Regression-adjusted Monte Carlo Estimators for Shapley Values and Probabilistic Values

R. Teal Witter
Claremont McKenna College
rtealwitter@cmc.edu

Yurong Liu
New York University
yurong.liu@nyu.edu

Christopher Musco
New York University
cmusco@nyu.edu

Abstract

With origins in game theory, probabilistic values like Shapley values, Banzhaf values, and semi-values have emerged as a central tool in explainable AI. They are used for feature attribution, data attribution, data valuation, and more. Since all of these values require exponential time to compute exactly, research has focused on efficient approximation methods using two techniques: Monte Carlo sampling and linear regression formulations. In this work, we present a new way of combining both of these techniques. Our approach is more flexible than prior algorithms, allowing for linear regression to be replaced with any function family whose probabilistic values can be computed efficiently. This allows us to harness the accuracy of tree-based models like XGBoost, while still producing unbiased estimates. From experiments across eight datasets, we find that our methods give state-of-the-art performance for estimating probabilistic values. For Shapley values, the error of our methods can be $6.5\times$ lower than Permutation SHAP (the most popular Monte Carlo method), $3.8\times$ lower than Kernel SHAP (the most popular linear regression method), and $2.6\times$ lower than Leverage SHAP (the prior state-of-the-art Shapley value estimator). For more general probabilistic values, we can obtain error $215\times$ lower than the best estimator from prior work.

1 Introduction

As AI becomes more prevalent across health care, education, finance, and the legal system, underlying algorithmic mechanisms are growing increasingly complex. Sophisticated computational models frequently make decisions that are opaque and challenging to comprehend. This is unacceptable in contexts where decisions can have profound consequences for individuals: the ability to clearly understand and explain how an algorithmic system reaches its conclusions is paramount.

One tool that has arisen to address the challenge of understanding model behavior are *probabilistic values*, which include Shapley values, Banzhaf values, and semi-values as special cases [SK10, LL17, LEC+20, WJ23]. Originating from game theory [Sha51], probabilistic values quantify the contribution of a player by measuring how its addition to a set of other players changes the value of the game. Formally, consider a *value function* $v : 2^{[n]} \to \mathbb{R}$ defined on sets $S \subseteq [n]$, where $[n]$ denotes $\{1, \dots, n\}$. The probabilistic value for player $i \in [n]$ is

$$\phi_i(v) = \sum_{S \subseteq [n] \setminus \{i\}} p_{|S|}[v(S \cup \{i\}) - v(S)] \tag{1}$$

where $\mathbf{p} = [p_0, \dots, p_{n-1}] \in [0, 1]^n$ is a set of probabilistic weights that satisfy $\sum_{\ell=0}^{n-1} \binom{n-1}{\ell} p_\ell = 1$. We can interpret the ith probabilistic value as the *average* marginal contribution of player i to random set S, where the distribution over set sizes is specified by $\mathbf{p}$. Different choices of $\mathbf{p}$ yield different variants of probabilistic values [KZ22a, KZ22b, LY24c]. For example, to obtain the ubiquitous Shapley values, set $p_\ell = \frac{1}{n}\binom{n-1}{\ell}^{-1}$, and to obtain Banzhaf values, set $p_\ell = 1/2^{n-1}$ for all ℓ.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Our paper addresses the problem of computing $\phi_1, \ldots, \phi_n$ in full generality for any $\mathbf{p}$. The topic of which weights are best for a given application has received significant attention. Some prior work focuses on axiomatic approaches for choosing $\mathbf{p}$. For example, all probabilistic values satisfy three desirable properties: *null player*, *symmetry*, and *linearity* (see [Web88] for a detailed discussion). Shapley values satisfy an additional *efficiency* property [Sha51] and Banzhaf values satisfy a *2-efficiency* property that might be desirable when there are non-linear interactions between players [Pen46, BI64]. Generalizations of these values include Beta Shapley [KZ22a] values and weighted Banzhaf values [LY24c]. See Appendix B for more on these generalizations.

Regardless of how $\mathbf{p}$ is chosen, the meaning of the probabilistic values depends on how the value function, v, is defined. For example, a common task in explainable AI is to attribute a model prediction (for a given input) to features [LL17]. Here, $v(S)$, is the prediction made when using just the subset of features corresponding to S.[1] Probabilistic values are also used in data attribution tasks, where $v(S)$ corresponds to the model loss when training with a given subset of data [GZ19, WMSJ25]. In these applications and others, evaluating v is expensive, as it requires re-running or possibly even re-training a model. As in prior work on efficient probabilistic value estimation, we thus focus on algorithms that estimate $\phi_1, \ldots, \phi_n$ using as few evaluations of v as possible. We view these evaluations as black-box, designing algorithms that are agnostic to the particular value function v, and can thus be applied in a wide range of downstream applications.

1.1 Efficiently Computing Probabilistic Values

For general value functions, exactly computing probabilistic values requires exponential time, as the summation in Equation (1) involves $O(2^n)$ terms. When v is a highly structured function, like a linear function or decision tree, more efficient algorithms exist [LL17, LEL18, LEC+20, KMM+22]. However, given the complexity of modern machine learning models, most prior work focuses on approximation algorithms.

The standard method is to approximate the summation in Equation (1) via a Monte Carlo estimate obtained from a weighted sample of sets that do not contain i [KZ22a, KZ22b, LY24b]. Concretely, assume for simplicity that we sample a collection of subsets, $\mathcal{S}_i$, by drawing samples with replacement from a distribution with density $\mathcal{D} : 2^{[n]\setminus\{i\}} \to [0, 1]$.[2] We then compute the unbiased estimate:

$$\tilde{\phi}_i^{\text{MC}} = \frac{1}{|\mathcal{S}_i|} \sum_{S \in \mathcal{S}_i} [v(S \cup \{i\}) - v(S)] \frac{p_{|S|}}{\mathcal{D}(S)} \tag{2}$$

We have that $\mathbb{E}[\tilde{\phi}_i^{\text{MC}}] = \phi_i$, and the estimator's variance depends on the choice of sampling distribution $\mathcal{D}$, as well as $[v(S \cup \{i\}) - v(S)]^2$ for all $S \subseteq [n]$. In addition to high-variance in practice[3], a downside of Monte Carlo estimators is that it is difficult to "reuse" samples between indices $1, \ldots, n$, as each term in Equation (2) requires evaluating both $v(S \cup \{i\})$ and $v(S)$ for a particular i. Several methods address this issue via "sample reuse" [CGT09]. One technique especially relevant to our work is the *maximum sample reuse* (MSR) method, which was originally applied to Banzhaf values [WJ23], but generalizes naturally to all probabilistic values [KBMH24, LY24a, LY24b]. The MSR method draws a single collection of subsets, $\mathcal{S}$, according to $\mathcal{D} : 2^{[n]} \to [0, 1]$, and computes the estimate:

$$\tilde{\phi}_i^{\text{MSR}} = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} v(S) \frac{p_{|S|-1} \mathbb{1}[i \in S] - p_{|S|} \mathbb{1}[i \notin S]}{\mathcal{D}(S)}. \tag{3}$$

It can be checked that we still have $\mathbb{E}[\tilde{\phi}_i^{\text{MSR}}] = \phi_i$ for all i. Moreover, every evaluation of the value function, $v(S)$, contributes to the estimate for *all* $i \in [n]$, so we achieve maximum sample reuse. However, the variance of MSR methods scales as a weighted sum of $[v(S)]^2$, which is generally much larger than the difference between nearby values $[v(S \cup \{i\}) - v(S)]^2$.

Beyond Monte Carlo. Given the high variance of Monte Carlo methods, an alternate approach based on *regression* has become popular for the special case of Shapley values. In particular, Shapley values

[1] Since most models in machine learning require a full set of input features, features not in S are replaced with either a mean value or random value from the training dataset as a baseline [JMB20, LEL18].

[2] In order to efficiently sample, $\mathcal{D}$ typically assigns the same density to subsets of the same size.

[3] In general, variance scales with $1/\sqrt{|\mathcal{S}_i|}$, i.e., only as the inverse root of the number of samples.

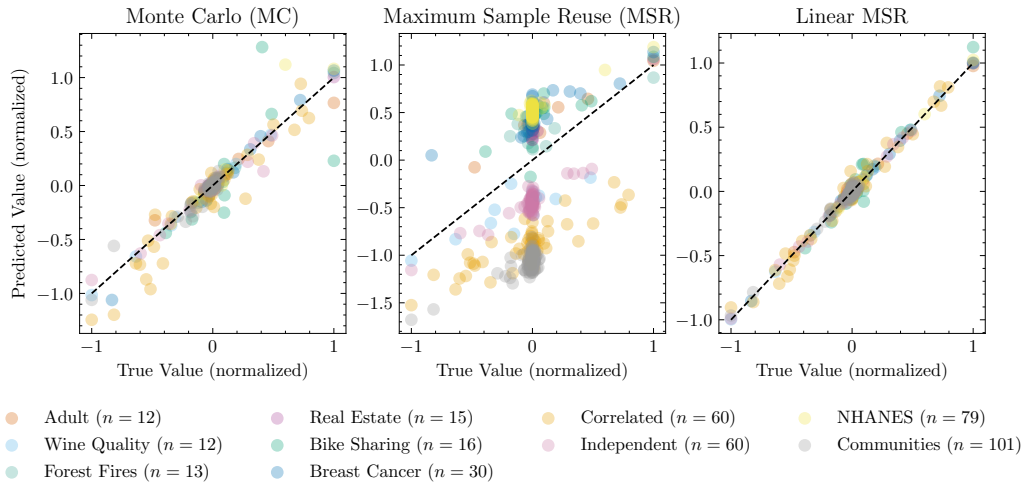

Figure 1: Predicted versus true (normalized) Shapley values for three unbiased estimators given a fixed number of black-box evaluations of the value function, v. Each point represents one feature's estimated vs true Shapley value on one dataset. The Monte Carlo estimator uses each sample to estimate only one Shapley value, but has variance that depends on the difference in values between neighboring sets, i.e., $[v(S \cup \{i\}) - v(S)]^2$. The Maximum Sample Reuse (MSR) estimator reuses samples, but has larger variance that depends on the magnitude of the values, i.e., $[v(S)]^2$. Our Regression MSR estimators reuse samples *and* have smaller variance that depends on how well a learned function f fits the value function v, i.e., $[v(S) - f(S)]^2$. Even taking f to be linear gives excellent performance (we call this method Linear MSR). Taking f to be a decision-tree model (Tree MSR) can produce even better estimates for large sample sizes, as shown in Figure 2.

are the unique solution to a particular overdetermined linear regression problem [CGKR88]:

$$\boldsymbol{\phi} = [\phi_1, \ldots, \phi_n] = \underset{\mathbf{x}:\langle \mathbf{x}, \mathbf{1} \rangle = v([n]) - v(\emptyset)}{\arg\min} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_W, \tag{4}$$

where $\mathbf{A} \in \mathbb{R}^{2^n \times n}$ is a specific structured matrix whose rows correspond to sets $S \subseteq [n]$, $\mathbf{b} \in \mathbb{R}^{2^n}$ is vector whose entries equal $v(S) - v(\emptyset)$, and $\|\cdot\|_W$ is a weighted ℓ_2 norm.

The ubiquitous Kernel SHAP algorithm [LL17, CL21] takes advantage of the regression formulation by *approximately solving* Equation (4) using a subsample of constraints (and corresponding entries in $\mathbf{b}$), each of which requires evaluating $v(S)$ for a single subset S. This approach was recently improved by incorporating leverage score sampling [Sar06, SS11], resulting in the state-of-the-art Leverage SHAP method [MW25]. In addition to inherent sample reuse, the empirical effectiveness of Kernel SHAP and Leverage SHAP seems related to the fact that the accuracy of both methods depends on how well v is approximated by a linear function. Indeed, it can be shown that if v is exactly linear, both methods return exact Shapley values after just n function evaluations [MW25]. However, even when v is not linear, there are theoretical guarantees on the performance of Kernel SHAP and Leverage SHAP [MW25, CSV$^+$25].

The Kernel SHAP approach has been extended to Banzhaf values [LWK$^+$25], thanks to a similarly elegant regression formulation [HH92]. However, extensions to more general probabilistic values have been less effective, failing to outperform Monte Carlo methods [LZL$^+$22, LY24a, LY24b]. A key challenge is that, due to the lack of an efficiency property, generalized linear regression formulations for probabilistic values typically require estimating the *sum* of these values, which introduces another source of error [RVZ98]. Moreover, even for Shapley and Banzhaf values, a drawback of regression-based methods is that they fail to provide an unbiased estimate for each ϕ_i. Attempts to fix this issue have generally led to estimates with much higher variance [CL21].

