Revisiting RegressionMSR: A Reproducibility Study

Anonymous authors
Paper under double-blind review

Abstract

Regression-adjusted Monte Carlo estimators for Shapley values and probabilistic values combine surrogate modeling with maximum-sample-reuse estimation to reduce the cost of feature-attribution computation. We present a reproducibility study of RegressionMSR in the interventional feature-attribution setting, focusing on artifact fidelity, practical algorithmic choices, and empirical robustness. We first identify implementation-level differences between the estimator displayed in the paper and the released artifact: the code computes separated conditional means over coalitions containing and excluding each feature, rather than the displayed all-row signed residual average, and in the large-n with-replacement sampler, it stores a size-level denominator rather than the paper's full coalition-level sampling probability. We characterize these conventions algebraically and verify them with synthetic diagnostics. Holding the released denominator fixed, the all-row versus separated-means discrepancy is measurable but has limited effect on final attribution error in our tested settings. We then test the authors' practical single-surrogate simplification, finding that it substantially reduces runtime while maintaining and often slightly improving accuracy. Next, we qualitatively reproduce the headline Shapley benchmark: TreeMSR remains the lowest-error estimator on average, although rank-stability metrics give a complementary ranking. Finally, we extend the original Gaussian-noise experiment to Laplacian and coalition-size-dependent heteroskedastic noise, finding that TreeMSR's low-noise advantage persists across other low-noise regimes. Overall, our results support RegressionMSR's empirical value while clarifying implementation details needed for reproducible use.

1 Introduction

Feature-attribution methods based on cooperative-game values are widely used to explain black-box model predictions. Given a model and an input of interest, these methods define a coalition-value function over subsets of features and assign each feature an attribution based on its average contribution across coalitions. Shapley values are the most prominent example, but they belong to a broader family of probabilistic values that includes Banzhaf values, Beta-Shapley values, weighted Banzhaf values, and other semivalues (Shapley, 1953; Weber, 1988; Kwon & Zou, 2022; Wang & Jia, 2023). These quantities are useful in explainable AI and data valuation, but exact computation requires evaluating exponentially many coalitions (Štrumbelj & Kononenko, 2014; Lundberg & Lee, 2017).

A large literature therefore studies Monte Carlo and regression-based estimators for Shapley and related values. Classical Monte Carlo methods estimate marginal contributions from sampled coalitions or permutations (Castro et al., 2009; Štrumbelj & Kononenko, 2014), while maximum-sample-reuse (MSR) estimators rewrite probabilistic values as linear functionals of coalition values so that each sampled coalition can update every player's estimate (Wang & Jia, 2023; Li & Yu, 2024; Kolpaczki et al., 2024). Regression-based methods, such as KernelSHAP and LeverageSHAP, instead exploit weighted least-squares characterizations of Shapley values (Charnes et al., 1988; Lundberg & Lee, 2017; Musco & Witter, 2025).

Regression-adjusted Monte Carlo estimators for Shapley values and probabilistic values, which we refer to as RegressionMSR, combine these two perspectives (Witter et al., 2025). The key identity is that probabilistic

values are linear in the coalition-value function:

$$\phi(v) = \phi(f) + \phi(v - f),$$

where f is a learned surrogate for v. RegressionMSR estimates $\phi(f)$ using a surrogate class whose probabilistic values can be computed efficiently, and then applies MSR only to the residual game $v - f$. When f approximates v well, the residual correction can have much lower variance than applying MSR directly to v. The original paper reports large empirical improvements over PermutationSHAP, KernelSHAP, and LeverageSHAP for Shapley-value estimation, and further claims strong performance for broader probabilistic values (Witter et al., 2025).

This paper presents a reproducibility study of RegressionMSR. Rather than only rerunning benchmark tables, we study the estimator at three levels: the released artifact, the practical algorithmic choices used in experiments, and the empirical claims reported in the original paper. First, we audit the implementation and identify a residual-correction convention in the released code that differs from the all-row correction displayed in the paper. Second, we test the practical single-surrogate simplification against cross-fitted variants, measuring error, empirical bias, variance, and runtime. Third, we reproduce the headline Shapley-value benchmark and add rank-stability metrics. Finally, we extend the original Gaussian-noise stress test to Laplacian and heteroskedastic noise. Overall, we find that the main qualitative conclusions of RegressionMSR are robust, while exact numerical reproduction depends on implementation and stochastic-pipeline details.

2 Background and Related Work

2.1 Probabilistic values for attribution

Let $N = [n]$ denote a set of players and let $v : 2^N \to \mathbb{R}$ be a cooperative-game value function. In feature attribution, the players are input features and $v(S)$ is the model output when features in S are observed while the remaining features are masked, marginalized, or set to a baseline. The attribution $\phi_i(v)$ then measures the contribution of feature i to a particular prediction.

A probabilistic value assigns each player i a weighted average of marginal contributions:

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

where the weights $p_0, \ldots, p_{n-1}$ are nonnegative and normalized so that

$$\sum_{s=0}^{n-1} \binom{n-1}{s} p_s = 1.$$

Shapley weights $p_s = 1/(n\binom{n-1}{s})$ recover the Shapley value (Shapley, 1953). Uniform weights $p_s = 2^{-(n-1)}$ recover the Banzhaf value (Banzhaf, 1965). Other choices yield Beta-Shapley values, weighted Banzhaf values, and broader semivalue families (Kwon & Zou, 2022; Wang & Jia, 2023; Weber, 1988; Li & Yu, 2024). These values differ in how they weight small, large, or intermediate coalitions, but they share desirable structural properties such as linearity in v, symmetry, and the null-player property.

2.2 Monte Carlo, sample reuse, and regression estimators

Evaluating equation 1 exactly requires summing over $O(2^n)$ coalitions for each player. Since $v(S)$ may require a model forward pass, a marginalization procedure, or even model retraining in data valuation, exact computation is often infeasible. Classical Monte Carlo estimators draw coalitions or player orderings and average the resulting marginal contributions with importance weights. PermutationSHAP is a standard example, estimating Shapley values by sampling random player orderings and reading off marginal contributions along the induced prefixes (Castro et al., 2009; Štrumbelj & Kononenko, 2014).

Maximum-sample-reuse estimators exploit the fact that probabilistic values are linear functionals of coalition values. Define

$$\rho_i(S) = p_{|S|-1} \mathbf{1}[i \in S] - p_{|S|} \mathbf{1}[i \notin S],$$

where out-of-range terms such as p_{-1} and p_n are treated as zero. Then Equation equation 1 can be rewritten as

$$\phi_i(v) = \sum_{S \subseteq N} \rho_i(S) v(S). \tag{2}$$

Thus, a sampled coalition value $v(S)$ can update every player's estimate simultaneously. If coalitions are sampled from a distribution D over 2^N with sample size m, the corresponding MSR estimator is

$$\widehat{\phi}_i^{\mathrm{MSR}}(v) = \frac{1}{m} \sum_{t=1}^{m} v(S_t) \frac{\rho_i(S_t)}{D(S_t)}. \tag{3}$$

This sample reuse reduces value-function queries, but the estimator can have high variance because its fluctuations are driven by the magnitude of $v(S)$, rather than by smaller marginal differences.

A parallel line of work estimates Shapley values through regression. The Shapley value admits a weighted least-squares characterization over coalition indicators (Charnes et al., 1988). KernelSHAP uses this characterization by fitting a linear model to sampled pairs $(S_t, v(S_t))$ with Shapley-kernel weights (Lundberg & Lee, 2017). Later work studied the statistical behavior of this estimator and introduced practical improvements and uncertainty refinements (Covert & Lee, 2021). LeverageSHAP refines the sampling law using leverage-score ideas and provides non-asymptotic guarantees for Shapley estimation (Musco & Witter, 2025).

2.3 RegressionMSR

RegressionMSR combines regression adjustment with maximum sample reuse (Witter et al., 2025). The method uses the linearity of probabilistic values: for any surrogate game $f : 2^N \to \mathbb{R}$,

$$\phi_i(v) = \phi_i(f) + \phi_i(v - f). \tag{4}$$

The estimator fits f on sampled coalition values, computes $\phi_i(f)$ using a surrogate class with tractable probabilistic values, and estimates only the residual term by MSR:

$$\widehat{\phi}_i^{\mathrm{RegMSR}}(v) = \phi_i(f) + \frac{1}{m} \sum_{t=1}^{m} \big[v(S_t) - f(S_t) \big] \frac{\rho_i(S_t)}{D(S_t)}. \tag{5}$$

When f approximates v well, the residual $v(S) - f(S)$ is small, so the Monte Carlo correction has lower variance than direct MSR. The original paper instantiates this idea with linear surrogates, called LinearMSR, and tree-based surrogates, called TreeMSR. To preserve the unbiasedness argument for the residual correction, the theoretical algorithm uses cross-fitting: each surrogate is trained on one subset of sampled coalitions and evaluated on a held-out subset.

The original paper also discusses a practical simplification in which a single surrogate is trained on all sampled coalitions and used for the residual correction on the same sample. This breaks the theoretical independence condition and introduces bias, but the authors state that it is faster and generally more accurate in practice. One of our main empirical contributions is to test this practical recommendation directly.

2.4 Developments since RegressionMSR

Several recent papers build on or benchmark against RegressionMSR. PolySHAP extends KernelSHAP by fitting higher-degree polynomial approximations to the coalition-value function, thereby capturing interaction structure that a linear surrogate cannot represent (Fumagalli et al., 2026b). In addition to reporting attribution-error metrics, PolySHAP evaluates rank-based agreement metrics such as Spearman correlation and Precision@k, reflecting the fact that many practical uses of feature attribution depend on the induced feature ranking rather than only on squared ℓ_2 error. We adopt similar rank-stability metrics in our reproduction study, inspired by PolySHAP.

