Equitable Data Valuation Meets the Right to Be Forgotten in Model Markets (Technical Report)

Haocheng Xia, Jinfei Liu, Jian Lou, Zhan Qin, Kui Ren
Zhejiang University
{xiahc,jinfeiliu,jian.lou}@zju.edu.cn
{qinzhan,kuiren}@zju.edu.cn

Yang Cao
Hokkaido University
yang@ist.hokudai.ac.jp

Li Xiong
Emory University
lxiong@emory.edu

ABSTRACT

The increasing demand for data-driven machine learning (ML) models has led to the emergence of model markets, where a broker collects personal data from data owners to produce high-usability ML models. To incentivize data owners to share their data, the broker needs to price data appropriately while protecting their privacy. For *equitable data valuation*, which is crucial in data pricing, *Shapley value* has become the most prevalent technique because it satisfies all four desirable properties in fairness: balance, symmetry, zero element, and additivity. For *the right to be forgotten*, which is stipulated by many data privacy protection laws to allow data owners to unlearn their data from trained models, the *sharded structure* in ML model training has become a de facto standard to reduce the cost of future unlearning by avoiding retraining the entire model from scratch. In this paper, we explore how the sharded structure for the right to be forgotten affects Shapley value for equitable data valuation in model markets. To adapt Shapley value for the sharded structure, we propose S-Shapley value, a sharded structure-based Shapley value, which satisfies four desirable properties for data valuation. Since we prove that computing S-Shapley value is #P-complete, two sampling-based methods are developed to approximate S-Shapley value. Furthermore, to efficiently update valuation results after data owners unlearn their data, we present two delta-based algorithms that estimate the change of data value instead of the data value itself. Experimental results demonstrate the efficiency and effectiveness of the proposed algorithms.

PVLDB Artifact Availability:
The source code, data, and/or other artifacts are available at https://anonymous.4open.science/r/rtbf-C5FC. The original technique report is available at https://anonymous.4open.science/r/rtbf-C5FC/original_rtbf.pdf.

1 INTRODUCTION

In the era of big data, machine learning (ML) models are being applied in an ever-growing number of businesses and governments to promote financial gains and social welfare [27, 28, 47]. Enormous model buyers seek ML models for their demands. High-usability ML models are powered by large amounts of high-quality training data, which indicates that the data is valuable. Nowadays, personal data has become one of the most significant data sources [18], but high-quality data is sparsely distributed among different data owners [3, 13]. To bridge the gap between data owners and model buyers, model markets have emerged [2, 33]. A model market consists of three entities: data owners, a broker, and model buyers. The broker collects data from multiple data owners, then builds and sells various ML models to interested model buyers. To incentivize more data owners to join model markets and share their data, pricing their data properly [8, 9] and protecting their privacy adequately are essential [30, 44].

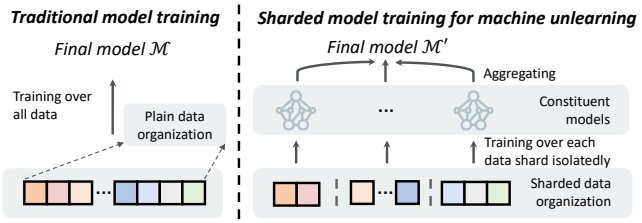


Figure 1: Comparison between traditional model training and sharded model training for machine unlearning – Model $\mathcal{M}$ is trained directly on all data while model $\mathcal{M}'$ is aggregated from the constituent models trained on corresponding disjoint shards.

Equitable data valuation is one of the most desirable abilities in model markets, which is pivotal for data pricing. An equitable data valuation strategy helps the broker assign more payoff to data owners whose data contributes to better model performance [30]. To approach the equitability goal, many data valuation strategies are developed, including leave-one-out (LOO) score [23], *Shapley value* [16], reinforcement learning-based value [37], etc. Among these, Shapley value has become the most prevalent strategy by virtue of its unique four properties for equitable payoff allocation: *balance*, *symmetry*, *zero element*, and *additivity* [33, 36]. Despite being an intriguing equitable valuation strategy, one shortfall of Shapley value is that it may not best suit all desiderata of specific data valuation tasks rising in their particular application scenarios. Consequently, various desiderata have given birth to different Shapley variants at the expense of partial properties [25, 35, 37]. For example, Beta Shapley [25] value better captures the influence of individual data points by removing the balance property. From another perspective, the intensive computing workload is a major hurdle for the application of Shapley value. Specifically, the naïve calculation's computational complexity is exponential in the number of data owners, so numerous approaches have been proposed to improve efficiency by sampling approximation [16, 46] or task-specific simplification [23, 31].

The right to be forgotten, which is stipulated by data privacy protection laws including GDPR [14], CCPA [21], and PIPL [7], has become a mandatory part of the personal data protection standard. The right to be forgotten mandates that data owners shall have the right to erasure their personal data from service providers (e.g., companies and institutions). Recent research argues that deleting personal data from databases is not enough to promise the removal

of personal data [34]. For example, ML models trained on the previous dataset are also regulated because these models can be used to infer the training data [45]. To make ML models satisfy the right to be forgotten with less overhead than the naïve baseline of "retraining-from-scratch", a new research direction "machine unlearning" [5] has emerged and quickly garnered growing research interest recently. A common idea in machine unlearning is limiting the impact of a data point on the model in the training process to support efficient updates in the future. Following this idea, the authors [4, 17, 34] adopt *a sharded structure* in model training. As an example shown in Figure 1, the training process of Model $\mathcal{M}'$ applies the sharded structure. Unlike model $\mathcal{M}$ trained by the traditional training approach, the training data of model $\mathcal{M}'$ is divided into several disjoint shards. Different constituent models on the corresponding shard are aggregated for the final model. For example, the aggregation strategy can be a simple *label-based majority vote* [4] for classification problems. And only the shards involving the data to be forgotten need to retrain corresponding constituent models.

Since the broker builds ML models with the personal data of data owners, these ML models in model markets are regulated by the right to be forgotten naturally. However, existing model markets have not incorporated the right to be forgotten. In this paper, to enable model markets to respect this widely enforced regulation, we explore *the right to be forgotten in model markets* from the perspective of data valuation.

Gaps and Challenges. Though efforts have been made to develop different variants of Shapley value [25, 35, 37], how Shapley value for equitable data valuation should respond to the sharded structure for the right to be forgotten is still understudied. It is therefore tempting to ask: how can we design a variant of Shapley value, which can simultaneously satisfy the equitable data valuation and suit the sharded structure for the right to be forgotten? We summarize the gaps and challenges as follows.

- *Shapley value over the sharded structure.* When data owners cooperate with a sharded structure in model markets, traditional Shapley value faces a dilemma in payoff allocation. Consider an ML task that has applied the sharded structure in a model market, each data shard contains the data of one or several data owner(s). We refer to a player as either a single data owner or a data shard consisting of several data owners to compute Shapley value. Consequently, Shapley value of a data shard is not equal to the sum of its data owners' Shapley values. This inequality creates complexities and conflicts for payoff allocation. As data owners are profit-driven, they tend to choose a payoff allocation that can obtain more profit. If a data shard's Shapley value is more (resp. less) than the sum of its data owners' Shapley value, the data owners in this shard may request to adopt data shards' (resp. data owners') Shapley value to allocate payoff. Therefore, the challenge is: *How to design a variant of Shapley value for equitable data valuation over the sharded structure?*
- *Efficient computation and update.* Computing Shapley value or existing variants including Beta-Shapley value, CS-Shapley value, and Data Banzhaf value is known to be #P-complete [16, 35, 40]. What is more, the right to be forgotten guarantees data owners the freedom to exit the model market upon their removal requests. Once data owners exit and request to remove their data, the previous valuations become inapplicable since the data distribution changed. Shapley value computation requires large amounts of utility function evaluations (e.g., model accuracy in ML) whose number is exponential in the number of players (e.g., data owners or data shards). The time-consuming training of machine learning models will further increase the computational resource overhead. Therefore, blindly reevaluating the data value for each data owner from scratch is inefficient. The challenge to be addressed is: *How to efficiently compute the initial data value and update it when data removal requests occur?*

Contributions. In this paper, we address the identified challenges by proposing the **S**harded structure-based Shapley (S-Shapley) value, combined with a series of efficient approximation algorithms for estimating initial S-Shapley value and updating S-Shapley value.

For the first challenge, we extend and define four desirable properties (Section 3.3) for equitable data valuation given the sharded structure. Then we propose the sharded structure-based Shapley (S-Shapley) value, as a metric to quantify the value of data given the sharded structure. S-Shapley value satisfies the four desirable properties. We prove that computing S-Shapley value is #P-complete. Moreover, we perform an evaluation using two classifiers, six datasets, and five baseline methods. The results demonstrate that S-Shapley value gives more insights into the data importance in learning performance than other existing data valuation strategies under sharded structures.

For the second challenge, we develop two algorithms to approximate the initial S-Shapley value and two algorithms to update it in polynomial time. For approximating initial S-Shapley value: (i) we develop a simple algorithm with Monte Carlo simulation as the baseline; (ii) to achieve higher efficiency, we develop a utility sampling-based algorithm to reuse the evaluated utilities. For updating S-Shapley value: (i) we develop an algorithm to estimate the change of utility as the change of S-Shapley value when a data owner exits; (ii) for the case of multiple data owners exiting, we introduce a batched algorithm to reduce asymptotic error. Both algorithms reduce the time cost by at least an order of magnitude. We briefly summarize our contributions as follows.

- We present four desirable properties for the data valuation with the sharded structure and propose S-Shapley value to measure the contribution of data. In addition, we theoretically show that computing S-Shapley value is #P-complete.
- We develop two approximation algorithms for efficiently estimating S-Shapley value with Monte Carlo simulation and utility evaluation reuse.
- We present two efficient algorithms for updating S-Shapley value on the new datasets when one or multiple data owners exit, respectively.
- Our experimental studies show that S-Shapley value gives more insights than existing methods in the importance of data under the sharded structure. The effectiveness and efficiency of our proposed algorithms for approximating and updating S-Shapley value are demonstrated.

Organization. The rest of the paper is organized as follows. In Section 2, we provide a comprehensive review of the related work on data valuation and machine unlearning. Section 3 discusses the

preliminaries, while Section 4 proposes the S-Shapley value. We present a simple Monte Carlo simulation algorithm for estimating the S-Shapley value as the baseline and an efficient algorithm by reusing utilities in Section 5. Section 6 presents the algorithms based on the change of value for fast updating the S-Shapley value. We report the experimental results and findings in Section 7. Finally, we conclude the paper in Section 8.

2 RELATED WORK

In this section, we discuss related work on data valuation and the right to be forgotten, respectively.

Data valuation. In model markets, a common way for payoff allocation is based on the importance of the data. Data valuation methods quantize the importance of data by assigning a larger value to data that is more important for a given task, e.g, improving the performance of ML models [16].

Some existing data valuation strategies in ML such as LOO score [23], influence-function-based method [33], and reinforcement-learning-based value [37], have simple intuition and do not depend on the concept of Shapley value. Compared with Shapley-value-based methods, these methods are usually more computationally efficient as they require less or even no training of models but cannot provide theoretical guarantees of fairness desired in data valuation [16]. Shapley-value-based methods include Shapley value [16] and its variants with the partially relaxed Shapley properties [15, 25, 35, 37, 40]. Ghorbani and Zou [16] first utilized Shapley value to quantify the contributions of data points. Subsequent work has tried to design variants of Shapley value according to different scenarios, such as D-Shapley [15] based on stable data distributions, Beta Shapley [25] highlighting the importance of individual data, CS-Shapley [35] for classifiers, and Data Banzhaf [40] for robust value ranking. However, directly applying existing Shapley-value-based methods under the sharded structure to facilitate machine unlearning for the right to be forgotten may introduce unfairness. In contrast, S-Shapley value builds on the sharded structure to ensure desirable properties for data valuation. Under this new definition, not all coalitions (subsets of data owners) can be used for computing S-Shapley value, hence existing approximation algorithms cannot be directly applied. In contrast to existing work on data valuation in federated learning [41, 42], S-Shapley value differs significantly in several ways: (i) it focuses on record-level valuation rather than client-level valuation; (ii) it performs a continuous valuation due to the requirement of the right to be forgotten rather than a one-time valuation; and (iii) since all participants are trusted, the broker has direct access to all data from data owners while the coordinating server cannot access the data at each client in federated learning.

The right to be forgotten. Recent laws (e.g., Article 17 of GDPR) stipulated the right to be forgotten [14] which require companies and institutions to delete user data upon request. In model markets, the right to be forgotten protects the right of data owners to exit markets and cancel data transactions. Cao and Yang [5] initiated the study of the right to be forgotten in ML and came up with a strict definition of machine unlearning which can entirely remove certain sensitive data from trained statistical query learning models [24]. Moreover, researchers further consider the unlearning problem for other prevalent ML models, such as k-means clustering [17], decision trees [34], and even neural networks [4]. To update the ML model and avoid retraining the model from scratch after removing the data to be forgotton, they coincidentally applied the idea of the sharded structure which only requires retraining of the model for the corresponding shard and updating the aggregated model. Specifically, Ginart et al. [17] identified the principle of modularity in unlearning which restricts the model parameters depending on specific partitions of the dataset. Schelter et al. [34] developed unlearning algorithms on Extremely Randomised Trees [38] by learning an ensemble representation. Bourtoule et al. [4] introduced SISA training framework and systematically explored the impact of the sharded structure on model performance and unlearning speed-up. Recently, Chen et al. [10] extended the sharded structure into graph unlearning by employing balanced graph partitioning algorithms that rely on community detection and embedding clustering. Ginart et al. [17] introduced approximate unlearning without model retraining using the influence function under a relaxed unlearning definition. However, recent studies [19, 20] on influence-function-based machine unlearning methods are mostly limited to linear models or fine-tuning settings, making it challenging to apply them to more general models such as deep neural networks. In contrast, shard-based unlearning surpasses in terms of applicability. Our target application scenario is a model market that utilizes existing shard-based machine unlearning methods to ensure the right to be forgotten of data owners.

3 PRELIMINARIES

In this section, we review the concept of Shapley value in Section 3.1 and describe the sharded structure in Section 3.2. We list four desiderata for data valuation based on the sharded structure in Section 3.3. Table 1 summarizes the frequently used notations throughout the paper.

Table 1: The summary of frequently used notations.

Notation	Definition
n	the number of data owners
m	the number of data shards
$\mathcal{U}(\cdot)$	utility function
D_i	the i^{th} data owner
z_i	the i^{th} data set
d_j	the j^{th} data shard
$\mathcal{S}$	a coalition of data sets
$\mathcal{L}$	sharded structure
L_k	the partitions in k^{th} level of $\mathcal{L}$
$\mathcal{SV}$	Shapley value
$\mathcal{SSV}$	S-Shapley value
τ	the total number of samples

3.1 Shapley Value

Consider n data owners $D_1, \ldots, D_n$ such that data owner D_i owns data set z_i ($1 \leq i \leq n$). They aim to solve a task by training an ML model with a joint effort. To quantify the contribution of each data owner towards solving the task, we assume a utility function $\mathcal{U}(\mathcal{S})$ ($\mathcal{S} \subseteq \{z_1, \ldots, z_n\}$) that evaluates the utility of a coalition $\mathcal{S}$,

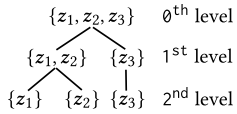
which consists of data sets from multiple data owners. The utility of coalition $\mathcal{S}$ can be the performance of the ML model trained over $\mathcal{S}$. Shapley value is a measure that can be used to evaluate data importance for payoff allocation, which uniquely satisfies four equitable properties, including balance, symmetry, zero element, and additivity. Shapley value of data owner D_i measures *the marginal utility improvement (i.e., marginal contribution)* contributed by z_i averaged over all possible $n!$ permutations $\mathfrak{S}_n$ on $\{1, \ldots, n\}$.

$$\mathcal{SV}_i = \frac{1}{n!} \sum_{\pi \in \mathfrak{S}_n} \left[\mathcal{U}\left(\mathcal{S}_\pi^{z_i} \cup \{z_i\}\right) - \mathcal{U}\left(\mathcal{S}_\pi^{z_i}\right) \right], \tag{1}$$

where π is a permutation in $\mathfrak{S}_n$ and $\mathcal{S}_\pi^{z_i}$ is the coalition of data set(s) whose index before i in permutation π ($\mathcal{S}_\pi^{z_i} = \emptyset$ if i is the first element in π). Computing the exact Shapley value has to enumerate all coalitions' utilities and thus is prohibitively expensive. More precisely, computing Shapley value is #P-complete [16]. In the following, we use z_i to represent both data owner D_i and its data set z_i.

3.2 Sharded Structure

Under the regulation of the right to be forgotten, to reduce the computational overhead of data removal, data points are sharded to train constituent models correspondingly, and then these models are ensembled by model aggregation (e.g., majority vote). We represent the sharded structure with a sequence of partitions over $\{z_1, \ldots, z_n\}$, $\mathcal{L} = \{L_0, L_1, L_2\}$ such that $L_0 = \{\{z_1, \ldots, z_n\}\}$, $L_1 = \{d_1, \ldots, d_m\}$ ($d_j \subseteq \{z_1, \ldots, z_n\}$ and $\bigcap_{j=1}^{m} d_j = \emptyset$), and $L_2 = \{\{z_1\}, \ldots, \{z_n\}\}$. We call L_k ($0 \le k \le 2$) the partitions in the k^{th} level. Partitions of a level are coarser than the partitions of the next level, which indicates that one partition in a level consists of the partition(s) in the next level. We note that each partition in L_1, d_j ($1 \le j \le m$), is called a *data shard* which is a given coalition of data owners. An example of the sharded structure is shown as follows.



$$\mathcal{U}(\mathcal{S}) = \begin{cases} 10, & \text{if } \mathcal{S} = \{z_1, z_2\}, \{z_2, z_3\} \\ 12, & \text{if } \mathcal{S} = \{z_1, z_2, z_3\} \\ 2, & \text{otherwise} \end{cases}$$

Figure 2: Sharded structure.

Example 3.1. Consider three data owners D_1, D_2, D_3 such that D_1 and D_2 cooperate to train a constituent model while D_3 trains a constituent model independently. The sharded structure can be represented with $\mathcal{L} = \{L_0, L_1, L_2\}$, where $L_0 = \{\{z_1, z_2, z_3\}\}$, $L_1 = \{\{z_1, z_2\}, \{z_3\}\}$, and $L_2 = \{\{z_1\}, \{z_2\}, \{z_3\}\}$ as shown in Figure 2. Utility of each coalition $\mathcal{S}$, $\mathcal{U}(\mathcal{S})$, is the performance of the final model trained over $\mathcal{S}$ with sharded structure $\mathcal{L}$.

Following the existing work in machine unlearning [4, 10, 17], we assume the sharded structure is given by the specific sharding mechanisms. How to design the sharded structure is beyond the scope of this paper.

3.3 Desiderata for Data Valuation

Following the celebrated properties of Shapley value (A1-A4), we list four desirable properties (P1-P4) under the constraints of the sharded structure. P1-P4 will be the design guidance to S-Shapley value for equitable data valuation over the sharded structure.

A1. Balance: The sum of the payoff to data owners should be equal to the utility of all the data owners. That is, the total payoff is fully distributed to all data owners. Formally, $\sum_{i=1}^{n} \mathcal{SV}_i = \mathcal{U}(\{z_1, \ldots, z_n\})$.

P1. Sharded balance: The total payoff should be fully distributed to all data owners and each data shard's payoff should be fully distributed to its data owners. Formally, S-Shapley value of a data shard $d_j \in L_1$, $\mathcal{SSV}_{d_j}$, should be the sum of the S-Shapley value of data owners in d_j, that is $\mathcal{SSV}_{d_j} = \sum_{z_i \in d_j} \mathcal{SSV}_i$, and $\sum_{i=1}^{n} \mathcal{SSV}_i = \mathcal{U}(\{z_1, \ldots, z_n\})$, where $\mathcal{SSV}_i$ is the S-Shapley value of z_i.