1.2 Our Contributions

We introduce a method called *Regression MSR* for leveraging regression to approximate probabilistic values. In contrast to previous work on regression methods, Regression MSR leads to estimates that

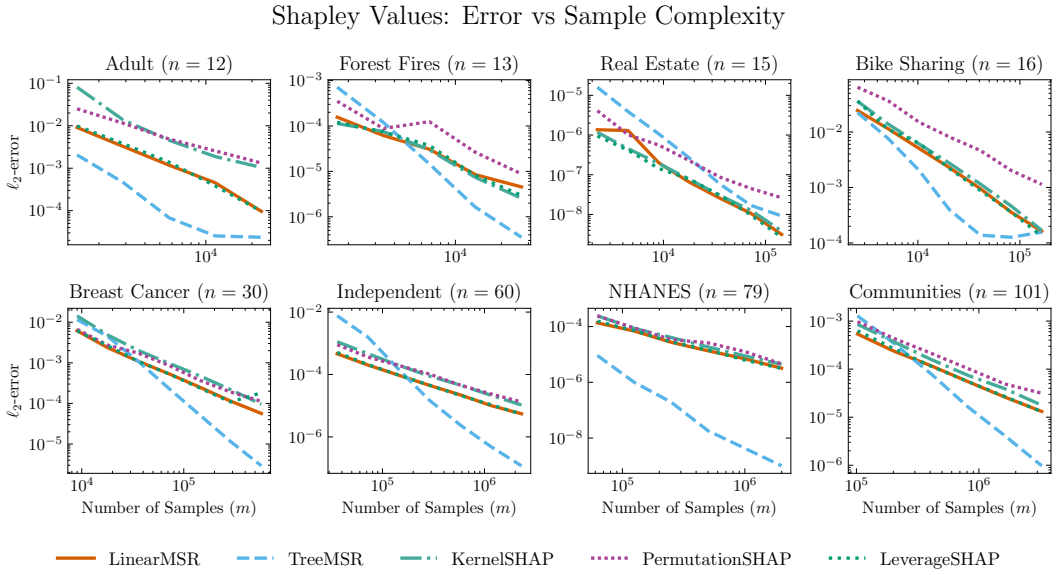

Figure 2: Average ℓ_2-error between estimated and true Shapley values as a function of sample size m (number of evaluations of v) for various datasets. The lines report the mean error over 100 runs, and $m = 10n, 20n, 40n, 80n, 160n, 320n, 640n$. Linear MSR consistently performs comparably to the prior state-of-the-art Leverage SHAP. Meanwhile, the performance of Tree MSR depends on how well the tree-based model approximates the value function; with more samples, it can even outperform Leverage SHAP by several orders of magnitude.

are unbiased and easily extend to all probabilistic values. Moreover, the method can take advantage of non-linear regression methods like XGBoost [CG16] and other decision-tree models.

Instead of starting with a custom linear regression formulation for a given type of probabilistic value, Regression MSR uses regression as a *variance reduction* method for Monte Carlo approximation, and specifically, for the Maximum Sample Reuse method introduced earlier. Concretely, learning from a small number of random subsets, we start by approximating the value function v using a simpler function, f. Using the fact that the probabilistic values are linear — i.e., $\phi_i(v) = \phi_i(f) + \phi_i(v - f)$ for any f — we propose to return the estimator:

$$\tilde{\phi}_i = \phi_i(f) + \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} [v(S) - f(S)] \frac{p_{|S|-1} \mathbb{1}[i \in S] - p_{|S|} \mathbb{1}[i \notin S]}{\mathcal{D}(S)}. \tag{5}$$

Since the MSR estimator (second term) is consistent — i.e., returns the true Shapley values when run on all subsets — the Regression MSR estimator is too. Further, it can be checked that $\mathbb{E}\left[\tilde{\phi}_i\right] = \phi_i$ for any fixed f (e.g., one learned using samples not in $\mathcal{S}$). That is, the method is unbiased. Moreover, like the biased Kernel and Leverage SHAP methods, the variance of $\tilde{\phi}_i$ depends on $[v(S) - f(S)]^2$ (see Section 2 for details). So, our method is more accurate than the standard MSR method when we can obtain a good approximation to v.

The benefit of our approach is clear in Figure 1, where we take f to be a linear approximation to v and use Equation (5) to estimate Shapley values. However, there is also the potential to go beyond linear approximations. Observe that, to evaluate $\tilde{\phi}_i$, f does not need to be a linear. Indeed, we can use *any approximation for which the $\phi_i(f)$ term in Equation* (5) *can be computed efficiently*. I.e., any function family that admits efficient probabilistic value computation. Importantly, this includes a wide variety of functions based on decision trees. Concretely, in Appendix C, we show how to efficiently compute probabilistic values for any linear mixture of decision trees.[4]

[4]Efficient methods for computing Shapley and Banzhaf values for decision trees were previously known [LEL18, LEC+20, KMM+22]. However, they were based on a particular summation property that does not hold for general probabilistic values; if a value function only has contributions from $n' < n$ players, the

We leverage this observation to learn tree-based approximations to v using powerful models like XGBoost (we call this method Tree MSR). For the well-studied Shapley values, we find that Tree MSR achieves state-of-the-art performance, especially when there are enough samples for the tree-based model to learn an accurate fit, see e.g., Figure 2. In particular, Tree MSR can yield estimates with average error that is $2.6\times$ lower than the prior state-of-the-art Leverage SHAP estimator (see Table 1). For general probabilistic values, Tree MSR gives up to $215\times$ lower average error than the best estimator from prior work (see Figure 6 and Table 3).

Concurrent to our work, [BAK+25] introduce Proxy SPEX for estimating probabilistic values. Like Tree MSR, they fit gradient boosted trees to the value function v. However, instead of computing the probabilistic values of the trees, they extract the most influential Fourier terms and compute the probabilistic values of the Fourier representation. In terms of performance, Proxy SPEX outperforms Kernel SHAP for the low sample regime with budget $m \leq 5n$, but Kernel SHAP is more accurate for moderate and larger sample regimes [BAK+25]. Tree MSR underperforms Kernel SHAP (and hence Proxy SPEX) in the low sample regime, but generally outperforms Leverage SHAP in larger sample regimes (see e.g., Figure 2). Proxy SPEX is neither consistent nor unbiased, unlike Regression MSR.

2 Regression MSR

In this section, we present our Regression MSR method, which combines the benefits of Monte Carlo and regression-based estimators. In particular, Regression MSR produces estimates that are unbiased (like Monte Carlo methods), reuses every sample for each estimate (like Maximum Sample Reuse and regression-based methods), and achieves lower variance when a learned approximation is accurate (like regression-based methods). Unlike prior linear regression-based methods, Regression MSR successfully extends to any probabilistic value, and can harness the accuracy of richer function classes like regression trees.

The pseudocode of Regression MSR appears in Algorithm 1. We separate the samples used to train from the samples in the final prediction; this both ensures the estimator is unbiased, and allows us to give strong theoretical guarantees in Theorem 2.1. First, the algorithm partitions m samples into k collections of samples $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(k)}$. The algorithm then proceeds in three phases, repeated for each $\mathcal{S}^{(\ell)}$: During the first phase, Regression MSR learns an approximation $f^{(\ell)}$ to the value function v, on all samples that are *not* in $\mathcal{S}^{(\ell)}$. In the second phase, the probabilistic values $\phi_i(f^{(\ell)})$ are computed for all i. (We run Regression MSR with linear or tree-based methods so that computing their probabilistic values is efficient.) Finally, the algorithm uses the learned function to reduce the variance of the MSR estimates on the samples in $\mathcal{S}^{(\ell)}$.

Theorem 2.1 gives theoretical guarantees on the performance of Regression MSR. For a constant error constraint $\epsilon > 0$ and failure probability $\delta > 0$, Regression MSR uses a linear number of samples to produces estimates with ℓ_2-norm error that depends on a natural weighted squared error between the value function and our worst learned function. We present the guarantee for any sampling distribution $\mathcal{D}$ over subsets, and, below, discuss our suggested choice of this distribution.

Theorem 2.1 (Regression-Adjustment Guarantee)**.** *The estimates produced by Algorithm 1 are unbiased estimates of the probabilistic values. Further, let $\epsilon, \delta > 0$, and $f_{\max}$ be the learned function $f^{(\ell)}$ with largest generalization error over $\ell \in [k]$. When run with $m = O(n\frac{1}{\epsilon\delta})$ samples, Algorithm 1 produces estimates that satisfy, with probability $1 - \delta$,*

$$\|\tilde{\boldsymbol{\phi}} - \boldsymbol{\phi}\|_2^2 \leq \epsilon \sum_{S \subseteq [n]} [v(S) - f_{\max}(S)]^2 \frac{p_{|S|}^2(1 - \frac{|S|}{n}) + p_{|S|-1}^2 \frac{|S|}{n}}{\mathcal{D}(S)}. \tag{6}$$

Algorithm 1 can be used to make the estimates of any regression-based estimator unbiased while preserving its variance. To this end, we purposefully do not specify the sampling distribution $\mathcal{D}$ or the function class f. We next discuss two choices for how to select the model, f, and collect samples.

Linear MSR The simplest choice for f is a linear model. As discussed in the introduction, there is extensive prior work on special linear regression formulations for Shapley values [CGKR88, LL17,

Shapley/Banzhaf value on the induced game of those n' players is the same as the Shapley/Banzhaf value on the original value function with all n players. Our approach in Appendix C is based on an alternative way of viewing tree-based models that avoids the need for this property.

Algorithm 1 Regression Maximum Sample Reuse

Input: number of players n, number of samples m, value function $v : 2^{[n]} \to \mathbb{R}$, probabilistic weights $\mathbf{p} \in [0,1]^n$, probability density function for sampling $\mathcal{D} : 2^{[n]} \to [0,1]$, number of splits k

Output: Estimated probabilistic values $\tilde{\phi}_1, \ldots, \tilde{\phi}_n$

Sample $\mathcal{S}$, consisting of m subsets drawn with (or without) replacement from $\mathcal{D}$.

Randomly partition $\mathcal{S}$ into $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(k)}$.

for $\ell \in \{1, \ldots, k\}$ **do**

  For $i \in [n]$, initialize $\tilde{\phi}_i^{(\ell)} \leftarrow 0$.

  Learn $f^{(\ell)} : 2^{[n]} \to \mathbb{R}$ to minimize loss

$$\sum_{S \in \cup_{\ell' \neq \ell} \mathcal{S}^{(\ell')}} [v(S) - f(S)]^2.$$

  For all $i \in [n]$, compute probabilistic values $\phi_i(f^{(\ell)})$. ▷ Efficient for linear/tree-based models.

  For all $i \in [n]$, compute

$$\tilde{\phi}_i^{(\ell)} \leftarrow \phi_i(f^{(\ell)}) + \frac{1}{|\mathcal{S}^{(\ell)}|} \sum_{S \in \mathcal{S}^{(\ell)}} [v(S) - f^{(\ell)}(S)] \frac{p_{|S|-1} \mathbb{1}[i \in S] - p_{|S|} \mathbb{1}[i \notin S]}{\mathcal{D}(S)}.$$

end for

For all $i \in [n]$, compute final estimate $\tilde{\phi}_i \leftarrow \frac{1}{k} \sum_\ell \tilde{\phi}_i^{(\ell)}$.

return $\tilde{\phi}_1, \ldots, \tilde{\phi}_n$

Table 1: Summary statistics of the average ℓ_2-norm error between estimated and true Shapley values for all datasets listed in Appendix G. All estimators are run with $m = 40n$ samples. Tree MSR achieves average error that is $6.5\times$ lower than Permutation SHAP, $3.8\times$ lower than Kernel SHAP, and $2.6\times$ lower than the prior state-of-the-art Leverage SHAP. We emphasize that Tree MSR gives even better performance for larger sample sizes, as shown in Figure 2. We follow Olympic medal convention: gold, silver and bronze signify first, second and third best performance, respectively.