OddSHAP provides another recent perspective on Shapley estimation. It shows that Shapley values depend on the odd component of a set function and uses this observation to motivate an estimator based on

Table 1: Scope of reproducibility. Original statements from Witter et al. (2025)

Claim or issue	Original statement	Our evaluation
Cross-fitting and practical simplification	Increasing k improves accuracy but raises computational cost; the single-surrogate simplification introduces a small bias but is faster and generally more accurate.	Experiment 1: Compare cross-fitting and a single-fit simplification using error, empirical bias, variance, and runtime.
Shapley benchmark performance	RegressionMSR reports lower Shapley-estimation error than PermutationSHAP, KernelSHAP, and LeverageSHAP, including 6.5×, 3.8×, and 2.6× headline improvements.	Experiment 2: Reproduce Table 1, compare against released artifacts, and evaluate rank stability.
Noise robustness	TreeMSR gives the best performance in noisy value-function experiments, especially when Gaussian noise is small.	Experiment 3: Rerun the Gaussian-noise experiment and extend to heteroskedastic and Laplacian noise.
Implementation audit	Algorithm 1 displays an all-row residual correction using coalition-level sampling probabilities $D(S)$.	Appendix A.1–A.2: Show that the released code uses separated conditional means and a size-level denominator convention in the large-n with-replacement sampler; characterize both discrepancies and empirically diagnose the averaging difference.

polynomial regression in the odd subspace (Fumagalli et al., 2026a). OddSHAP also treats paired sampling as a central design feature, which is directly relevant to our implementation audit because the released RegressionMSR code uses complement-pair sampling for symmetric probabilistic values.

EASE provides a broader statistical view of probabilistic-value estimation (Liu et al., 2026). It shows that weighted-average estimators, self-normalized estimators, regression-adjusted estimators, and weighted least-squares estimators share a common augmented inverse-probability-weighted leading term. Under this view, the first-order mean-squared error depends jointly on the sampling law and the surrogate. EASE therefore proposes to choose both the surrogate and sampling distribution to minimize a first-order MSE criterion. This is conceptually complementary to our study: we focus on reproducing and auditing RegressionMSR itself, rather than implementing a new efficiency-aware sampling design.

To our knowledge, no prior work has provided a dedicated reproducibility study of RegressionMSR, including an artifact-level analysis of the released implementation. Subsequent papers such as PolySHAP and OddSHAP use RegressionMSR as an important benchmark, but their primary goal is to introduce new estimators rather than to reproduce the original RegressionMSR artifact. The implementation for RegressionMSR used in the PolySHAP codebase, for instance, exclusively uses a practical, single-sample style estimator (Fumagalli et al., 2026b). Our study is complementary: we reproduce the original RegressionMSR claims, audit the released code, and evaluate implementation choices that affect reproducibility.

3 Scope of Reproducibility

We evaluate three claims from the original paper and one implementation issue uncovered during reproduction. Table 1 summarizes the scope.

We also release a seeded reproduction repository implementing the released estimator, an All-Row diagnostic variant holding the released denominator convention fixed, and the single-fit practical simplification.

4 Experimental Protocol

Our experiments follow the feature-attribution protocol of Witter et al. (2025). We reproduce the original setting closely where possible, building directly on top of the original codebase, but make the pipeline seeded and control for appending so that repeated runs produce stable output files. Just as in the original paper, our experiments are run on a machine with an Apple M2 chip and 8GB RAM. The empirical experiments 1-3 evaluate the original released RegressionMSR artifact. We summarize the shared protocol here.

Value function. For each dataset (see A.5), we train a predictive model and explain the model's prediction at an explicand point $x^e \in \mathbb{R}^n$ relative to a baseline point $x^b \in \mathbb{R}^n$. The baseline point is the feature-wise empirical mean of the processed dataset. For a coalition $S \subseteq [n]$, define x^S by

$$x_i^S = \begin{cases} x_i^e, & i \in S, \\ x_i^b, & i \notin S. \end{cases}$$

The value function is the interventional feature-attribution value

$$v(S) = f_{\text{model}}(x^S).$$

Thus, each estimator is given black-box access to coalition values $v(S)$ and must estimate the corresponding probabilistic-value vector $\phi(v)$. Throughout the paper, n denotes the number of features and m denotes the requested number of coalition-value queries passed to an estimator.

Ground truth. Following the original paper, we divide datasets by feature dimension. For small datasets with $n < 30$, exact probabilistic values are computed by enumerating all coalitions. For larger datasets with $n \geq 30$, exact enumeration is infeasible, so we use the tree-based exact computation procedure from the original codebase when the ground-truth predictive model is a random forest.

Metrics and uncertainty. The primary error metric is the normalized squared ℓ_2 error

$$\frac{\|\widehat{\phi} - \phi\|_2^2}{\|\phi\|_2^2}.$$

For Experiment 1, we additionally decompose mean error into empirical squared bias and empirical variance over sample seeds:

$$\frac{\|\mathbb{E}[\widehat{\phi}] - \phi\|_2^2}{\|\phi\|_2^2}, \qquad \frac{\mathbb{E}\|\widehat{\phi} - \mathbb{E}[\widehat{\phi}]\|_2^2}{\|\phi\|_2^2}.$$

For Experiments 2 and 3, we also report rank-based metrics, including Spearman correlation, Kendall's τ, Precision@3, and Precision@5. Confidence intervals are computed by nonparametric percentile bootstrap over the relevant aggregation unit, which is specified in each experiment (Efron & Tibshirani, 1993).

5 Artifact and Implementation Audit

Before comparing RegressionMSR variants, we audit the released implementation used in the original experiments. The released cross-fitted estimator is not algebraically identical to the residual correction displayed in Algorithm 1 of the original paper, so we treat it as a distinct estimator variant.

Algorithm 1 presents the residual correction as a single signed average over the held-out fold. For held-out fold $\mathcal{H}_\ell$, surrogate $f^{(\ell)}$ trained out-of-fold, and stored denominator $d(S)$, this all-row correction can be written as

$$\widehat{R}_{i,\ell}^{\text{all}} = \frac{1}{|\mathcal{H}_\ell|} \sum_{S \in \mathcal{H}_\ell} \left[v(S) - f^{(\ell)}(S) \right] \frac{p_{|S|-1}\mathbf{1}[i \in S] - p_{|S|}\mathbf{1}[i \notin S]}{d(S)}. \tag{6}$$

Here $d(S)$ denotes the denominator stored by the implementation. In the displayed estimator, this denominator corresponds to the sampling density $D(S)$; below we hold $d(S)$ fixed to isolate the averaging convention.

The released implementation instead computes two conditional means:

$$\widehat{R}^{\text{sep}}_{i,\ell} = \frac{1}{|\mathcal{H}^+_{\ell,i}|} \sum_{S \in \mathcal{H}^+_{\ell,i}} \left[v(S) - f^{(\ell)}(S) \right] \frac{p_{|S|-1}}{d(S)} - \frac{1}{|\mathcal{H}^-_{\ell,i}|} \sum_{S \in \mathcal{H}^-_{\ell,i}} \left[v(S) - f^{(\ell)}(S) \right] \frac{p_{|S|}}{d(S)}. \tag{7}$$

Thus, the released code averages the positive and negative sides separately, rather than averaging one signed term over the entire held-out fold.

Let $q_{\ell,i} = |\mathcal{H}^+_{\ell,i}|/|\mathcal{H}_\ell|$, and let $A_{\ell,i}$ and $B_{\ell,i}$ denote the positive and negative all-row components of equation 6. Then

$$\widehat{R}^{\text{all}}_{i,\ell} = A_{\ell,i} - B_{\ell,i}, \qquad \widehat{R}^{\text{sep}}_{i,\ell} = \frac{1}{q_{\ell,i}} A_{\ell,i} - \frac{1}{1 - q_{\ell,i}} B_{\ell,i}.$$

For symmetric values such as Shapley values, the released implementation uses complement-pair sampling and preserves complement pairs within folds. Therefore $q_{\ell,i} = 1/2$ for every feature and fold, yielding

$$\widehat{R}^{\text{sep}}_{i,\ell} = 2\widehat{R}^{\text{all}}_{i,\ell}$$

when both corrections are computed with the same stored denominator. Full derivations, denominator interpretation details, and edge cases are provided in Appendix A.1.

We verified this relationship with an end-to-end synthetic diagnostic using the released `MSR` estimator. Because `MSR` uses a null surrogate, its output isolates the residual-correction term. We injected a deterministic synthetic coalition-value model $v(S)$, ran the released sampling and explanation pipeline, reconstructed the sampled coalition matrix and stored denominators, and compared the actual estimator output $\widehat{R}^{\text{actual}}$, the separated-mean correction $\widehat{R}^{\text{sep}}$, and the all-row correction $\widehat{R}^{\text{all}}$.

Table 2: Diagnostic of the released residual correction. Values are maximum absolute differences across features.