Comparison: The intuition is that if any data shard fails to meet the balance property, it will result in a conflict in the distribution of the payoff. If $\mathcal{SV}_{d_j} \neq \sum_{z_i \in d_j} \mathcal{SV}_i$, where d_j is a data shard in L_1 and $\mathcal{SV}_{d_j}$ is Shapley value of d_j, data owners tend to choose the higher one which brings more payoff. The difference between A1 and P1 is that in addition to following Shapley value's balance property, P1 further introduces the balance property for each shard, which requires that the payoff of each data shard should be equal to the total payoff of its data owners.

A2. Symmetry: The same contribution brings the same payoff. Formally, for two data owners z_i and $z_{i'}$ and any subset of data owners $\mathcal{S} \subseteq \{z_1, \ldots, z_n\} \backslash \{z_i, z_{i'}\}$, if $\mathcal{U}(\mathcal{S} \cup \{z_i\}) = \mathcal{U}(\mathcal{S} \cup \{z_{i'}\})$, then $\mathcal{SV}_i = \mathcal{SV}_{i'}$.

P2. Sharded symmetry: Two shards with the same contributions receive the same payoff; two data owners with the same contributions within a shard receive the same payoff. Formally, for two data shards d_j and $d_{j'}$ and any coalition of data shards $\mathcal{DS} \subseteq L_1 \backslash \{d_j, d_{j'}\}$, if $\mathcal{U}(\mathcal{DS} \cup \{d_j\}) = \mathcal{U}(\mathcal{DS} \cup \{d_{j'}\})$, then $\mathcal{SSV}_{d_j} = \mathcal{SSV}_{d_{j'}}$; for two data sets z_i and $z_{i'}$ in the same data shard $d_j \in L_1$ and any coalition of data owners $\mathcal{S} \in d_j \backslash \{z_i, z_{i'}\}$, if $\mathcal{U}(\mathcal{S} \cup \{z_i\}) = \mathcal{U}(\mathcal{S} \cup \{z_{i'}\})$, then $\mathcal{SSV}_i = \mathcal{SSV}_{i'}$.

Comparison: The intuition is that even the same data may have considerably different contributions when belonging to different shards. Therefore, symmetry for data owners is defined on individual shards in P2 rather than globally in A2.

A3. Zero element: No contribution, no payoff. Formally, for a data owner z_i and any subset of data owners $\mathcal{S} \subseteq \{z_1, \ldots, z_n\} \backslash \{z_i\}$, if $\mathcal{U}(\mathcal{S} \cup \{z_i\}) = \mathcal{U}(\mathcal{S})$, then $\mathcal{SV}_i = 0$.

P3. Sharded zero element: No contribution for the overall model, no payoff for the data shard; no contribution in the data shard, no payoff for the data owner. Formally, for a data shard d_j and any coalition of data shards $\mathcal{DS} \subseteq L_1 \backslash \{d_j\}$, if we have $\mathcal{U}(\mathcal{DS} \cup \{d_j\}) = \mathcal{U}(\mathcal{DS})$, then $\mathcal{SSV}_{d_j} = 0$; for a data set z_i in data shard d_j and any subset $\mathcal{S} \subseteq d_j \backslash \{z_i\}$, if we have $\mathcal{U}(\mathcal{S} \cup \{z_i\}) = \mathcal{U}(\mathcal{S})$, then $\mathcal{SSV}_i = 0$.

Comparison: The intuition is that when a data set has no contribution to the corresponding data shard it will also have no contribution to the final model. The difference between A3 and P3 is that when judging whether a data owner has no contribution, we examine all possible subsets in the corresponding data shard rather than all possible subsets of data owners.

A4. Additivity: If data owners' data can be used for two ML tasks T_1 and T_2 with payoff $\mathcal{SV}_1$ and $\mathcal{SV}_2$, respectively, then the payoff to complete both tasks $T_1 + T_2$ is $\mathcal{SV}_1 + \mathcal{SV}_2$.

P4. Additivity: The same as A4.

Comparison: Additivity property is independent of data's cooperation structure, so it remains unchanged with a sharded structure.

Compared to Shapley value, S-Shapley value has more favorable properties when applied to sharded data by additionally requiring balance property within each data shard (P1), reducing the scope of symmetry property and zero element property to each of data shard (P2 & P3), and preserving additivity property (P4). Since the shared structure has limited the combination of data owners, it will be more efficient to compute S-Shapley value.

4 S-SHAPLEY VALUE

In this section, we define S-Shapley value based on four desirable properties (P1-P4) in Section 3.3 for equitable data valuation over the sharded structure and prove the problem of computing S-Shapley value is #P-complete.

S-Shapley value. Sharded balance (P1) can be expressed as that the total payoff should be fully distributed to all data shards and then each data shard's payoff should be fully distributed to its data owners. It follows the idea of *top-down* allocation, which is broadly applied in allocating investments and budgets. We first consider the payoff allocation from L_0 to L_1. For each data shard's $\mathcal{SSV}$, the requirements of sharded balance (P1), sharded symmetry (P2), sharded zero element (P3), and additivity (P4) at these two levels will degenerate to the same requirements of Shapley value's corresponding properties. Therefore, *all possible coalitions of data shards need to be considered to measure contribution.* When we allocate payoff from L_1 to L_2, sharded balance, sharded symmetry, sharded zero element, and additivity become the Shapley value's corresponding properties restricted to each data shard. Therefore, we need to further *consider all possible coalitions of data owners in their corresponding data shards to measure contribution* based on the previous coalitions of data shards.

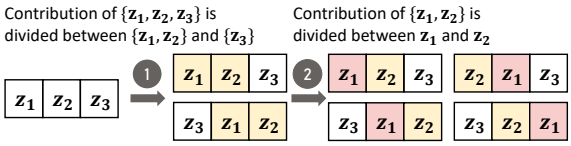


Figure 3: Illustration of top-down allocation.

Example 4.1. Consider Example 3.1's setting. For payoff allocation from L_0 to L_1, we evaluate the average marginal contribution of data shards as Shapley value does by considering all permutations over $\{\{z_1, z_2\}, \{z_3\}\}$ (i.e., $[\{z_1, z_2\}, \{z_3\}]$ and $[\{z_3\}, \{z_1, z_2\}]$). For payoff allocation from L_1 to L_2, we measure average marginal contribution of the data owners inside $\{z_1, z_2\}$ by consider all permutations over $\{z_1, z_2\}$ in permutations $[\{z_1, z_2\}, \{z_3\}]$ and $[\{z_3\}, \{z_1, z_2\}]$ as shown in Figure 3.

Compared to all $n!$ permutations $\mathfrak{S}_n$ over $\{1, \ldots, n\}$ that need to be considered in computing Shapley value, some permutations cannot be involved in computing S-Shapley value. Given sharded structure $\mathcal{L}$, we refer to the permutations for computing S-Shapley value as $\mathfrak{S}_n(\mathcal{L})$.

$$\mathfrak{S}_n(\mathcal{L}) = \{\pi \in \mathfrak{S}_n : \text{for all data shard } d_j \in L_1, \forall z_i, z_{i'} \in d_j, \\ \text{if } \pi.\text{idx}(i) < \pi.\text{idx}(i'') < \pi.\text{idx}(i') \text{ then } z_{i''} \in d_j\},$$

where $\pi.\text{idx}(i)$ is the index of element i in π $(1 \le i \le n, 1 \le \pi.\text{idx}(i) \le n)$.

While Equation 1 computes Shapley value by averaging a data owner's marginal contribution in all permutations of $\mathfrak{S}_n$, $\mathcal{SSV}$ is computed by averaging a data owner's marginal contribution in all permutations of $\mathfrak{S}_n(\mathcal{L})$, so we have the following formulation.

Proposition 4.2. *$\mathcal{SSV}_i$ uniquely satisfies sharded balance, sharded symmetry, sharded zero element, and additivity (P1-P4) presented in Section 3.3.*

$$\mathcal{SSV}_i = \frac{1}{|\mathfrak{S}_n(\mathcal{L})|} \sum_{\pi \in \mathfrak{S}_n(\mathcal{L})} \left(\mathcal{U}\left(\mathcal{S}_\pi^{z_i} \cup \{z_i\}\right) - \mathcal{U}\left(\mathcal{S}_\pi^{z_i}\right)\right), \quad (2)$$

where $\mathcal{S}_\pi^{z_i} = \{z_{\pi(1)}, \ldots, z_{\pi.\text{idx}(i)}\} \backslash \{z_i\}$.

Proof. The expression of $\mathcal{SSV}_i$ in Equation 2 is a particular case of the levels structure value in game theory except for the utility function. The proof follows from the uniqueness of the game theoretic levels structure value by reducing the ML model training to a cooperative game. Levels structure value uniquely satisfies five properties including efficiency, coalitional symmetry, individual symmetry, additivity, and null player (Theorem 2 in [43]). We prove that these properties and the desiderata for data valuation in Section 3.3 can be mapped in both directions. Additivity is trivially consistent. Null player and sharded zero element have the same effect logically.

First, we show that if the value satisfies efficiency, coalitional symmetry, and individual symmetry, it satisfies sharded balance and sharded symmetry. Balance property implies that the sum of all partitions in each level is equal to $\mathcal{U}(\{z_1, \ldots, z_n\})$ and individual symmetry property indicates that the value is independent of the order of players (i.e., data). Coalitional symmetry property denotes that if two data owners are in the same data shard, and they are symmetric in the data shard, then the value for two data owners are equal. We assume that $\exists\, d_j \in L_1, \sum_{z_i \in d_j} \mathcal{SSV}_i \neq \mathcal{SSV}_{d_j}$. When all data shards in L_1 are symmetric and $z_1, \ldots, z_n$ are symmetric, then either $\sum_{i=1}^n \mathcal{SSV}_i \neq \mathcal{U}(\{z_1, \ldots, z_n\})$ or $\sum_{d_j \in L_1} \mathcal{SSV}_{d_j} \neq \mathcal{U}(\{z_1, \ldots, z_n\})$. There is a conflict with the balance property, therefore, sharded balance is satisfied, $\sum_{z_i \in d_j} \mathcal{SSV}_i = \mathcal{SSV}_{d_j}$. Coalitional symmetry property denotes that if two data owners $z_i, z_{i'} \in d_j$ are symmetric in d_j, then the value for z_i and $z_{i'}$ are equal. If $z_i, z_{i'} \in d_j$ are symmetric in d_j, $z_i, z_{i'}$ must contribute the same to any coalition of the partitions in d_j. Therefore, sharded symmetry is satisfied.

Second, we show that if the value satisfies sharded balance and sharded symmetry, it satisfies balance, coalitional symmetry, and individual symmetry. Sharded balance contains the requirement of efficiency property. In addition, sharded balance indicates that for any data shard $d_j \in L_1$, $\sum_{z_i \in d_j} \mathcal{SSV}_i$ is equal to the value of d_j in the corresponding level. Combined with sharded symmetry, each data shard can be seen as an individual game independent

from the order of players (i.e. data) and individual symmetry is satisfied. Since the top-down value allocation among a data shard $d_j \in L_1$ is independent, the effect of sharded symmetry is the same as coalitional symmetry. □

Example 4.3. According to Equation 2, we evaluate the data owners' data in Example 3.1 as follows. Firstly, generate the permutation set $\mathfrak{S}_n(\mathcal{L})$. Then we average the marginal contribution of

Table 2: Marginal contribution.

$\mathfrak{S}_n$	$\mathfrak{S}_n(\mathcal{L})$	z_1	z_2	z_3
$[1,2,3]$	$[1,2,3]$	2	8	2
$[1,3,2]$	~~$[1,3,2]$~~	2	10	0
$[2,1,3]$	$[2,1,3]$	8	2	2
$[2,3,1]$	~~$[2,3,1]$~~	2	2	8
$[3,1,2]$	$[3,1,2]$	0	10	2
$[3,2,1]$	$[3,2,1]$	2	8	2

z_i over the permutations in $\mathfrak{S}_n(\mathcal{L})$ shown in Table 2 to compute $\mathcal{SSV}_i$. The yielded $\mathcal{SSV}$ for (z_1, z_2, z_3) is $\left(\frac{1}{4} \times (2+8+0+2) = 3, \frac{1}{4} \times (8+2+10+8) = 7, \frac{1}{4} \times (2+2+2+2) = 2\right)$, which is different from Shapley value $\left(\frac{1}{6} \times (2+2+8+2+0+2) = \frac{8}{3}, \frac{1}{6} \times (8+10+2+2+10+8) = \frac{20}{3}, \frac{1}{6} \times (2+0+2+8+2+2) = \frac{8}{3}\right)$ for (z_1, z_2, z_3).

Interpretation of S-Shapley value. $\mathcal{SSV}_i$ can be interpreted as the average marginal contribution of z_i in permutations of $\mathfrak{S}_n(\mathcal{L})$. Besides, $\mathcal{SSV}_i$ is the average marginal Shapley value contribution of z_i to the corresponding data shard. An example to compute $\mathcal{SSV}_i$ started with precomputed Shapley value is shown as follows.

Example 4.4. Following Example 4.3, the standard Shapley value for (z_1, z_2, z_3) is $(\frac{8}{3}, \frac{20}{3}, \frac{8}{3})$. Shapley value for the data shards in L_1, $(\{z_1, z_2\}, \{z_3\})$, is $(\frac{1}{2} \times (10+10) = 10, \frac{1}{2} \times (2+2) = 2)$. Then we can calculate $\mathcal{SSV}$ correspondingly. $\mathcal{SSV}_3$ inherits Shapley value of $\{z_3\}$ in L_1, thus $\mathcal{SSV}_3 = 2$. We determine $\mathcal{SSV}_1$ and $\mathcal{SSV}_2$. There are two possible permutations within the data shard $\{z_1, z_2\}$, i.e., $[z_1, z_2]$ and $[z_2, z_1]$. Averaging z_1's marginal contribution for Shapley value in these permutations, we have $\mathcal{SSV}_1 = \frac{1}{2} \times (\frac{8}{3} - 0 + 10 - \frac{20}{3}) = 3$. Similarly, we have $\mathcal{SSV}_2 = 7$.

It is worth noting that computing S-Shapley value is #P-complete.

Theorem 4.5. *Computing S-Shapley value with an arbitrary sharded structure $\mathcal{L}$ in n-person weighted voting [29] games is #P-complete.*

Proof. Consider n-person weighted voting games (WVGs). Here we are given non-negative voting weight $w_i \in \mathbb{R}_{\geq 0}$ for each player $z_i (1 \leq i \leq n)$ and a positive quota $q \in \mathbb{R}_{>0}$. The utility value of a coalition $\mathcal{S}(\mathcal{S} \subseteq \{z_1, \ldots, z_n\})$ is one if $\sum_{z_i \in \mathcal{S}} w_i > q$, and zero otherwise. Computing $|\mathfrak{S}_0(\mathcal{L})| \cdot \mathcal{SSV}_i$ in a n-person WVG can be considered as counting the number of accepting computations of a nondeterministic Turing machine, so the problem is in class #P.

Then, to show completeness, we use a variant of *the subset sum problem* that we call *SUBSET-SUM-EQ*. A *SUBSET-SUM-EQ* instance is given by a set of non-negative integers $U = \{x_1, \ldots, x_n\}$, and positive integers M. The question is to determine if there exists a set of integers $U' \subseteq U$ whose elements sum to exactly M. It can be seen from the reductions given in [11, 32] that *SUBSET-SUM-EQ* is NP-complete.

Here are the details of a reduction from *SUBSET-SUM-EQ* to the problem that finding a valid coalition $\mathcal{S}$ where the i^{th} player's marginal contribution $(1 \leq i \leq n)$ is one with given $\mathcal{L}$ in polynomial time. The reduction maps an instance of *SUBSET-SUM-EQ* to a WVG given $\mathcal{L}$. We focus on the n^{th} player, z_n, and let n^{th} player in m^{th} shard, $z_n \in d_m$. Given an instance of *SUBSET-SUM-EQ* with a set of $(m + |d_m| - 2)$ non-negative integers $U = \{x_1, \ldots, x_{m+|d_m|-2}\}$ and $M = \sum_{j=1}^{m+|d_m|-2} x_j/2$. According the instance, we construct a n-person WVG with the given $\mathcal{L}$. For $j = 1$ to $m-1$, for any $z_i \in d_j$, $w_i = x_j/|d_j|$. For any $z_i \in d_m$, $w_i = x_{m-1+\alpha_i}$ where $i \neq n$ and α_i is the order of z_i in d_m. And $w_n = 1$.

We show the reduction works as follows.

First, we need to show that *SUBSET-SUM-EQ* has a solution if there is a valid coalition $\mathcal{S}$ where the n^{th} player's marginal contribution is one with $\mathcal{L}$. It is easy to see that any valid coalition $\mathcal{S}$ of players satisfying $\mathcal{U}(\mathcal{S}) - \mathcal{U}(\mathcal{S}\backslash\{z_n\})$ is one if and only if $\sum_{z_i \in \mathcal{S}} w_i > M$ and $\sum_{z_i \in \mathcal{S}\backslash\{z_n\}} w_i \leq M$. Since $w_n = 1$, we have $\sum_{z_i \in \mathcal{S}\backslash\{z_n\}} w_i = M$. Thus $\mathcal{S}\backslash\{z_n\}$ has a corresponding solution to the instance of *SUBSET-SUM-EQ*.

Second, if there is a solution for the instance of *SUBSET-SUM-EQ*, we show that there is a corresponding valid coalition $\mathcal{S}$ where n^{th} player's marginal contribution is one with $\mathcal{L}$. If the sum of s numbers equals the target sum, $x_{a_1} + \ldots + x_{a_s} = M$, we can construct a valid coalition $\mathcal{S}$. First, we initialize an empty coalition $\mathcal{S}$. For x_j in $\{x_{a_1}, \ldots, x_{a_s}\}$, if $j <= m-1$, add the element(s) in d_j into $\mathcal{S}$; otherwise, add the corresponding player z_i such that $j = m - 1 + \alpha_i$ into $\mathcal{S}$. Then we have $\mathcal{U}(\mathcal{S}) - \mathcal{U}(\mathcal{S}\backslash\{z_n\}) = 1$. This completes the proof. □

5 COMPUTING S-SHAPLEY VALUE

In this section, we propose two approximation algorithms to estimate S-Shapley value in polynomial time based on sampling: Monte Carlo sampling (Section 5.1) and utility sampling (Section 5.2).

5.1 Monte Carlo Sampling

We directly sample the marginal contribution to approximate S-Shapley value of z_i, $\mathcal{SSV}_i$. We rewrite Equation 2 by regarding $\mathcal{SSV}_i$ as the expectation of z_i's marginal contribution. That is

$$\mathcal{SSV}_i = E_{\pi \sim \Pi}\left[\mathcal{U}\left(\mathcal{S}_\pi^{z_i} \cup \{z_i\}\right) - \mathcal{U}\left(\mathcal{S}_\pi^{z_i}\right)\right], \tag{3}$$

where Π is the uniform distribution over all permutations in $\mathfrak{S}_n(\mathcal{L})$.

Given that $\mathcal{SSV}_i$ is the expectation of marginal contribution, Monte Carlo simulation can be used to estimate $\mathcal{SSV}_i$, following previous work on Shapley value computation [16, 23]. First, we randomly sample a permutation in $\mathfrak{S}_n(\mathcal{L})$. Then, we scan the permutation from the first element to the last element and calculate the marginal contribution of each data owner. Repeating the same procedure over multiple permutations, the final estimation of $\mathcal{SSV}_i$ is simply the average of all the calculated marginal contributions.