	Adult	Forest Fires	Real Estate	Bike Sharing	Breast Cancer	Independent	NHANES	Communities	Mean
LinearMSR									
Mean	1.18×10^{-3}	3.07×10^{-5}	2.00×10^{-7}	5.07×10^{-3}	1.09×10^{-3}	9.49×10^{-5}	2.73×10^{-5}	1.17×10^{-4}	9.51×10^{-4}
1st Quartile	2.99×10^{-4}	4.72×10^{-7}	2.46×10^{-8}	9.72×10^{-4}	2.14×10^{-4}	5.38×10^{-5}	2.18×10^{-10}	4.76×10^{-5}	1.98×10^{-4}
2nd Quartile	7.67×10^{-4}	2.75×10^{-6}	5.91×10^{-8}	2.85×10^{-3}	1.02×10^{-3}	6.64×10^{-5}	2.49×10^{-6}	9.46×10^{-5}	6.00×10^{-4}
3rd Quartile	1.52×10^{-3}	6.00×10^{-6}	1.82×10^{-7}	6.37×10^{-3}	1.71×10^{-3}	1.06×10^{-4}	2.88×10^{-5}	1.45×10^{-4}	1.24×10^{-3}
TreeMSR									
Mean	6.77×10^{-5}	1.45×10^{-5}	1.07×10^{-6}	2.04×10^{-3}	1.08×10^{-3}	1.47×10^{-4}	1.95×10^{-7}	7.93×10^{-5}	4.29×10^{-4}
1st Quartile	1.79×10^{-5}	1.32×10^{-6}	9.50×10^{-8}	6.23×10^{-4}	2.37×10^{-4}	2.40×10^{-5}	2.99×10^{-10}	1.78×10^{-5}	1.15×10^{-4}
2nd Quartile	4.12×10^{-5}	3.55×10^{-6}	1.97×10^{-7}	1.28×10^{-3}	5.51×10^{-4}	8.20×10^{-5}	8.89×10^{-10}	3.58×10^{-5}	2.50×10^{-4}
3rd Quartile	9.03×10^{-5}	1.01×10^{-5}	1.40×10^{-6}	2.44×10^{-3}	1.23×10^{-3}	1.70×10^{-4}	3.91×10^{-9}	5.70×10^{-5}	5.00×10^{-4}
KernelSHAP									
Mean	4.55×10^{-3}	2.98×10^{-5}	1.93×10^{-7}	6.12×10^{-3}	2.01×10^{-3}	1.97×10^{-4}	4.08×10^{-5}	1.59×10^{-4}	1.64×10^{-3}
1st Quartile	5.14×10^{-4}	3.08×10^{-7}	1.04×10^{-9}	1.40×10^{-3}	6.87×10^{-4}	1.09×10^{-4}	1.60×10^{-16}	7.03×10^{-5}	3.47×10^{-4}
2nd Quartile	8.59×10^{-4}	3.05×10^{-6}	3.50×10^{-8}	4.00×10^{-3}	1.89×10^{-3}	1.64×10^{-4}	3.10×10^{-6}	1.27×10^{-4}	8.81×10^{-4}
3rd Quartile	2.84×10^{-3}	7.30×10^{-6}	1.59×10^{-7}	7.91×10^{-3}	2.98×10^{-3}	2.80×10^{-4}	3.97×10^{-5}	2.25×10^{-4}	1.79×10^{-3}
PermutationSHAP									
Mean	4.86×10^{-3}	1.25×10^{-4}	5.58×10^{-7}	1.51×10^{-2}	1.73×10^{-3}	1.96×10^{-4}	3.43×10^{-5}	2.14×10^{-4}	2.78×10^{-3}
1st Quartile	1.65×10^{-3}	8.54×10^{-7}	3.64×10^{-9}	3.13×10^{-3}	2.97×10^{-4}	6.96×10^{-5}	1.60×10^{-16}	5.87×10^{-5}	6.50×10^{-4}
2nd Quartile	3.84×10^{-3}	4.83×10^{-6}	4.90×10^{-8}	5.97×10^{-3}	1.05×10^{-3}	1.70×10^{-4}	2.10×10^{-6}	1.61×10^{-4}	1.40×10^{-3}
3rd Quartile	7.68×10^{-3}	1.52×10^{-5}	2.69×10^{-7}	1.92×10^{-2}	1.97×10^{-3}	2.77×10^{-4}	2.09×10^{-5}	2.78×10^{-4}	3.68×10^{-3}
LeverageSHAP									
Mean	1.38×10^{-3}	3.71×10^{-5}	1.44×10^{-7}	6.32×10^{-3}	1.08×10^{-3}	9.62×10^{-5}	2.83×10^{-5}	1.15×10^{-4}	1.13×10^{-3}
1st Quartile	3.35×10^{-4}	3.07×10^{-7}	7.88×10^{-10}	1.05×10^{-3}	2.74×10^{-4}	5.32×10^{-5}	1.60×10^{-16}	4.41×10^{-5}	2.20×10^{-4}
2nd Quartile	6.62×10^{-4}	2.22×10^{-6}	3.21×10^{-8}	2.73×10^{-3}	1.09×10^{-3}	7.30×10^{-5}	2.70×10^{-6}	9.36×10^{-5}	5.81×10^{-4}
3rd Quartile	1.62×10^{-3}	5.13×10^{-6}	1.29×10^{-7}	7.03×10^{-3}	1.46×10^{-3}	1.08×10^{-4}	2.62×10^{-5}	1.52×10^{-4}	1.30×10^{-3}

CL21, MW25]. Using a linear function for variance reduction in MSR offers a natural alternative to these methods, and adds negligible computational overhead, yet tends to show superior performance on most datasets (see, e.g., Table 1). When applying the method to Shapley values specifically, we use the existing state-of-the art Leverage SHAP method (which samples via leverage scores) to fit the learned function. We similarly use the linear regression-based Kernel Banzhaf algorithm [LWK+25] when estimating Banzhaf values. For general probabilistic values, Linear MSR fits a linear model with the sampling distribution described below, and uses its predictions to adjust the final estimates.

Tree MSR Beyond linear models, tree-based regression models like XGBoost can learn more accurate approximations. As discussed in the introduction, it is known how to efficiently compute the Shapley values [LEC+20] and Banzhaf values [KMM+22] of trees; however, it was previously unclear how to generalize these approaches to probabilistic values: the Shapley/Banzhaf methods use the property that the probabilistic value of a value function on n players is the probabilistic value of the extended value function on $n' > n$ players, as long as the additional $n' - n$ players always contribute nothing. This property unfortunately does not hold for all probabilistic values; e.g., consider probabilistic weights $\mathbf{p}$ that are independently sampled (and normalized) for each number of players n and n'. Instead, efficiently computing the probabilistic values of trees requires a subtly different approach, which we describe in Appendix C. With this approach in hand, we can fit v with a tree-based model like XGBoost and efficiently compute its probabilistic values for the final estimate.

Sampling Distribution When Algorithm 1 is not run on top of another regression-based estimator, we choose the sampling distribution so that the function f is directly trained to minimize the error bound in Theorem 2.1. That is, we sample each set with probability proportional to

$$\sqrt{p_{|S|}^2(1 - \frac{|S|}{n}) + p_{|S|-1}^2 \frac{|S|}{n}}.$$

Then the error bound in Theorem 2.1 is proportional to

$$\sum_{S \subseteq [n]} [v(S) - f(S)]^2 \sqrt{p_{|S|}^2(1 - \frac{|S|}{n}) + p_{|S|-1}^2 \frac{|S|}{n}}$$

which, by design, is the *expected* loss used to train f.

Bias vs. Accuracy vs. Runtime Regression MSR produces unbiased estimates by training k functions, each on a $(k-1)/k$ fraction of the available samples and evaluating on the held-out $1/k$. This creates a trade-off: increasing k improves accuracy (each function has a larger training set) but raises computational cost. In our experiments, we set $k = 10$, meaning that each function is trained on 90% of the data. Thanks to efficient solvers (e.g., least squares or XGBoost), this setup maintains fast runtimes while delivering high accuracy.

Practical Simplification Unless a small bias term (similar to that of Leverage SHAP or Kernel SHAP) is unacceptable, we recommend simplifying the Regression MSR algorithm: Train a single function—and build the final estimate with it—on all samples. While this introduces a small bias (and breaks the theoretical guarantees), the resulting algorithm runs faster and is generally more accurate.

3 Experiments

In this section, we describe our experiments on eight datasets. Overall, we find that Linear MSR and Tree MSR give state-of-the-art performance for almost all datasets and sample budgets.

Value Function For evaluation, we focus on the explainable AI feature attribution task, but emphasize that our methods can be applied to any application involving probabilistic values, as we only require black-box access to the value function v. Concretely, we train a model on a dataset, and attribute the prediction the model makes on a given *explicand* point $\mathbf{x}^e \in \mathbb{R}^n$ to its n input features. We consider the *interventional* definition of v, where the explanation is relative to a *baseline* point $\mathbf{x}^b \in \mathbb{R}^n$: For a set S, let $\mathbf{x}^S$ be the point where the ith feature is x_i^e if $i \in S$ and x_i^b otherwise. Then, the value function $v(S)$ is the model's prediction on $\mathbf{x}^S$. There is also a conditional version of feature attribution, where the features not in S are drawn from a background dataset [LL17, LEL18]. However, we choose to focus on the interventional version since it is more efficient to compute $v(S)$, and the resulting probabilistic values are more interpretable [JMB20].

Ground Truth Probabilistic Values For small datasets with $n < 30$, we use a neural network model and compute the true probabilistic values through enumeration. For larger datasets with $n \geq 30$ where exact enumeration is infeasible, we use a random forest model and compute the true probabilistic values using the algorithm described in Appendix C. (Please see Appendix G for a summary of the datasets in our experiments.) We emphasize that our method for computing the probabilistic values of trees enables the first experiments on medium to large datasets where we can compare the estimates

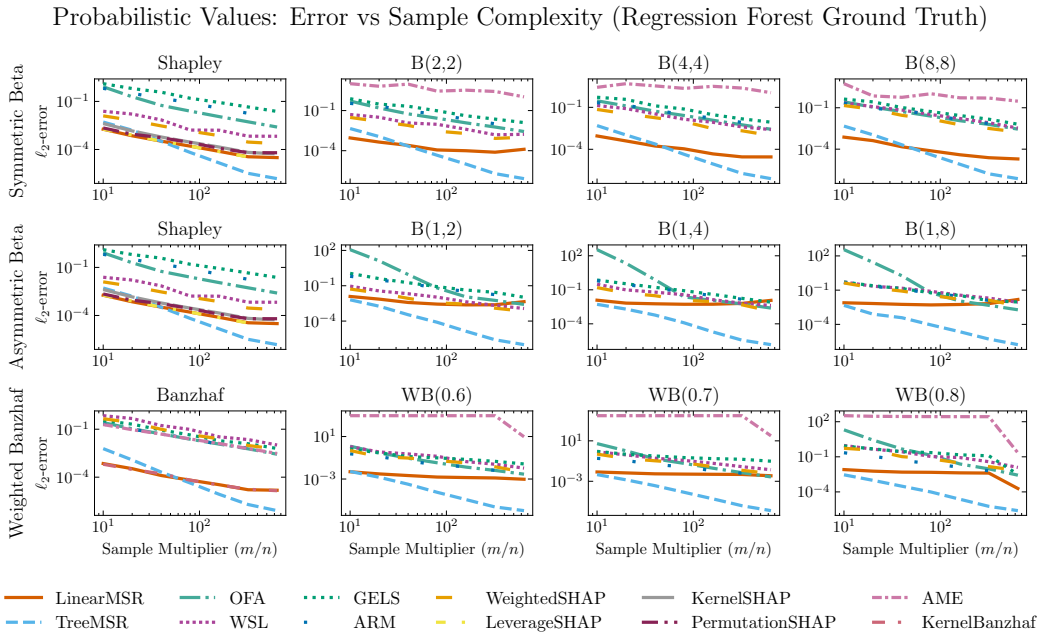

Figure 3: Average error between the estimated and true probabilistic values as a function of sample size. Each subplot shows results for a different probabilistic value with the error averaged over all large datasets ($n \geq 30$), for which we used the tree-based method described in Appendix C. The lines report the mean error over 10 runs. Tree MSR gives the best performance, often by several orders of magnitude when m is large.

Table 2: Summary statistics of the ℓ_2-norm error between estimated and true probabilistic values when $m = 40n$. We summarize the error over large datasets ($n \geq 30$), for which we use the tree-based method described in Appendix C to compute the true probabilistic values. On average over all probabilistic values, Tree MSR produces estimates with mean error that is $215\times$ lower than the best estimator from prior work.

