and I.(b). We can see that DATASIFTER outperforms all other methods in the detection rate and significantly reduces the attack accuracy after filtered out bad data points.

D.6.2 Data Poisoning Attack

We discuss two popular types of clean-label data poisoning attacks. Feature collision attack [47] crafts poison images that collide with a target image in feature space, thus making it difficult for a model to discriminate between the two. Influence function-based poisoning attack [10] identifies the most influential training data points for the target image and generates the adversarial training perturbation that causes the most increase in the loss on the target image. The Attack Success Rate is

---

[5]https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
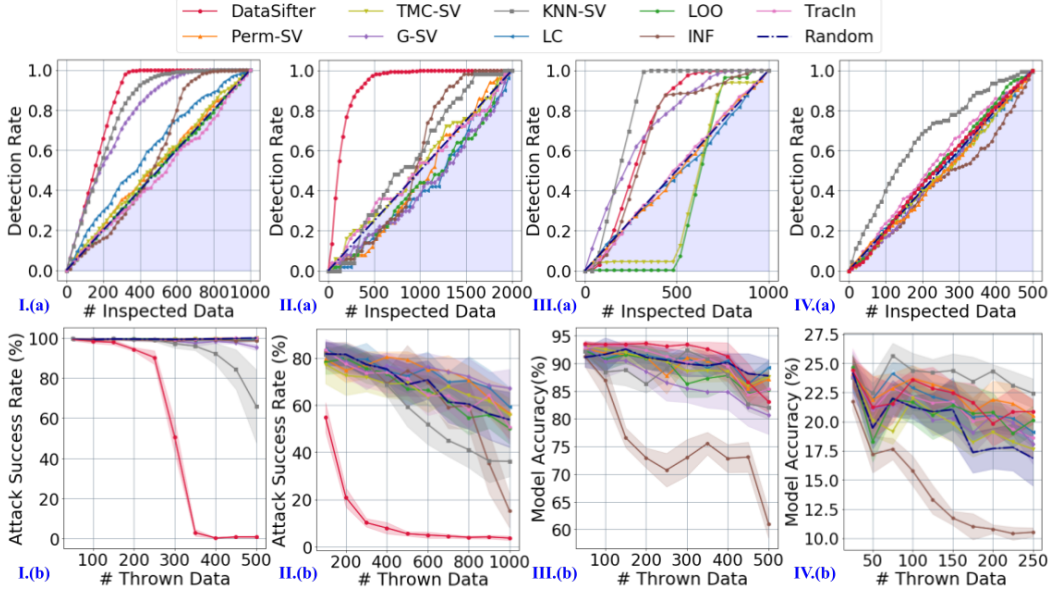
Figure 7: The experimental results and comparisons of the DATASIFTER under the case of filtering out harmful data (application I-IV). The light blue region in each (a) graph represents the area that a method is no better than a random selection. For I.(b) and II.(b), we depict the Attack Success Rate (ASR), where a lower ASR indicates a more effective detection. For III.(b) and IV.(b), we show the model test accuracy, where a higher accuracy means a better selection.

measured by the model's confidence on the prediction of poisoned data (with respect to the target label).

Figure 7 II.(a) (b) show the results for influence function-based attack on Dog vs. Cat dataset, where 50 data points of class 'cat' are perturbed to increase the model loss on a 'dog' sample in the test set. As we can see, DATASIFTER is a more effective approach to detect poisoned data points than all other baselines.

D.6.3 Noisy Feature

We follow the same evaluation method for noisy data detection as in Section 5 with another setting: LeNet model trained on noise polluted MNIST. We randomly select 1000 data points and corrupt 25% of them with white noise. As shown in Figure 7 III.(a) (b), we can see that although KNN-Shapley can achieve slightly better performance in detecting noisy data points, DATASIFTER still retains a higher performance for model accuracy. Besides, similar to the case for CIFAR10, we find that the KNN-SV approach only starts finding the noisy data points until filtering out a certain amount of clean data. This is mainly because all noisy data points are out-of-distribution (OOD), as shown in Figure 8 (b). The mechanism of KNN-SV, however, tends to assign 0 values to OOD data points while assign negative values to clean data points that are in-distribution but have different labels from their neighbors. Figure 8 (c) gives a visualization of the distribution of KNN-Shapley values.