Setting	Pair	$\max_i \lvert\widehat{R}^{\text{actual}}_i - \widehat{R}^{\text{sep}}_i\rvert$	$\max_i \lvert\widehat{R}^{\text{sep}}_i - 2\widehat{R}^{\text{all}}_i\rvert$	$\max_i \lvert\widehat{R}^{\text{actual}}_i\rvert$
Shapley, $n = 8$, without repl.	Yes	2.08×10^{-17}	2.43×10^{-17}	3.32×10^{-2}
Shapley, $n = 60$, with repl.	Yes	9.21×10^{-15}	8.32×10^{-15}	5.79
Non-symmetric Beta-Shapley	No	1.39×10^{-17}	N/A	1.81×10^{-2}

Table 2 confirms that the released estimator output matches the separated-mean correction to numerical precision. In the paired Shapley settings, the released correction is also exactly twice the all-row correction computed with the same stored denominator. In the non-symmetric Beta-Shapley setting, pair sampling is disabled and the constant factor-of-two relationship no longer applies; instead, the released correction matches the general $q_{\ell,i}$-dependent formula above.

We also evaluate this discrepancy more generally in Appendix A.2 on actual datasets to gauge its practical implications. However, the final estimator errors remain similar, and the released correction is slightly more accurate on average in our runs.

The denominator convention differs as well. In the paper, $D(S)$ is the probability of sampling the specific coalition S. For $n < 50$, the released without-replacement sampler stores a size-specific inclusion probability that closely matches the natural design denominator. For $n \geq 50$, however, the with-replacement sampler samples a size s and then a coalition uniformly within that size, so the full coalition probability includes $\binom{n}{|S|}^{-1}$; `actual_sample_prob` stores only the size-level denominator. This difference is negligible for small n but notable for large n (Appendix A.1).

6 Experiment 1: Cross-Fitting and Practical Simplification

The theoretical RegressionMSR estimator uses cross-fitting to preserve the independence condition needed for the residual-correction argument: each surrogate is trained on one subset of sampled coalitions and

evaluated on a held-out subset. The original paper also describes a practical simplification that trains a single surrogate on all sampled coalitions and reuses the same samples for the residual correction. This single-fit variant breaks the theoretical guarantee, but is claimed to run faster and generally improve empirical accuracy.

We test this claim under the released estimators. This isolates the practical-simplification question from the distinctions studied in Section 5. We compare cross-fitted variants CF-k for $k \in \{2, 5, 10, 20\}$ against the single-fit variant SF. All methods use Shapley values and sample budget $m = 40n$. Variants are compared on matched sampled coalitions, so differences between variants are not due to drawing different coalitions, and we can instead isolate the effect of how they split them. We evaluate both LinearMSR and TreeMSR.

Table 3: Cross-fitting versus single-fit under the released Separated-Means correction. Results are averaged over four datasets, 10 explicands per dataset, and five sample seeds per explicand, each with requested budget $m = 40n$. Runtime is mean estimator runtime in seconds. Brackets denote 95% bootstrap confidence intervals. Gold, silver, and bronze mark the best, second-best, and third-best point estimates within each surrogate group for Error and Runtime. The medal shading is descriptive only; confidence intervals often overlap.

Variant	Surrogate	Error	Bias^2	Variance	Runtime	Err./SF
CF-2	Linear	8.85×10^{-4} [4.88×10^{-4}, 1.37×10^{-3}]	1.68×10^{-4} [9.17×10^{-5}, 2.64×10^{-4}]	7.18×10^{-4} [3.94×10^{-4}, 1.11×10^{-3}]	0.087 [0.067, 0.109]	1.13 [1.10, 1.16]
	Tree	8.26×10^{-4} [5.70×10^{-4}, 1.13×10^{-3}]	6.94×10^{-4} [4.82×10^{-4}, 9.48×10^{-4}]	1.33×10^{-4} [8.29×10^{-5}, 1.88×10^{-4}]	0.158 [0.134, 0.184]	4.31 [3.84, 4.93]
	Overall	8.56×10^{-4} [6.07×10^{-4}, 1.13×10^{-3}]	4.31×10^{-4} [3.11×10^{-4}, 5.59×10^{-4}]	4.25×10^{-4} [2.48×10^{-4}, 6.34×10^{-4}]	0.122 [0.105, 0.142]	1.76 [1.49, 2.22]
CF-5	Linear	7.88×10^{-4} [4.32×10^{-4}, 1.18×10^{-3}]	1.53×10^{-4} [8.12×10^{-5}, 2.35×10^{-4}]	6.35×10^{-4} [3.44×10^{-4}, 9.65×10^{-4}]	0.256 [0.184, 0.334]	1.01 [1.00, 1.01]
	Tree	2.79×10^{-4} [1.78×10^{-4}, 3.87×10^{-4}]	2.23×10^{-4} [1.44×10^{-4}, 3.15×10^{-4}]	5.54×10^{-5} [3.11×10^{-5}, 8.23×10^{-5}]	0.355 [0.296, 0.416]	1.46 [1.39, 1.53]
	Overall	5.34×10^{-4} [3.39×10^{-4}, 7.49×10^{-4}]	1.88×10^{-4} [1.31×10^{-4}, 2.46×10^{-4}]	3.45×10^{-4} [1.95×10^{-4}, 5.24×10^{-4}]	0.305 [0.257, 0.357]	1.10 [1.06, 1.15]
CF-10	Linear	7.83×10^{-4} [4.19×10^{-4}, 1.21×10^{-3}]	1.54×10^{-4} [7.99×10^{-5}, 2.47×10^{-4}]	6.30×10^{-4} [3.38×10^{-4}, 9.69×10^{-4}]	0.555 [0.377, 0.742]	1.00 [0.999, 1.00]
	Tree	2.20×10^{-4} [1.46×10^{-4}, 3.07×10^{-4}]	1.67×10^{-4} [1.09×10^{-4}, 2.40×10^{-4}]	5.28×10^{-5} [3.05×10^{-5}, 7.98×10^{-5}]	0.682 [0.564, 0.802]	1.15 [1.12, 1.18]
	Overall	5.02×10^{-4} [3.23×10^{-4}, 7.33×10^{-4}]	1.60×10^{-4} [1.15×10^{-4}, 2.14×10^{-4}]	3.41×10^{-4} [1.98×10^{-4}, 5.30×10^{-4}]	0.619 [0.520, 0.724]	1.03 [1.02, 1.05]
CF-20	Linear	7.83×10^{-4} [4.13×10^{-4}, 1.19×10^{-3}]	1.54×10^{-4} [8.22×10^{-5}, 2.39×10^{-4}]	6.29×10^{-4} [3.28×10^{-4}, 9.66×10^{-4}]	1.16 [0.794, 1.54]	1.00 [1.00, 1.00]
	Tree	1.98×10^{-4} [1.32×10^{-4}, 2.77×10^{-4}]	1.46×10^{-4} [9.63×10^{-5}, 2.12×10^{-4}]	5.17×10^{-5} [2.93×10^{-5}, 7.78×10^{-5}]	1.33 [1.09, 1.58]	1.03 [1.02, 1.06]
	Overall	4.91×10^{-4} [3.10×10^{-4}, 7.16×10^{-4}]	1.50×10^{-4} [1.06×10^{-4}, 2.03×10^{-4}]	3.40×10^{-4} [1.96×10^{-4}, 5.30×10^{-4}]	1.25 [1.03, 1.47]	1.01 [1.00, 1.01]
SF	Linear	7.83×10^{-4} [4.20×10^{-4}, 1.20×10^{-3}]	1.55×10^{-4} [7.91×10^{-5}, 2.44×10^{-4}]	6.28×10^{-4} [3.40×10^{-4}, 9.60×10^{-4}]	0.116 [0.088, 0.149]	1.00
	Tree	1.92×10^{-4} [1.27×10^{-4}, 2.66×10^{-4}]	1.32×10^{-4} [8.61×10^{-5}, 1.83×10^{-4}]	6.00×10^{-5} [3.48×10^{-5}, 8.99×10^{-5}]	0.119 [0.098, 0.143]	1.00
	Overall	4.87×10^{-4} [2.94×10^{-4}, 7.08×10^{-4}]	1.43×10^{-4} [9.62×10^{-5}, 1.94×10^{-4}]	3.44×10^{-4} [1.98×10^{-4}, 5.21×10^{-4}]	0.117 [0.098, 0.136]	1.00

Table 3 supports the practical single-fit simplification. For LinearMSR, cross-fitting provides little measurable accuracy benefit once $k \geq 5$: CF-5, CF-10, CF-20, and SF all have essentially identical normalized error. Runtime, however, grows with k, reaching 1.16 seconds for CF-20 compared with 0.116 seconds for SF. Thus, for LinearMSR, cross-fitting does not seem to noticeably improve the runtime–accuracy tradeoff.

For TreeMSR, increasing k more largely improves the cross-fitted estimator: error falls from 8.26×10^{-4} at CF-2 to 1.98×10^{-4} at CF-20. This confirms the original intuition that larger training folds can improve surrogate quality. However, the single-fit variant remains slightly more accurate than CF-20, with error 1.92×10^{-4}, and is far faster: CF-20 is roughly $11\times$ slower than SF. The bias–variance decomposition shows that the single-fit TreeMSR variant does not exhibit larger empirical squared bias in this experiment; instead, it has lower empirical bias than all cross-fitted variants and only slightly higher variance than CF-20.

Averaging over LinearMSR and TreeMSR gives the same conclusion. SF achieves the lowest mean error, 4.87×10^{-4}, and the lowest runtime, 0.117 seconds. CF-20 nearly matches SF in error but is more than $10\times$ slower. We therefore find strong empirical support for the original paper's practical recommendation: although single-fit breaks the theoretical independence condition, it gives the best or near-best accuracy at a substantially lower computational cost in this controlled reproduction.