	B(1,1)	B(2,2)	B(4,4)	B(8,8)	B(1,2)	B(1,4)	B(1,8)	WB(0.5)	WB(0.6)	WB(0.7)	WB(0.8)	WB(0.9)	Mean
LinearMSR													
Mean	3.03×10^{-4}	2.50×10^{-4}	1.72×10^{-4}	1.48×10^{-4}	3.80×10^{-3}	5.85×10^{-3}	5.66×10^{-3}	1.45×10^{-4}	2.13×10^{-3}	5.46×10^{-3}	4.91×10^{-3}	1.13×10^{-3}	2.50×10^{-3}
1st Quartile	6.26×10^{-6}	8.70×10^{-6}	5.99×10^{-6}	6.65×10^{-6}	5.59×10^{-4}	5.25×10^{-4}	3.25×10^{-4}	5.29×10^{-6}	7.66×10^{-4}	1.53×10^{-3}	2.51×10^{-4}	1.02×10^{-4}	3.41×10^{-4}
2nd Quartile	5.90×10^{-5}	4.74×10^{-5}	4.42×10^{-5}	3.16×10^{-5}	1.06×10^{-3}	1.03×10^{-3}	7.56×10^{-4}	3.81×10^{-5}	1.54×10^{-3}	4.27×10^{-3}	7.39×10^{-4}	2.79×10^{-4}	8.25×10^{-4}
3rd Quartile	1.46×10^{-4}	1.23×10^{-4}	1.37×10^{-4}	8.92×10^{-5}	3.01×10^{-3}	3.11×10^{-3}	2.47×10^{-3}	8.76×10^{-5}	2.94×10^{-3}	7.17×10^{-3}	5.21×10^{-3}	8.08×10^{-4}	2.11×10^{-3}
TreeMSR													
Mean	2.64×10^{-4}	2.21×10^{-4}	2.41×10^{-4}	2.06×10^{-4}	3.47×10^{-4}	5.91×10^{-4}	3.83×10^{-4}	2.09×10^{-4}	3.25×10^{-4}	3.67×10^{-4}	3.06×10^{-4}	2.18×10^{-4}	3.07×10^{-4}
1st Quartile	2.48×10^{-6}	1.29×10^{-6}	9.50×10^{-7}	1.23×10^{-6}	1.77×10^{-6}	4.10×10^{-6}	1.08×10^{-5}	1.17×10^{-6}	8.75×10^{-7}	7.87×10^{-7}	5.09×10^{-6}	7.61×10^{-6}	3.18×10^{-6}
2nd Quartile	4.77×10^{-5}	3.61×10^{-5}	2.91×10^{-5}	2.94×10^{-5}	5.24×10^{-5}	8.58×10^{-5}	8.84×10^{-5}	2.97×10^{-5}	3.14×10^{-5}	3.11×10^{-5}	3.82×10^{-5}	3.59×10^{-5}	4.46×10^{-5}
3rd Quartile	3.28×10^{-4}	2.31×10^{-4}	2.12×10^{-4}	1.72×10^{-4}	3.37×10^{-4}	5.80×10^{-4}	3.54×10^{-4}	1.85×10^{-4}	2.36×10^{-4}	2.85×10^{-4}	3.23×10^{-4}	1.97×10^{-4}	2.87×10^{-4}
OFA													
Mean	5.85×10^{-2}	5.67×10^{-2}	5.27×10^{-2}	5.31×10^{-2}	9.00×10^{-1}	7.67×10^{-1}	1.80	4.65×10^{-2}	9.05×10^{-2}	1.66×10^{-1}	4.66×10^{-1}	1.24	4.75×10^{-1}
1st Quartile	4.61×10^{-2}	4.96×10^{-2}	4.59×10^{-2}	4.64×10^{-2}	5.88×10^{-2}	5.58×10^{-2}	5.62×10^{-2}	4.14×10^{-2}	5.10×10^{-2}	5.07×10^{-2}	5.59×10^{-2}	6.37×10^{-2}	5.18×10^{-2}
2nd Quartile	5.91×10^{-2}	5.69×10^{-2}	5.36×10^{-2}	5.18×10^{-2}	1.03×10^{-1}	7.27×10^{-2}	8.48×10^{-2}	4.56×10^{-2}	5.76×10^{-2}	6.21×10^{-2}	8.95×10^{-2}	9.57×10^{-2}	6.94×10^{-2}
3rd Quartile	6.99×10^{-2}	6.51×10^{-2}	5.89×10^{-2}	5.99×10^{-2}	3.07×10^{-1}	1.83×10^{-1}	3.30×10^{-1}	5.15×10^{-2}	6.63×10^{-2}	9.03×10^{-2}	1.62×10^{-1}	2.13×10^{-1}	1.38×10^{-1}
WSL													
Mean	6.33×10^{-3}	1.19×10^{-2}	3.37×10^{-2}	6.80×10^{-2}	2.10×10^{-2}	6.20×10^{-2}	1.48×10^{-1}	2.32×10^{-1}	2.30×10^{-1}	1.92×10^{-1}	2.70×10^{-1}	4.10×10^{-1}	1.40×10^{-1}
1st Quartile	3.93×10^{-4}	5.20×10^{-3}	1.17×10^{-2}	2.62×10^{-2}	7.76×10^{-3}	1.53×10^{-2}	5.72×10^{-2}	5.58×10^{-2}	6.13×10^{-2}	7.05×10^{-2}	1.23×10^{-1}	1.41×10^{-1}	4.80×10^{-2}
2nd Quartile	1.55×10^{-3}	8.08×10^{-3}	2.47×10^{-2}	4.58×10^{-2}	1.47×10^{-2}	5.21×10^{-2}	8.81×10^{-2}	1.37×10^{-1}	1.30×10^{-1}	1.42×10^{-1}	2.52×10^{-1}	2.69×10^{-1}	9.71×10^{-2}
3rd Quartile	5.09×10^{-3}	1.55×10^{-2}	4.56×10^{-2}	8.15×10^{-2}	3.15×10^{-2}	9.59×10^{-2}	1.79×10^{-1}	2.54×10^{-1}	2.46×10^{-1}	1.95×10^{-1}	3.79×10^{-1}	5.05×10^{-1}	1.69×10^{-1}
GELS													
Mean	2.80×10^{-1}	2.11×10^{-1}	1.28×10^{-1}	1.04×10^{-1}	1.81×10^{-1}	1.65×10^{-1}	1.37×10^{-1}	1.10×10^{-1}	1.30×10^{-1}	3.18×10^{-1}	2.79×10^{-1}	5.74×10^{-2}	1.75×10^{-1}
1st Quartile	1.53×10^{-1}	1.14×10^{-1}	7.55×10^{-2}	6.24×10^{-2}	7.95×10^{-2}	7.40×10^{-2}	6.76×10^{-2}	5.64×10^{-2}	1.01×10^{-1}	2.09×10^{-1}	3.06×10^{-2}	2.85×10^{-2}	8.75×10^{-2}
2nd Quartile	2.02×10^{-1}	1.73×10^{-1}	1.01×10^{-1}	7.45×10^{-2}	1.41×10^{-1}	1.20×10^{-1}	9.04×10^{-2}	7.20×10^{-2}	1.33×10^{-1}	3.33×10^{-1}	4.99×10^{-2}	3.97×10^{-2}	1.27×10^{-1}
3rd Quartile	3.03×10^{-1}	2.73×10^{-1}	1.55×10^{-1}	1.15×10^{-1}	2.38×10^{-1}	2.41×10^{-1}	1.27×10^{-1}	1.39×10^{-1}	1.60×10^{-1}	3.95×10^{-1}	2.33×10^{-1}	6.49×10^{-2}	2.04×10^{-1}
ARM													
Mean	1.26×10^{-1}	9.30×10^{-2}	6.68×10^{-2}	6.45×10^{-2}	1.70×10^{-1}	1.41×10^{-1}	1.23×10^{-1}	5.94×10^{-2}	5.06×10^{-2}	4.64×10^{-2}	4.82×10^{-2}	5.23×10^{-2}	8.69×10^{-2}
1st Quartile	6.85×10^{-2}	4.78×10^{-2}	3.78×10^{-2}	3.22×10^{-2}	7.25×10^{-2}	8.41×10^{-2}	7.84×10^{-2}	3.18×10^{-2}	3.18×10^{-2}	3.12×10^{-2}	3.39×10^{-2}	4.02×10^{-2}	4.92×10^{-2}
2nd Quartile	9.96×10^{-2}	6.47×10^{-2}	4.48×10^{-2}	4.36×10^{-2}	1.16×10^{-1}	1.08×10^{-1}	9.55×10^{-2}	4.09×10^{-2}	4.43×10^{-2}	4.17×10^{-2}	4.29×10^{-2}	4.67×10^{-2}	6.57×10^{-2}
3rd Quartile	1.40×10^{-1}	9.12×10^{-2}	7.38×10^{-2}	8.65×10^{-2}	1.72×10^{-1}	1.63×10^{-1}	1.42×10^{-1}	6.22×10^{-2}	6.25×10^{-2}	5.81×10^{-2}	5.49×10^{-2}	6.38×10^{-2}	9.75×10^{-2}
WeightedSHAP													
Mean	2.19×10^{-3}	7.74×10^{-3}	1.81×10^{-2}	2.83×10^{-2}	8.50×10^{-3}	3.02×10^{-2}	8.27×10^{-2}	8.75×10^{-2}	9.45×10^{-2}	1.06×10^{-1}	1.03×10^{-1}	2.25×10^{-1}	6.62×10^{-2}
1st Quartile	2.40×10^{-4}	3.04×10^{-3}	6.62×10^{-3}	8.16×10^{-3}	4.10×10^{-3}	8.39×10^{-3}	2.47×10^{-2}	3.21×10^{-2}	2.53×10^{-2}	2.53×10^{-2}	3.61×10^{-2}	6.98×10^{-2}	2.03×10^{-2}
2nd Quartile	8.74×10^{-4}	4.91×10^{-3}	1.11×10^{-2}	2.12×10^{-2}	7.68×10^{-3}	2.51×10^{-2}	4.33×10^{-2}	5.43×10^{-2}	7.04×10^{-2}	6.18×10^{-2}	6.52×10^{-2}	1.31×10^{-1}	4.14×10^{-2}
3rd Quartile	2.49×10^{-3}	9.36×10^{-3}	1.78×10^{-2}	4.14×10^{-2}	1.27×10^{-2}	3.98×10^{-2}	1.07×10^{-1}	1.10×10^{-1}	1.07×10^{-1}	1.56×10^{-1}	1.26×10^{-1}	2.81×10^{-1}	8.43×10^{-2}

to the *ground truth* probabilistic values. Such experiments were previously done for Shapley and Banzhaf values [MW25, LWK$^+$25], but not for other probabilistic values.

Baselines We compare Linear MSR and Tree MSR to a wide variety of probabilistic value estimators from prior work. For the popular task of estimating Shapley values, we focus on the most effective estimators for general value functions i.e., Permutation SHAP [CGT09], Kernel SHAP [LL17, CL21],

and Leverage SHAP [MW25]. We use the optimized implementations of Permutation SHAP and Kernel SHAP in the ubiquitous SHAP library for parity [LL17]. For estimating probabilistic values, there has been substantial recent interest in designing estimators [KZ22a, KZ22b, LZL$^+$22, WJ23, KBMH24, LY24a, LY24b]. These estimators generally use the standard Monte Carlo approach, apply the Maximum Sample Reuse idea, or extend linear regression-based methods. We provide a description of each in Appendix D.

Error and Uncertainty We measure the error between the true probabilistic values $\boldsymbol{\phi}$ and the estimated probabilistic values $\tilde{\boldsymbol{\phi}}$ with the ℓ_2-norm error $\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}\|_2^2 / \|\boldsymbol{\phi}\|_2^2$. All of our tables and figures report summary statistics over at least 10 runs. In the tables, we report the mean, first quartile, median, and third quartiles of the error. (We do not report $+/-$ standard deviation because these are often negative for the small errors in our experiments.) In the figures, we report the mean error.

Implementation Details We use scikit-learn [PVG$^+$11] and XGBoost [CG16] for training our models. For the implementations of Permutation SHAP and Kernel SHAP, we use the SHAP library [LL17]. Please see Appendix G for details on how we accessed each dataset. All of our experiments are run on a machine with an Apple M2 chip and 8GB RAM.

We first describe our experiments on the popular task of estimating Shapley values. Figure 2 shows estimator performance by sample size, and Table 1 highlights the corresponding uncertainty statistics when each estimator is run with $m = 40n$ samples. We find that Linear MSR generally improves the prior state-of-the-art Leverage SHAP by making its estimates unbiased, but the gain is marginal. In contrast, the performance Tree MSR depends on how well the tree-based approximation f fits the underlying value function v. When the number of samples is smaller, Tree MSR is comparable to prior Shapley value estimators; however, as the number of samples grows, the tree-based model becomes more accurate and Tree MSR often gives the best performance, sometimes by orders of magnitude. As can be seen in Table 1, **Tree MSR can give average error that is $2.6\times$ lower than the prior state-of-the-art Leverage SHAP**, when $m = 40n$. For larger sample sizes, Tree MSR outperforms Leverage SHAP by an even wider margin, as shown in Figure 2.

Beyond estimating Shapley values, we run experiments on estimating the more general beta Shapley values and weighted Banzhaf values (see Appendix B for definitions). Figure 6 shows estimator performance by sample size for small datasets (where we can feasibly compute the true probabilistic values of neural networks), and Table 3 highlights the corresponding uncertainty statistics when each estimator is run with $m = 40n$ samples. We present the analogous results for smaller datasets (where we can exactly compute the probabilistic values of neural networks) in Figure 6 and Table 3 in Appendix E. We find that Linear MSR generally plateaus as the number of samples increases. In contrast, Tree MSR gives the best performance across the board, with the gap to the next best estimator widening with the number of samples. We confirm this finding in Figure 4. As can be seen in Table 3, **Tree MSR can give average error that is $215\times$ lower than the best probabilistic value estimator from prior work**.

We provide additional experiments on the effect of noisy access to the value function in Appendix F. Figures 7 and 8 suggest that Tree MSR is particularly resilient to noise.

Limitations and Broader Impacts

The performance of our methods depends on the underlying fit of the learned model. When the dataset is structured or the number of samples is large, the learned model is accurate and the estimators are, too. However, for datasets with less structure or in sample-constrained settings, the performance of our estimators can worsen.

The primary application of our work is in explainable AI, where we seek to understand how features and data points contribute to the performance of machine learning models. We expect the broader impact of our work to be better model explanations, and we do not see substantial negative risks as a result of our research.