D.6.4 Mislabeled Data

We conduct another experiment on noisy label detection: a small CNN model trained on 500 data points from the CIFAR-10 dataset. The noise flipping ratio is 25%. The performance of mislabel detection is shown in Figure 7 IV.(a). As we can see, no DQM techniques are particularly effective in detecting mislabeled data for this task. Only KNN-SV achieves a slightly better performance than other approaches. We conjecture that the difficulty of mislabel detection on CIFAR-10 dataset is due to the following reason: since an oracle for detecting mislabeled data points can also be used to implement a classifier, the difficulty of mislabeling detection is at least as difficult as classification. A classifier directly trained on the 500 clean data points in this experiment, however, can only attain
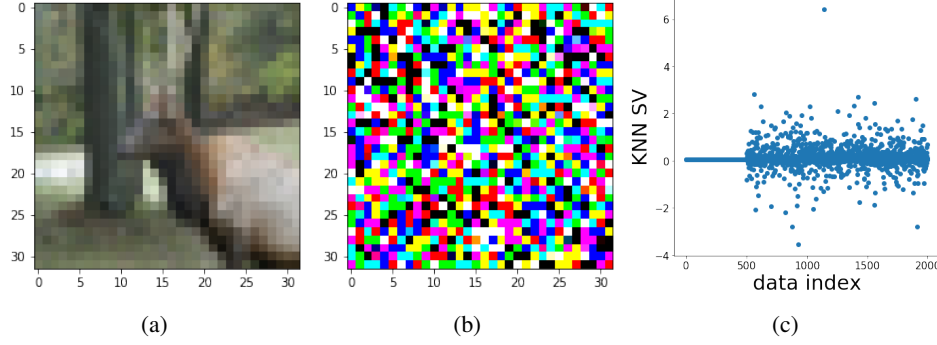
18

Figure 8: (a) a normal image from CIFAR-10, (b) an example of noisy data image, (c) a sample of KNN-Shapley values, where data points with index < 500 are noisy. A data point with a higher KNN-Shapley value is considered more important.

around 28% test classification accuracy. Nevertheless, Figure 7 IV.(b) shows that DATASIFTER only achieves slightly worse model accuracy than KNN-SV after filtering out selected bad data points.

D.6.5 Data Summarization

As another setting for the data summarization application we consider, we use the patient CT images from COVID-CT dataset for a binary classification task, which aims to determine whether an individual is diagnosed with COVID-19 or not. The CNN model trained on the dataset achieves around 72% classification accuracy. Figure 9 V. shows the results for selecting up to 400 data points with different DQM techniques. As we can see, DATASIFTER achieves the best model accuracies on the selected data points along with KNN-SV.
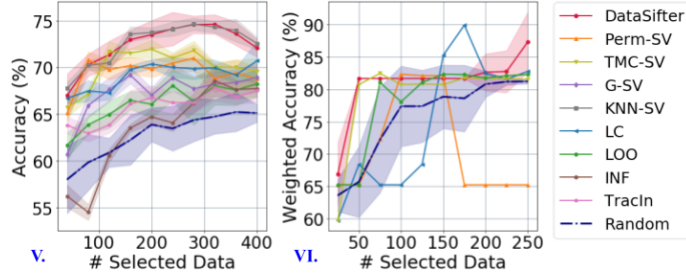

Figure 9: The experimental results and comparison of the DATASIFTER under the case of selecting high-quality data (application V and VI). We depict the validation accuracy for both cases. A higher accuracy indicates a better performance.

D.6.6 Data Debiasing

We introduce another data debiasing experiment on the criminal recidivism prediction (COMPAS) task, where races are considered as the sensitive attribute. The utility metric we adopted here is the average accuracy across different race groups. The learning algorithm we use is SVM with RBF kernel. Baselines including G-SV, KNN-SV, and Influence-based techniques are not applicable for this application due to the utility metric and learning algorithm we use. Figure 9 VI. shows the results for DATASIFTER and the remaining five baselines. We can see that DATASIFTER again achieves the top-tire performance.

D.6.7 Large Datasets

We follow the same protocol as in Section 5.3 for comparing the scalability between DATASIFTER and other baselines on a different setting: noisy data detection on a 20,000-size CIFAR-10 subset. The corruption ratio is 25%. Again, for DATASIFTER, we use the learned utility model from Section 5.3 to select data points on the 20,000-size set. We remove the LOO, the Least core, and all the Shapley value-based approaches except KNN-SV from comparison, as they did not terminate in 24 hours for 4000 utility sampling on the 20,000-size set. As we can see from Figure 10 (a) and (b),

DATASIFTER significantly outperforms all other baseline techniques. The results demonstrate that although the utility sampling step could be expensive, the scalability of DATASIFTER can be boosted by the predictive power of the learned utility model.
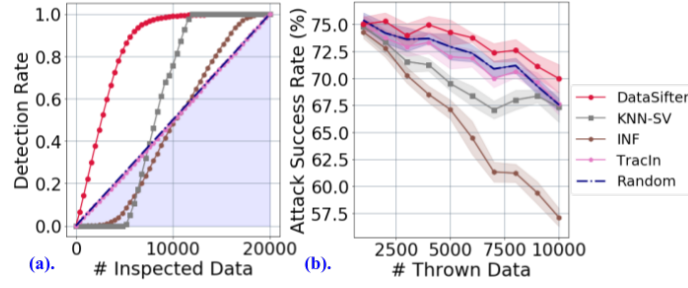

Figure 10: The experimental results and comparison of the DATASIFTER and baseline algorithms for detecting noisy data on larger datasets.