7 Experiment 2: Shapley Benchmark Reproduction and Rank Stability

The original paper's most prominent empirical claim is its Shapley-value benchmark. Table 1 of Witter et al. (2025) reports normalized squared ℓ_2 error for LinearMSR, TreeMSR, KernelSHAP, PermutationSHAP, and LeverageSHAP at sample budget $m = 40n$, and states that TreeMSR achieves average error $6.5\times$, $3.8\times$, and $2.6\times$ lower than PermutationSHAP, KernelSHAP, and LeverageSHAP respectively, achieving SOTA performance. We reproduce this benchmark using our controlled pipeline and the released RegressionMSR implementation. We follow the original evaluation metric, and report means and quartiles over runs.

Table 4 shows that the main qualitative conclusion reproduces: TreeMSR remains the lowest-error estimator on average. The reproduced average error for TreeMSR is 2.95×10^{-4}, compared with 4.82×10^{-4} for LinearMSR, 5.21×10^{-4} for LeverageSHAP, 1.23×10^{-3} for PermutationSHAP, and 4.05×10^{-3} for KernelSHAP. The ranking of the top three estimators is therefore the same as in the original paper: TreeMSR first, LinearMSR second, and LeverageSHAP third.

However, the exact numerical ratios do not reproduce. In our run, TreeMSR is $4.2\times$ lower-error than PermutationSHAP, $13.7\times$ lower-error than KernelSHAP, and $1.8\times$ lower-error than LeverageSHAP. Thus, the direction of the original claim is preserved for all three baselines, but the magnitude of the improvement changes. The KernelSHAP ratio is especially sensitive to one dataset: our KernelSHAP Adult mean is substantially larger than the corresponding value in the original table, which inflates the reproduced KernelSHAP average. Full dataset-level statistics as in Table 1 of the original paper are reported in Appendix A.3.

Since the benchmark depends on stochastic model training, stochastic explicand selection, coalition sampling, estimator randomness, dependency versions, and SHAP-library behavior, we consequently view Table 4 as a qualitative reproduction of the main ordering rather than an exact numerical reconstruction of Table 1.

Table 4: Reproduction of the original Shapley benchmark at $m = 40n$ with 10 explicands per dataset. We use $k = 10$ folds. Original values are from Table 1 of Witter et al. (2025); reproduced values are from our controlled seeded pipeline. Lower error is better.

Estimator	Original mean	Reproduced mean	Reprod./Orig.	Reproduced rank
LinearMSR	9.51×10^{-4}	4.82×10^{-4}	0.51	2
TreeMSR	4.29×10^{-4}	2.95×10^{-4}	0.69	1
KernelSHAP	1.64×10^{-3}	4.05×10^{-3}	2.47	5
PermutationSHAP	2.78×10^{-3}	1.23×10^{-3}	0.44	4
LeverageSHAP	1.13×10^{-3}	5.21×10^{-4}	0.46	3

Rank stability. The original benchmark evaluates attribution-vector magnitude error. In many feature-attribution use cases, however, the induced feature ordering is also important. We therefore additionally compute Spearman correlation, Kendall's τ, Precision@3, and Precision@5 between the estimated and exact attribution vectors.

Table 5 shows that lower ℓ_2 error does not necessarily imply better ranking performance. TreeMSR has the lowest average magnitude error, but PermutationSHAP has the highest overall Spearman and Kendall rank correlations, despite having the highest error in the original study and second highest in our reproduction. At the same time, top-k precision is nearly saturated for all methods: Precision@3 is at least 0.988 for every estimator, and Precision@5 is at least 0.970. This suggests that the top few features are usually recovered

Table 5: Rank-stability metrics for the Shapley benchmark. Values are averaged over datasets, explicands, and runs; brackets denote 95% bootstrap confidence intervals. Higher is better.

Estimator	Spearman	Kendall's τ	Precision@3	Precision@5
LinearMSR	0.828 [0.788, 0.866]	0.753 [0.707, 0.795]	0.988 [0.971, 1.000]	0.970 [0.950, 0.988]
TreeMSR	0.854 [0.809, 0.892]	0.795 [0.749, 0.841]	1.000 [1.000, 1.000]	0.975 [0.955, 0.990]
KernelSHAP	0.839 [0.800, 0.876]	0.765 [0.718, 0.809]	0.988 [0.971, 1.000]	0.973 [0.950, 0.993]
PermutationSHAP	0.992 [0.986, 0.996]	0.974 [0.963, 0.984]	0.996 [0.987, 1.000]	0.985 [0.972, 0.995]
LeverageSHAP	0.825 [0.783, 0.864]	0.748 [0.706, 0.791]	0.992 [0.979, 1.000]	0.978 [0.960, 0.993]

Table 6: Headline TreeMSR improvement factors at $m = 40n$. Values report baseline mean error divided by TreeMSR mean error. Larger values indicate a larger TreeMSR advantage.

Baseline	Original factor	Reproduced factor
PermutationSHAP	6.5×	4.2×
KernelSHAP	3.8×	13.7×
LeverageSHAP	2.6×	1.8×

reliably by all estimators in this benchmark, while the full ranking and attribution magnitudes distinguish the methods more strongly. Thus, TreeMSR appears generally best when accurate attribution magnitudes are required, but rank-based metrics reveal a different and complementary view of estimator quality.

8 Experiment 3: Robustness to Noisy Value Queries

The original paper studies noisy value-function access by adding Gaussian noise to observed coalition values in Appendix F (Witter et al., 2025). We reproduce this using the released implementation and extend it to two additional noise models: Laplacian, which introduces heavier-tailed perturbations (Kotz et al., 2001), and heteroskedastic Gaussian, where variance depends on coalition size. In all cases, estimators observe

$$\widetilde{v}(S) = v(S) + \epsilon(S),$$

but are evaluated against the clean ground-truth probabilistic values $\phi(v)$. Thus, the experiment measures whether an estimator can recover the original clean attribution vector when its sampled value queries are noisy.

We use requested budget $m = 40n$ and noise magnitudes

$$\sigma \in \{10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1\}.$$

For Gaussian noise,

$$\epsilon(S) \sim \mathcal{N}(0, \sigma^2).$$

For Laplacian noise,

$$\epsilon(S) \sim \text{Laplace}\left(0, \frac{\sigma}{\sqrt{2}}\right),$$

so that the variance matches the Gaussian setting. This isolates the effect of heavier tails while keeping the noise scale comparable. For heteroskedastic Gaussian noise,

$$\epsilon(S) \sim \mathcal{N}\big(0, \sigma^2 h(S)^2\big), \qquad h(S)^2 = 1 + 4\left(\frac{|S|}{n} - \frac{1}{2}\right)^2.$$

Thus middle-sized coalitions receive noise standard deviation σ, while extreme coalitions receive noise standard deviation up to $\sqrt{2}\sigma$. We leave this multiplier unnormalized and interpret the setting as a moderate coalition-size-dependent stress test.

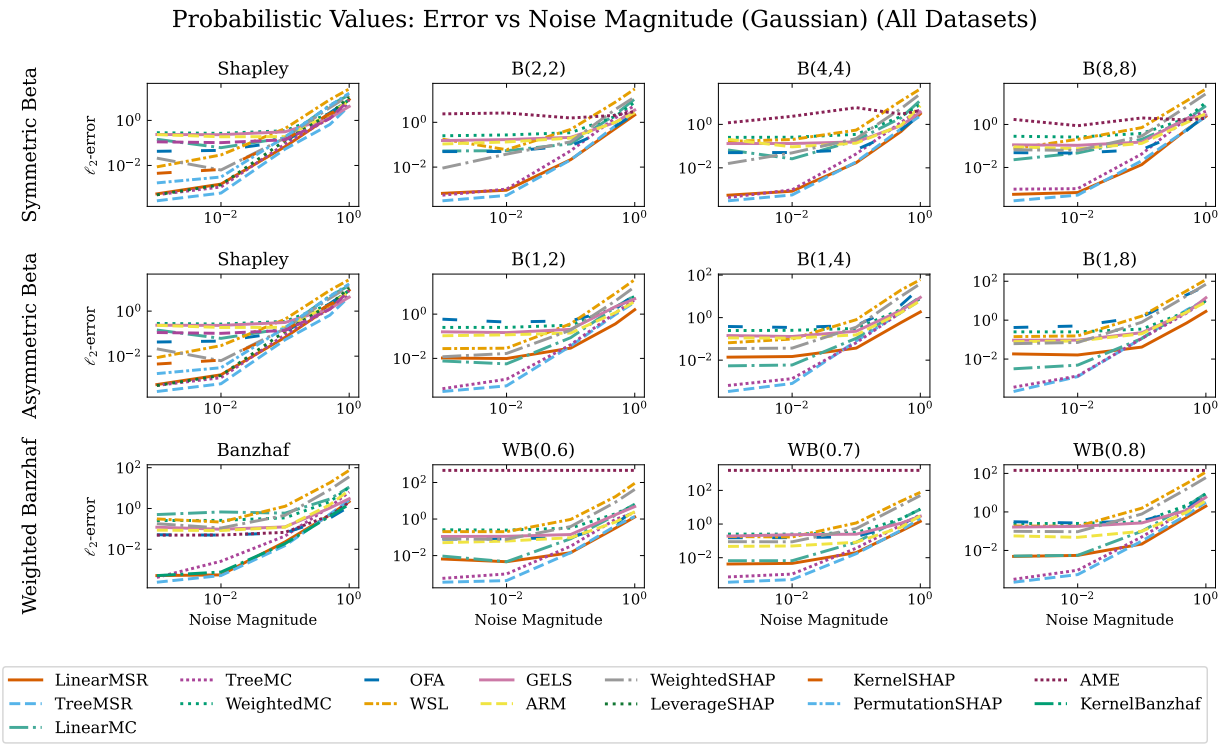

Figure 1: Error versus Gaussian noise magnitude for probabilistic-value estimation. $m = 40n$. Each panel corresponds to one probabilistic value, and curves are averaged over 10 explicands per dataset. Lower is better. The Laplacian and heteroskedastic analogues are shown in Appendix A.4.