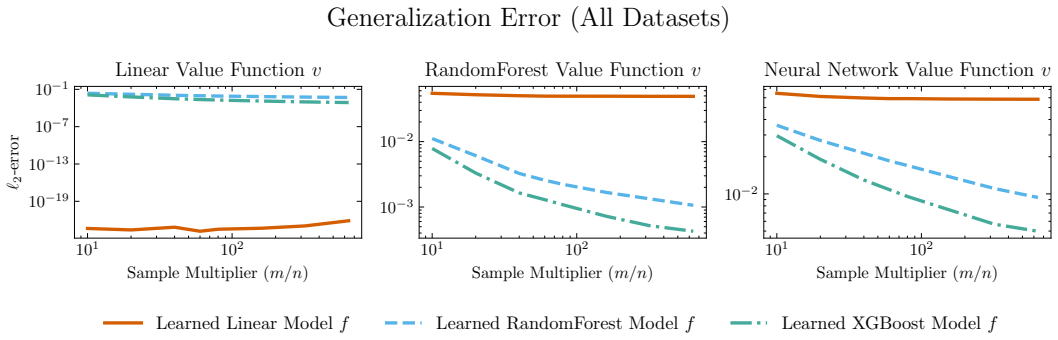

Figure 4: Generalization error between value function v and learned model f by sample size, averaged over all datasets. When the base model is linear, the learned linear model quickly fits it to machine precision. When the base model is a random forest or a neural network, the error of the linear model plateaus while the random forest and XGBoost learned models continue to improve. This phenomenon is reflected in Figures 3 and 6; the performance of Tree MSR continues to improve with the number of samples while Linear MSR plateaus.

Acknowledgments and Disclosure of Funding

Witter was supported by NSF Graduate Research Fellowship Grant No. DGE-2234660. Liu was partially supported by NSF Awards IIS-2106888 and the DARPA ASKEM and ARPA-H BDF programs. Musco was partially supported by NSF Award CCF-2045590.

References

[BAK+25] Landon Butler, Abhineet Agarwal, Justin Singh Kang, Yigit Efe Erginbas, Bin Yu, and Kannan Ramchandran. Proxyspex: Inference-efficient interpretability via sparse feature interactions in llms. *arXiv preprint arXiv:2505.17495*, 2025.

[BI64] John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19:317, 1964.

[Cen23] Centers for Disease Control and Prevention. National health and nutrition examination survey (nhanes). `https://wwwn.cdc.gov/nchs/nhanes/`, 2023. `https://wwwn.cdc.gov/nchs/nhanes/`.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[CGKR88] A Charnes, B Golany, M Keane, and J Rousseau. Extremal principle solutions of games in characteristic function form: core, chebychev and shapley value generalizations. *Econometrics of planning and efficiency*, pages 123–133, 1988.

[CGT09] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

[CL21] Ian Covert and Su-In Lee. Improving KernelSHAP: Practical Shapley value estimation using linear regression. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 3457–3465, 2021.

[CM07] Paulo Cortez and Aníbal Morais. A data mining approach to predict forest fires using meteorological data. *Proceedings of the 13th EPIA*, 7:512–523, 2007.

[CSV+25] Tyler Chen, Akshay Seshadri, Mattia J Villani, Pradeep Niroula, Shouvanik Chakrabarti, Archan Ray, Pranav Deshpande, Romina Yalovetzky, Marco Pistoia, and Niraj Kumar. A unified framework for provably efficient algorithms to estimate shapley values. *arXiv preprint arXiv:2506.05216*, 2025.

[FTG14] Hadi Fanaee-T and João Gama. Event labeling combining ensemble detectors and background knowledge. In *Progress in Artificial Intelligence (EPIA)*, pages 1–15. Springer, 2014.

[GZ19] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.

[HH92] Peter L Hammer and Ron Holzman. Approximations of pseudo-boolean functions; applications to game theory. *Zeitschrift für Operations Research*, 36(1):3–21, 1992.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[JMB20] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable ai: A causal problem. In *International Conference on artificial intelligence and statistics*, pages 2907–2916. PMLR, 2020.

[KBMH24] Patrick Kolpaczki, Viktor Bengs, Maximilian Muschalik, and Eyke Hüllermeier. Approximating the shapley value without marginal contributions. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 38, pages 13246–13255, 2024.

[KMM+22] Adam Karczmarz, Tomasz Michalak, Anish Mukherjee, Piotr Sankowski, and Piotr Wygocki. Improved feature importance computation for tree models based on the banzhaf value. In *Uncertainty in Artificial Intelligence*, pages 969–979. PMLR, 2022.

[Koh96] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 202–207, 1996.

[KZ22a] Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 8780–8802. PMLR, 28–30 Mar 2022.

[KZ22b] Yongchan Kwon and James Y Zou. Weightedshap: analyzing and improving shapley based feature attributions. *Advances in Neural Information Processing Systems*, 35:34363–34376, 2022.

[LEC+20] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67, 2020.

[LEL18] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[LWK+25] Yurong Liu, R. Teal Witter, Flip Korn, Tarfah Alrashed, Dimitris Paparas, Christopher Musco, and Juliana Freire. Kernel banzhaf: A fast and robust estimator for banzhaf values, 2025.

[LY24a] Weida Li and Yaoliang Yu. Faster approximation of probabilistic and distributional values via least squares. In *The Twelfth International Conference on Learning Representations*, 2024.

[LY24b] Weida Li and Yaoliang Yu. One sample fits all: Approximating all probabilistic values simultaneously and efficiently. *Advances in Neural Information Processing Systems*, 2024.

[LY24c] Weida Li and Yaoliang Yu. Robust data valuation with weighted banzhaf values. *Advances in Neural Information Processing Systems*, 36, 2024.

[LZL+22] Jinkun Lin, Anqi Zhang, Mathias Lécuyer, Jinyang Li, Aurojit Panda, and Siddhartha Sen. Measuring the effect of training data on deep learning predictions via randomized experiments. In *International Conference on Machine Learning*, pages 13468–13504. PMLR, 2022.

[MW25] Christopher Musco and R Teal Witter. Provably accurate shapley value estimation via leverage score sampling. In *International conference on learning representations*, 2025.

[Pen46] Lionel S Penrose. The elementary statistics of majority voting. *Journal of the Royal Statistical Society*, 109(1):53–57, 1946.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RC02] Michael Redmond and Corinna Cortes. Communities and crime data set. Technical report, UCI Machine Learning Repository, 2002. `https://archive.ics.uci.edu/ml/datasets/communities+and+crime`.

[RVZ98] Luis M Ruiz, Federico Valenciano, and Jose M Zarzuelo. The family of least square values for transferable utility games. *Games and Economic Behavior*, 24(1-2):109–130, 1998.

[Sar06] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[Sha51] Lloyd S. Shapley. Notes on the n-person game—ii: The value of an n-person game. Research Memorandum RM-670, RAND Corporation, 1951.

[SK10] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

[SS11] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[SWM93] WN Street, WH Wolberg, and OL Mangasarian. Nuclear feature extraction for breast tumor diagnosis. *IS&TSPIE 1993 International Symposium on Electronic Imaging: Science and Technology*, 1905:861–870, 1993.

[Web88] Robert James Weber. *Probabilistic values for games*, page 101–120. Cambridge University Press, 1988.

[WJ23] Jiachen T. Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 6388–6421. PMLR, 25–27 Apr 2023.

[WMSJ25] Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. Data shapley in one training run. In *International conference on learning representations*, 2025.

[YH09] I-Cheng Yeh and Ting-Kuei Hsu. Comparisons of data mining techniques for predicting real-estate prices. *The International Journal of Artificial Intelligence Tools*, 18(03):435–446, 2009.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please see Sections 2 and 3, as well as Appendix A.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see the Limitations and Broader Impacts section immediately before the references.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Please see Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Please see Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is available in the supplementary material, and will be made accessible after publication.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please see Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Please see Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please see Section 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please see the Limitations and Broader Impacts section before the references.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not have datasets or models that have a high risk of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Please see Appendix G.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Please see the supplementary material.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We did not conduct research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We did not conduct research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this work does not involve LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Proof of Error Bound

Theorem 2.1 (Regression-Adjustment Guarantee)**.** *The estimates produced by Algorithm 1 are unbiased estimates of the probabilistic values. Further, let $\epsilon, \delta > 0$, and $f_{\max}$ be the learned function $f^{(\ell)}$ with largest generalization error over $\ell \in [k]$. When run with $m = O(n\frac{1}{\epsilon\delta})$ samples, Algorithm 1 produces estimates that satisfy, with probability $1-\delta$,*

$$\|\tilde{\boldsymbol{\phi}} - \boldsymbol{\phi}\|_2^2 \leq \epsilon \sum_{S\subseteq[n]} [v(S) - f_{\max}(S)]^2 \frac{p_{|S|}^2(1-\frac{|S|}{n}) + p_{|S|-1}^2\frac{|S|}{n}}{\mathcal{D}(S)}. \tag{6}$$

Proof of Theorem 2.1. We will analyze the variance when the samples are drawn with replacement. By Theorem 4 in [Hoe63], the variance can only decrease when samples are drawn without replacement.

Consider $\ell \in [k]$. For simplicity, suppose that $|\mathcal{S}^{(\ell)}| = m/k$. We will first show that each estimated probabilistic value $\tilde{\phi}_i^{(\ell)}$ is unbiased:

$$\begin{aligned}
\mathbb{E}[\tilde{\phi}_i^{(\ell)}] &= \phi_i(f^{(\ell)}) + \frac{k}{m}\mathbb{E}\left[\sum_{S'\in\mathcal{S}^{(\ell)}}\sum_{S\subseteq[n]} \frac{\mathbb{1}[S=S']}{\mathcal{D}(S)}[v(S) - f^{(\ell)}(S)]\left(p_{|S|-1}\mathbb{1}[i\in S] - p_{|S|}\mathbb{1}[i\notin S]\right)\right] \\
&= \phi_i(f^{(\ell)}) + \sum_{S\subseteq[n]}[v(S) - f^{(\ell)}(S)](p_{|S|-1}\mathbb{1}[i\in S] - p_{|S|}\mathbb{1}[i\notin S]) \\
&= \phi_i(f^{(\ell)}) + \sum_{S\subseteq[n]\setminus\{i\}}[v(S\cup\{i\}) - v(S)]p_{|S|} - \sum_{S\subseteq[n]\setminus\{i\}}[f^{(\ell)}(S\cup\{i\}) - f^{(\ell)}(S)]p_{|S|} \\
&= \sum_{S\subseteq[n]\setminus\{i\}}[v(S\cup\{i\}) - v(S)]p_{|S|} = \phi_i.
\end{aligned}$$

Since $\tilde{\phi}_i^{(\ell)}$ is unbiased, the final estimate $\mathbb{E}[\frac{1}{k}\sum_{\ell=1}^k \tilde{\phi}_i^{(\ell)}]$ is also unbiased by the linearity of expectation. Next, we will analyze the variance of each estimate:

$$\begin{aligned}
\text{Var}[\tilde{\phi}_i^{(\ell)}] &= \frac{k^2}{m^2}\text{Var}\left[\sum_{S'\in\mathcal{S}^{(\ell)}}\sum_{S\subseteq[n]}\mathbb{1}[S=S'][v(S) - f^{(\ell)}(S)]\frac{p_{|S|-1}\mathbb{1}[i\in S] - p_{|S|}\mathbb{1}[i\notin S]}{\mathcal{D}(S)}\right] \\
&\leq \frac{k}{m}\sum_{S\subseteq[n]}\mathcal{D}(S)[v(S) - f^{(\ell)}(S)]^2\left(\frac{p_{|S|-1}\mathbb{1}[i\in S] - p_{|S|}\mathbb{1}[i\notin S]}{\mathcal{D}(S)}\right)^2 \\
&= \frac{k}{m}\sum_{S\subseteq[n]}[v(S) - f^{(\ell)}(S)]^2\frac{p_{|S|-1}^2\mathbb{1}[i\in S] + p_{|S|}^2\mathbb{1}[i\notin S]}{\mathcal{D}(S)}.
\end{aligned} \tag{7}$$

Let $\boldsymbol{\phi} \in \mathbb{R}^n$ and $\tilde{\boldsymbol{\phi}}^{(\ell)} \in \mathbb{R}^n$ be vectors containing the true and estimated probabilistic values, respectively. We will analyze the random variable $\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\|^2$. By linearity of expectation, we have

$$\mathbb{E}[\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\|^2] = \mathbb{E}\left[\sum_{i=1}^n(\phi_i - \tilde{\phi}_i^{(\ell)})^2\right] = \sum_{i=1}^n\mathbb{E}\left[(\phi_i - \tilde{\phi}_i^{(\ell)})^2\right] = \sum_{i=1}^n\text{Var}[\phi_i - \tilde{\phi}_i^{(\ell)}] = \sum_{i=1}^n\text{Var}[\tilde{\phi}_i^{(\ell)}]$$

where the penultimate equality follows because $\mathbb{E}[\tilde{\phi}_i^{(\ell)}] = \phi_i$ and the final equality follows because ϕ_i is a constant with respect to the randomness of the estimator. Plugging in Equation (7), we get

$$\begin{aligned}
\mathbb{E}[\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\|^2] &\leq \sum_{i=1}^n\frac{k}{m}\sum_{S\subseteq[n]}[v(S) - f^{(\ell)}(S)]^2\frac{p_{|S|-1}^2\mathbb{1}[i\in S] + p_{|S|}^2\mathbb{1}[i\notin S]}{\mathcal{D}(S)} \\
&= \frac{k}{m}\sum_{S\subseteq[n]}[v(S) - f^{(\ell)}(S)]^2\frac{p_{|S|-1}^2|S| + p_{|S|}^2(n-|S|)}{\mathcal{D}(S)} \\
&\leq \frac{k}{m}\sum_{S\subseteq[n]}[v(S) - f_{\max}(S)]^2\frac{p_{|S|-1}^2|S| + p_{|S|}^2(n-|S|)}{\mathcal{D}(S)}
\end{aligned} \tag{8}$$

where

$$f_{\max} := f^{(\ell^*)}, \quad \text{where } \ell^* = \arg\max_{\ell \in [k]} \sum_{S \subseteq [n]} [v(S) - f^{(\ell)}(S)]^2 \frac{p_{|S|-1}^2 |S| + p_{|S|}^2 (n - |S|)}{\mathcal{D}(S)}.$$