To randomly sample a permutation in $\mathfrak{S}n(\mathcal{L})$, a multistage sampling approach is employed, which first permutes all data shards and then generates permutations for data owners within each data shard. Algorithm 1 outlines the pseudo-code for sampling a permutation in $\mathfrak{S}_n(\mathcal{L})$. Given a sharded structure $\mathcal{L}$, a permutation is first sampled over the data shards (Line 2), followed by the sampling of permutations of the corresponding data owners' indexes in each

Algorithm 1: π = **Sample**($\mathcal{L}$).

input : sharded structure $\mathcal{L}$
output: a permutation π in $\mathfrak{S}_n(\mathcal{L})$

1 initialize π as an empty list;
2 let π' be a random permutation from $1, \ldots, m$;
3 **for** j=1 *to* m **do**
4 $\quad a = \pi'(j)$;
5 $\quad$ append a random permutation of the indexes of d_a's data owners to π;
6 **return** π;

Algorithm 2: Monte Carlo Sampling Algorithm.

input : data sets from data owners $\{z_1, \ldots, z_n\}$, sharded structure $\mathcal{L}$, the number of permutations τ
output: estimated S-Shapley value $\widehat{\mathcal{SSV}}_i$ for each data owner ($1 \le i \le n$)

1 $\widehat{\mathcal{SSV}}_i \leftarrow 0$ ($1 \le i \le n$);
2 **for** t=1 *to* τ **do**
3 $\quad \pi^t \leftarrow$ **Sample**($\mathcal{L}$);
4 $\quad$ **for** i=1 *to* n **do**
5 $\quad\quad \widehat{\mathcal{SSV}}_{\pi^t(i)} += \frac{t-1}{t} \cdot \widehat{\mathcal{SSV}}_{\pi^t(i)} +$
6 $\quad\quad \frac{1}{t} \cdot \left(\mathcal{U}(\{z_{\pi^t(1)}, \ldots, z_{\pi^t(i)}\}) - \mathcal{U}(\{z_{\pi^t(1)}, \ldots, z_{\pi^t(i-1)}\})\right)$;
7 **return** $\widehat{\mathcal{SSV}}_1, \ldots, \widehat{\mathcal{SSV}}_n$;

data shard (Lines 4-5), where $\pi'(j)$ is the j^{th} element in permutation π' ($1 \le \pi'(j) \le m$). The data owners' indexes in different data shards do not overlap as $\bigcap_{j=1}^{m} d_j = \emptyset$. We note that Algorithm 1 will be used as a fundamental module for all algorithms.

Algorithm 2 outlines the pseudo-code for estimating $\mathcal{SSV}_i$ ($1 \le i \le n$) using Monte Carlo sampling. We randomly sample τ permutations in $\mathfrak{S}_n(\mathcal{L})$ based on Algorithm 1 (Lines 2-3) and calculate the average marginal contribution of each data owner (Lines 4-5), where $\pi^t(i)$ is the i^{th} element in permutation π^t ($1 \le \pi^t(i) \le n$). It is worth noting that the estimation of $\mathcal{SSV}_i$ (i.e., $\widehat{\mathcal{SSV}}_i$) in Algorithm 2 is unbiased.

Theorem 5.1. *Algorithm 2 gives an unbiased estimation of* $\mathcal{SSV}_i$ ($1 \le i \le n$), *i.e.,* $E[\widehat{\mathcal{SSV}}_i] = \mathcal{SSV}_i$.

Proof. $\widehat{\mathcal{SSV}}_i$ is the average of the marginal contribution samples, i.e., $\widehat{\mathcal{SSV}}_i = \frac{1}{\tau}\sum_{t=1}^{\tau} \mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i} \cup \{z_i\}\right) - \mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i}\right)$. We have

$$E[\widehat{\mathcal{SSV}}_i] = E[\frac{1}{\tau}\sum_{t=1}^{\tau} \mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i} \cup \{z_i\}\right) - \mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i}\right)]$$

$$= \frac{1}{\tau}\sum_{t=1}^{\tau} E[\mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i} \cup \{z_i\}\right) - \mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i}\right)] = \frac{1}{\tau}\sum_{t=1}^{\tau} \mathcal{SSV}_i = \mathcal{SSV}_i.$$

That is, $\widehat{\mathcal{SSV}}_i$ is an unbiased estimation of $\mathcal{SSV}_i$. □

In practice, we can conduct Monte Carlo sampling iteratively until the average empirically converges. The larger the number of sample permutations, the smaller error bound between $\widehat{\mathcal{SSV}}_i$ and $\mathcal{SSV}_i$ according to Hoeffding's inequality [22].

Theorem 5.2. *According to Hoeffding's inequality, given the range of a data set's marginal contributions* r *and the number of permutations* τ, *the error* $|\widehat{\mathcal{SSV}}_i - \mathcal{SSV}_i| \le \sqrt{2r^2 \log(2/\delta)/\tau}$ *in probability* $1-\delta$.

Proof. Let ϕ_t denote the t^{th} ($1 \le t \le \tau$) sampled marginal contribution. Since in any game with a finite number of data owners, the population of marginal contributions of a data owner's data (i.e., a data set) is finite. Let all marginal contributions be in the range $[-r, r]$ ($r \ge 0$). According to Equation 3 and Hoeffding's inequality, we have

$$Pr(|\tau\widehat{\mathcal{SSV}}_i - \sum_{t=1}^{\tau} E[\phi_t]| \ge \tau\epsilon) = Pr(|\widehat{\mathcal{SSV}}_i - E[\mathcal{SSV}_i]| \ge \epsilon)$$

$$\le 2\exp\left(-\frac{2\epsilon^2}{\sum_{t=1}^{\tau}(2r)^2}\right) = 2\exp\left(-\frac{\epsilon^2}{2\tau r^2}\right).$$

Then we fix the probability to be at most δ, we have

$$2\exp\left(-\frac{\epsilon^2}{2\tau r^2}\right) \le \delta \Rightarrow \tau \ge \frac{2\ln(2/\delta)r^2}{\epsilon^2}.$$

Move ϵ to the left side and we have $\epsilon \le \sqrt{2r^2\log(2/\delta)/\tau}$. This completes the proof. □

5.2 Utility Sampling

Although Monte Carlo sampling is simple to use, it does not fully exploit the shared utility computation between permutations. When we sample a permutation π with Algorithm 1 and calculate the marginal contribution of $z_{\pi(i)}$ for estimating $\mathcal{SSV}_{\pi(i)}$, i.e., $\mathcal{U}(\{z_{\pi(1)}, \ldots, z_{\pi(i)}\}) - \mathcal{U}(\{z_{\pi(1)}, \ldots, z_{\pi(i-1)}\})$, we evaluate the utilities of two coalitions $\{z_{\pi(1)}, \ldots, z_{\pi(i)}\}$ and $\{z_{\pi(1)}, \ldots, z_{\pi(i-1)}\}$. Since these utilities are also parts of other data owners' S-Shapley value, they can be reused for efficient computation as in Example 5.3.

Example 5.3. Consider Example 3.1's setting. When we scan z_3 in a sampled permutation of data sets $[z_2, z_1, z_3]$, $\mathcal{U}(\{z_1, z_2\})$ and $\mathcal{U}(\{z_1, z_2, z_3\})$ are evaluated for estimating $\mathcal{SSV}_3$. $\mathcal{U}(\{z_1, z_2\})$ can be reused for estimating $\mathcal{SSV}_2$ with $\mathcal{U}(\{z_1, z_2\}) - \mathcal{U}(\{z_1\})$.

In order to estimate S-Shapley value of multiple data owners simultaneously, we can treat S-Shapley value as the difference of two utility expectations and reuse utilities accordingly.

Theorem 5.4. *Given a sharded structure* $\mathcal{L}$, *the S-Shapley value of* z_i *is formed by two expectations of utilities* $\mathcal{SSV}_i^+$ *and* $\mathcal{SSV}_i^-$. *That is,*

$$\mathcal{SSV}_i = \underbrace{E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}_{\pi}^{z_i} \cup \{z_i\}\right)\right]}_{\mathcal{SSV}_i^+} - \underbrace{E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}_{\pi}^{z_i}\right)\right]}_{\mathcal{SSV}_i^-}. \quad (4)$$

Proof. There are two random variable in Equation 4, $\mathcal{U}\left(\mathcal{S}_{\pi}^{z_i} \cup \{z_i\}\right)$ and $\mathcal{U}\left(\mathcal{S}_{\pi}^{z_i}\right)$. Let $f\left(\mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i} \cup \{z_i\}\right), \mathcal{U}\left(\mathcal{S}_{\pi^{t'}}^{z_i}\right)\right)$ denotes the joint

probability function of these two random variables, then we have

$$\begin{aligned}
&\mathcal{SSV}_i = E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}_\pi^{z_i}\cup\{z_i\}\right)-\mathcal{U}\left(\mathcal{S}_\pi^{z_i}\right)\right]\\
&=\sum_t\sum_{t'}\left(\mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i}\cup\{z_i\}\right)-\mathcal{U}\left(\mathcal{S}_{\pi^{t'}}^{z_i}\right)\right)f\left(\mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i}\cup\{z_i\}\right),\mathcal{U}\left(\mathcal{S}_{\pi^{t'}}^{z_i}\right)\right)\\
&=\sum_j\sum_k\mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i}\cup\{z_i\}\right)f\left(\mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i}\cup\{z_i\}\right),\mathcal{U}\left(\mathcal{S}_{\pi^{t'}}^{z_i}\right)\right)\\
&\quad-\sum_j\sum_k\mathcal{U}\left(\mathcal{S}_{\pi^{t'}}^{z_i}\right)f\left(\mathcal{U}\left(\mathcal{S}_{\pi^t}^{z_i}\cup\{z_i\}\right),\mathcal{U}\left(\mathcal{S}_{\pi^{t'}}^{z_i}\right)\right)\\
&=E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}_\pi^{z_i}\cup\{z_i\}\right)\right]-E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}_\pi^{z_i}\right)\right].
\end{aligned}$$

□

We note that permutation π in $\mathcal{U}\left(\mathcal{S}_\pi^{z_i}\cup\{z_i\}\right)$ is not necessary the same as π in $\mathcal{U}\left(\mathcal{S}_\pi^{z_i}\right)$.

With Theorem 5.4, we estimate two utility expectations (i.e., $\mathcal{SSV}_i^+$ and $\mathcal{SSV}_i^-$) separately instead of directly estimating one marginal contribution expectation (i.e., $\mathcal{SSV}_i$) with Monte Carlo sampling. The range of utilities is generally larger than the range of marginal contributions which are differences between paired utilities. The more spread the data samples, the larger the variance is concerning the mean. Thus, although it enables us sufficiently reuse utilities, Theorem 5.4 enlarges the variance in estimation. To reduce the variance, we apply stratified sampling which divides utilities into homogeneous subgroups whose range is smaller in each subgroup and estimates them respectively.

Stratification. To appropriately stratify the utilities in $\mathcal{SSV}_i^+$ or $\mathcal{SSV}_i^-$ into homogeneous subgroups, we investigate the distribution of utilities. In traditional Shapley value computation, the strata are determined only by the number of data owners in a given coalition [6]. However, when computing S-Shapley value, using the same stratification approach leads to imbalanced strata and biased estimation. To this end, we considered not only the number of data owners but also the number of data shards in the coalition for stratification. To compute $\mathcal{SSV}_i^+$ or $\mathcal{SSV}_i^-$, a valid coalition must include the data of one or several data shards, and only the data shard containing z_i, denoted as $d[z_i]$, can be incomplete. Thus, we can categorize utilities into different strata based on the number of involved data shards in the corresponding coalition and the number of involved data owners in $d[z_i]$.

Definition 5.5. Given a sharded structure $\mathcal{L}$, denote by $d[z_i]$ the data shard containing z_i. If a coalition $\mathcal{S}$ involving b data shards contains z_i, for any data shard d_j ($d_j\neq d[z_i]$), $d_j\cap\mathcal{S}=d_j$ or $\emptyset$, and $|d[z_i]\cap\mathcal{S}|=c$ ($1\le c\le|d[z_i]|$), then $\mathcal{S}$ is called a (z_i,b,c)-coalition, where $|d[z_i]\cap\mathcal{S}|$ is the number of common data owners in $d[z_i]$ and $\mathcal{S}$, and $|d[z_i]|$ is the number of data owners in $d[z_i]$. Denote by $R^{z_i,b,c}$ the set of all (z_i,b,c)-coalitions, $\mathcal{SSV}_{z_i,b,c}^+$ the expected utilities of (z_i,b,c)-coalitions, and $\mathcal{SSV}_{z_i,b,c}^-$ the expected utilities of (z_i,b,c)-coalitions excluding z_i. That is,

$$\mathcal{SSV}_{z_i,b,c}^+=\sum_{\mathcal{S}\in R^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c-1}},\tag{5}$$

$$\mathcal{SSV}_{z_i,b,c}^-=\sum_{\mathcal{S}\in R^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S}\backslash\{z_i\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c-1}}.\tag{6}$$

We use Definition 5.5 to reformulate Equation 4.

Theorem 5.6. *Given a sharded structure $\mathcal{L}$, $\mathcal{SSV}_i^+=\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\mathcal{SSV}_{z_i,b,c}^+$ and $\mathcal{SSV}_i^-=\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\mathcal{SSV}_{z_i,b,c}^-$.*

Proof. We rewrite Equation 2 by regarding $\mathcal{SSV}_i$ as the weighted average of different coalitions' marginal contribution.

$$\begin{aligned}
\mathcal{SSV}_i&=\frac{1}{m|d[z_i]|}\sum_{\mathcal{DS}\subseteq L_1\backslash\{d[z_i]\}}\sum_{\mathcal{S}\subseteq d[z_i]\backslash\{z_i\}}\frac{\mathcal{U}(\mathcal{DS}\cup\mathcal{S}\cup\{z_i\})-\mathcal{U}(\mathcal{DS}\cup\mathcal{S})}{\binom{m-1}{|\mathcal{DS}|}\binom{|d[z_i]|-1}{|\mathcal{S}|}}\\
&=\underbrace{\frac{1}{m|d[z_i]|}\sum_{\mathcal{DS}\subseteq L_1\backslash\{d[z_i]\}}\sum_{\mathcal{S}\subseteq d[z_i]\backslash\{z_i\}}\frac{\mathcal{U}(\mathcal{DS}\cup\mathcal{S}\cup\{z_i\})}{\binom{m-1}{|\mathcal{DS}|}\binom{|d[z_i]|-1}{|\mathcal{S}|}}}_{\mathcal{SSV}_i^+}\\
&\quad-\underbrace{\frac{1}{m|d[z_i]|}\sum_{\mathcal{DS}\subseteq L_1\backslash\{d[z_i]\}}\sum_{\mathcal{S}\subseteq d[z_i]\backslash\{z_i\}}\frac{\mathcal{U}(\mathcal{DS}\cup\mathcal{S})}{\binom{m-1}{|\mathcal{DS}|}\binom{|d[z_i]|-1}{|\mathcal{S}|}}}_{\mathcal{SSV}_i^-},
\end{aligned}$$

where $\mathcal{DS}\subseteq L_1\backslash\{d[z_i]\}$ is a coalition of different data shards, $|\mathcal{DS}|$ is the number of data shards in $\mathcal{DS}$ ($0\le|\mathcal{DS}|\le m-1$), $\mathcal{S}\subseteq d[z_i]\backslash\{z_i\}$ is a coalition of data owners, and $|\mathcal{S}|$ is the number of data owners in $\mathcal{S}$ ($0\le\mathcal{S}\le|d[z_i]|-1$).

For $\mathcal{SSV}_i^+$, $\mathcal{DS}\subseteq L_1\backslash\{d[z_i]\}$, $\mathcal{S}\subseteq d[z_i]\backslash\{z_i\}$, any coalition $\mathcal{DS}\cup\mathcal{S}\cup\{z_i\}\in R^{z_i,|\mathcal{DS}|+1,|\mathcal{S}|+1}$ and any coalition $\mathcal{S}'\in R^{z_i,|\mathcal{DS}|+1,|\mathcal{S}|+1}$ has a corresponding coalition $\mathcal{DS}\cup\mathcal{S}\cup\{z_i\}$. we have

$$\mathcal{SSV}_i^+=\frac{1}{m|d[z_i]|}\sum_{\mathcal{S}'\in R^{z_i,|\mathcal{DS}|+1,|\mathcal{S}|+1}}\frac{\mathcal{U}(\mathcal{DS}\cup\mathcal{S}\cup\{z_i\})}{\binom{m-1}{|\mathcal{DS}|}\binom{|d[z_i]|-1}{|\mathcal{S}|}},$$

where the data owners in $\mathcal{S}'$ and $\mathcal{DS}\cup\mathcal{S}\cup\{z_i\}$ are the same. Then we get

$$\mathcal{SSV}_i^+=\frac{1}{m|d[z_i]|}\sum_{\mathcal{S}'\in R^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S}')}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c-1}},$$

where $1\le b\le m$, $1\le c\le|d[z_i]|$.

Similarly for $\mathcal{SSV}_i^-$, we have

$$\mathcal{SSV}_i^-=\frac{1}{m|d[z_i]|}\sum_{\mathcal{S}'\in R^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S}'\backslash\{z_i\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c-1}},$$

where $1\le b\le m$, $1\le c\le|d[z_i]|$. □

According to Theorem 5.6, approximating $\mathcal{SSV}_i^+$ (similarly for $\mathcal{SSV}_i^-$) becomes a stratified sampling process. The stratification design is to divide all utilities in $\mathcal{SSV}_i^+$ into $m|d[z_i]|$ strata such that utilities of all (z_i,b,c)-coalitions are in the $(b|d[z_i]|+c)^{th}$ stratum. Then, to approximate $\mathcal{SSV}_i^+$, we can first estimate $\mathcal{SSV}_{z_i,b,c}^+$ by sampling with replacement. Let $\mathcal{U}_{z_i,b,c}^+$ be a random variable with uniform distribution on set $\{\mathcal{U}(\mathcal{S})|\mathcal{S}\in R^{z_i,b,c}\}$. The expectation of $\mathcal{U}_{z_i,b,c}^+$ is $\mathcal{SSV}_{z_i,b,c}^+$. Given $\tau_{z_i,b,c}^+$ samples of $\mathcal{U}_{z_i,b,c}^+$, $\{\mathcal{U}(\mathcal{S}_1),\dots,\mathcal{U}(\mathcal{S}_{\tau_{z_i,b,c}^+})\}$, where $\mathcal{S}_1,\dots,\mathcal{S}_{\tau_{z_i,b,c}^+}\in R^{z_i,b,c}$, the mean over $\mathcal{U}(\mathcal{S}_1),\dots,\mathcal{U}(\mathcal{S}_{\tau_{z_i,b,c}^+})$ is an estimation of $\mathcal{SSV}_{z_i,b,c}^+$, $\widehat{\mathcal{SSV}}_{z_i,b,c}^+$

Algorithm 3: Utility Sampling Algorithm.

input : data sets from data owners $\{z_1, \ldots, z_n\}$, sharded structure $\mathcal{L}$, the number of permutations τ

output: estimated S-Shapley value $\widehat{\mathcal{SSV}}_i$ for each z_i $(1 \le i \le n)$

1 $\widehat{\mathcal{SSV}}_i \leftarrow 0$ $(1 \le i \le n)$; $\widehat{\mathcal{SSV}}^+_{z_i,b,c}, \widehat{\mathcal{SSV}}^-_{z_i,b,c}, \tau^+_{z_i,b,c}, \tau^-_{z_i,b,c} \leftarrow 0$ $(1 \le i \le n, 1 \le b \le m, 1 \le c \le |d[z_i]|)$;
2 **for** *t=1 to* τ **do**
3 $\pi^t \leftarrow$ **Sample**$(\mathcal{L})$;
4 let i be a random value drawn from $1, \ldots, n$;
5 $\mathcal{S} \leftarrow \{z_{\pi^t(1)}, \ldots, z_{\pi^t(i)}\}$;
6 $u \leftarrow \mathcal{U}(\mathcal{S})$;
7 $b \leftarrow 0$;
8 **for** *j=1 to m* **do**
9 **if** $\mathcal{S} \cap d_j \neq \emptyset$ **then**
10 b+=1;
11 $c \leftarrow |\mathcal{S} \cap d[z_{\pi^t(i)}]|$;
12 **if** $c = |d[z_{\pi^t(i)}]|$ **then**
13 **for** *i'=1 to n* **do**
14 **if** $z_{i'} \in \mathcal{S}$ **then**
15 $\widehat{\mathcal{SSV}}^+_{z_{i'},b,|d[z_{i'}]|} += u$; $\tau^+_{z_{i'},b,|d[z_{i'}]|} += 1$;
16 **else**
17 $\widehat{\mathcal{SSV}}^-_{z_{i'},b+1,1} += u$; $\tau^-_{z_{i'},b+1,1} += 1$;
18 **else**
19 **foreach** $z_{i'} \in d[z_{\pi^t(i)}]$ **do**
20 **if** $z_{i'} \in \mathcal{S}$ **then**
21 $\widehat{\mathcal{SSV}}^+_{z'_i,b,c} += u$; $\tau^+_{z'_i,b,c} += 1$;
22 **else**
23 $\widehat{\mathcal{SSV}}^-_{z'_i,b,c+1} += u$; $\tau^-_{z'_i,b,c+1} += 1$;
24 **for** *i=1 to n* **do**
25 $\widehat{\mathcal{SSV}}_i = \frac{1}{m|d[z_i]|} \sum_{b=1}^{m} \sum_{c=1}^{|d[z_i]|} [\widehat{\mathcal{SSV}}^+_{z_i,b,c}/\tau^+_{z_i,b,c} - \widehat{\mathcal{SSV}}^-_{z_i,b,c}/\tau^-_{z_i,b,c}]$;
26 **return** $\widehat{\mathcal{SSV}}_1, \ldots, \widehat{\mathcal{SSV}}_n$;

$= \frac{1}{\tau^+_{z_i,b,c}} \sum_{t=1}^{\tau^+_{z_i,b,c}} \mathcal{U}(\mathcal{S}_t)$. Then, an estimation of $\mathcal{SSV}^+_i$ is $\widehat{\mathcal{SSV}}^+_i = \frac{1}{m|d[z_i]|} \sum_{b=1}^{m} \sum_{c=1}^{|d[z_i]|} \widehat{\mathcal{SSV}}^+_{z_i,b,c}$. Finally, we get an estimation of $\mathcal{SSV}_i$ by $\widehat{\mathcal{SSV}}^+_i - \widehat{\mathcal{SSV}}^-_i$.