We report the normalized squared ℓ_2 error and Spearman rank correlation between $\widehat{\phi}$ and $\phi(v)$. The main error curves follow the format of the original Appendix F, with one panel for each of 12 probabilistic value types. Figure 1 shows the Gaussian setting; the corresponding Laplacian, heteroskedastic, and Spearman grids are provided in Appendix A.4.

Figure 1 qualitatively reproduces the original Gaussian-noise finding: TreeMSR is strongest in the low-noise regime for most probabilistic values, often by a large margin. The same pattern appears under Laplacian and heteroskedastic noise. Thus, TreeMSR's low-noise advantage does not appear to be an artifact of Gaussian perturbations alone. At the same time, the high-noise regime is more nuanced. As σ increases, all estimators degrade and the curves become less separated. In several high-noise settings, simpler or more stable estimators such as LinearMSR, OFA, GELS, or KernelBanzhaf can match or outperform TreeMSR.

To quantify this pattern, Table 7 reports the ratio best non-TreeMSR mean error/TreeMSR mean error for three representative probabilistic values: Shapley, Banzhaf, and $B(1, 8)$. These represent the symmetric Beta, weighted Banzhaf, and asymmetric Beta families, respectively. Values greater than 1 mean that TreeMSR has lower error than every non-TreeMSR estimator; values below 1, shown in bold, mean that some other estimator has lower error than TreeMSR.

Table 7 confirms the main visual trend. At low noise, TreeMSR is the lowest-error estimator for all three representative values and all three noise models. This supports the original paper's claim that TreeMSR performs best especially when the noise magnitude is small. The non-Gaussian extensions also support the same conclusion: the low-noise advantage persists under both Laplacian and heteroskedastic perturbations.

At medium and high noise, however, the picture changes. For Shapley values, TreeMSR remains competitive and often best, but the gap narrows and some high-noise settings are won by GELS or TreeMC. For Banzhaf

Table 7: Best non-TreeMSR estimator relative to TreeMSR under noisy value queries. Each entry reports best non-TreeMSR error/TreeMSR error, with the winning non-TreeMSR estimator in parentheses. Values greater than 1 indicate that TreeMSR is best; bold values below 1 indicate that a non-TreeMSR estimator has lower error. Low, medium, and high correspond to $\sigma = 10^{-3}$, 10^{-1}, and 1, respectively.

Value	Noise model	Low	Medium	High
Shapley	Gaussian	1.84 (LevSHAP)	1.29 (LinearMSR)	**0.95 (GELS)**
Shapley	Laplace	1.82 (TreeMC)	2.03 (LevSHAP)	1.03 (GELS)
Shapley	Heteroskedastic	1.62 (LevSHAP)	**0.76 (TreeMC)**	**0.45 (GELS)**
Banzhaf	Gaussian	1.72 (TreeMC)	1.24 (KernelBanzhaf)	**0.49 (OFA)**
Banzhaf	Laplace	1.52 (KernelBanzhaf)	**0.56 (KernelBanzhaf)**	**0.76 (LinearMSR)**
Banzhaf	Heteroskedastic	1.78 (LinearMSR)	**0.66 (LinearMSR)**	**0.93 (LinearMSR)**
$B(1,8)$	Gaussian	1.67 (TreeMC)	**0.35 (LinearMSR)**	**0.25 (LinearMSR)**
$B(1,8)$	Laplace	1.86 (TreeMC)	**0.39 (LinearMSR)**	**0.17 (LinearMSR)**
$B(1,8)$	Heteroskedastic	1.81 (TreeMC)	**0.25 (LinearMSR)**	**0.17 (LinearMSR)**

values, KernelBanzhaf, OFA, and LinearMSR can overtake TreeMSR as noise grows. For the asymmetric $B(1,8)$ value, LinearMSR becomes substantially better than TreeMSR at medium and high noise under all three noise models. This suggests that the flexible tree surrogate is most valuable when the residual signal is learnable, while simpler estimators can be more stable once observation noise dominates, perhaps due to avoiding overfitting to the noise.

We also look at rank stability in Appendix A.4 for a complementary view. Spearman correlations decrease steadily as noise increases for nearly all estimators and probabilistic values. At low noise, TreeMSR and other tree- or regression-adjusted estimators typically preserve feature rankings well. At high noise, the ranking metrics become less separated, and TreeMSR is not uniformly best.

Thus, the original Gaussian-noise robustness claim qualitatively reproduces, and the same low-noise TreeMSR advantage extends to Laplacian and heteroskedastic noise. The stronger claim that TreeMSR uniformly dominates under all noise magnitudes is not supported. Instead, TreeMSR is best interpreted as highly effective in low-to-moderate noise regimes, while high-noise settings can become noise-dominated and may favor simpler or value-specific estimators.

9 Discussion, Limitations, and Future Directions

9.1 What was easy and difficult to reproduce

What was easy. The released RegressionMSR artifact was a useful basis for reproduction. The main code path exposes the datasets, probabilistic-value weightings, estimators, and table-generation scripts used in the original experiments. The saved output logs also made it possible to trace how reported table entries were computed; in particular, we could regenerate the released Shapley benchmark table from the stored outputs to the displayed scientific-notation precision. The codebase further includes the exact-value routines needed for the original ground-truth protocol, using enumeration for small-n models and tree-based exact probabilistic values for larger random-forest models.

What was difficult. The main difficulty was exact determinism. The repository did not include a detailed README or fully pinned environment, and the experimental pipeline is only partially seeded. Although some scripts set a global NumPy seed and some models use fixed random states, many estimator-local generators are created with `np.random.default_rng()` without an explicit seed, so `np.random.seed(42)` does not make the full run deterministic. Several model-fitting components, including small-n `MLPRegressor` fits and tree surrogates based on `XGBRegressor`, are also unseeded. The noise experiments and SHAP wrappers introduce additional untracked randomness. We also found that some scripts append to existing output files, which can cause generated tables to aggregate different numbers of rows across estimator–dataset

cells. These issues motivated our seeded, appending-aware reproduction pipeline and help explain why we focus on qualitative reproduction and controlled within-pipeline comparisons rather than exact recovery of every original number.

9.2 Limitations

Our experiments focus on the interventional feature-attribution setting used in the original paper, and do not test conditional feature attribution, data valuation, or settings where evaluating $v(S)$ requires model retraining. Our All-Row comparison isolates the averaging convention while holding the released stored denominators fixed; it should therefore be read as a paper-style reconstruction rather than a complete independent reimplementation of every theoretical sampling detail. The non-Gaussian noise experiments are intended as robustness stress tests rather than exhaustive models of noisy value access; in particular, the heteroskedastic form is synthetic and depends only on coalition size.

9.3 Conclusions

Our study supports the main qualitative conclusions of RegressionMSR while clarifying several implementation and reproducibility details. First, our artifact audit shows that the released implementation differs from the estimator displayed in Algorithm 1 in both residual-correction normalization and denominator convention: it computes a Separated-Means correction rather than the displayed All-Row correction, and in the large-n with-replacement regime stores a size-level denominator rather than the full coalition-level probability $D(S)$. The averaging difference is algebraically real and becomes measurable at the residual-correction level, especially for asymmetric probabilistic values where complement-pair sampling is disabled. Holding the released denominator fixed, however, this averaging difference does not undermine empirical performance in our experiments: final attribution errors remain similar, and the released correction is often slightly more accurate in finite samples. Second, our cross-fitting ablation supports the original authors' practical recommendation. The single-fit simplification breaks the theoretical independence condition, but provides the best or near-best accuracy at substantially lower runtime in our controlled runs. Third, our Shapley benchmark reproduction largely preserves the original ordering: TreeMSR remains the lowest-error estimator on average, although the exact improvement factors differ, indicating sensitivity to implementation, dependency, and randomness details. Finally, the noise experiments suggest that TreeMSR's low-noise advantage is not specific to Gaussian perturbations and extends qualitatively to Laplacian and coalition-size-dependent heteroskedastic noise, though this advantage diminishes as noise increases.

9.4 Future Directions

The RegressionMSR artifact would benefit from a reference implementation that explicitly exposes the All-Row, Separated-Means, cross-fitted, and single-fit variants as named estimator choices, together with pinned environments and deterministic output generation. The empirical comparison could be expanded to newer Shapley estimators such as PolySHAP and OddSHAP under a unified value-function interface once stable reference implementations are available. Further, the noisy-value setting deserves a more systematic treatment: in applications where $v(S)$ is itself estimated by simulation, resampling, or retraining, the noise may be correlated, heteroskedastic, biased, or budget-dependent. Understanding when surrogate adjustment denoises value queries versus overfits noisy observations is an important next step.

9.5 Broader Impact

The broader impact of this work is primarily methodological. Probabilistic-value estimators are often used to support explanations of model predictions, so implementation details that change attribution estimates can affect how users interpret model behavior. By documenting these details and providing seeded reproduction code, this study aims to make RegressionMSR easier to evaluate, compare, and use responsibly. At the same time, our results reinforce a general caution for explainable AI: lower attribution error under one metric does not guarantee better behavior under every downstream criterion, such as feature ranking, robustness to noisy queries, or interpretability for end users.