We now apply Markov's inequality to $\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\|^2$ for each ℓ:

$$\Pr\left(\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\|^2 \geq \frac{1}{\delta'} \mathbb{E}[\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\|^2] \right) \leq \delta'.$$

We are interested in the final estimate $\tilde{\boldsymbol{\phi}} = \frac{1}{k} \sum_{\ell=1}^{k} \tilde{\boldsymbol{\phi}}^{(\ell)}$. By the Union Bound, we have, with probability at most $k\delta'$,

$$\sum_{\ell=1}^{k} \|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\| \geq \sum_{\ell=1}^{k} \sqrt{\frac{1}{\delta'} \mathbb{E}[\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\|^2]}.$$

By the triangle inequality, setting $\delta' = \delta/k$, and taking the complement, we have, with probability $1 - \delta$,

$$\begin{aligned}
k\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}\| &\leq \sum_{\ell=1}^{k} \|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\| && \text{(by triangle inequality)} \\
&\leq \sum_{\ell=1}^{k} \sqrt{\frac{k}{\delta} \mathbb{E}[\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}^{(\ell)}\|^2]} && \text{(by setting } \delta' = \delta/k\text{)} \\
&\leq k \sqrt{\frac{k}{\delta} \frac{k}{m} \sum_{S \subseteq [n]} [v(S) - f_{\max}(S)]^2 \frac{p_{|S|-1}^2 |S| + p_{|S|}^2 (n - |S|)}{\mathcal{D}(S)}}. && \text{(by Equation (8))}
\end{aligned}$$

Then, with probability $1 - \delta$,

$$\|\boldsymbol{\phi} - \tilde{\boldsymbol{\phi}}\|^2 \leq \frac{k}{\delta} \frac{k}{m} \sum_{S \subseteq [n]} [v(S) - f_{\max}(S)]^2 \frac{p_{|S|-1}^2 |S| + p_{|S|}^2 (n - |S|)}{\mathcal{D}(S)}.$$

The theorem statement follows by setting $m = k^2 \frac{n}{\epsilon\delta} = O(\frac{n}{\epsilon\delta})$. □

B Beta Shapley and Weighted Banzhaf Values

Probabilistic values satisfy three intuitive properties:

- Linearity: The probabilistic value of a linear combination of value functions is the linear combination of the probabilistic values for each value function i.e., $\phi_i(av+bw) = a\phi_i(v) + b\phi_i(w)$ for real values $a, b \in \mathbb{R}$ and games $v, w : 2^{[n]} \to \mathbb{R}$.
- Null Player: The probabilistic value for a player that has no effect on any coalition is 0 i.e., if $v(S \cup \{i\}) = v(S)$ for all S then $\phi_i = 0$.
- Symmetry: If two players contribute equally to all coalitions then they have the same probabilistic value i.e., if $v(S \cup \{i\}) = v(S \cup \{j\})$ for all S then $\phi_i = \phi_j$.

In addition to these three properties, Shapley and Banzhaf values satisfy the efficiency and 2-efficiency properties, respectively:

- Efficiency: The sum of probabilistic values is the difference between the whole coalition and the empty set i.e., $\sum_{i=1}^{n} \phi_i = v([n]) - v(\emptyset)$. Efficiency is desirable in settings where we want to attribute the value $v([n]) - v(\emptyset)$ to each player e.g., model prediction explanation or cost-sharing.
- 2-Efficiency: Let v' be a game where i and j are combined into a single player (i, j). That is, v' is defined on $n-1$ players where the i and j players in v are always considered together; effectively, v' is only defined on subsets that contain both i and j or contain neither i nor j. The 2-efficiency property requires that $\phi_i(v) + \phi_j(v) = \phi_{(i,j)}(v')$ for all i, j. 2-Efficiency is desirable in settings where players can be combined into subgroups e.g., federated learning with client aggregation or games with alliances between players.

Shapley values are the most popular probabilistic value, in part because they are the only probabilistic value to satisfy the efficiency property [Sha51]. The probabilistic weights for Shapley values are $p_{|S|} = \frac{1}{n}\binom{n-1}{|S|}^{-1}$, which weights all *set sizes* equally.

The efficiency property is useful in settings where we want to allocate the total value of the game to the players. However, efficiency is not always appropriate especially when there are non-linear interactions between players that cannot be attributed to individuals. For these cases, a more appropriate property may be 2-efficiency which requires that probabilistic values add if players are merged. The only probabilistic value that satisfies 2-efficiency is the Banzhaf value [Pen46, BI64]. The probabilistic weight for Banzhaf values is $p_{|S|} = 1/2^{n-1}$, which weights all *sets* equally.

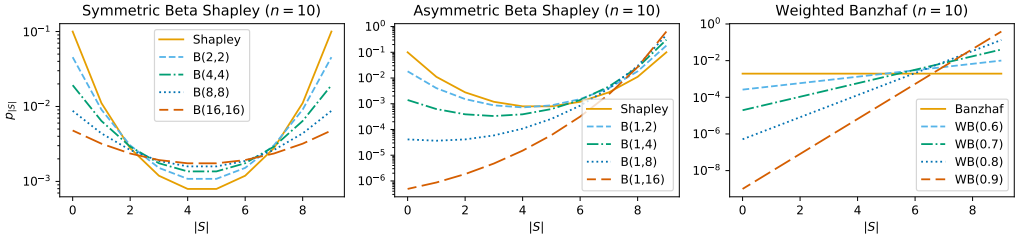

Figure 5: Probabilistic values by subset size for $n = 10$. Beta Shapley values $\text{B}(\alpha, \beta)$ generalize Shapley values for $\alpha, \beta \in [1, \infty)$; increasing both α and β flattens beta Shapley values while increasing just α (or just β) tilts beta Shapley values. Weighted Banzhaf values $\text{WB}(p)$ generalize Banzhaf values for $p \in (0, 1)$; increasing (or decreasing) p tilts weighted Banzhaf values.

Both Shapley and Banzhaf values have been generalized to beta Shapley values [KZ22a] and weighted Banzhaf values [LY24c], respectively. Figure 5 plots $p_{|S|}$ for various beta Shapley and weighted Banzhaf values when $n = 10$. Beta Shapley values are defined by two parameters $\alpha, \beta \in [1, \infty)$. In particular, the probabilistic weight is

$$\frac{\text{Beta}(|S| + \beta, n - |S| - 1 + \alpha)}{\text{Beta}(\alpha, \beta)}.$$

Setting $\alpha = \beta = 1$ recovers Shapley values. Weighted Banzhaf values are defined by one parameter $p \in (0, 1)$. In particular, the probabilistic weight is

$$p^{|S|}(1-p)^{n-|S|-1}.$$

Setting $p = \frac{1}{2}$ recovers Banzhaf values.

C Computing Probabilistic Values of Tree-based Models

In this section, we show how to efficiently compute the probabilistic values of a tree when the value function is interventional feature attribution. Unfortunately, we cannot directly generalize analogous algorithms for Shapley and Banzhaf values [LL17, LEC+20, KMM+22], since they uses a property which does not hold for all probabilistic values. In particular, prior approaches assume that if a value function only has contributions from $n' < n$ players, the Shapley/Banzhaf value on the induced game of those n' players is the same as the Shapley/Banzhaf value on the original value function with all n players. Our approach is based on an alternative way of viewing tree-based models that avoids the need for this property.

Consider the value function $v : 2^{[n]} \to \mathbb{R}$ induced by a tree with explicand $\mathbf{x}^e$ and baseline $\mathbf{x}^b$. We decompose v into a sum of *path value functions* $\{v^P\}_P$, where each $v^P : 2^{[n]} \to S$ corresponds to a distinct root-to-leaf path P. In particular,

$$v(S) \;=\; \sum_P v^P(S), \qquad \text{where} \quad v^P(S) = \begin{cases} \text{leaf value of } P & \text{if } S \text{ follows path } P \text{ on } \mathbf{x}^e, \mathbf{x}^b, \\ 0 & \text{otherwise.} \end{cases}$$

By the linearity property of probabilistic values, the contribution of feature i to the full tree model v can be expressed as the sum of its contributions to each path model v^P. Specifically,

$$\phi_i\big(v\big) \;=\; \phi_i\Big(\sum_P v^P\Big) \;=\; \sum_P \phi_i\big(v^P\big).$$

Therefore, it suffices to compute $\phi_i(v^P)$ for each path model v^P and aggregate their contributions over all paths. To this end, we will introduce the following notation. Given probabilistic weights $\mathbf{p} = [p_0, \ldots, p_{n-1}] \in [0,1]^n$, for each path P, define

- S_P as the set of features in P whose conditions are satisfied by $\mathbf{x}^e$ but not by $\mathbf{x}^b$,
- N_P as the set of features in P whose conditions are satisfied by $\mathbf{x}^b$ but not by $\mathbf{x}^e$,
- ℓ_P as the final leaf value on path P.

Recall we can write the ith probabilistic value as

$$\phi_i(v) = \sum_{S \subseteq [n] : i \in S} p_{|S|-1} v(S) - \sum_{S \subseteq [n] : i \notin S} p_{|S|} v(S).$$

Using the definition of v^P, S_P, and N_P, we can consider $\phi_i(v^P)$ in three cases:

- **Case 1:** $i \in S_P$**:** We need $i \in S$ in order to reach the leaf i.e., $v^P(S) = 0$ unless $S_P \subseteq S \subseteq [n] \setminus N_P$ and $i \in S$. Then,

$$\phi_i(v^P) = \sum_{S_P \subseteq S \subseteq [n] \setminus N_P} p_{|S|-1} \cdot \ell_P = \sum_{l=|S_P|}^{n-|N_P|} p_{l-1} \binom{n - |N_P| - |S_P|}{l - |S_P|} \cdot \ell_P$$

- **Case 2:** $i \in N_P$**:** We need $i \notin S$ in order to reach the leaf i.e., $v^P(S) = 0$ unless $S_P \subseteq S \subseteq [n] \setminus N_P$ and $i \notin S$. Then,

$$\phi_i(v^P) = -\sum_{l=|S_P|}^{n-|N_P|} p_l \binom{n - |N_P| - |S_P|}{l - |S_P|} \cdot \ell_P$$

- **Case 3:** $i \notin N_P$ **and** $i \notin S_P$**:** We reach the leaf whether $i \in S$ or not i.e., $v^P(S) = 0$ unless $S_P \subseteq S \subseteq [n] \setminus N_P$. Then,

$$\begin{aligned} \phi_i(v^P) = &\sum_{l=|S_P|+1}^{n-|N_P|} p_{l-1} \binom{n - |N_P| - |S_P| - 1}{l - |S_P| - 1} \ell_P \\ &- \sum_{l=|S_P|}^{n-|N_P|-1} p_l \binom{n - |N_P| - |S_P| - 1}{l - |S_P|} \ell_P = 0 \end{aligned}$$

C.1 TreeProb Pseudocode

Algorithm 2 efficiently explores all root-to-leaf paths, maintaining counters for how many times each feature has been "seen" under the explicand (ef_seen) or the baseline (bf_seen). When a branching feature is encountered for the first time on a path, the algorithm branches into two recursive calls—one following $\mathbf{x}^e$, the other $\mathbf{x}^b$—and updates the *cardinalities* $s_P := |S_P|$ or $n_P := |N_P|$ accordingly, depending on which feature value is consistent with the split. At each leaf node, the algorithm computes the contribution based on the current (s_P, n_P) and aggregates these into the overall attribution vector ϕ. Algorithm 2 preserves the original complexity and traversal logic of Tree SHAP, while generalizing the feature-contribution calculation to the probabilistic formulation described above, making it applicable to any probabilistic value.