Algorithm 3 outlines the pseudo-code for estimating $\mathcal{SSV}_i$ $(1 \le i \le n)$ using utility sampling. The first step is to randomly generate a coalition $\mathcal{S}$, calculate the corresponding utility u, assign the value u to the relevant $\mathcal{SSV}^+_{z_i,b,c}$ (resp. $\mathcal{SSV}^-_{z_i,b,c}$), and update the associated counts of $\tau^+_{z_i,b,c}$ (resp. $\tau^-_{z_i,b,c}$) (Lines 3-23). After τ samples have been drawn, the estimation of $\mathcal{SSV}^+_i$ or $\mathcal{SSV}^-_i$ is obtained as the average of the corresponding utility means. The final estimation of $\mathcal{SSV}$ is obtained as the difference between $\widehat{\mathcal{SSV}}^+_i$ and $\widehat{\mathcal{SSV}}^-_i$ (Lines 24-25). It is worth noting that the estimation of $\mathcal{SSV}_i$ in Algorithm 3 is unbiased.

Theorem 5.7. *Algorithm 3 gives an unbiased estimation of* $\mathcal{SSV}_i$ $(1 \le i \le n)$, *i.e.,* $E[\widehat{\mathcal{SSV}}_i] = \mathcal{SSV}_i$.

Proof. Denote by $\{\mathcal{U}(\mathcal{S}_1), \ldots, \mathcal{U}(\mathcal{S}_{\tau^+_{z_i,b,c}})\}$ $\tau^+_{z_i,b,c}$ samples of $\mathcal{U}^+_{z_i,b,c}$ $(1 \le i \le n, 1 \le b \le m, 1 \le c \le |d[z_i]|)$ drawn by Algorithm 3. The expectation of the samples $\widehat{\mathcal{SSV}}^+_{z_i,b,c} = \frac{1}{\tau^+_{z_i,b,c}} \sum_{t=1}^{\tau^+_{z_i,b,c}} \mathcal{U}(\mathcal{S}_t)$. We have

$$E[\widehat{\mathcal{SSV}}^+_{z_i,b,c}] = E[\frac{1}{\tau^+_{z_i,b,c}} \sum_{t=1}^{\tau^+_{z_i,b,c}} \mathcal{U}(\mathcal{S}_t)] = \frac{1}{\tau^+_{z_i,b,c}} \sum_{t=1}^{\tau^+_{z_i,b,c}} E[\mathcal{U}(\mathcal{S}_t)].$$

According to Equation 5, $E[\mathcal{U}(\mathcal{S}_t)] = \mathcal{SSV}^+_{z_i,b,c}$. Thus, $E[\widehat{\mathcal{SSV}}^+_{z_i,b,c}] = \mathcal{SSV}^+_{z_i,b,c}$.

Now, consider the estimate $\widehat{\mathcal{SSV}}_i$ produced by Algorithm 3. We have

$$\begin{aligned} E[\widehat{\mathcal{SSV}}^+_i] &= E[\frac{1}{m|d[z_i]|} \sum_{b=1}^{m} \sum_{c=1}^{|d[z_i]|} \widehat{\mathcal{SSV}}^+_{z_i,b,c}] \\ &= \frac{1}{m|d[z_i]|} \sum_{b=1}^{m} \sum_{c=1}^{|d[z_i]|} E[\widehat{\mathcal{SSV}}^+_{z_i,b,c}] \\ &= \frac{1}{m|d[z_i]|} \sum_{b=1}^{m} \sum_{c=1}^{|d[z_i]|} \mathcal{SSV}^+_{z_i,b,c} = \mathcal{SSV}^+_i. \end{aligned}$$

Similarly, we have $E[\widehat{\mathcal{SSV}}^-_i] = \mathcal{SSV}^-_i$. Thus,

$$\begin{aligned} &E[\widehat{\mathcal{SSV}}_i] = E[\widehat{\mathcal{SSV}}^+_i - \widehat{\mathcal{SSV}}^-_i] = E[\widehat{\mathcal{SSV}}^+_i] - E[\widehat{\mathcal{SSV}}^-_i] \\ &= \mathcal{SSV}^+_i - \mathcal{SSV}^-_i = \mathcal{SSV}_i \end{aligned}$$

That is, $\widehat{\mathcal{SSV}}_i$ is an unbiased estimation of $\mathcal{SSV}_i$. □

Theorem 5.8. *According to Hoeffding's inequality, given the range of utilities* r *and the minimum value of sample size for utility in different strata* τ, *the error* $|\widehat{\mathcal{SSV}}_i - \mathcal{SSV}_i| \le 2\sqrt{2r^2 \log(2/\delta)/\tau}$ *in probability* $1 - \delta$.

Proof. According to Theorems 5.4 and 5.6, the application of stratification to utility sampling results in the estimation of $2m|d[z_i]|$ variables when estimating a single variable $\mathcal{SSV}_i$. Let ϕ_t denote the t^{th} $(1 \le t \le \tau)$ utility sample in a specific stratum corresponding to $\mathcal{SSV}^+_{z_i,b,c}$. Let all utilities be in the range $[-r, r]$ $(r \ge 0)$. According to Hoeffding's inequality, we have

$$\begin{aligned} &Pr(|\tau\widehat{\mathcal{SSV}}^+_{z_i,b,c} - \sum_{t=1}^{\tau} E[\phi_t]| \ge \tau\epsilon) \\ =&Pr(|\widehat{\mathcal{SSV}}^+_{z_i,b,c} - E[\mathcal{SSV}^+_{z_i,b,c}]| \ge \epsilon) \\ \le&2\exp\left(-\frac{2\epsilon^2}{\sum_{t=1}^{\tau}(2r)^2}\right) = 2\exp\left(-\frac{\epsilon^2}{2\tau r^2}\right). \end{aligned}$$

Then we fix the probability to be at most δ, we have

$$2\exp\left(-\frac{\epsilon^2}{2\tau r^2}\right) \le \delta \Rightarrow \tau \ge \frac{2\ln(2/\delta)r^2}{\epsilon^2}.$$

By moving ϵ to the left side, we obtain the inequality $\epsilon \le \sqrt{2r^2\log(2/\delta)/\tau}$. It is worth noting that τ represents the minimum sample size required for utility estimation across different strata. Consequently, the estimation error of each of the $2m|d[z_i]|$ variables is bounded by $\sqrt{2r^2\log(2/\delta)/\tau}$. According to Theorem 5.6 and triangle inequality, we have

$$\begin{aligned}
&|\widehat{\mathcal{SSV}}_i^+ - \mathcal{SSV}_i^+| \\
=&\left|\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}(\widehat{\mathcal{SSV}}^+_{z_i,b,c} - \mathcal{SSV}^+_{z_i,b,c})\right| \\
\le&\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\left|\widehat{\mathcal{SSV}}^+_{z_i,b,c} - \mathcal{SSV}^+_{z_i,b,c}\right| \le \sqrt{2r^2\log(2/\delta)/\tau}.
\end{aligned}$$

Similarly, we have

$$|\widehat{\mathcal{SSV}}_i^- - \mathcal{SSV}_i^-| \le \sqrt{2r^2\log(2/\delta)/\tau}.$$

According to Equation 4, we have

$$\begin{aligned}
&|\widehat{\mathcal{SSV}}_i - \mathcal{SSV}_i| \le |\widehat{\mathcal{SSV}}_i^+ - \mathcal{SSV}_i^+| + |\widehat{\mathcal{SSV}}_i^- - \mathcal{SSV}_i^-| \\
\le&2\sqrt{2r^2\log(2/\delta)/\tau}.
\end{aligned}$$

This completes the proof. □

6 UPDATING S-SHAPLEY VALUE UPON UNLEARNING

In this section, we propose two approximation algorithms to update S-Shapley value when one (Section 6.1) or multiple data owners exit the model market (Section 6.2), respectively.

6.1 Single Data Owner Exit

Once a data owner wants to exit the model market and unlearn her data, S-Shapley value needs to be updated to reflect the latest contributions of the remaining data owners. Rather than recomputing the S-Shapley value from scratch by Algorithm 2 or 3, which can be time-consuming upon each unlearning request, we introduce a method that requires a smaller sample to achieve the same accuracy. Specifically, we represent the difference between the new $\mathcal{SSV}_i^+$ (resp. new $\mathcal{SSV}_i^-$) and the precomputed $\mathcal{SSV}_i^+$ (resp. precomputed $\mathcal{SSV}_i^-$) using the differential utility. It is noteworthy that the magnitude of the differential utility, indicating the change in utility, is typically smaller than that of the utility function. This property enables us to attain stability using a smaller sample size, as anticipated by Hoeffding's inequality [22] when updating S-Shapley value.

Given the precomputed $\mathcal{SSV}_i$ $(1 \le i \le n)$ which is the difference of $\mathcal{SSV}_i^+$ and $\mathcal{SSV}_i^-$, the key idea is to compute the relative changes of $\mathcal{SSV}_i^+$ and $\mathcal{SSV}_i^-$, respectively. The difference between the precomputed $\mathcal{SSV}^+$ (resp. $\mathcal{SSV}^-$) and the new $\mathcal{SSV}^+$ (resp. $\mathcal{SSV}^-$) can be represented formally as Lemma 6.1.

Lemma 6.1. *Given a sharded structure $\mathcal{L}$, suppose that z_p is the data of exited data owner. Defining $R^{z_i,b,c}_{-z_p} = \{\mathcal{S} | \mathcal{S} \in R^{z_i,b,c}, z_p \notin \mathcal{S}\}$ $(1 \le i \le n, i \ne p)$, the difference between the new $\mathcal{SSV}_i^+$ and the precomputed $\mathcal{SSV}_i^+$ of z_i is*

$$\Delta\mathcal{SSV}_i^+ = \begin{cases} \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{c}{|d[z_i]|-1}\Delta\mathcal{SSV}^+_{z_i,b,c} & \text{if } z_p \in d[z_i], \\ \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{b}{m-1}\Delta\mathcal{SSV}^+_{z_i,b,c} & \text{otherwise,} \end{cases}$$

where

$$\Delta\mathcal{SSV}^+_{z_i,b,c} = \begin{cases} \sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}} \frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-2}{c-1}} & \text{if } z_p \in d[z_i], \\ \sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}} \frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup d[z_p])}{\binom{m-2}{b-1}\binom{|d[z_i]|-1}{c-1}} & \text{otherwise.} \end{cases}$$

The difference between the new $\mathcal{SSV}^-$ and the precomputed $\mathcal{SSV}^-$ of z_i is

$$\Delta\mathcal{SSV}_i^- = \begin{cases} \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{c}{|d[z_i]|-1}\Delta\mathcal{SSV}^-_{z_i,b,c} & \text{if } z_p \in d[z_i], \\ \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{b}{m-1}\Delta\mathcal{SSV}^-_{z_i,b,c} & \text{otherwise,} \end{cases}$$

where

$$\Delta\mathcal{SSV}^-_{z_i,b,c} = \begin{cases} \sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}} \frac{\mathcal{U}(\mathcal{S}\setminus\{z_i\})-\mathcal{U}(\mathcal{S}\setminus\{z_i\}\cup\{z_p\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-2}{c-1}} & \text{if } z_p \in d[z_i], \\ \sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}} \frac{\mathcal{U}(\mathcal{S}\setminus\{z_i\})-\mathcal{U}(\mathcal{S}\setminus\{z_i\}\cup d[z_p])}{\binom{m-2}{b-1}\binom{|d[z_i]|-1}{c-1}} & \text{otherwise.} \end{cases}$$

Proof. When a data set z_p is removed, the change of $\mathcal{SSV}_i$ $(1 \le i \le n, i \ne q)$ can be divided into two cases: (i) z_p and z_i are in the same data shard, $z_p \in d[z_i]$; (ii) z_p and z_i are in different data shards, $z_p \notin d[z_i]$.

In the first case, the change of $\mathcal{SSV}_i^+$ can be formulated as follows.

$$\begin{aligned}
\Delta\mathcal{SSV}_i^+ =& \frac{1}{m(|d[z_i]|-1)}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|-1}\sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}}\frac{\mathcal{U}(\mathcal{S})}{\binom{m-1}{b-1}\binom{|d[z_i]|-2}{c-1}} \\
&-\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c-1}} \\
=&\frac{1}{m(|d[z_i]|-1)}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|-1}\sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}}\frac{\mathcal{U}(\mathcal{S})}{\binom{m-1}{b-1}\binom{|d[z_i]|-2}{c-1}} \\
&-\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|-1}\sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}}\frac{\mathcal{U}(\mathcal{S})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c-1}} \\
&-\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|-1}\sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}}\frac{\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c}} \\
=&\frac{1}{m}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|-1}\sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}}\frac{(c-1)!(|d[z_i]|-c)!}{|d[z_i]|!}\frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-1}{b-1}} \\
&-\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|-1}\sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}}\frac{\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c}} \\
=&\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|-1}\sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}}\frac{c}{|d[z_i]|-1}\frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-2}{c-1}} \\
=&\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R^{z_i,b,c}_{-z_p}}\frac{c}{|d[z_i]|-1}\frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-2}{c-1}}.
\end{aligned}$$

In the second case, the change of $\mathcal{SSV}_i^+$ can be formulated as follows.

$$\Delta\mathcal{SSV}_i^+ = \frac{1}{(m-1)|d[z_i]|}\sum_{b=1}^{m-1}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S})}{\binom{m-2}{b-1}\binom{|d[z_i]|-1}{c-1}}$$
$$-\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c-1}}$$
$$=\frac{1}{(m-1)|d[z_i]|}\sum_{b=1}^{m-1}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S})}{\binom{m-2}{b-1}\binom{|d[z_i]|-1}{c-1}}$$
$$-\frac{1}{m|d[z_i]|}\sum_{b=1}^{m-1}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c-1}}$$
$$-\frac{1}{m|d[z_i]|}\sum_{b=1}^{m-1}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c}}$$

$$=\frac{1}{|d[z_i]|}\sum_{b=1}^{m-1}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{(b-1)!(m-b)!}{m!}\frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup d[z_p])}{\binom{|d[z_i]|-1}{c-1}}$$
$$-\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S}\cup d[z_p])}{\binom{m-1}{b-1}\binom{|d[z_i]|-1}{c}}$$
$$=\frac{1}{m|d[z_i]|}\sum_{b=1}^{m-1}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{b}{m-1}\frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-2}{b-1}\binom{|d[z_i]|-1}{c-1}}$$
$$=\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{b}{m-1}\frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-2}{b-1}\binom{|d[z_i]|-1}{c-1}}.$$

Let

$$\Delta\mathcal{SSV}_{z_i,b,c}^+ = \begin{cases}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup\{z_p\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-2}{c-1}} & \text{if } z_p\in d[z_i],\\ \sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup d[z_p])}{\binom{m-2}{b-1}\binom{|d[z_i]|-1}{c-1}} & \text{otherwise.}\end{cases}$$

Thus

$$\Delta\mathcal{SSV}_i^+ = \begin{cases}\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{c}{|d[z_i]|-1}\Delta\mathcal{SSV}_{z_i,b,c}^+ & \text{if } z_p\in d[z_i],\\ \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{b}{m-1}\Delta\mathcal{SSV}_{z_i,b,c}^+ & \text{otherwise.}\end{cases}$$

Similarly, let

$$\Delta\mathcal{SSV}_{z_i,b,c}^- = \begin{cases}\sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S}\setminus\{z_i\})-\mathcal{U}(\mathcal{S}\setminus\{z_i\}\cup\{z_p\})}{\binom{m-1}{b-1}\binom{|d[z_i]|-2}{c-1}} & \text{if } z_p\in d[z_i],\\ \sum_{\mathcal{S}\in R_{-z_p}^{z_i,b,c}}\frac{\mathcal{U}(\mathcal{S}\setminus\{z_i\})-\mathcal{U}(\mathcal{S}\setminus\{z_i\}\cup d[z_p])}{\binom{m-2}{b-1}\binom{|d[z_i]|-1}{c-1}} & \text{otherwise.}\end{cases}$$

We have

$$\Delta\mathcal{SSV}_i^- = \begin{cases}\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{c}{|d[z_i]|-1}\Delta\mathcal{SSV}_{z_i,b,c}^- & \text{if } z_p\in d[z_i],\\ \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{b}{m-1}\Delta\mathcal{SSV}_{z_i,b,c}^- & \text{otherwise.}\end{cases}$$

□

According to Lemma 6.1, approximating $\Delta\mathcal{SSV}_i^+$ (similarly for $\Delta\mathcal{SSV}_i^-$) becomes a stratified sampling process. The stratification design is to divide all differential utilities in $\Delta\mathcal{SSV}_i^+$ into $m|d[z_i]|$ strata such that the utility changes from z_p leaving (z_i,b,c)-coalitions are in the $(b|d[z_i]|+c)^{th}$ stratum. Then, to approximate $\Delta\mathcal{SSV}_i^+$, we can first estimate $\Delta\mathcal{SSV}_{z_i,b,c}^+$ by sampling a permutation in $\mathfrak{S}_n(\mathcal{L})$ and enumerating all possible positions of z_p. Let $\Delta\mathcal{U}_{z_i,b,c}^+$ be a random variable with uniform distribution on the set $\{\mathcal{U}(\mathcal{S})-\mathcal{U}(\mathcal{S}\cup\{z_p\})|\mathcal{S}\in R_{-z_p}^{z_i,b,c}\}$. The expectation of $\Delta\mathcal{U}_{z_i,b,c}^+$ is $\Delta\mathcal{SSV}_{z_i,b,c}^+$. Given $\tau_{z_i,b,c}^+$ samples of $\Delta\mathcal{U}_{z_i,b,c}^+$, $\{\mathcal{U}(\mathcal{S}_1)-\mathcal{U}(\mathcal{S}_1\cup\{z_p\}),\ldots,\mathcal{U}(\mathcal{S}_{\tau_{z_i,b,c}^+})-\mathcal{U}(\mathcal{S}_{\tau_{z_i,b,c}^+}\cup\{z_p\})\}$, where $\mathcal{S}_1,\ldots,\mathcal{S}_{\tau_{z_i,b,c}^+}\in R_{-z_p}^{z_i,b,c}$, the mean over $\mathcal{U}(\mathcal{S}_1)-\mathcal{U}(\mathcal{S}_1\cup\{z_p\}),\ldots,\mathcal{U}(\mathcal{S}_{\tau_{z_i,b,c}^+})-\mathcal{U}(\mathcal{S}_{\tau_{z_i,b,c}^+}\cup\{z_p\})$ is an estimation of $\Delta\mathcal{SSV}_{z_i,b,c}^+$, $\widehat{\Delta\mathcal{SSV}}_{z_i,b,c}^+ = \frac{1}{\tau_{z_i,b,c}^+}\sum_{t=1}^{\tau_{z_i,b,c}^+}[\mathcal{U}(\mathcal{S}_t)-\mathcal{U}(\mathcal{S}_t\cup\{z_p\})]$. Then, if $z_p\in d[z_i]$ (resp. $z_p\notin d[z_i]$), an estimation of $\Delta\mathcal{SSV}_i^+$ is $\widehat{\Delta\mathcal{SSV}}_i^+ = \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{c}{|d[z_i]|-1}\widehat{\Delta\mathcal{SSV}}_{z_i,b,c}^+$ (resp. $\widehat{\Delta\mathcal{SSV}}_i^- = \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{b}{m-1}\widehat{\Delta\mathcal{SSV}}_{z_i,b,c}^-$). Finally, we get an estimation of $\Delta\mathcal{SSV}_i$ by $\widehat{\Delta\mathcal{SSV}}_i^+ - \widehat{\Delta\mathcal{SSV}}_i^-$. The precomputed $\mathcal{SSV}_i$ can be updated by adding the estimation of $\Delta\mathcal{SSV}_i$.