References

John F. Banzhaf, III. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, 19(2): 317–343, 1965.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. ISSN 0305-0548. doi: 10.1016/j.cor.2008.04.004. URL `https://doi.org/10.1016/j.cor.2008.04.004`.

Abraham Charnes, Bracha Golany, Michael Keane, and John Rousseau. Extremal principle solutions of games in characteristic function form: Core, Chebychev and Shapley value generalizations. In *Econometrics of Planning and Efficiency*, pp. 123–133. Springer Nature, 1988.

Ian Covert and Su-In Lee. Improving kernelshap: Practical shapley value estimation using linear regression. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1993.

Fabian Fumagalli, Landon Butler, Justin Singh Kang, Kannan Ramchandran, and R. Teal Witter. An odd estimator for shapley values. *arXiv preprint arXiv:2602.01399*, 2026a.

Fabian Fumagalli, R. Teal Witter, and Christopher Musco. PolySHAP: Extending KernelSHAP with interaction-informed polynomial regression. In *International Conference on Learning Representations*, 2026b.

Patrick Kolpaczki, Viktor Bengs, Maximilian Muschalik, and Eyke Hüllermeier. Approximating the shapley value without marginal contributions. In *AAAI Conference on Artificial Intelligence*, 2024.

Samuel Kotz, Tomasz J. Kozubowski, and Krzysztof Podgórski. *The Laplace Distribution and Generalizations: A Revisit with Applications to Communications, Economics, Engineering, and Finance*. Birkhäuser, Boston, MA, 2001. ISBN 978-1-4612-6646-4. doi: 10.1007/978-1-4612-0173-1. URL `https://doi.org/10.1007/978-1-4612-0173-1`.

Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, 2022.

Weida Li and Yaoliang Yu. One sample fits all: Approximating all probabilistic values simultaneously and efficiently. *arXiv preprint arXiv:2410.23808*, 2024.

Ziqi Liu, Kiljae Lee, Yuan Zhang, and Weijing Tang. First-order efficiency for probabilistic value estimation via a statistical viewpoint. *arXiv preprint arXiv:2605.02827*, 2026.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017.

Christopher Musco and R. Teal Witter. Provably accurate shapley value estimation via leverage score sampling. In *International Conference on Learning Representations*, 2025.

Lloyd S. Shapley. A value for n-person games. In *Contributions to the Theory of Games II*. Princeton University Press, 1953.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, 2014. doi: 10.1007/s10115-013-0679-x. URL `https://doi.org/10.1007/s10115-013-0679-x`.

Jiachen T. Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, 2023.

Robert J. Weber. Probabilistic values for games. In *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

R. Teal Witter, Yurong Liu, and Christopher Musco. Regression-adjusted monte carlo estimators for shapley values and probabilistic values. In *Advances in Neural Information Processing Systems*, 2025.

A Appendix

A.1 Released Implementation of RegressionMSR and Residual-Correction Scaling

This section documents an implementation-level discrepancy we observed while auditing the released RegressionMSR code. The theoretical estimator in Algorithm 1 of the original paper computes a single importance-weighted residual average over the held-out fold. In contrast, the released code computes two separate conditional means: one over held-out coalitions that contain feature i and one over held-out coalitions that do not contain feature i. This distinction affects the scaling of the residual correction. The cross-fitting structure itself is implemented: the code trains the surrogate on samples outside the held-out fold and computes residuals on the held-out fold. The issue discussed here is only the normalization of the residual correction.

A.1.1 Notation

Let $N = \{1, \dots, n\}$ denote the feature set. For a coalition $S \subseteq N$, let $v(S)$ be the coalition value and let $f^{(\ell)}(S)$ denote the surrogate trained for fold ℓ. Define the held-out residual game

$$r^{(\ell)}(S) = v(S) - f^{(\ell)}(S). \tag{8}$$

For a fixed feature i, the exact residual contribution for a probabilistic value with weights $p_0, \dots, p_{n-1}$ is

$$\phi_i(r^{(\ell)}) = \sum_{S: i \in S} r^{(\ell)}(S) p_{|S|-1} - \sum_{S: i \notin S} r^{(\ell)}(S) p_{|S|}. \tag{9}$$

Here and below, terms with invalid indices are absent; for example, the $p_{|S|-1}$ term only appears when $i \in S$.

Let $\mathcal{H}_\ell$ be the held-out fold used for residual correction in split ℓ, and let $m_\ell = |\mathcal{H}_\ell|$. Define

$$\mathcal{H}^+_{\ell,i} = \{S \in \mathcal{H}_\ell : i \in S\}, \qquad N^+_{\ell,i} = |\mathcal{H}^+_{\ell,i}|, \tag{10}$$

$$\mathcal{H}^-_{\ell,i} = \{S \in \mathcal{H}_\ell : i \notin S\}, \qquad N^-_{\ell,i} = |\mathcal{H}^-_{\ell,i}|. \tag{11}$$

Thus $N^+_{\ell,i} + N^-_{\ell,i} = m_\ell$. We also define the empirical containment fraction

$$q_{\ell,i} = \frac{N^+_{\ell,i}}{m_\ell}. \tag{12}$$

When paired complement sampling is used and complement pairs remain intact inside the fold, $N^+_{\ell,i} = N^-_{\ell,i} = m_\ell/2$, so $q_{\ell,i} = 1/2$ for every feature i.

Let $d(S)$ denote the denominator stored in the released code as `prob_sampled`. We do not assume at first that $d(S)$ is the paper's probability density $D(S)$; the goal is to compare the normalizations while holding the same stored denominator fixed.

A.1.2 Theoretical residual correction in the paper

The paper's fold-level residual correction is a single average over the full held-out fold:

$$\widehat{R}^{\text{paper}}_{\ell,i} = \frac{1}{m_\ell} \sum_{S \in \mathcal{H}_\ell} r^{(\ell)}(S) \frac{p_{|S|-1}\mathbf{1}[i \in S] - p_{|S|}\mathbf{1}[i \notin S]}{D(S)}. \tag{13}$$

Using the code's stored denominator $d(S)$ instead of $D(S)$ for direct comparison, the analogous all-row correction is

$$\widehat{R}^{\text{allrow}}_{\ell,i} = \frac{1}{m_\ell} \left[\sum_{S \in \mathcal{H}^+_{\ell,i}} r^{(\ell)}(S) \frac{p_{|S|-1}}{d(S)} - \sum_{S \in \mathcal{H}^-_{\ell,i}} r^{(\ell)}(S) \frac{p_{|S|}}{d(S)} \right]. \tag{14}$$

This is the direct code-level translation of the displayed paper formula, modulo replacing $D(S)$ by the stored denominator $d(S)$.

Algorithm 1 Released RegressionMSR residual correction, as implemented

Require: Sampled coalition matrix X, values $y(S) = v(S)$, stored denominators $d(S)$, weights p, number of folds k

Pair sampling: if p is symmetric, sampled coalitions are stored in complement pairs (S, S^c) with the same denominator $d(S)$.
Partition rows into k folds, keeping complement pairs adjacent whenever the fold boundary permits.
for $\ell = 1, \ldots, k$ **do**
  Train surrogate $f^{(\ell)}$ on rows outside held-out fold $\mathcal{H}_\ell$.
  Compute surrogate values $f^{(\ell)}(S)$ for each $S \in \mathcal{H}_\ell$.
  Compute surrogate attributions $\phi_i(f^{(\ell)})$ for all $i \in N$.
  for $i = 1, \ldots, n$ **do**
    $\mathcal{H}^+_{\ell,i} \leftarrow \{S \in \mathcal{H}_\ell : i \in S\}; \quad \mathcal{H}^-_{\ell,i} \leftarrow \{S \in \mathcal{H}_\ell : i \notin S\}.$
    if $|\mathcal{H}^+_{\ell,i}| > 0$ **then**
      $$M^+_{\ell,i} \leftarrow \frac{1}{|\mathcal{H}^+_{\ell,i}|} \sum_{S \in \mathcal{H}^+_{\ell,i}} [v(S) - f^{(\ell)}(S)] \frac{p_{|S|-1}}{d(S)}.$$
    else
      $M^+_{\ell,i} \leftarrow 0.$
    end if
    if $|\mathcal{H}^-_{\ell,i}| > 0$ **then**
      $$M^-_{\ell,i} \leftarrow \frac{1}{|\mathcal{H}^-_{\ell,i}|} \sum_{S \in \mathcal{H}^-_{\ell,i}} [v(S) - f^{(\ell)}(S)] \frac{p_{|S|}}{d(S)}.$$
    else
      $M^-_{\ell,i} \leftarrow 0.$
    end if
    $\widehat{\phi}_i^{(\ell)} \leftarrow \phi_i(f^{(\ell)}) + M^+_{\ell,i} - M^-_{\ell,i}.$
  end for
end for
return $\widehat{\phi}_i \leftarrow \frac{1}{k} \sum_{\ell=1}^{k} \widehat{\phi}_i^{(\ell)}$ for all $i \in N$.