Algorithm 2 TreeProb with Interventional Feature Perturbation

Input: n: number of players; $\mathbf{p} \in [0,1]^n$: probabilistic weights
Output: Exact probabilistic values $\phi_1, \ldots, \phi_n$

function RECURSE(node, s_P, n_P, ef_seen, bf_seen)
if node is a leaf **then**
    pos_term $\leftarrow$ node.value $\cdot \sum_{l=s_P}^{n-n_P} p_{l-1} \binom{n-n_P-s_P}{l-s_P}$
    neg_term $\leftarrow -$ node.value $\cdot \sum_{l=s_P}^{n-n_P} p_{l} \binom{n-n_P-s_P}{l-s_P}$
    return (pos_term, neg_term)
end if
x_e_child $\leftarrow \begin{cases} \text{node.leftchild} & \text{if } x_e[\text{node.feat}] < \text{node.t} \\ \text{node.rightchild} & \text{otherwise} \end{cases}$
x_b_child $\leftarrow \begin{cases} \text{node.leftchild} & \text{if } x_b[\text{node.feat}] < \text{node.t} \\ \text{node.rightchild} & \text{otherwise} \end{cases}$
if ef_seen[node.feat] > 0 **then**
    return RECURSE(x_e_child, s_P, n_P, ef_seen, bf_seen)
end if
if bf_seen[node.feat] > 0 **then**
    return RECURSE(x_b_child, s_P, n_P, ef_seen, bf_seen)
end if
if x_e_child $= x_b$_child **then**
    return RECURSE(x_e_child, s_P, n_P, ef_seen, bf_seen)
else
    ef_seen[node.feat] $\leftarrow$ ef_seen[node.feat] $+ 1$
    (pos$_e$, neg$_e$) $\leftarrow$ RECURSE(x_e_child, $s_P + 1$, n_P, ef_seen, bf_seen)
    ef_seen[node.feat] $\leftarrow$ ef_seen[node.feat] $- 1$
    bf_seen[node.feat] $\leftarrow$ bf_seen[node.feat] $+ 1$
    (pos$_b$, neg$_b$) $\leftarrow$ RECURSE(x_b_child, s_P, $n_P + 1$, ef_seen, bf_seen)
    bf_seen[node.feat] $\leftarrow$ bf_seen[node.feat] $- 1$
    ϕ_{temp}[node.feat] $\leftarrow \phi_{\text{temp}}$[node.feat] $+$ (pos$_e$ $+$ neg$_b$)
    return (pos$_e$ $+$ pos$_b$, neg$_e$ $+$ neg$_b$)
end if
end function

Initialize $\boldsymbol{\phi} \leftarrow \mathbf{0}^n$
for each tree t in the ensemble **do**
    for each baseline $\mathbf{x}^b$ in baselines **do**
        $\phi_{\text{temp}} \leftarrow \mathbf{0}^n$
        RECURSE(t.root, 0, 0, $\mathbf{0}^n$, $\mathbf{0}^n$)
        $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \phi_{\text{temp}}$
    end for
end for
return $\boldsymbol{\phi}$/(number of trees $\times$ number of baselines)

D Baselines

In this section, we describe the baselines we compare against for estimating probabilistic values. These baselines broadly fall into three categories: standard Monte Carlo methods, maximum sample reuse methods, and regression methods.

For a more technical description of many of these baselines, please refer to Appendix D of [LY24b].

Monte Carlo Methods The standard Monte Carlo estimator estimates each probabilistic value individually by sampling each term in the summation with probability proportional to its weight.

Weighted Sampling Lift (WSL) [KZ22a] is the standard Monte Carlo, but where subsets are sampled according to the Shapley weights and reweighted to produce unbiased estimates of the probabilistic values.

Permutation SHAP [CGT09] is similar to Monte Carlo estimates except that subsets are sampled in permutations; that is, the first element in the sampled permutation is one sampled subset, the first two elements are another sampled subset, and so on. Because each set *size* is weighted equally by Shapley values, sampling permutations without any reweighting gives estimates that are the Shapley values in expectation. Permutation SHAP gives close to state-of-the-art performance.

Weighted SHAP estimator [KZ22b] generalizes permutation sampling approach to probabilistic value. Random permutations are drawn as before but now reweighted by the probabilistic value weights so that the final estimates are unbiased.

Maximum Sample Reuse Methods Monte Carlo methods are unbiased but inefficient in the sense that they only use each sample to compute the estimates of one or two probabilistic values. The key observation of Maximum Sample Reuse (MSR) estimators is that each probabilistic value ϕ_i can be written as two summations, one over sets that include i and one over sets that do not. Then the MSR methods use each sample to update every summation. First used for Banzhaf values [WJ23], MSR has since been generalized to other probabilistic values with different sampling distributions.

Approximation without Requesting Marginals (ARM) [KBMH24] is a kind of MSR estimator. Half the samples are drawn with probability $p_{|S|-1}$ while the other half are drawn with probability $p_{|S|}$. In order to avoid the numerical instability of reweighting, the final estimate only includes the first half of samples if $i \in S$ and the second if $i \notin S$.

One sample Fits All (OFA) [LY24b] similarly uses maximum sample reuse but samples according to a more complicated distribution.

Regression Methods A parallel line of work fits linear models f to v then returns the probabilistic values of f. These approaches originate for estimating Shapley values, and are based on a linear regression problem that exactly recovers the Shapley values when solved exactly [CGKR88].

Kernel SHAP [LL17] samples subsets from this regression problem with probability proportional to their weighting in the regression problem.

Leverage SHAP [MW25] similarly samples subsets but with probability proportional to their *statistical leverage* in the regression problem, resulting in state-of-the-art Shapley value estimates and error bounds that depend on the fit of f to v.

Kernel Banzhaf [LWK$^+$25] is similar to Kernel SHAP and Leverage SHAP but estimates Banzhaf values, and is based on a regression formulation specific to Banzhaf values [HH92].

There is a known generalization of the Shapley value regression problem [RVZ98], which, when solved exactly, recovers probabilistic values up to additive constant. Since Shapley values satisfy efficiency, this constant is efficient to exactly compute for Shapley values. However, for general probabilistic values, the constant depends on the entire value function v and must be estimated, introducing another source of error.

The *Generic Estimator based on Least Squares (GELS)* [LY24a] estimates the constant by adding a dummy variable with probabilistic value 0. Instead of fitting a linear function f, GELS considers the closed-form solution to the regression problem and effectively applies a maximum sample reuse estimator to the underlying matrix-vector multiplication. The final estimates are adjusted by subtracting the value of the dummy variable.

The *Average Marginal Effect (AME)* [LZL+22] is another regression estimator that uses a different regression formulation. For probabilistic values that satisfy a specific condition, the probabilistic values can be written as an infinitely tall regression problem. The estimator samples this regression problem and solves the approximate version.

E Experiments on Small Datasets with Neural Network Models

Table 3: Summary statistics of the ℓ_2-norm error between estimated and true probabilistic values when $m = 40n$. We summarize the error over small datasets ($n < 30$), for which the probabilistic values of a neural network model can be feasibly computed. On average over all probabilistic values, Tree MSR produces estimates with mean error that is $150\times$ lower than the best estimator from prior work.

	B(1,1)	B(2,2)	B(4,4)	B(8,8)	B(1,2)	B(1,4)	B(1,8)	WB(0.5)	WB(0.6)	WB(0.7)	WB(0.8)	WB(0.9)	Mean
LinearMSR													
Mean	9.41×10^{-4}	1.42×10^{-3}	1.14×10^{-3}	1.30×10^{-3}	1.41×10^{-2}	2.16×10^{-2}	2.74×10^{-2}	1.29×10^{-3}	6.80×10^{-3}	4.67×10^{-3}	5.61×10^{-3}	1.45×10^{-2}	8.39×10^{-3}
1st Quartile	1.41×10^{-7}	1.81×10^{-7}	1.54×10^{-7}	1.03×10^{-7}	5.85×10^{-5}	1.33×10^{-4}	1.56×10^{-4}	1.42×10^{-7}	2.08×10^{-5}	1.70×10^{-5}	1.68×10^{-5}	4.36×10^{-5}	3.72×10^{-5}
2nd Quartile	7.43×10^{-6}	1.31×10^{-5}	1.16×10^{-5}	7.83×10^{-6}	7.26×10^{-4}	1.25×10^{-3}	7.70×10^{-4}	1.08×10^{-5}	1.58×10^{-4}	2.27×10^{-4}	2.85×10^{-4}	4.15×10^{-4}	3.23×10^{-4}
3rd Quartile	7.27×10^{-4}	2.41×10^{-3}	1.97×10^{-3}	1.11×10^{-3}	1.50×10^{-2}	2.60×10^{-2}	4.47×10^{-2}	1.77×10^{-3}	3.71×10^{-3}	4.29×10^{-3}	5.67×10^{-3}	2.43×10^{-2}	1.10×10^{-2}
TreeMSR													
Mean	3.83×10^{-4}	5.36×10^{-4}	5.96×10^{-4}	6.46×10^{-4}	4.58×10^{-4}	3.59×10^{-4}	2.90×10^{-4}	6.61×10^{-4}	4.51×10^{-4}	6.36×10^{-4}	3.50×10^{-4}	1.26×10^{-4}	4.58×10^{-4}
1st Quartile	7.28×10^{-7}	1.08×10^{-6}	1.90×10^{-6}	1.08×10^{-6}	5.35×10^{-7}	2.60×10^{-7}	1.11×10^{-7}	1.09×10^{-6}	7.71×10^{-7}	6.55×10^{-7}	3.69×10^{-7}	4.36×10^{-8}	7.19×10^{-7}
2nd Quartile	2.63×10^{-5}	2.11×10^{-5}	3.08×10^{-5}	3.28×10^{-5}	1.21×10^{-5}	9.25×10^{-6}	7.16×10^{-6}	3.13×10^{-5}	2.13×10^{-5}	1.53×10^{-5}	1.14×10^{-5}	5.51×10^{-6}	1.87×10^{-5}
3rd Quartile	2.16×10^{-4}	2.45×10^{-4}	2.41×10^{-4}	1.69×10^{-4}	1.81×10^{-4}	1.34×10^{-4}	1.57×10^{-4}	1.78×10^{-4}	1.67×10^{-4}	2.44×10^{-4}	1.08×10^{-4}	6.15×10^{-5}	1.75×10^{-4}
OFA													
Mean	3.17×10^{-2}	3.72×10^{-2}	4.79×10^{-2}	4.30×10^{-2}	3.04×10^{-2}	2.08×10^{-2}	1.48×10^{-2}	4.48×10^{-2}	2.18×10^{-1}	1.96×10^{-1}	4.94×10^{-2}	1.55×10^{-1}	7.42×10^{-2}
1st Quartile	2.29×10^{-2}	2.95×10^{-2}	3.27×10^{-2}	3.12×10^{-2}	2.35×10^{-2}	1.47×10^{-2}	1.04×10^{-2}	2.82×10^{-2}	4.67×10^{-2}	3.56×10^{-2}	2.80×10^{-2}	1.51×10^{-2}	2.65×10^{-2}
2nd Quartile	3.04×10^{-2}	3.60×10^{-2}	4.95×10^{-2}	4.09×10^{-2}	2.97×10^{-2}	2.07×10^{-2}	1.27×10^{-2}	4.00×10^{-2}	6.16×10^{-2}	4.90×10^{-2}	3.70×10^{-2}	2.43×10^{-2}	3.60×10^{-2}
3rd Quartile	3.82×10^{-2}	4.33×10^{-2}	5.92×10^{-2}	5.27×10^{-2}	3.70×10^{-2}	2.55×10^{-2}	1.72×10^{-2}	5.59×10^{-2}	7.55×10^{-2}	6.84×10^{-2}	5.05×10^{-2}	6.45×10^{-2}	4.90×10^{-2}
WSL													
Mean	1.07×10^{-2}	3.67×10^{-2}	4.26×10^{-2}	3.34×10^{-1}	5.00×10^{-2}	7.18×10^{-2}	1.31×10^{-1}	5.73×10^{-1}	1.24×10^{-1}	8.35×10^{-2}	1.16×10^{-1}	2.01×10^{-1}	1.48×10^{-1}
1st Quartile	6.31×10^{-5}	2.76×10^{-3}	6.75×10^{-3}	1.76×10^{-2}	8.82×10^{-3}	1.49×10^{-2}	3.40×10^{-2}	2.60×10^{-2}	9.91×10^{-3}	7.67×10^{-3}	2.85×10^{-2}	5.32×10^{-2}	1.75×10^{-2}
2nd Quartile	3.76×10^{-4}	1.13×10^{-2}	2.30×10^{-2}	4.77×10^{-2}	2.38×10^{-2}	3.86×10^{-2}	8.61×10^{-2}	7.16×10^{-2}	5.91×10^{-2}	5.65×10^{-2}	7.62×10^{-2}	1.08×10^{-1}	5.02×10^{-2}
3rd Quartile	1.35×10^{-2}	2.42×10^{-2}	5.95×10^{-2}	9.04×10^{-2}	4.18×10^{-2}	8.59×10^{-2}	1.88×10^{-1}	1.27×10^{-1}	1.29×10^{-1}	1.06×10^{-1}	1.57×10^{-1}	2.30×10^{-1}	1.04×10^{-1}
GELS													
Mean	1.82×10^{-1}	1.16×10^{-1}	1.14×10^{-1}	1.15×10^{-1}	8.83×10^{-2}	8.02×10^{-2}	6.83×10^{-2}	1.20×10^{-1}	7.42×10^{-2}	5.65×10^{-2}	4.82×10^{-2}	5.59×10^{-2}	9.33×10^{-2}
1st Quartile	8.97×10^{-2}	5.91×10^{-2}	6.20×10^{-2}	5.60×10^{-2}	4.79×10^{-2}	3.41×10^{-2}	3.54×10^{-2}	5.94×10^{-2}	3.33×10^{-2}	3.03×10^{-2}	2.88×10^{-2}	3.07×10^{-2}	4.72×10^{-2}
2nd Quartile	1.42×10^{-1}	9.13×10^{-2}	8.73×10^{-2}	8.24×10^{-2}	7.61×10^{-2}	5.63×10^{-2}	5.23×10^{-2}	9.24×10^{-2}	5.07×10^{-2}	4.33×10^{-2}	3.85×10^{-2}	4.27×10^{-2}	7.13×10^{-2}
3rd Quartile	2.26×10^{-1}	1.26×10^{-1}	1.35×10^{-1}	1.45×10^{-1}	1.11×10^{-1}	9.56×10^{-2}	7.34×10^{-2}	1.63×10^{-1}	8.37×10^{-2}	6.92×10^{-2}	6.24×10^{-2}	7.33×10^{-2}	1.14×10^{-1}
ARM													
Mean	2.06×10^{-1}	1.04×10^{-1}	1.68×10^{-1}	1.27×10^{-1}	7.41×10^{-2}	5.13×10^{-2}	4.91×10^{-2}	9.18×10^{-2}	6.13×10^{-2}	4.86×10^{-2}	5.65×10^{-2}	4.04×10^{-2}	8.98×10^{-2}
1st Quartile	4.13×10^{-2}	3.69×10^{-2}	3.32×10^{-2}	2.80×10^{-2}	3.72×10^{-2}	3.36×10^{-2}	2.77×10^{-2}	2.83×10^{-2}	2.80×10^{-2}	3.30×10^{-2}	3.54×10^{-2}	2.20×10^{-2}	3.20×10^{-2}
2nd Quartile	6.79×10^{-2}	6.28×10^{-2}	4.76×10^{-2}	4.57×10^{-2}	5.77×10^{-2}	4.82×10^{-2}	4.28×10^{-2}	3.69×10^{-2}	4.33×10^{-2}	4.01×10^{-2}	4.64×10^{-2}	3.02×10^{-2}	4.75×10^{-2}
3rd Quartile	9.67×10^{-2}	9.79×10^{-2}	8.27×10^{-2}	6.68×10^{-2}	9.99×10^{-2}	5.86×10^{-2}	6.62×10^{-2}	5.70×10^{-2}	7.98×10^{-2}	6.26×10^{-2}	6.33×10^{-2}	4.62×10^{-2}	7.31×10^{-2}
WeightedSHAP													
Mean	1.24×10^{-1}	1.04×10^{-2}	5.98×10^{-2}	1.33×10^{-1}	1.58×10^{-2}	3.36×10^{-2}	7.52×10^{-2}	1.51×10^{-1}	5.43×10^{-2}	4.40×10^{-2}	6.43×10^{-2}	6.64×10^{-2}	6.93×10^{-2}
1st Quartile	1.82×10^{-5}	1.47×10^{-3}	2.45×10^{-3}	4.07×10^{-3}	1.01×10^{-3}	4.46×10^{-3}	1.22×10^{-2}	1.23×10^{-2}	6.90×10^{-3}	3.82×10^{-3}	1.38×10^{-2}	6.67×10^{-3}	5.76×10^{-3}
2nd Quartile	2.94×10^{-4}	3.48×10^{-3}	1.26×10^{-2}	1.20×10^{-2}	4.98×10^{-3}	1.85×10^{-2}	5.33×10^{-2}	2.96×10^{-2}	1.90×10^{-2}	3.07×10^{-2}	3.59×10^{-2}	2.85×10^{-2}	2.07×10^{-2}
3rd Quartile	4.58×10^{-3}	1.42×10^{-2}	3.72×10^{-2}	3.56×10^{-2}	1.74×10^{-2}	4.74×10^{-2}	1.10×10^{-1}	5.53×10^{-2}	7.07×10^{-2}	6.71×10^{-2}	8.94×10^{-2}	9.13×10^{-2}	5.33×10^{-2}