Algorithm 4 outlines the pseudo-code for estimating the change of $\mathcal{SSV}_i$ $(1\le i\le n, i\ne p)$ to update $\mathcal{SSV}_i$ when a single data owner with data set z_p exits. The first step is to randomly generate a coalition $\mathcal{S}$ excluding z_p, calculate the corresponding differential utility Δu, assign the value Δu to the relevant $\Delta\mathcal{SSV}_{z_i,b,c}^+$ (resp. $\Delta\mathcal{SSV}_{z_i,b,c}^-$), and update the associated counts of $\tau_{z_i,b,c}^+$ (resp. $\tau_{z_i,b,c}^-$) (Lines 3-11). After τ samples have been drawn, the estimation of $\Delta\mathcal{SSV}_i^+$ or $\Delta\mathcal{SSV}_i^-$ is obtained as the weighted average of the corresponding utility means. The final estimation of $\Delta\mathcal{SSV}_i$ is obtained as the difference between $\widehat{\Delta\mathcal{SSV}}_i^+$ and $\widehat{\Delta\mathcal{SSV}}_i^-$ (Lines 12-16). It is worth noting that the estimation of $\Delta\mathcal{SSV}_i$ in Algorithm 4 is unbiased.

Theorem 6.2. *Given the exited data owner with z_p, Algorithm 4 gives an unbiased estimation of $\Delta\mathcal{SSV}_i$ $(1\le i\le n, i\ne p)$, i.e., $\boldsymbol{E}[\widehat{\Delta\mathcal{SSV}}_i]=\Delta\mathcal{SSV}_i$.*

Proof. Denote by $\{\mathcal{U}(\mathcal{S}_1)-\mathcal{U}(\mathcal{S}_1\cup\{z_p\}),\ldots,\mathcal{U}(\mathcal{S}_{\tau_{z_i,b,c}^+})-\mathcal{U}(\mathcal{S}_{\tau_{z_i,b,c}^+}\cup\{z_p\})\}$ $\tau_{z_i,b,c}^+$ samples of $\Delta\mathcal{U}_{z_i,b,c}^+$ $(1\le i\le n, 1\le b\le m, 1\le c\le|d[z_i]|)$ drawn by Algorithm 4. The expectation of the samples $\widehat{\Delta\mathcal{SSV}}_{z_i,b,c}^+ = \frac{1}{\tau_{z_i,b,c}^+}\sum_{t=1}^{\tau_{z_i,b,c}^+}\Delta\mathcal{U}(\mathcal{S}_t)$. We have

$$E[\widehat{\Delta\mathcal{SSV}}_{z_i,b,c}^+] = E[\frac{1}{\tau_{z_i,b,c}^+}\sum_{t=1}^{\tau_{z_i,b,c}^+}\Delta\mathcal{U}(\mathcal{S}_t)] = \frac{1}{\tau_{z_i,b,c}^+}\sum_{t=1}^{\tau_{z_i,b,c}^+}E[\Delta\mathcal{U}(\mathcal{S}_t)].$$

According to Lemma 6.1, $E[\Delta\mathcal{U}(\mathcal{S}_t)]=\Delta\mathcal{SSV}_{z_i,b,c}^+$. Thus, $E[\widehat{\Delta\mathcal{SSV}}_{z_i,b,c}^+]=\Delta\mathcal{SSV}_{z_i,b,c}^+$.

Now, consider the estimate $\widehat{\Delta\mathcal{SSV}_i}$ produced by Algorithm 4. If $z_p \in d[z_i]$, we have

$$\begin{aligned}E[\widehat{\Delta\mathcal{SSV}_i^+} =& E[\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{c}{|d[z_i]|-1}\widehat{\Delta\mathcal{SSV}}^+_{z_i,b,c}]\\ =&\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{c}{|d[z_i]|-1}E[\widehat{\Delta\mathcal{SSV}}^+_{z_i,b,c}]\\ =&\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{c}{|d[z_i]|-1}\Delta\mathcal{SSV}^+_{z_i,b,c}=\Delta\mathcal{SSV}_i^+.\end{aligned}$$

If $z_p \notin d[z_i]$, we have

$$\begin{aligned}E[\widehat{\Delta\mathcal{SSV}_i^+} =& E[\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{b}{m-1}\widehat{\Delta\mathcal{SSV}}^+_{z_i,b,c}]\\ =&\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{b}{m-1}E[\widehat{\Delta\mathcal{SSV}}^+_{z_i,b,c}]\\ =&\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{b}{m-1}\Delta\mathcal{SSV}^+_{z_i,b,c}=\Delta\mathcal{SSV}_i^+.\end{aligned}$$

Similarly, we have $E[\widehat{\Delta\mathcal{SSV}_i^-}] = \Delta\mathcal{SSV}_i^-$. Thus,

$$\begin{aligned}&E[\widehat{\Delta\mathcal{SSV}_i}] = E[\widehat{\Delta\mathcal{SSV}_i^+} - \widehat{\Delta\mathcal{SSV}_i^-}] = E[\widehat{\Delta\mathcal{SSV}_i^+}] - E[\widehat{\Delta\mathcal{SSV}_i^-}]\\ =&\Delta\mathcal{SSV}_i^+ - \Delta\mathcal{SSV}_i^- = \Delta\mathcal{SSV}_i\end{aligned}$$

That is, $\widehat{\Delta\mathcal{SSV}_i}$ is an unbiased estimation of $\Delta\mathcal{SSV}_i$. □

Corollary 6.3. *Given the precomputed S-Shapley value, Algorithm 4 can provide an unbiased estimation of the updated S-Shapley value and preserve P1-P4 in Section 3.3 in expectation.*

Proof. The precomputed S-Shapley value $\widehat{\mathcal{SSV}_i}$ is an unbiased estimator of S-Shapley value of original data owners. According to Theorem 6.2, the estimated change of S-Shapley value is also unbiased. Given two unbiased estimators $\widehat{\mathcal{SSV}_i}$ and $\widehat{\Delta\mathcal{SSV}_i}$ for estimating the new S-Shapley value $\mathcal{NSSV}_i$, we want to show that the sum of the estimators, denoted as $\widehat{\mathcal{NSSV}_i} = \widehat{\mathcal{SSV}_i} + \widehat{\Delta\mathcal{SSV}_i}$, is also unbiased. That is, we want to show

$$E[\widehat{\mathcal{NSSV}_i}] = \mathcal{NSSV}_i.$$

To prove this, we can use the linearity of expectation to write

$$E[\widehat{\mathcal{NSSV}_i}] = E[\widehat{\mathcal{SSV}_i}] + E[\widehat{\Delta\mathcal{SSV}_i}].$$

Since $\widehat{\mathcal{SSV}_i}$ and $\widehat{\Delta\mathcal{SSV}_i}$ are unbiased estimators, we have

$$E[\widehat{\mathcal{SSV}_i}] = \mathcal{SSV}_i \quad \text{and} \quad E[\widehat{\Delta\mathcal{SSV}_i}] = \Delta\mathcal{SSV}_i.$$

According to Lemma 6.1, we have $\Delta\mathcal{SSV}_i = \mathcal{NSSV}_i - \mathcal{SSV}_i$. Substituting these values into the equation above, we have

$$E[\widehat{\mathcal{NSSV}_i}] = \mathcal{SSV}_i + \Delta\mathcal{SSV}_i = \mathcal{NSSV}_i.$$

Algorithm 4: Delta-based Algorithm.

input : data sets of data owners $\{z_1, \ldots, z_n\}$, the index of exited data owner p, sharded structure $\mathcal{L}$, the number of permutations τ

output : $\widehat{\Delta\mathcal{SSV}_i}$ for each z_i $(1 \le i \le n, i \ne p)$

$\widehat{\Delta\mathcal{SSV}_i} \leftarrow 0$ $(1 \le i \le n)$;
$\widehat{\Delta\mathcal{SSV}}^+_{z_i,b,c}, \widehat{\Delta\mathcal{SSV}}^-_{z_i,b,c}, \tau^+_{z_i,b,c}, \tau^-_{z_i,b,c} \leftarrow 0$ $(1 \le i \le n, 1 \le b \le m, 1 \le c \le |d[z_i]|)$;
for *t=1 to τ* **do**
  $\pi^t \leftarrow$ **Sample**$(\mathcal{L})$ removing p;
  let i be a random value drawn from $1, \ldots, n-1$;
  $\mathcal{S} \leftarrow \{z_{\pi^t(1)}, \ldots, z_{\pi^t(i)}\}$;
  if $z_p \in d[z_{\pi^t(i)}]$ **then**
    $\Delta u = \mathcal{U}(\mathcal{S}) - \mathcal{U}(\mathcal{S} \cup \{z_p\})$;
  else
    $\Delta u = \mathcal{U}(\mathcal{S} \backslash d[z_p]) - \mathcal{U}(\mathcal{S} \cup d[z_p])$;
  compute b and c as Algorithm 3;
  add Δu to the corresponding stratum of $\mathcal{U}(\mathcal{S})$ in Algorithm 3 and update associated counts;
for *i=1 to n* **do**
  if $z_p \in d[z_i]$ **then**
    $\widehat{\Delta\mathcal{SSV}_i} = \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{c}{|d[z_i]|-1}$
    $\left(\frac{\widehat{\Delta\mathcal{SSV}}^+_{z_i,b,c}}{\tau^+_{z_i,b,c}} - \frac{\widehat{\Delta\mathcal{SSV}}^-_{z_i,b,c}}{\tau^-_{z_i,b,c}}\right)$;
  else
    $\widehat{\Delta\mathcal{SSV}_i} = \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\frac{b}{m-1}$
    $\left(\frac{\widehat{\Delta\mathcal{SSV}}^+_{z_i,b,c}}{\tau^+_{z_i,b,c}} - \frac{\widehat{\Delta\mathcal{SSV}}^-_{z_i,b,c}}{\tau^-_{z_i,b,c}}\right)$;
return $\widehat{\Delta\mathcal{SSV}}_1, \ldots, \widehat{\Delta\mathcal{SSV}}_n$;

The updated S-Shapley value given by Algorithm 4 is unbiased and therefore it preserves P1-P4 in Section 3.3 in expectation like the recalculated S-Shapley value. □

Theorem 6.4. *According to Hoeffding's inequality, given the range of differential utilities r and the minimum value of sample size for differential utility in different strata τ, the error $|\widehat{\Delta\mathcal{SSV}_i} - \Delta\mathcal{SSV}_i| \le \frac{2m|d[z_i]|-2}{m|d[z_i]|}\sqrt{2r^2\log(2/\delta)/\tau}$ in probability $1-\delta$.*

Proof. According to Theorems 5.4 and 5.6, the application of stratification to utility sampling results in the estimation of $2m|d[z_i]|$ variables when estimating a single variable $\Delta\mathcal{SSV}_i$. Let ϕ_t denote the t^{th} $(1 \le t \le \tau)$ differential utility sample in a specific stratum corresponding to $\Delta\mathcal{SSV}^+_{z_i,b,c}$. Let all utilities be in the range $[-r, r]$ $(r \ge 0)$. According to Hoeffding's inequality, we have

$$Pr(|\tau\widehat{\Delta\mathcal{SSV}}^{+}_{z_i,b,c} - \sum_{t=1}^{\tau} E[\phi_t]| \geq \tau\epsilon)$$
$$= Pr(|\widehat{\Delta\mathcal{SSV}}^{+}_{z_i,b,c} - E[\Delta\mathcal{SSV}^{+}_{z_i,b,c}]| \geq \epsilon)$$
$$\leq 2\exp\left(-\frac{2\epsilon^2}{\sum_{t=1}^{\tau}(\frac{m|d[z_i]|-1}{m|d[z_i]|})^2(2r)^2}\right) = 2\exp\left(-\frac{\epsilon^2}{2(\frac{m|d[z_i]|-1}{m|d[z_i]|})^2\tau r^2}\right).$$

Then we fix the probability to be at most δ, we have

$$2\exp\left(-\frac{\epsilon^2}{2(\frac{m|d[z_i]|-1}{m|d[z_i]|})^2\tau r^2}\right) \leq \delta \Rightarrow \tau \geq \frac{2(m|d[z_i]|-1)\ln(2/\delta)r^2}{(m|d[z_i]|)\epsilon^2}.$$

By moving ϵ to the left side, we obtain the inequality $\frac{m|d[z_i]|-1}{m|d[z_i]|}\sqrt{2r^2\log(2/\delta)/\tau}$. It is worth noting that τ represents the minimum sample size required for utility estimation across different strata. Consequently, the estimation error of each of the $2m|d[z_i]|$ variables is bounded by $\sqrt{2r^2\log(2/\delta)/\tau}$. According to Theorem 5.6 and triangle inequality, we have

$$|\widehat{\mathcal{SSV}}^{+}_i - \mathcal{SSV}^{+}_i|$$
$$= \left|\frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}(\widehat{\mathcal{SSV}}^{+}_{z_i,b,c} - \mathcal{SSV}^{+}_{z_i,b,c})\right|$$
$$\leq \frac{1}{m|d[z_i]|}\sum_{b=1}^{m}\sum_{c=1}^{|d[z_i]|}\left|\widehat{\mathcal{SSV}}^{+}_{z_i,b,c} - \mathcal{SSV}^{+}_{z_i,b,c}\right|$$
$$\leq \frac{m|d[z_i]|-1}{m|d[z_i]|}\sqrt{2r^2\log(2/\delta)/\tau}.$$

Similarly, we have

$$|\widehat{\mathcal{SSV}}^{-}_i - \mathcal{SSV}^{-}_i| \leq \frac{m|d[z_i]|-1}{m|d[z_i]|}\sqrt{2r^2\log(2/\delta)/\tau}.$$

According to Equation 4, we have

$$|\widehat{\mathcal{SSV}}_i - \mathcal{SSV}_i| \leq |\widehat{\mathcal{SSV}}^{+}_i - \mathcal{SSV}^{+}_i| + |\widehat{\mathcal{SSV}}^{-}_i - \mathcal{SSV}^{-}_i|$$
$$\leq 2\frac{m|d[z_i]|-1}{m|d[z_i]|}\sqrt{2r^2\log(2/\delta)/\tau} = \frac{2m|d[z_i]|-2}{m|d[z_i]|}\sqrt{2r^2\log(2/\delta)/\tau}.$$

This completes the proof. □

6.2 Multi Data Owners Exit

Another common practice in machine unlearning is that multiple data deletion requests within a certain period are responded to together by a single machine unlearning execution (also known as batch unlearning) since data owners will not always exit the model market one by one. Moreover, applying the delta-based algorithm (Algorithm 4) progressively to update S-Shapley value can be more time-consuming and inaccurate with more data owners' exit. To this end, we propose a solution to update S-Shapley value in one batch when multiple data owners exit the model market.

The difference between the precomputed $\mathcal{SSV}^{+}_i$ (resp. $\mathcal{SSV}^{-}_i$) and the new $\mathcal{SSV}^{+}_i$ (resp. $\mathcal{SSV}^{-}_i$) can be represented formally as Lemma 6.5. And they consist of the difference of $\mathcal{SSV}_i$.

Lemma 6.5. *Given a sharded structure $\mathcal{L}$, suppose that there are q data owners exiting with $\{z_{p_1},\ldots,z_{p_q}\}$ $(1 \leq q \leq n)$. We have the difference between the new $\mathcal{SSV}_i$ and the precomputed $\mathcal{SSV}_i$ of z_i*

$$\Delta\mathcal{SSV}_i = \underbrace{E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\cup\{z_i\}\backslash\{z_{p_1},\ldots,z_{p_q}\}\right) - \mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\cup\{z_i\}\right)\right]}_{\Delta\mathcal{SSV}^{+}_i}$$
$$\underbrace{- E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\backslash\{z_{p_1},\ldots,z_{p_q}\}\right) - \mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\right)\right]}_{\Delta\mathcal{SSV}^{-}_i},$$

where $i \notin \{p_1,\ldots,p_q\}$.

Proof. According to Equation 3, we have

$$\begin{aligned}\Delta\mathcal{SSV}_i =& E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\cup\{z_i\}\backslash\{z_{p_1},\ldots,z_{p_q}\}\right) - \mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\backslash\{z_{p_1},\ldots,z_{p_q}\}\right)\right]\\ &- E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\cup\{z_i\}\right) - \mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\right)\right]\\ =& E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\cup\{z_i\}\backslash\{z_{p_1},\ldots,z_{p_q}\}\right) - \mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\backslash\{z_{p_1},\ldots,z_{p_q}\}\right)\right.\\ &\left.- \mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\cup\{z_i\}\right) + \mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\right)\right]\\ =& E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\cup\{z_i\}\backslash\{z_{p_1},\ldots,z_{p_q}\}\right) - \mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\cup\{z_i\}\right)\right]\\ &- E_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\backslash\{z_{p_1},\ldots,z_{p_q}\}\right) - \mathcal{U}\left(\mathcal{S}^{z_i}_{\pi}\right)\right].\end{aligned}$$

□

According to Lemma 6.5, we estimate two differential utility expectations (i.e., $\Delta\mathcal{SSV}^{+}_i$ and $\Delta\mathcal{SSV}^{-}_i$) separately to estimate the change of S-Shapley value (i.e., $\Delta\mathcal{SSV}_i$). Algorithm 5 outlines the pseudo-code for estimating the change of $\mathcal{SSV}_i$ $(1 \leq i \leq n, i \notin \{p_1,\ldots,p_q\})$ to update $\mathcal{SSV}_i$ when a coalition of multiple data owner $\{z_{p_1},\ldots,z_{p_q}\}$ exits. The first step is to randomly generate a coalition $\mathcal{S}$, calculate the corresponding differential utility Δu, assign the value Δu to the relevant $\Delta\mathcal{SSV}^{+}_i$ (resp. $\Delta\mathcal{SSV}^{-}_i$), and update the associated counts of τ^{+}_i (resp. τ^{-}_i) (Lines 3-18). After τ samples have been drawn, the estimation of $\Delta\mathcal{SSV}^{+}_i$ or $\Delta\mathcal{SSV}^{-}_i$ is obtained as the average of the corresponding utilities. The final estimation of $\Delta\mathcal{SSV}$ is obtained as the difference between $\widehat{\Delta\mathcal{SSV}}^{+}_i$ and $\widehat{\Delta\mathcal{SSV}}^{-}_i$ (Lines 19-20). The stratification design in Section 5.2 is applicable for Algorithm 5. It is worth noting that the estimation of $\Delta\mathcal{SSV}_i$ in Algorithm 5 is unbiased.