A.1.3 Residual correction in the released code

The released code computes the residual correction as two separate conditional means. In mathematical form, the fold-level correction is

$$\widehat{R}^{\text{code}}_{\ell,i} = M^+_{\ell,i} - M^-_{\ell,i}, \tag{15}$$

where

$$M^+_{\ell,i} = \begin{cases} \dfrac{1}{N^+_{\ell,i}} \displaystyle\sum_{S \in \mathcal{H}^+_{\ell,i}} r^{(\ell)}(S) \frac{p_{|S|-1}}{d(S)}, & N^+_{\ell,i} > 0, \\ 0, & N^+_{\ell,i} = 0, \end{cases} \tag{16}$$

$$M^-_{\ell,i} = \begin{cases} \dfrac{1}{N^-_{\ell,i}} \displaystyle\sum_{S \in \mathcal{H}^-_{\ell,i}} r^{(\ell)}(S) \frac{p_{|S|}}{d(S)}, & N^-_{\ell,i} > 0, \\ 0, & N^-_{\ell,i} = 0. \end{cases} \tag{17}$$

The fold estimate implemented in the released code is then

$$\widehat{\phi}^{\text{code}}_{\ell,i}(v) = \phi_i(f^{(\ell)}) + \widehat{R}^{\text{code}}_{\ell,i}, \tag{18}$$

and the final estimate averages these fold estimates over $\ell = 1, \ldots, k$.

A.1.4 Finite-sample relationship between the two corrections

For compactness, define

$$A_{\ell,i} = \frac{1}{m_\ell} \sum_{S \in \mathcal{H}^+_{\ell,i}} r^{(\ell)}(S) \frac{p_{|S|-1}}{d(S)}, \tag{19}$$

$$B_{\ell,i} = \frac{1}{m_\ell} \sum_{S \in \mathcal{H}^-_{\ell,i}} r^{(\ell)}(S) \frac{p_{|S|}}{d(S)}. \tag{20}$$

Then the all-row paper-style correction with the same denominator is

$$\widehat{R}^{\text{allrow}}_{\ell,i} = A_{\ell,i} - B_{\ell,i}. \tag{21}$$

If $0 < N^+_{\ell,i} < m_\ell$, then

$$\widehat{R}^{\text{code}}_{\ell,i} = \frac{1}{q_{\ell,i}} A_{\ell,i} - \frac{1}{1 - q_{\ell,i}} B_{\ell,i}. \tag{22}$$

Therefore,

$$\widehat{R}^{\text{code}}_{\ell,i} - \widehat{R}^{\text{allrow}}_{\ell,i} = \left(\frac{1}{q_{\ell,i}} - 1 \right) A_{\ell,i} - \left(\frac{1}{1 - q_{\ell,i}} - 1 \right) B_{\ell,i}. \tag{23}$$

Equation 23 is a finite-sample identity. It does not depend on any asymptotics.

Balanced paired folds. If complement pairs remain intact in the fold, then $q_{\ell,i} = 1/2$ for all i. Equation 22 gives

$$\widehat{R}^{\text{code}}_{\ell,i} = 2A_{\ell,i} - 2B_{\ell,i} = 2\widehat{R}^{\text{allrow}}_{\ell,i}. \tag{24}$$

Thus, using the same denominator $d(S)$, the released separated-mean correction is exactly twice the all-row correction whenever the held-out fold is pair-balanced.

Unbalanced folds. If a fold boundary splits complement pairs or if the held-out fold otherwise has $q_{\ell,i} \neq 1/2$, then there is generally no single constant c such that $\widehat{R}^{\text{code}}_{\ell,i} = c\widehat{R}^{\text{allrow}}_{\ell,i}$. The positive and negative sides are rescaled separately by $1/q_{\ell,i}$ and $1/(1 - q_{\ell,i})$.

Empty-side edge cases. If $N^+_{\ell,i} = 0$, the released code sets $M^+_{\ell,i} = 0$ and computes only the negative-side mean. In this case $q_{\ell,i} = 0$ and equation 22 is not defined, but the implementation is well-defined by convention. Similarly, if $N^-_{\ell,i} = 0$, the code sets $M^-_{\ell,i} = 0$. These cases are unusual under paired Shapley sampling but can occur under small unpaired samples or if folds are pathological.

A.1.5 Expectation-level bias under joint-density interpretation

Suppose, for this subsection, that held-out coalitions are sampled iid from a distribution $P(S)$ and that the code denominator is interpreted as the joint sampling density, i.e. $d(S) = P(S)$. Let

$$G^+_i = \sum_{S: i \in S} r(S) p_{|S|-1}, \tag{25}$$

$$G^-_i = \sum_{S: i \notin S} r(S) p_{|S|}, \tag{26}$$

$$q_i = P(i \in S). \tag{27}$$

The target residual attribution is

$$\phi_i(r) = G^+_i - G^-_i. \tag{28}$$

The separated-mean estimator converges to

$$\mathbb{E}\left[r(S) \frac{p_{|S|-1}}{P(S)} \mid i \in S \right] - \mathbb{E}\left[r(S) \frac{p_{|S|}}{P(S)} \mid i \notin S \right] = \frac{G^+_i}{q_i} - \frac{G^-_i}{1 - q_i}. \tag{29}$$

Thus the asymptotic bias is

$$\mathrm{Bias}_i = \left(\frac{1}{q_i} - 1\right) G_i^+ - \left(\frac{1}{1-q_i} - 1\right) G_i^- . \tag{30}$$

This is generally nonzero. In the balanced case $q_i = 1/2$, the separated-mean estimator converges to $2(G_i^+ - G_i^-) = 2\phi_i(r)$, not $\phi_i(r)$.

A.1.6 When do the two forms coincide?

The separated-mean correction can coincide with the all-row paper-style correction only under special conditions.

Conditional-density denominators. If the denominator in the positive stratum were

$$d_i^+(S) = P(S \mid i \in S) = \frac{P(S)}{q_i}, \tag{31}$$

and the denominator in the negative stratum were

$$d_i^-(S) = P(S \mid i \notin S) = \frac{P(S)}{1-q_i}, \tag{32}$$

then the separated means would be unbiased for G_i^+ and G_i^- separately. However, this requires a feature-specific conditional denominator. The released implementation stores a single row-level value `prob_sampled`; it is not feature-specific.

Balanced sampling and doubled density. If $q_i = 1/2$ for every feature and the stored denominator satisfies

$$d(S) = 2P(S), \tag{33}$$

then the separated-mean correction equals the all-row correction. Pair sampling provides the first condition, $q_i = 1/2$, when complement pairs remain intact. It does not by itself guarantee the second condition.

Special residual cancellation. For a fixed residual game, the bias in equation 30 can vanish by cancellation. This requires

$$(1-q_i)^2 G_i^+ = q_i^2 G_i^- . \tag{34}$$

This is a property of a particular residual game and sampling distribution; it is not a general unbiasedness guarantee.

Zero residual value. If $\phi_i(r) = 0$ and the positive and negative residual sums cancel in the appropriate way, then the scaling difference may have no effect for feature i. More generally, if the surrogate is highly accurate and $r(S)$ is small, the practical effect of the correction scaling can be small even when the formulae are not identical.

A.1.7 Summary for reproducibility analysis

The released implementation matches several major elements of the paper's algorithm: it samples coalitions, uses paired complements for symmetric weights, evaluates $v(S)$ on sampled coalitions, uses cross-fitting, trains the surrogate on non-held-out folds, computes held-out residuals, and adds surrogate attributions $\phi_i(f^{(\ell)})$. The discrepancy arises in the residual-correction normalization. The paper's displayed estimator uses one all-row importance-weighted residual average over the held-out fold. The released implementation uses separate conditional means over coalitions containing and excluding feature i. When paired folds are balanced, this separated-mean correction is exactly twice the corresponding all-row correction if the same denominator is used. When folds are unbalanced, the two sides are rescaled by $1/q_{\ell,i}$ and $1/(1-q_{\ell,i})$, respectively. Therefore, the released implementation should be treated as a distinct residual-correction variant unless the stored denominator can be justified as a feature-conditional density.

For symmetric probabilistic weights such as Shapley, the released implementation uses paired complement sampling. The implementation first rounds the sampled row count to a multiple of $2k$, then shuffles complement-pair identifiers and expands each pair into adjacent rows before forming folds. Thus, complement pairs are not split across folds. Consequently, for every fold ℓ and every feature i, the number of held-out coalitions containing i equals the number not containing i:

$$N^{+}_{\ell,i} = N^{-}_{\ell,i} = |\mathcal{S}^{(\ell)}|/2.$$

Therefore, the separated-mean residual correction used in the implementation is exactly twice the full-fold average computed with the same stored denominator. This factor-of-two identity concerns only the averaging convention. The stored denominator in the implementation is not generally the per-coalition density $D(S)$ appearing in the paper formula: in the with-replacement sampler it is a size-level weight, while in the without-replacement sampler it is an inclusion probability. Thus the implemented correction differs from the displayed estimator both in its averaging convention and in the interpretation of the sampling denominator. We isolate the effect of the averaging convention here rather than reimplementing the exact theoretical sampling denominator.

As a brief aside, looking into the effects of the denominator implementation, we ran a diagnostic on synthetic coalitions comparing the denominator used in the code with a reconstructed paper style $D(S)$ on the same coalitions. We find that the discrepancy is small for without-replacement sampler, but large for the with-replacement sampler.

Table 8: Denominator diagnostic comparing the released stored denominator $d(S)$ with a reconstructed coalition-level design denominator $D(S)$. The stored/design ratio is averaged over sampled coalitions.

n	Weighting	Sampler	Mean d/D
8	Shapley	w/o replacement. paired	1.00
8	Beta(2,5)	w/o replacement.	1.005
60	Shapley	w/ replacment. paired	74.04
60	Beta(2,5)	w/ replacement.	306.11

Finally, for LinearMSR with Shapley or Beta$(1,1)$ weights, the implementation also adds an efficiency adjustment

$$(v([n]) - \sum_j \phi_j(f) - v(\emptyset))/n$$

to each feature. Our factor-of-two discussion applies to the residual-correction term, not to this additional efficiency adjustment.