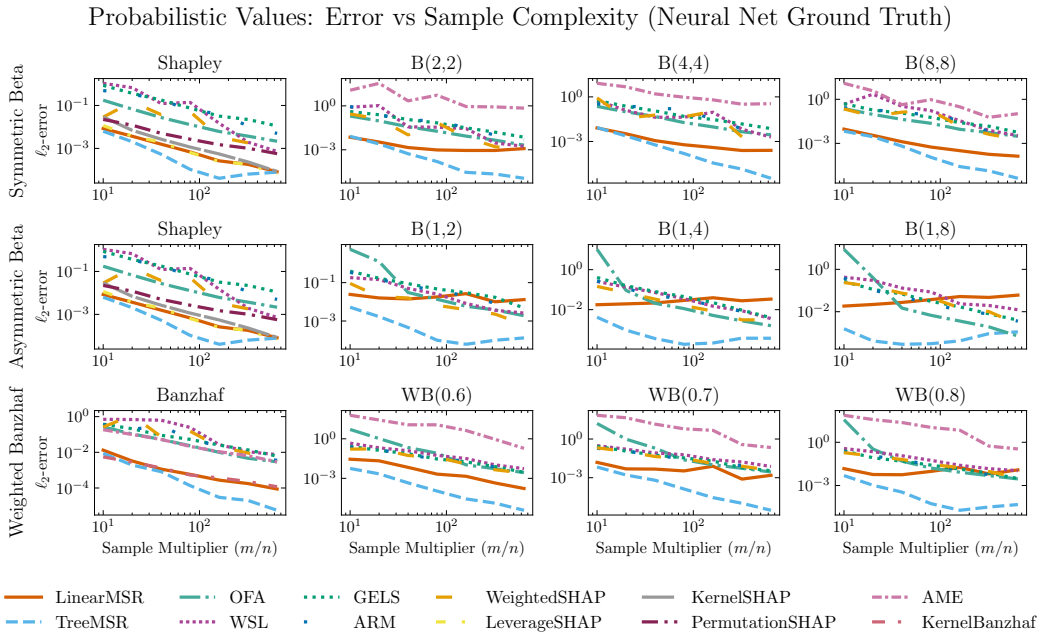

Figure 6: Error between the estimated and true probabilistic values by complexity. Each subplot shows results for a different probabilistic value with the error averaged over all large datasets ($n \geq 30$). The lines report the mean error over 10 runs. Tree and Linear MSR give the best performance, often by several orders of magnitude especially when the number of samples is large.

F Experiments by Noise

In many settings, access to the value function is noisy. For example, v may be the expectation over a distribution that is expensive to exactly compute. Instead, we may estimate the expectation, and hence the values we observe are noisy. In this experiment, we add normally distributed noise to the values passed into each estimator. The plots show the performance of each estimator by the magnitude of this noise.

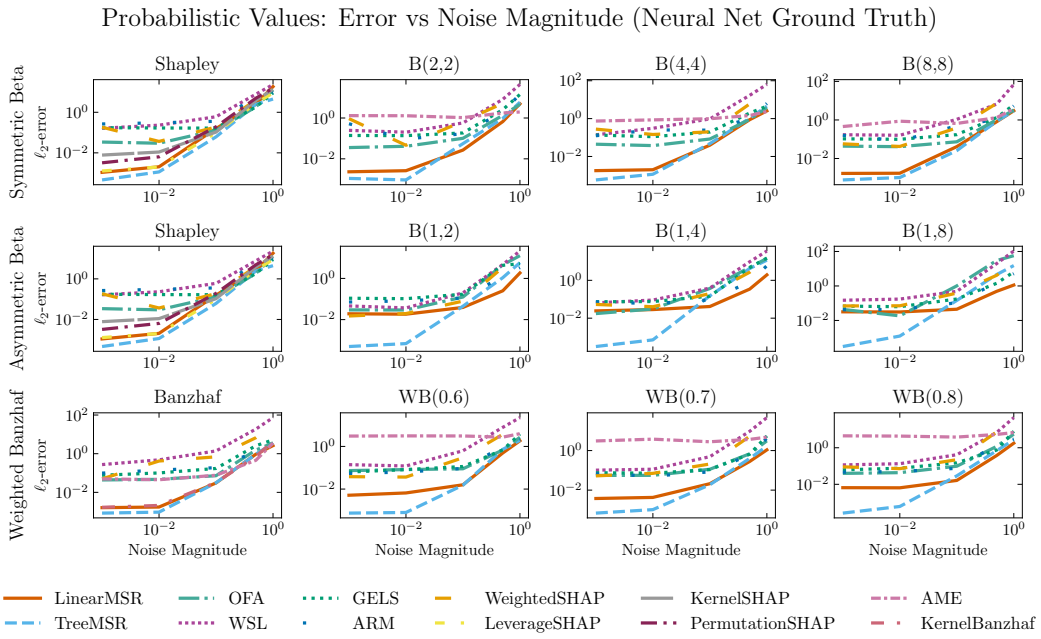

Figure 7: Error between the estimated and true probabilistic values as a function of noise magnitude. Each subplot shows results for a different probabilistic value with the error averaged over all small datasets ($n < 30$). The lines report the mean error over 10 runs. Tree MSR gives the best performance, often by several orders of magnitude especially when the magnitude of the noise is small.

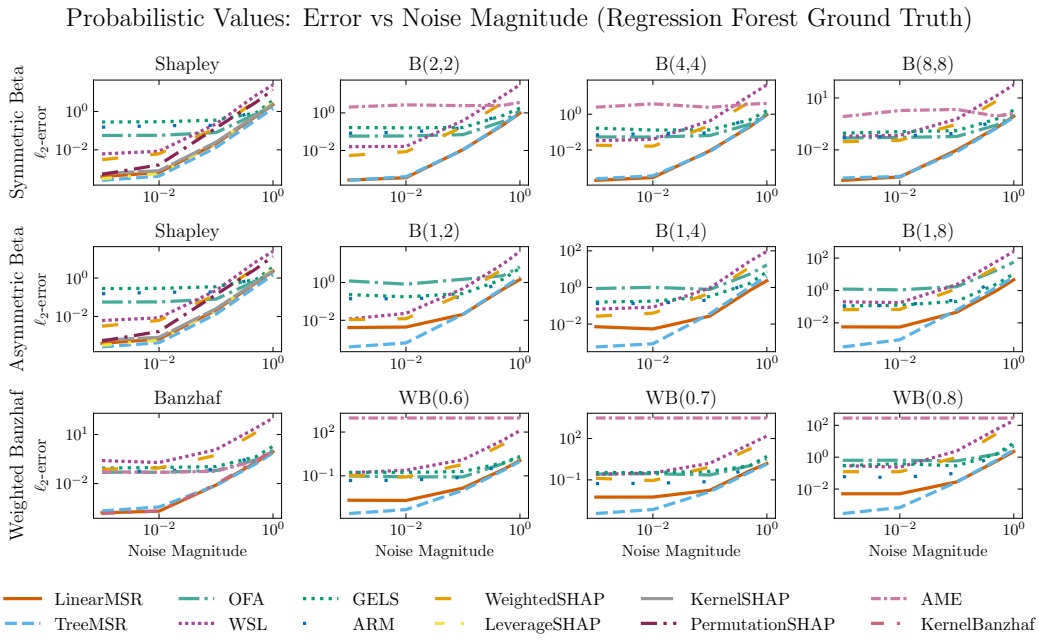

Figure 8: Error between the estimated and true probabilistic values as a function of noise magnitude. Each subplot shows results for a different probabilistic value with the error averaged over all large datasets ($n \geq 30$). The lines report the mean error over 10 runs. Tree MSR gives the best performance, often by several orders of magnitude especially when the magnitude of the noise is small.

G Dataset Descriptions

Table 4: A summary of the datasets used in our experiments, including source, access method, license, and number of features n.

Dataset	n	Source / Citation	Access Method	License
Adult	12	[Koh96]	`shap.datasets`	CC-BY 4.0
Forest Fires	13	[CM07]	UCI ML Repo[5]	CC-BY 4.0
Real Estate	15	[YH09]	UCI ML Repo[6]	CC-BY 4.0
Bike Sharing	16	[FTG14]	OpenML[7]	Public Domain
Breast Cancer	30	[SWM93]	`sklearn.datasets`	CC-BY 4.0
Independent	60	[LL17]	`shap.datasets`	MIT
NHANES	79	[Cen23]	`shap.datasets`	Public Domain
Communities	101	[RC02]	`shap.datasets`	CC-BY 4.0

[5] `https://archive.ics.uci.edu/ml/datasets/forest+fires`

[6] `https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set`

[7] `https://www.openml.org/d/42712`