Theorem 6.6. *Given the exited data owners with $\{z_{p_1},\ldots,z_{p_q}\}$, algorithm 5 gives an unbiased estimation of $\Delta\mathcal{SSV}_i$ $(1 \leq i \leq n, i \notin \{p_1,\ldots,p_q\})$, i.e., $E[\widehat{\Delta\mathcal{SSV}}_i] = \Delta\mathcal{SSV}_i$.*

Proof. Denote by $\{\mathcal{U}(\mathcal{S}_1\backslash\{z_{p_1},\ldots,z_{p_q}\})-\mathcal{U}(\mathcal{S}_1),\ldots,\mathcal{U}(\mathcal{S}_{\tau^{+}_i}\backslash\{z_{p_1},\ldots,z_{p_q}\})-\mathcal{U}(\mathcal{S}_{\tau^{+}_i})\}$ τ^{+}_i samples for estimating $\Delta\mathcal{SSV}^{+}_i$ drawn by Algorithm 5. The expectation of the samples $\widehat{\Delta\mathcal{SSV}}^{+}_i = \frac{1}{\tau^{+}_i}\sum_{t=1}^{\tau^{+}_i}[\mathcal{U}(\mathcal{S}_t\backslash\{z_{p_1},\ldots,z_{p_q}\}) - \mathcal{U}(\mathcal{S}_t)]$. We have

$$E[\widehat{\Delta\mathcal{SSV}}^{+}_i] = E[\frac{1}{\tau^{+}_i}\sum_{t=1}^{\tau^{+}_i}[\mathcal{U}(\mathcal{S}_t\backslash\{z_{p_1},\ldots,z_{p_q}\}) - \mathcal{U}(\mathcal{S}_t)]]$$
$$= \frac{1}{\tau^{+}_i}\sum_{t=1}^{\tau^{+}_i}E[\mathcal{U}(\mathcal{S}_t\backslash\{z_{p_1},\ldots,z_{p_q}\}) - \mathcal{U}(\mathcal{S}_t)].$$

Algorithm 5: Batched Delta-based Algorithm.

input : data sets of data owners $\{z_1, \ldots, z_n\}$, the index of exited data owners $\{p_1, \ldots, p_q\}$, sharded structure $\mathcal{L}$, the number of permutations τ

output : $\widehat{\Delta\mathcal{SSV}}_i$ for each z_i $(1 \le i \le n, i \notin \{p_1, \ldots, p_q\})$

$\widehat{\Delta\mathcal{SSV}}_i, \widehat{\Delta\mathcal{SSV}}_i^+, \widehat{\Delta\mathcal{SSV}}_i^-, \tau_i^+, \tau_i^- \leftarrow 0$ $(1 \le i \le n)$;
for *t=1 to* τ **do**
  $\pi^t \leftarrow$ **Sample**$(\mathcal{L})$;
  let i be a random value drawn from $1, \ldots, n$;
  $\mathcal{S} \leftarrow \{z_{\pi^t(1)}, \ldots, z_{\pi^t(i)}\}$;
  $\Delta u \leftarrow \mathcal{U}(\mathcal{S} \backslash \{z_{p_1}, \ldots, z_{p_q}\}) - \mathcal{U}(\mathcal{S})$;
  if $\mathcal{S} \cap d[z_{\pi^t(i)}] = d[z_{\pi^t(i)}]$ **then**
    for i'*=1 to* n **do**
      if $z_{i'} \in \mathcal{S}$ **then**
        $\widehat{\Delta\mathcal{SSV}}_{i'}^+ += \Delta u$; $\tau_{i'}^+ += 1$;
      else
        $\widehat{\Delta\mathcal{SSV}}_{i'}^- += \Delta u$; $\tau_{i'}^- += 1$;
  else
    foreach $z_{i'} \in d[z_{\pi^t(i)}]$ **do**
      if $z_{i'} \in \mathcal{S}$ **then**
        $\widehat{\Delta\mathcal{SSV}}_{i'}^+ += \Delta u$; $\tau_{i'}^+ += 1$;
      else
        $\widehat{\Delta\mathcal{SSV}}_{i'}^- += \Delta u$; $\tau_{i'}^- += 1$;
for *i=1 to* n **do**
  $\widehat{\Delta\mathcal{SSV}}_i = \widehat{\Delta\mathcal{SSV}}_i^+ / \tau_i^+ - \widehat{\Delta\mathcal{SSV}}_i^- / \tau_i^-$;
return $\widehat{\Delta\mathcal{SSV}}_1, \ldots, \widehat{\Delta\mathcal{SSV}}_n$;

According to Lemma 6.5, $E[\mathcal{U}(\mathcal{S}_t \backslash \{z_{p_1}, \ldots, z_{p_q}\}) - \mathcal{U}(\mathcal{S}_t)] = \Delta\mathcal{SSV}_i^+$. Thus, $E[\widehat{\Delta\mathcal{SSV}}_i^+] = \Delta\mathcal{SSV}_i^+$.

Similarly, we have $E[\widehat{\Delta\mathcal{SSV}}_i^-] = \Delta\mathcal{SSV}_i^-$. Thus,

$$E[\widehat{\Delta\mathcal{SSV}}_i] = E[\widehat{\Delta\mathcal{SSV}}_i^+ - \widehat{\Delta\mathcal{SSV}}_i^-] = E[\widehat{\Delta\mathcal{SSV}}_i^+] - E[\widehat{\Delta\mathcal{SSV}}_i^-]$$
$$= \Delta\mathcal{SSV}_i^+ - \Delta\mathcal{SSV}_i^- = \Delta\mathcal{SSV}_i.$$

That is, $\widehat{\Delta\mathcal{SSV}}_i$ is an unbiased estimation of $\Delta\mathcal{SSV}_i$. □

Corollary 6.7. *Given the precomputed S-Shapley value, Algorithm 5 can provide an unbiased estimation of the updated S-Shapley value and preserve P1-P4 in Section 3.3 in expectation.*

Proof. The precomputed S-Shapley value $\widehat{\mathcal{SSV}}_i$ is an unbiased estimator or the accurate value of S-Shapley value with original data owners. According to Theorem 6.6, the estimated change of S-Shapley value is also unbiased. Given two unbiased estimators $\widehat{\mathcal{SSV}}_i$ and $\widehat{\Delta\mathcal{SSV}}_i$ for estimating the new S-Shapley value $\mathcal{NSSV}_i$, we want to show that the sum of the estimators, denoted as $\widehat{\mathcal{NSSV}}_i = \widehat{\mathcal{SSV}}_i + \widehat{\Delta\mathcal{SSV}}_i$, is also unbiased. That is, we want to show

$$E[\widehat{\mathcal{NSSV}}_i] = \mathcal{NSSV}_i.$$

To prove this, we can use the linearity of expectation to write

$$E[\widehat{\mathcal{NSSV}}_i] = E[\widehat{\mathcal{SSV}}_i] + E[\widehat{\Delta\mathcal{SSV}}_i].$$

Since $\widehat{\mathcal{SSV}}_i$ and $\widehat{\Delta\mathcal{SSV}}_i$ are unbiased estimators, we have

$$E[\widehat{\mathcal{SSV}}_i] = \mathcal{SSV}_i \quad \text{and} \quad E[\widehat{\Delta\mathcal{SSV}}_i] = \Delta\mathcal{SSV}_i.$$

According to Lemma 6.5, we have $\Delta\mathcal{SSV}_i = \mathcal{NSSV}_i - \mathcal{SSV}_i$. Substituting these values into the equation above, we have

$$E[\widehat{\mathcal{NSSV}}_i] = \mathcal{SSV}_i + \Delta\mathcal{SSV}_i = \mathcal{NSSV}_i.$$

The updated S-Shapley value given by Algorithm 5 is unbiased and therefore it preserves P1-P4 in Section 3.3 in expectation like the recalculated S-Shapley value. □

Theorem 6.8. *According to Hoeffding's inequality, given the range of differential utilities r and the minimum value of sample size for differential utility in different strata τ, the error $|\widehat{\Delta\mathcal{SSV}}_i - \Delta\mathcal{SSV}_i| \le 2\sqrt{2r^2 \log(2/\delta)/\tau}$ in probability $1-\delta$.*

Proof. The proof is similar to Theorem 5.8. □

7 EXPERIMENTS

In this section, we conduct comprehensive experiments to evaluate the effectiveness of S-Shapley value (SSV) and the efficiency of the proposed algorithms.

7.1 Experiment Setup

Experiments are conducted on a server comprising two 16-core Montage(R) Jintide(R) C6226R @ 2.90GHz and four Geforce RTX 3090 GPUs, running Ubuntu 18.04 LTS 64-bit with 256GB memory.

7.1.1 Implementation. We implement all algorithms using PyTorch v1.12.1 (CUDA 11.6) and Python 3.9. In order to take advantage of the benefits of unlearning in terms of time cost, we employ incremental training within each data shard to expedite the retraining process and marginal contribution computation. This is achieved by caching the model trained on each data shard, allowing us to directly invoke the trained model when the coalitions of data owners changed, and thus reducing unnecessary model training.

7.1.2 Datasets and ML Models. We employ four real-world datasets: Iris, Car Evalutation, Phoneme, and Creditcard from OpenML [39]. To evaluate the performance, we conduct experiments using two representative ML models, logistic regression and Support Vector Machines (SVMs) with the Radial Basis Function (RBF) kernel. Due to page limits, the experimental results on logistic regression are only shown in our technical report [1].

7.1.3 Methods. In Section 7.2, we compare S-Shapley value (SSV) against five baselines to demonstrate its effectiveness in evaluating data importance. The baselines include Shapley value (SV), LOO, random valuation (Random), and two representatives from Beta Shapley [25] with hyper-parameters (1,16) and (16,1), respectively. In Section 7.3, we compare the efficiency of the utility sampling algorithm (US) with the Monte Carlo sampling algorithm (MCS)

as a baseline. Additionally, we introduce the paired utility sampling algorithm (PUS), which simply applies the paired sampling strategy [12] to Algorithm 3 by sampling a coalition $\mathcal{S}$ with its complementary coalition $\{z_1, \ldots, z_n\} \backslash \mathcal{S}$. In Section 7.4, we compare the efficiency of the Delta-based algorithm (Delta) and the batched Delta-based algorithm (BDelta) using the stratification design of Algorithm 3 with MCS, US, and PUS recomputing from scratch.

7.1.4 Metrics. In Sections 7.3 and 7.4, we employ the following metrics to measure the quality of the estimated S-Shapley value.

- **Average error ratio.** Given the benchmark S-Shapley value $\mathcal{SSV}_i$ and the estimated S-Shapley value $\widehat{\mathcal{SSV}}_i$ $(1 \leq i \leq n)$, the average error ratio $\overline{ER}$ of the estimated S-Shapley value compared to the benchmark S-Shapley value is

$$\overline{ER} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\widehat{\mathcal{SSV}}_i - \mathcal{SSV}_i}{\mathcal{SSV}_i} \right|.$$

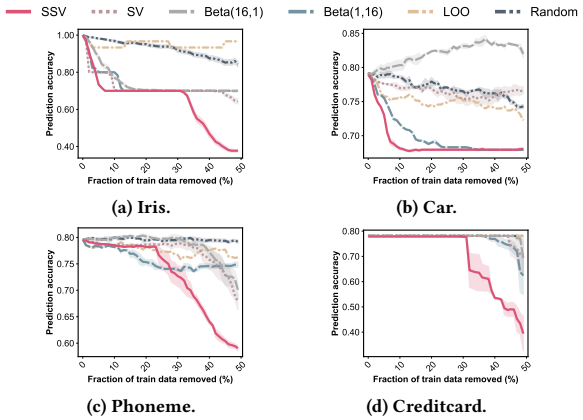
- **Average coefficient of variation.** Given a set of estimated S-Shapley value $\{\widehat{\mathcal{SSV}}_i^1, \ldots, \widehat{\mathcal{SSV}}_i^k\}(1 \leq i \leq n)$ obtained by computing k times using the same algorithm under the same setting, where $\widehat{\mathcal{SSV}}_i^j$ denotes the j^{th} estimated S-Shapley value of z_i computed by the algorithm, the average coefficient of variation $\overline{CV}$ of the estimated S-Shapley value is

$$\overline{CV} = \frac{1}{n} \sum_{i=1}^{n} \frac{\sqrt{\frac{1}{k} \sum_{j=1}^{k} \left(\widehat{\mathcal{SSV}}_i^j - \frac{1}{k} \sum_{j=1}^{k} \widehat{\mathcal{SSV}}_i^j \right)^2}}{\left| \frac{1}{k} \sum_{j=1}^{k} \widehat{\mathcal{SSV}}_i^j \right|}.$$

Computing the exact S-Shapley value for evaluation purposes is prohibitively expensive because it grows exponentially with the number of data owners. Therefore, we use the estimated S-Shapley value computed by Algorithm 2 with 100,000 permutations as the benchmark S-Shapley value for all experiments.

7.2 Effectiveness

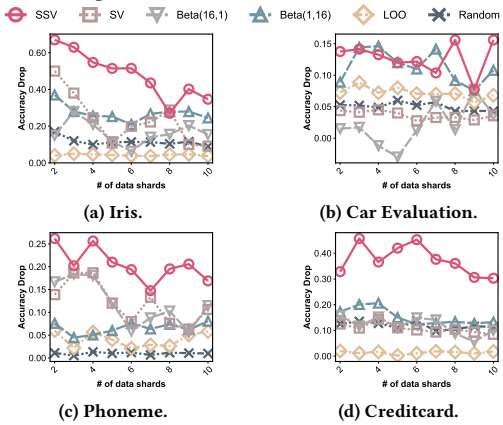
We experimentally study the effectiveness of the proposed S-Shapley value. We conduct point removal experiments to compare the importance of high-value data in each valuation method. Specifically, we evaluate the data value of each data owner with S-Shapley value and several baselines. Then we gradually remove data owners from the highest value to the lowest value. At each removal step, we retrain the classifier and evaluate predictive performance on the test dataset. Training instances with higher value estimates should be helpful for model performance, so we measure the performance of each method with the accuracy drop following their removal. We follow prior work [25] and plot the accuracy drop for up to 50% train data removed. For each data set, we randomly sample 60%, 20%, and 20% of the total data as training dataset, validation dataset, and test dataset, respectively. Wang and Jia [40] pointed out that limited by the numerical issue, Beta Shapley does NOT applicable for datasets with more than 1,000 data points (Phoneme and Creditcard). To overcome this problem, we randomly sample 1,000 data points in the training dataset instead. SVM is employed as the constituent model and the utility function is set to the accuracy score of the final model which is aggregated from constituent models on the validation dataset. The training dataset is randomly divided into five data shards, except for Iris, which is divided into three data shards. Figures 4 (a)(b)(c)(d) investigate the accuracy



(a) Iris. (b) Car.

(c) Phoneme. (d) Creditcard.

Figure 4: The effectiveness of SSV.



(a) Iris. (b) Car Evaluation.

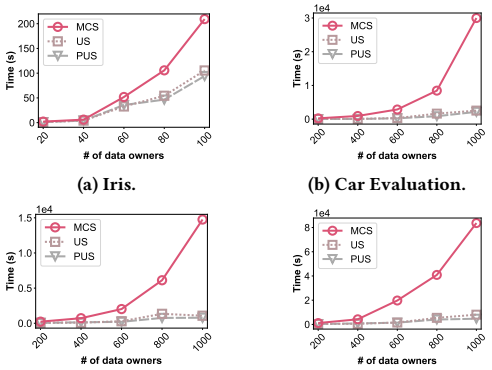
(c) Phoneme. (d) Creditcard.

Figure 5: The impact of different number of data shards.

drop for up to 50% train data removed. We denote a 95% confidence band based on 50 repetitions. The data removal with SSV outperforms all baselines, especially on Car Evaluation and Creditcard datasets. Figures 4 (c)(b) demonstrate a gradual decrease in accuracy followed by a sudden and significant drop in the S-Shapley value scheme's accuracy. This phenomenon occurs when some data is removed from the training dataset, especially the corresponding learning task is relatively simple. At beginning, the model may lose some of its ability to generalize to new data, leading to a slow or no decrease in accuracy. However, if too much data is removed, the model may become overfitted to the remaining data, leading to a sudden and significant drop in accuracy.

Impact of Sharding. In machine unlearning, it is well-understood that increasing the number of shards will improve the expected unlearning efficiency. Here, we wish to understand the impact of sharding on the S-Shapley valuation results. Figures 5(a)(b)(c)(d) show the accuracy drops with different numbers of data shards when 50% of data is removed. Overall, the accuracy drop decreases with an increase in the number of data shards. SSV, SV, and Beta(1,16) exhibit similar performance, outperforming other methods. Notably, SSV outperforms most methods in the majority of cases, indicating its superior effectiveness in evaluating data importance.

Impact of Removing Data Type. We experimentally analyze the effects of removing outliers and boundary data points on the model's performance and the S-Shapley value of the data shard,



(a) Iris. (b) Car Evaluation.

(c) Phoneme. (d) Creditcard.

Figure 6: The efficiency of SSV computation.

using the Lympho dataset [26] commonly used for outlier detection. An SVM classifier was trained, and its support vectors were used as boundary data points. Random partitioning into three data shards was performed, with five data points removed from different data types. The results presented in Table 3 indicate that removing outliers can improve the model's performance and the S-Shapley value of the corresponding data shard, whereas removing boundary data points can lead to a significant decrease in both.

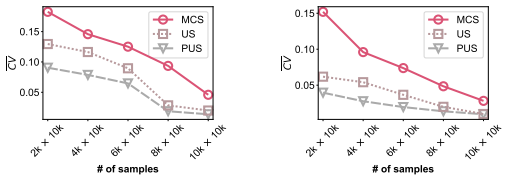
Table 3: The effects of removing different types of data points.

Data type	Accuracy change	SSV change
Outliers	6.7567e-03	2.2522e-03
Boundary data points	-4.7297e-02	-2.0270e-02
Random data points	-3.3783e-03	-9.0090e-03

7.3 Efficiency of Computation

We experimentally study the efficiency of the proposed algorithms in approximating S-Shapley value. For Car Evaluation, Phoeneme, and Creditcard datasets (resp. Iris dataset), we randomly sample 200, 400, 600, 800, and 1000 data subsets (resp. 20, 40, 60, 80, and 100 data subsets) to form different data owners and adopt the accuracy of the SVM model on the test dataset of size 500 (resp. 50) as the utility function. Figures 6(a)(b)(c)(d) investigate the time cost for the algorithms to achieve an average error ratio $\overline{ER} \leq 10\%$. The time cost required for the baseline, MCS, increases sharply with the increasing number of data owners, while US and PUS require significantly less time to achieve the same approximation error ratio, which verifies the efficiency of utility sampling strategy which reuses utilities between permutations. The paired sampling strategy significantly accelerates convergence in practice by reducing variance according to Theorem C.1.

Since it is hard to obtain a sufficiently accurate S-Shapley value as the benchmark Shapley value for comparison in tolerable time on large datasets (Phonemet and Creditcard), we perform an analysis of the proposed algorithms by measuring the average coefficient of variation $\overline{CV}$. We randomly sample 10000 data subsets to form data owners and compute S-Shapley value, and randomly sample 1000 data points as the test dataset. SVM is employed as the constituent model and the utility function is set to the accuracy score of the final model on the test dataset. Figures 7(a)(b) investigate $\overline{CV}$ of MCS, US, PUS with $2000n$, $4000n$, $6000n$, $8000n$, and $10000n$ samples, where n is the number of data subsets. $\overline{CV}$ of US and PUS are much smaller than MCS, which confirms the convergence of the estimated S-Shapley value computed by US.



(a) Phoneme. (b) Creditcard.

Figure 7: The scalability of SSV computation.

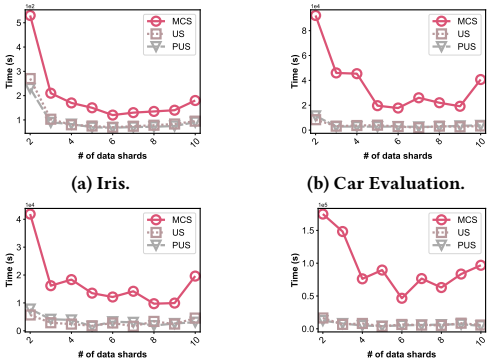
Impact of Sharding. We examine the impact of sharding on the efficiency of approximating S-Shapley value. Figures 8(a)(b)(c)(d) demonstrate the time costs required to achieve an average error ratio $\leq 10\%$ to the benchmark S-Shapley value with varying numbers of data shards. Generally, the time costs decrease initially with an increase in the number of data shards, and then increase again as the number of shards continues to increase.

7.4 Efficiency of Update

7.4.1 Single Data Owner Exit. We experimentally study the efficiency of the proposed algorithms in updating S-Shapley value when a single data owner exits. For Car Evaluation, Phoeneme, and Creditcard datasets (resp. Iris dataset), we randomly sample 200, 400, 600, 800, and 1000 data subsets (resp. 20, 40, 60, 80, and 100 data subsets) to form different data owners and adopt the accuracy of the SVM model on the test dataset of size 500 (resp. 50) as the utility function. Given the precomputed benchmark S-Shapley value on these datasets, Figures 9(a)(b)(c)(d) investigate the time cost for the algorithms to update S-Shapley value and achieve $\overline{ER} \leq 10\%$ when a data set in the corresponding dataset is removed. Delta and BDelta both significantly outperform all baselines while Delta takes the least time, which verifies the efficiency of our algorithms.

To verify the scalability, we randomly sample 10000 data subsets from Phonemet and Creditcard datasets to form data owners and compute S-Shapley value, and randomly sample 1000 data points as the test dataset. SVM is employed as the constituent model and the utility function is set to the accuracy score of the final model on the test dataset. Figures 10(a)(b) investigate the average coefficient of variation $\overline{CV}$ of MCS, US, PUS, Delta, BDelta with $2000n$, $4000n$, $6000n$, $8000n$, and $10000n$ samples. $\overline{CV}$ of Delta and BDelta is smaller than MCS, US, and PUS, which confirms that estimating the change of S-Shapley value leads to faster convergence.

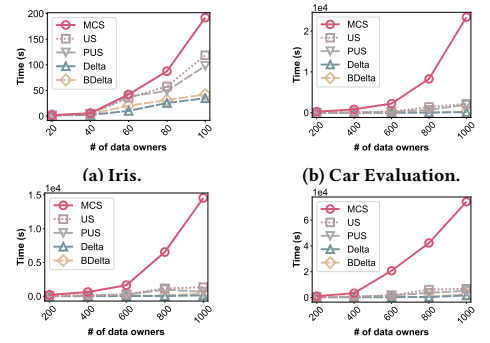
Update Quality. We experimentally study the approximation quality of the updated S-Shapley value with varying numbers of samples. For Car Evaluation, Phoeneme, and Creditcard datasets (resp. Iris dataset), we randomly sample 1000 (resp. 100) data subsets to form different data owners and adopt the accuracy of the SVM model on the test dataset of size 500 (resp. 50) as the utility function. Figures 11(a)(b)(c)(d) investigate $\overline{ER}$ of MCS, US, PUS, Delta, and BDelta with $2000n$, $4000n$, $6000n$, $8000n$, and $10000n$ samples when a data subset is removed, where n is the number of data subsets. Our results show that $\overline{ER}$ of Delta and BDelta is significantly smaller



(a) Iris. (b) Car Evaluation.

(c) Phoneme. (d) Creditcard.

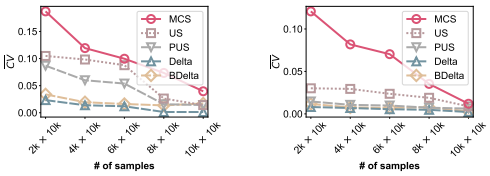
Figure 8: The impact of different number of data shards.



(a) Iris. (b) Car Evaluation.

(c) Phoneme. (d) Creditcard.

Figure 9: The efficiency of SSV update (single data owner exit).



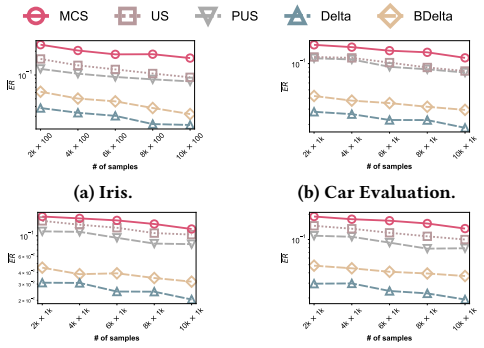
(a) Phoneme. (b) Creditcard.

Figure 10: The scalability of SSV update (single data owner exit).

than MCS, US, and PUS, demonstrating that the proposed algorithms, especially Delta, can yield better approximation in updating S-Shapley value when a single data owner exits.

Impact of Sharding. We examine the impact of sharding on the efficiency of approximating S-Shapley value. Figures 12(a)(b)(c)(d) demonstrate the time costs required to achieve an average error ratio $\leq 10\%$ to the benchmark S-Shapley value with varying numbers of data shards. Generally, the time costs decrease initially with an increase in the number of data shards, and then increase again as the number of shards continues to increase.

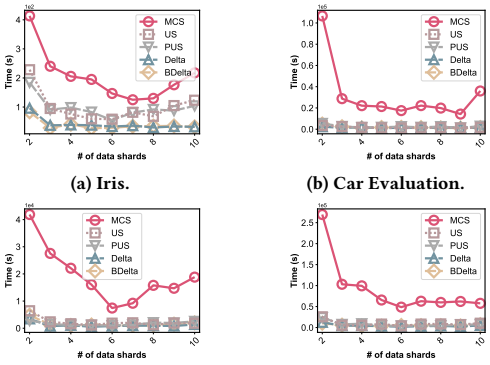
7.4.2 Multiple Data Owners Exit. We experimentally study the efficiency of the proposed algorithms in updating S-Shapley value when multiple data owners exit. For Car Evaluation, Phoeneme, and Creditcard datasets (resp. Iris dataset), we randomly sample 200, 400, 600, 800, and 1000 data subsets (resp. 20, 40, 60, 80, and 100



(a) Iris. (b) Car Evaluation.

(c) Phoneme. (d) Creditcard.

Figure 11: The quaility of SSV update (single data owner exit).



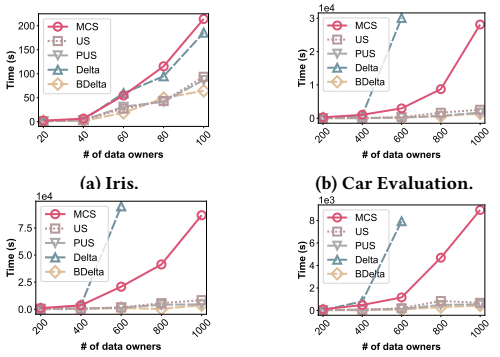
(a) Iris. (b) Car Evaluation.

(c) Phoneme. (d) Creditcard.

Figure 12: The impact of different number of data shards.

data subsets) to form different data owners, adopt the accuracy of the SVM model on the test dataset of size 500 (resp. 50) as the utility function and need to remove 5% data subsets. Given the benchmark S-Shapley value on these datasets, MCS, US, and PUS recompute S-Shapley value. Delta needs to update S-Shapley value step by step, while BDelta directly estimates the change of S-Shapley value for updating. Since Delta incrementally updates S-Shapley value, resulting in high time cost and high cumulative error, we omit some experimental results for Delta. Figures 13(a)(b)(c)(d) investigate the time cost for the algorithms to achieve $\overline{ER} \leq 10\%$ to the benchmark S-Shapley value on the remaining data. The time cost of BDelta is significantly lower than the baselines.

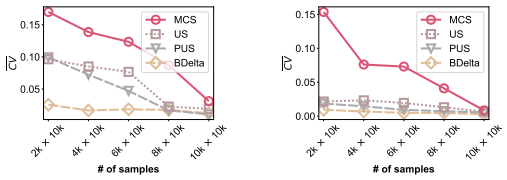
To verify the scalability, we randomly sample 10000 data subsets from Phonemet and Creditcard datasets to form data owners and compute S-Shapley value, and randomly sample 1000 data points as the test dataset. SVM is employed as the constituent model and the utility function is set to the accuracy score of the final model on the test dataset. Figures 14(a)(b) investigate the average coefficient of variation $\overline{CV}$ of MCS, US, PUS, and BDelta with $2000n$, $4000n$, $6000n$, $8000n$, and $10000n$ samples. The results of Delta are omitted due to high time costs. $\overline{CV}$ of BDelta is smaller than all baselines, which confirms that BDelta can well estimate the change of S-Shapley value when multiple data owners exit. Due to the time cost and error accumulation, Delta is not suitable for scenarios with multiple exited data owners. Our experiments show that it fails to attain the required average error ratio within a reasonable time frame, hence it is excluded from the results.



(a) Iris. (b) Car Evaluation.

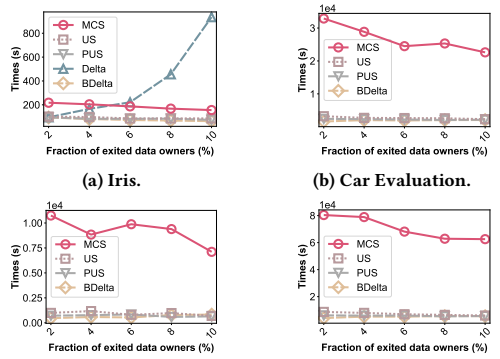
(c) Phoneme. (d) Creditcard.

Figure 13: The efficiency of SSV update (multiple data owners exit).



(a) Phoneme. (b) Creditcard.

Figure 14: The scalability of SSV update (multiple data owners exit).



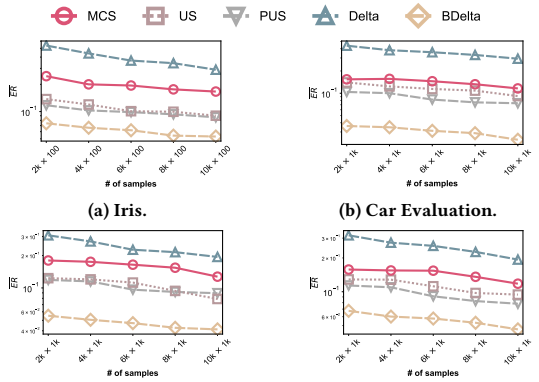
(a) Iris. (b) Car Evaluation.

(c) Phoneme. (d) Creditcard.

Figure 15: The impact of different number of exited data owners.

Impact of exited data owners' number. We tested algorithms for updating S-Shapley value when a data shard exits, using four datasets. The Delta algorithm has a significantly lower time cost than the others. The results are shown in Table 4, with 1000 subsets for Car Evaluation, Phoeneme, and Creditcard, and 100 for Iris.

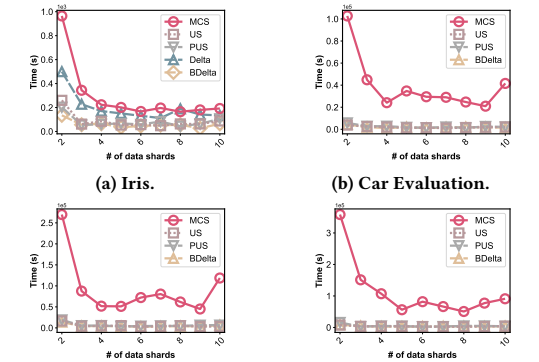
Update Quality. We experimentally study the approximation quality of the updated S-Sharply values varying numbers of samples. For Car Evaluation, Phoeneme, and Creditcard datasets (resp. Iris dataset), we randomly sample 1000 (resp. 100) data subsets to form different data owners and adopt the accuracy of the SVM model on the test dataset of size 500 (resp. 50) as the utility function. Figures 16(a)(b)(c)(d) investigate $\overline{ER}$ of MCS, US, PUS, Delta, and BDelta with 2000n, 4000n, 6000n, 8000n, and 10000n samples when 5% data subsets are removed, where n is the number of data subsets. The performance of Delta is poor due to time cost and error



(a) Iris. (b) Car Evaluation.

(c) Phoneme. (d) Creditcard.

Figure 16: The quality of SSV update (multi data owner exit).



(a) Iris. (b) Car Evaluation.

(c) Phoneme. (d) Creditcard.

Figure 17: The impact of different number of data shards.

accumulation in progressive updates. $\overline{ER}$ of BDelta is much smaller than other methods, which confirms that the batched Delta-based algorithm can yield better approximation in updating S-Shapley value when multiple data owners exit.

Impact of Sharding. We examine the impact of sharding on the efficiency of approximating S-Shapley value. Figures 17(a)(b)(c)(d) demonstrate the time costs required to achieve an average error ratio $\leq 10\%$ to the benchmark S-Shapley value with varying numbers of data shards. Generally, the time costs decrease initially with an increase in the number of data shards, and then increase again as the number of shards continues to increase.

7.4.3 Entire Data Shard Exit. We experimentally study the efficiency of the proposed algorithms in updating S-Shapley value when a data shard exits entirely. For Car Evaluation, Phoeneme, and Creditcard datasets (resp. Iris dataset), we randomly sample 1000 data subsets (resp. 100 data subsets) to form different data owners and follow the previous sharded structures. Table 4 presents the time cost for the algorithms to achieve $\overline{ER} \leq 10\%$ to the benchmark S-Shapley value on the remaining data. The time cost of Delta is significantly lower than others.

8 CONCLUSION AND FUTURE WORK

In this paper, we explored the problem of how the sharded structure for the right to be forgotten affects Shapley value for equitable data valuation. Guided by the four desirable properties for data valuation,

Table 4: The efficiency of SSV update (a data shard exit). Delta costs the least time to achieve the required accuracy. The time unit is second here.

Dataset	MCS	US	PUS	Delta	BDelta
Iris	1.3582e02	7.7413e01	6.8678e01	2.6038e01	2.9756e01
Car Evaluation	2.1199e04	2.0419e03	2.2108e03	6.6379e02	7.6956e02
Phoneme	1.2398e04	1.9421e03	1.8424e03	1.0276e03	1.0745e03
Creditcard	5.1268e04	8.5933e03	8.3685e03	3.6812e03	4.3506e03

we proposed the sharded structure-based Shapley value, S-Shapley value, to evaluate the data contribution equitably given the sharded structure. We established the #P-completeness of computing S-Shapley value and proposed two sampling-based approximation algorithms. Additionally, we proposed two efficient algorithms that estimate the change of S-Shapley value such that we can update S-Shapley value efficiently when data owners exit. Experimental results on real-world datasets show the effectiveness of S-Shapley value and the computational efficiency of the proposed algorithms. As different data sharding mechanisms can result in various sharded structures, it is interesting to investigate the impact of different sharded structures on S-Shapley value in future work.

REFERENCES

[1] 2023. Technical Report. https://anonymous.4open.science/r/rtbf-C5FC/rtbf.pdf

[2] Magdalena Balazinska, Bill Howe, and Dan Suciu. 2011. Data Markets in the Cloud: An Opportunity for the Database Community. *Proc. VLDB Endow.* 4, 12 (2011), 1482–1485. http://www.vldb.org/pvldb/vol4/p1482-balazinska.pdf

[3] Johes Bater, Yongjoo Park, Xi He, Xiao Wang, and Jennie Rogers. 2020. SAQE: Practical Privacy-Preserving Approximate Query Processing for Data Federations. *Proc. VLDB Endow.* 13, 11 (2020), 2691–2705. http://www.vldb.org/pvldb/vol13/p2691-bater.pdf

[4] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine Unlearning. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 141–159. https://doi.org/10.1109/SP40001.2021.00019

[5] Yinzhi Cao and Junfeng Yang. 2015. Towards Making Systems Forget with Machine Unlearning. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 463–480. https://doi.org/10.1109/SP.2015.35

[6] Javier Castro, Daniel Gómez, Elisenda Molina, and Juan Tejada. 2017. Improving polynomial estimation of the Shapley value by stratified random sampling with optimum allocation. *Comput. Oper. Res.* 82 (2017), 180–188. https://doi.org/10.1016/j.cor.2017.01.019

[7] Jihong Chen and Jiabin Sun. 2021. Understanding the Chinese Data Security Law. *International Cybersecurity Law Review* 2, 2 (2021), 209–221.

[8] Lingjiao Chen, Paraschos Koutris, and Arun Kumar. 2019. Towards Model-based Pricing for Machine Learning in a Data Marketplace. In *SIGMOD*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 1535–1552. https://doi.org/10.1145/3299869.3300078

[9] Lingjiao Chen, Hongyi Wang, Leshang Chen, Paraschos Koutris, and Arun Kumar. 2019. Demonstration of Nimbus: Model-based Pricing for Machine Learning in a Data Marketplace. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 1885–1888. https://doi.org/10.1145/3299869.3320231

[10] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph Unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM, 499–513. https://doi.org/10.1145/3548606.3559352

[11] H Christos. 1994. PAPADIMITRIOU: Computational complexity. *Addison-Wesley* 2, 3 (1994), 4.

[12] Ian Covert and Su-In Lee. 2021. Improving KernelSHAP: Practical Shapley Value Estimation Using Linear Regression. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event (Proceedings of Machine Learning Research)*, Arindam Banerjee and Kenji Fukumizu (Eds.), Vol. 130. PMLR, 3457–3465. http://proceedings.mlr.press/v130/covert21a.html

[13] Raul Castro Fernandez, Pranav Subramaniam, and Michael J. Franklin. 2020. Data Market Platforms: Trading Data Assets to Solve Data Problems. *Proc. VLDB Endow.* 13, 11 (2020), 1933–1947. http://www.vldb.org/pvldb/vol13/p1933-fernandez.pdf

[14] GDPR.eu. 2018. Article 17: Right to be forgotten. (2018). https://gdpr.eu/article-17-right-to-beforgotten

[15] Amirata Ghorbani, Michael P. Kim, and James Zou. 2020. A Distributional Framework For Data Valuation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research)*, Vol. 119. PMLR, 3535–3544. http://proceedings.mlr.press/v119/ghorbani20a.html

[16] Amirata Ghorbani and James Y. Zou. 2019. Data Shapley: Equitable Valuation of Data for Machine Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 2242–2251. http://proceedings.mlr.press/v97/ghorbani19c.html

[17] Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. 2019. Making AI Forget You: Data Deletion in Machine Learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 3513–3526. https://proceedings.neurips.cc/paper/2019/hash/cb79f8fa58b91d3af6c9c991f63962d3-Abstract.html

[18] Behzad Golshan, Alon Y. Halevy, George A. Mihaila, and Wang-Chiew Tan. 2017. Data Integration: After the Teenage Years. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts (Eds.). ACM, 101–106. https://doi.org/10.1145/3034786.3056124

[19] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. 2021. Amnesiac Machine Learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 11516–11524. https://ojs.aaai.org/index.php/AAAI/article/view/17371

[20] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. 2020. Certified Data Removal from Machine Learning Models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research)*, Vol. 119. PMLR, 3832–3842. http://proceedings.mlr.press/v119/guo20c.html

[21] Elizabeth Liz Harding, Jarno J Vanto, Reece Clark, L Hannah Ji, and Sara C Ainsworth. 2019. Understanding the scope and impact of the California Consumer Privacy Act of 2018. *Journal of Data Protection & Privacy* 2, 3 (2019), 234–253.

[22] Wassily Hoeffding. 1994. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*. Springer, 409–426.

[23] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gürel, Bo Li, Ce Zhang, Costas J. Spanos, and Dawn Song. 2019. Efficient Task-Specific Data Valuation for Nearest Neighbor Algorithms. *Proc. VLDB Endow.* 12, 11 (2019), 1610–1623. https://doi.org/10.14778/3342263.3342637

[24] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Goldberg. 2016. ActiveClean: Interactive Data Cleaning For Statistical Modeling. *Proc. VLDB Endow.* 9, 12 (2016), 948–959. https://doi.org/10.14778/2994509.2994514

[25] Yongchan Kwon and James Zou. 2022. Beta Shapley: a Unified and Noise-reduced Data Valuation Framework for Machine Learning. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event (Proceedings of Machine Learning Research)*, Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (Eds.), Vol. 151. PMLR, 8780–8802. https://proceedings.mlr.press/v151/kwon22a.html

[26] Aleksandar Lazarevic and Vipin Kumar. 2005. Feature bagging for outlier detection. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, Robert Grossman, Roberto J. Bayardo, and Kristin P. Bennett (Eds.). ACM, 157–166. https://doi.org/10.1145/1081870.1081891

[27] Guoliang Li and Xuanhe Zhou. 2022. Machine Learning for Data Management: A System View. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 3198–3201. https://doi.org/10.1109/ICDE53745.2022.00297

[28] Guoliang Li, Xuanhe Zhou, Ji Sun, Xiang Yu, Yue Han, Lianyuan Jin, Wenbo Li, Tianqing Wang, and Shifu Li. 2021. openGauss: An Autonomous Database System. *Proc. VLDB Endow.* 14, 12 (2021), 3028–3041. https://doi.org/10.14778/3476311.3476380

[29] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. 2014. A Confidence-Aware Approach for Truth Discovery on Long-Tail Data. *Proc. VLDB Endow.* 8, 4 (2014), 425–436. https://doi.org/10.14778/2735496.2735505

[30] Jinfei Liu, Jian Lou, Junxu Liu, Li Xiong, Jian Pei, and Jimeng Sun. 2021. Dealer: An End-to-End Model Marketplace with Differential Privacy. *Proc. VLDB Endow.* 14, 6 (2021), 957–969. http://www.vldb.org/pvldb/vol14/p957-liu.pdf

[31] Xuan Luo, Jian Pei, Zicun Cong, and Cheng Xu. 2022. On Shapley Value in Data Assemblage Under Independent Utility. *Proc. VLDB Endow.* 15, 11 (2022), 2761–2773. https://www.vldb.org/pvldb/vol15/p2761-luo.pdf

[32] Christos H Papadimitriou and Kenneth Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

[33] Jian Pei. 2021. A Survey on Data Pricing: from Economics to Data Science. *IEEE Trans. Knowl. Data Eng.* (2021). https://doi.org/10.1109/TKDE.2020.3045927

[34] Sebastian Schelter, Stefan Grafberger, and Ted Dunning. 2021. HedgeCut: Maintaining Randomised Trees for Low-Latency Machine Unlearning. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1545–1557. https://doi.org/10.1145/3448016.3457239

[35] Stephanie Schoch, Haifeng Xu, and Yangfeng Ji. 2022. CS-Shapley: Class-wise Shapley Values for Data Valuation in Classification. In *Advances in Neural Information Processing Systems 36 [Neural Information Processing Systems, NIPS 2022, Nov 29th - Dec 1st, 2022, New Orleans, USA]*. https://neurips.cc/Conferences/2022/ScheduleMultitrack?event=53147

[36] Lloyd S Shapley. 1953. A value for n-person games. *Contributions to the Theory of Games* 2, 28 (1953), 307–317.

[37] Rachael Hwee Ling Sim, Xinyi Xu, and Bryan Kian Hsiang Low. 2022. Data Valuation in Machine Learning: "Ingredients", Strategies, and Open Challenges. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, Luc De Raedt (Ed.). ijcai.org, 5607–5614. https://doi.org/10.24963/ijcai.2022/782

[38] Ikkyun Song, Yicheng Yang, Jongho Im, Tong Tong, Halil Ceylan, and In Ho Cho. 2020. Impacts of Fractional Hot-Deck Imputation on Learning and Prediction of Engineering Data. *IEEE Trans. Knowl. Data Eng.* 32, 12 (2020), 2363–2373. https://doi.org/10.1109/TKDE.2019.2922638

[39] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: networked science in machine learning. *SIGKDD Explorations* 15, 2 (2013), 49–60. https://doi.org/10.1145/2641190.2641198

[40] Tianhao Wang and Ruoxi Jia. 2023. Data Banzhaf: A Robust Data Valuation Framework for Machine Learning (Oral). In *International Conference on Artificial Intelligence and Statistics, AISTATS 2023, 25-27 April 2023, Palau de Congressos, Valencia, Spain (Proceedings of Machine Learning Research)*. PMLR.

[41] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. 2020. A Principled Approach to Data Valuation for Federated Learning. In *Federated Learning - Privacy and Incentive*, Qiang Yang, Lixin Fan, and Han Yu (Eds.). Lecture Notes in Computer Science, Vol. 12500. Springer, 153–167. https://doi.org/10.1007/978-3-030-63076-8_11

[42] Shuyue Wei, Yongxin Tong, Zimu Zhou, and Tianshu Song. 2020. Efficient and Fair Data Valuation for Horizontal Federated Learning. In *Federated Learning - Privacy and Incentive*, Qiang Yang, Lixin Fan, and Han Yu (Eds.). Lecture Notes in Computer Science, Vol. 12500. Springer, 139–152. https://doi.org/10.1007/978-3-030-63076-8_10

[43] Eyal Winter. 1989. A value for cooperative games with levels structure of cooperation. *International Journal of Game Theory* 18, 2 (1989), 227–240.

[44] Jiayao Zhang, Qiongqiong Lin, Jinfei Liu, Kui Ren, Jian Lou, Junxu Liu, Li Xiong, Jian Pei, and Jimeng Sun. 2021. Demonstration of Dealer: An End-to-End Model Marketplace with Differential Privacy. *Proc. VLDB Endow.* 14, 12 (2021), 2747–2750. http://www.vldb.org/pvldb/vol14/p2747-zhang.pdf

[45] Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. 2022. Protecting Decision Boundary of Machine Learning Model With Differentially Private Perturbation. *IEEE Trans. Dependable Secur. Comput.* 19, 3 (2022), 2007–2022. https://doi.org/10.1109/TDSC.2020.3043382

[46] Shuyuan Zheng, Yang Cao, and Masatoshi Yoshikawa. 2023. Secure Shapley Value for Cross-Silo Federated Learning. *Proc. VLDB Endow.* 16, 6 (2023).

[47] Xuanhe Zhou, Chengliang Chai, Guoliang Li, and Ji Sun. 2022. Database Meets Artificial Intelligence: A Survey. *IEEE Trans. Knowl. Data Eng.* 34, 3 (2022), 1096–1116. https://doi.org/10.1109/TKDE.2020.2994641

A IMPLEMENTATION DETAILS

Paired Utility Sampling. The key idea of paired utility sampling is applying paired sampling strategy which samples both a coalition $\mathcal{S}$ and its complementary coalition $\{z_1, \ldots, z_n\} \backslash \mathcal{S}$ in Algorithm 3 (Lines 6-7).

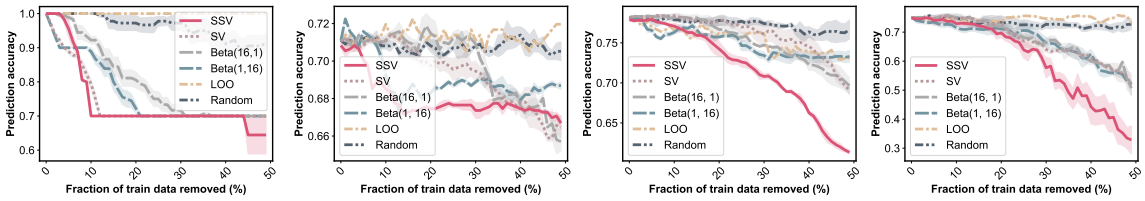
Algorithm 6: Paired Utility Sampling Algorithm.

input : data sets from data owners $\{z_1, \ldots, z_n\}$, sharded structure $\mathcal{L}$, the number of permutations τ
output: estimated S-Shapley value $\widehat{\mathcal{SSV}}_i$ for each z_i $(1 \leq i \leq n)$

$\widehat{\mathcal{SSV}}_i \leftarrow 0\ (1 \leq i \leq n)$; $\widehat{\mathcal{SSV}}^+_{z_i,b,c}, \widehat{\mathcal{SSV}}^-_{z_i,b,c}, \tau^+_{z_i,b,c}, \tau^-_{z_i,b,c} \leftarrow 0\ (1 \leq i \leq n, 1 \leq b \leq m, 1 \leq c \leq |d[z_i]|)$;
for t=1 *to* τ **do**
  $\pi^t \leftarrow$ **Sample**$(\mathcal{L})$;
  let i be a random value drawn from $1, \ldots, n$;
  $\mathcal{S} \leftarrow \{z_{\pi^t(1)}, \ldots, z_{\pi^t(i)}\}$;
  $u' \leftarrow \mathcal{U}(\mathcal{S})$; $u'' \leftarrow \mathcal{U}(\{z_1, \ldots, z_n\} \backslash \mathcal{S})$;
  foreach $u \in \{u', u''\}$ **do**
    $b \leftarrow 0$;
    for j=1 *to* m **do**
      if $\mathcal{S} \cap d_j \neq \emptyset$ **then**
        b+=1;
    $c \leftarrow |\mathcal{S} \cap d[z_{\pi^t(i)}]|$;
    if $c = |d[z_{\pi^t(i)}]|$ **then**
      for i'=1 *to* n **do**
        if $z_{i'} \in \mathcal{S}$ **then**
          $\widehat{\mathcal{SSV}}^+_{z_{i'},b,|d[z_{i'}]|} += u$; $\tau^+_{z_{i'},b,|d[z_{i'}]|} += 1$;
        else
          $\widehat{\mathcal{SSV}}^-_{z_{i'},b+1,1} += u$; $\tau^-_{z_{i'},b+1,1} += 1$;
    else
      foreach $z_{i'} \in d[z_{\pi^t(i)}]$ **do**
        if $z_{i'} \in \mathcal{S}$ **then**
          $\widehat{\mathcal{SSV}}^+_{z'_i,b,c} += u$; $\tau^+_{z'_i,b,c} += 1$;
        else
          $\widehat{\mathcal{SSV}}^-_{z'_i,b,c+1} += u$; $\tau^-_{z'_i,b,c+1} += 1$;
for i=1 *to* n **do**
  $\widehat{\mathcal{SSV}}_i = \frac{1}{m|d[z_i]|} \sum_{b=1}^{m} \sum_{c=1}^{|d[z_i]|} [\widehat{\mathcal{SSV}}^+_{z_i,b,c}/\tau^+_{z_i,b,c} - \widehat{\mathcal{SSV}}^-_{z_i,b,c}/\tau^-_{z_i,b,c}]$;
return $\widehat{\mathcal{SSV}}_1, \ldots, \widehat{\mathcal{SSV}}_n$;

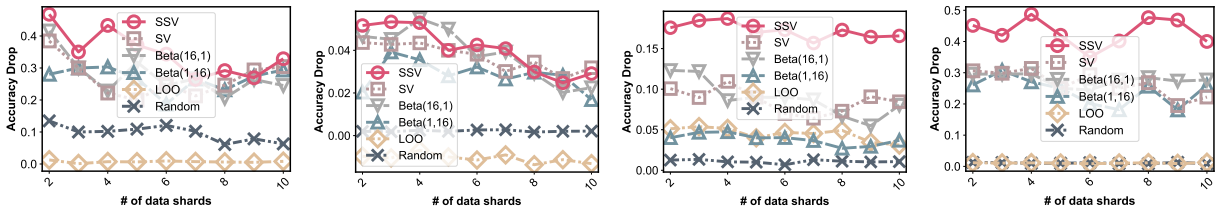
B ADDITIONAL RESULTS

In the following, we report the results across experiments in Section 7.2 for the logistic regression classifier. We report the accuracy drop for the logistic regression classifier and the accuracy drop change with different numbers of data shards.



(a) Iris. (b) Car Evaluation. (c) Phoneme. (d) Creditcard.

Figure 18: The effectiveness of S-Shapley value.



(a) Iris. (b) Car Evaluation. (c) Phoneme. (d) Creditcard.

Figure 19: The impact of different number of data shards.

C ADDITIONAL THEOREM

When sampling a coalition $\mathcal{S}$ of $\mathcal{D}$ to generate the utility, we suggest a *paired sampling* strategy [12] where each sample $\mathcal{S}$ is paired with it complement $\mathcal{D}\backslash\mathcal{S}$. For arbitrary games $\mathcal{V}$, we can guarantee that paired sampling approach leads to lower variance in the estimation of S-Shapley value in a probabilistic sense. According to Equation 4, the estimated S-Shapley value can be divided into two parts, A_1 and A_2,

$$\mathcal{SSV}_i = \underbrace{\boldsymbol{E}_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}_\pi^{z_i}\cup\{z_i\}\right)\right]}_{A_i^1} - \underbrace{\boldsymbol{E}_{\pi\sim\Pi}\left[\mathcal{U}\left(\mathcal{S}_\pi^{z_i}\right)\right]}_{A_i^2}.$$

Note that for each data point there is a term A_i^1 and A_i^2, thus we can vectorize them as follows

$$\overrightarrow{\mathcal{SSV}} = \overrightarrow{A^1} - \overrightarrow{A^2}.$$

Then we denote the estimation of $\overrightarrow{A^1}$ (*resp.* $\overrightarrow{A^2}$) through simple utility sampling by $\overrightarrow{\widetilde{A^1}}$ (*resp.* $\overrightarrow{\widetilde{A^2}}$) and the estimation of $\overrightarrow{A^1}$ (*resp.* $\overrightarrow{A^2}$) through paired utility sampling by $\overrightarrow{\check{A}^1}$ (*resp.* $\overrightarrow{\check{A}^2}$).

Theorem C.1. *The difference between the covariance matrices for the estimators* $\overrightarrow{\widetilde{A^1}}$, $\overrightarrow{\widetilde{A^2}}$, $\overrightarrow{\check{A}^1}$, *and* $\overrightarrow{\check{A}^2}$ *satisfies*

$$\left(\mathrm{Cov}(\overrightarrow{\widetilde{A^1}}) - \mathrm{Cov}(\overrightarrow{\check{A}^1})\right)_{ii} \geq 0 \quad \text{and} \quad \left(\mathrm{Cov}(\overrightarrow{\widetilde{A^2}}) - \mathrm{Cov}(\overrightarrow{\check{A}^2})\right)_{ii} \geq 0.$$

Proof. For simplification, let us reduce S-Shapley value computation into the Shapley value computation. First, we denote each subset $S \in \mathcal{D}$ using the corresponding binary vector $\vec{v} \in \{0,1\}^n$, and with tolerable abuse of notation we write $\mathcal{U}(\vec{v}) \equiv \mathcal{U}(S)$ for $S = \{z_i : v_i = 1\}$. Lastly, we denote a distribution over V using $p(\vec{v})$ ($p(\vec{v}) \propto \binom{n}{<\vec{1},\vec{v}>}^{-1}$ when $<\vec{1},\vec{v}>\in[0,n]$ and $p(\vec{v}) = 0$ otherwise). Then we have,

$$\overrightarrow{\widetilde{A^1}}_k = \frac{1}{k}\sum_{i=1}^{k} \vec{v_i}\mathcal{U}(\vec{v_i}).$$

we proposed a variance reduction technique that pairs each sample $\vec{v_i} \sim p(\vec{v})$ with its complement $\vec{1} - \vec{v_i}$ when estimating $\overrightarrow{A^1}$. To simplify our notation, we introduce three jointly distributed random variables, M^0, M^1, and $\bar{M}$, which are all functions of the random variable Z, as follows

$$\begin{aligned} M^0 &= V\mathcal{U}(V), \\ M^1 &= (\vec{1} - V)\mathcal{U}(\vec{1} - V), \\ \bar{M} &= \frac{1}{2}\left(M^0 + M^1\right). \end{aligned}$$

To understand $\bar{M}$'s covariance structure, we can decompose it using standard covariance properties and the fact that $p(\vec{v}) = p(\vec{1} - \vec{v})$ for all $\vec{v}$ as follows

$$\begin{aligned} \mathrm{Cov}(\bar{M},\bar{M})_{ij} &= \frac{1}{4}\mathrm{Cov}\left(M_i^0 + M_i^1, M_j^0 + M_j^1\right) \\ &= \frac{1}{4}\left(\mathrm{Cov}\left(M_i^0, M_j^0\right) + \mathrm{Cov}\left(M_i^1, \bar{M}_j^1\right) + \mathrm{Cov}\left(M_i^0, M_j^1\right) + \mathrm{Cov}\left(M_i^1, M_j^0\right)\right) \\ &= \frac{1}{2}\left(\mathrm{Cov}\left(M_i^0, M_j^0\right) + \mathrm{Cov}\left(M_i^0, M_j^1\right)\right). \end{aligned}$$

To account for each $\bar{M}$ sample requiring twice as many cooperative game evaluations as M^0, we compare the covariance $\mathrm{Cov}(\overrightarrow{\widetilde{A^1}}_{2k})$ to the covariance $\mathrm{Cov}(\overrightarrow{\check{A}^1}_k)$ as follows

$$n\left(\mathrm{Cov}\left(\bar{b}_{2n}\right) - \mathrm{Cov}\left(\check{b}_n\right)\right)_{ij} = -\frac{1}{2}\mathrm{Cov}\left(M_i^0, M_j^1\right).$$

Based on this, we define G_v as follows

$$\begin{aligned} G_v &= -\operatorname{Cov}\left(M_i^0, M_j^1\right) \\ &= -\operatorname{Cov}(Zv(Z) - \mathbb{E}[Z]v(\mathbf{0}), (\mathbf{1}-Z)v(\mathbf{1}-Z) - \mathbb{E}[\mathbf{1}-Z]v(\mathbf{0})) \\ &= -\operatorname{Cov}(Zv(Z), (\mathbf{1}-Z)v(\mathbf{1}-Z)). \end{aligned}$$

A necessary (but not sufficient) condition for $G_v \succeq 0$ is that its diagonal elements are non-negative. We can prove that this weaker condition holds for all games.

$$\begin{aligned} &(G_v)_{ii} = -\operatorname{Cov}\left(Z_i v(Z), (1-Z_i)\, v(\mathbf{1}-Z)\right) \\ &= -\mathbb{E}\left[Z_i\,(1-Z_i)\, v(Z) v(\mathbf{1}-Z)\right] + \mathbb{E}\left[Z_i v(Z)\right] \mathbb{E}\left[(1-Z_i)\, v(\mathbf{1}-Z)\right] \\ &= \mathbb{E}\left[Z_i v(Z)\right]^2 \\ &= \mathbb{E}[v(S) \mid i \in S]^2 \\ &\geq 0. \end{aligned}$$

□

Remark. *Theorem C.1 shows that the confidence boundary of* $\overrightarrow{\widehat{A^1}}$ *(resp.* $\overrightarrow{\widehat{A^2}}$*) extends* $\overrightarrow{\check{A}^1}$ *(resp.* $\overrightarrow{\check{A}^2}$*). In other words, the result implies the variance for each Shapley value estimate is lower when using the paired sampling technique.*