A.2 Residual-Correction Discrepancy in Practice

The implementation audit in Section 5 shows that the released code computes a separated-mean residual correction rather than the all-row signed average displayed in Algorithm 1 of the original paper. Here we test whether the discrepancy remains only a formal difference, or whether it becomes empirically visible for asymmetric probabilistic values where complement-pair sampling is disabled.

For fold ℓ and feature i, define

$$q_{\ell,i} = \frac{|\mathcal{H}^{+}_{\ell,i}|}{|\mathcal{H}_\ell|}, \qquad \mathcal{H}^{+}_{\ell,i} = \{S \in \mathcal{H}_\ell : i \in S\}.$$

Thus, $q_{\ell,i}$ is the fraction of held-out coalitions in fold ℓ that contain feature i. In the all-row correction, the positive and negative residual components enter as

$$\widehat{R}^{\text{all}}_{i,\ell} = A_{\ell,i} - B_{\ell,i}.$$

Table 9: Residual-correction discrepancy across symmetric and asymmetric probabilistic values. Results are averaged over datasets, explicands, and sample seeds.

Value group	Surrogate	Mean q	Δ_q	All Row error	Separated Means error	Residual discrepancy
Symmetric	Linear	0.500	0.0017	1.69×10^{-3}	1.69×10^{-3}	1.17×10^{-5}
Symmetric	Tree	0.500	0.0026	4.00×10^{-4}	3.99×10^{-4}	3.06×10^{-5}
Asymmetric	Linear	0.599	0.286	2.47×10^{-2}	2.45×10^{-2}	1.76×10^{-3}
Asymmetric	Tree	0.599	0.286	3.83×10^{-4}	3.68×10^{-4}	8.55×10^{-4}

The released separated-mean correction instead satisfies

$$\widehat{R}^{\text{sep}}_{i,\ell} = \frac{1}{q_{\ell,i}} A_{\ell,i} - \frac{1}{1-q_{\ell,i}} B_{\ell,i},$$

whenever both sides are nonempty. Therefore the two corrections differ by

$$\widehat{R}^{\text{sep}}_{i,\ell} - \widehat{R}^{\text{all}}_{i,\ell} = \left(\frac{1}{q_{\ell,i}} - 1\right) A_{\ell,i} - \left(\frac{1}{1-q_{\ell,i}} - 1\right) B_{\ell,i}.$$

For symmetric probabilistic values, the implementation uses complement-pair sampling and preserves pairs inside folds. Each pair (S, S^c) contains feature i exactly once, so $q_{\ell,i} = 1/2$. In this case,

$$\widehat{R}^{\text{sep}}_{i,\ell} = 2\widehat{R}^{\text{all}}_{i,\ell}$$

when both corrections are computed with the same stored denominator. For asymmetric values, pair sampling is disabled, and $q_{\ell,i}$ can deviate substantially from 1/2.

We test this effect directly by computing both corrections within the same cross-fitted RegressionMSR run. For each dataset, explicand, value, surrogate type, and sample seed, we train the same surrogate on the same folds and compute both the paper-style all-row correction and the released separated-mean correction using the same sampled coalitions, same stored denominators, and same predictions. We use $k = 10$ folds and sample budget $m = 40n$. We evaluate symmetric values, including Shapley, $B(2,2)$, $B(8,8)$, and weighted Banzhaf with $q = 0.5$, as well as asymmetric values, including $B(1,4)$, $B(1,8)$, $B(4,1)$, $B(8,1)$, and weighted Banzhaf with $q = 0.8$ and $q = 0.9$.

We report the normalized squared ℓ_2 error of the full estimator, empirical squared bias and variance over sample seeds, and the residual-only discrepancy

$$\frac{\|\widehat{R}^{\text{sep}} - \widehat{R}^{\text{all}}\|_2}{\|\phi\|_2}.$$

We also report

$$\Delta_q = \mathbb{E}_{\ell,i}\left|q_{\ell,i} - \frac{1}{2}\right|,$$

which measures how far the held-out folds are from balanced paired sampling.

Table 9 shows that the q-diagnostic behaves as expected. Symmetric values have mean $q = 0.5$ and near-zero Δ_q, confirming that complement-pair sampling balances feature inclusion inside held-out folds. Asymmetric values have much larger imbalance, with $\Delta_q = 0.286$ on average. The residual-only discrepancy correspondingly increases by roughly one to two orders of magnitude for asymmetric values.

Despite this larger residual discrepancy, the final estimator errors are only modestly affected. For symmetric values, PaperStyle-CF-10 and Released-CF-10 are nearly indistinguishable. For asymmetric values, Released-CF-10 is slightly more accurate on average in these runs: the released/paper error ratio is 0.992

for LinearMSR and 0.953 for TreeMSR. The improvement is mostly associated with slightly lower empirical bias, while variance remains similar. The largest differences occur for extreme asymmetric values such as $B(1, 8)$ and $B(8, 1)$, especially on the Independent dataset, where Δ_q exceeds 0.4 and the released/paper error ratio can fall near 0.62.

These results support two conclusions. First, the released residual correction is not merely a cosmetic implementation difference: once pair sampling is disabled, the discrepancy becomes measurable at the residual-correction level. Second, in our experiments this discrepancy does not undermine the empirical performance of RegressionMSR. The released correction often performs slightly better than the all-row correction computed with the same stored denominators, although this should be interpreted as a finite-sample empirical result rather than as a theoretical unbiasedness guarantee.

A.3 Experiment 2 extended results

Table 10: Full reproduction of the original Shapley benchmark (Table 1 of Witter et al. (2025)) at $m = 40n$, $k = 10$ folds, and 10 explicands per dataset. Gold, silver, bronze indicate 1st, 2nd, 3rd lowest error per dataset.

		Adult	Forest Fires	Real Estate	Bike Sharing	Breast Cancer	Independent	NHANES	Communities	Overall
LinearMSR										
	Mean	8.87×10^{-4}	6.35×10^{-5}	5.82×10^{-8}	1.66×10^{-3}	1.06×10^{-3}	1.04×10^{-4}	1.09×10^{-5}	6.48×10^{-5}	4.82×10^{-4}
	1st Quartile	4.62×10^{-4}	2.36×10^{-5}	3.44×10^{-9}	4.39×10^{-4}	8.88×10^{-5}	3.74×10^{-5}	1.20×10^{-7}	4.00×10^{-6}	5.50×10^{-6}
	2nd Quartile	7.77×10^{-4}	3.51×10^{-5}	4.65×10^{-9}	9.95×10^{-4}	1.12×10^{-3}	7.48×10^{-5}	2.84×10^{-6}	5.15×10^{-5}	7.36×10^{-5}
	3rd Quartile	1.29×10^{-3}	1.08×10^{-4}	1.77×10^{-8}	2.62×10^{-3}	1.85×10^{-3}	1.04×10^{-4}	1.34×10^{-5}	1.12×10^{-4}	4.62×10^{-4}
TreeMSR										
	Mean	6.62×10^{-5}	1.09×10^{-4}	2.89×10^{-7}	1.09×10^{-3}	8.86×10^{-4}	1.49×10^{-4}	1.45×10^{-7}	6.23×10^{-5}	2.95×10^{-4}
	1st Quartile	2.45×10^{-5}	1.19×10^{-5}	7.22×10^{-9}	3.20×10^{-4}	5.23×10^{-4}	3.43×10^{-5}	5.25×10^{-10}	2.85×10^{-5}	2.52×10^{-6}
	2nd Quartile	3.94×10^{-5}	6.52×10^{-5}	1.27×10^{-8}	6.91×10^{-4}	5.96×10^{-4}	1.41×10^{-4}	2.44×10^{-9}	4.75×10^{-5}	4.83×10^{-5}
	3rd Quartile	6.43×10^{-5}	1.80×10^{-4}	8.76×10^{-8}	1.59×10^{-3}	1.21×10^{-3}	1.94×10^{-4}	1.45×10^{-8}	5.20×10^{-5}	2.54×10^{-4}
KernelSHAP										
	Mean	2.74×10^{-2}	6.43×10^{-5}	2.08×10^{-8}	2.75×10^{-3}	1.95×10^{-3}	1.69×10^{-4}	1.56×10^{-5}	9.64×10^{-5}	4.05×10^{-3}
	1st Quartile	5.66×10^{-4}	2.31×10^{-5}	4.64×10^{-11}	9.92×10^{-4}	3.49×10^{-4}	7.57×10^{-5}	1.44×10^{-7}	6.82×10^{-6}	9.74×10^{-6}
	2nd Quartile	7.43×10^{-4}	3.97×10^{-5}	1.20×10^{-9}	1.39×10^{-3}	1.72×10^{-3}	1.51×10^{-4}	3.83×10^{-6}	6.57×10^{-5}	1.16×10^{-4}
	3rd Quartile	2.09×10^{-3}	1.10×10^{-4}	5.04×10^{-9}	3.61×10^{-3}	3.47×10^{-3}	2.37×10^{-4}	2.39×10^{-5}	1.66×10^{-4}	6.29×10^{-4}
PermutationSHAP										
	Mean	4.29×10^{-3}	1.47×10^{-4}	6.10×10^{-8}	3.77×10^{-3}	1.35×10^{-3}	2.19×10^{-4}	1.24×10^{-5}	8.90×10^{-5}	1.23×10^{-3}
	1st Quartile	1.19×10^{-3}	4.92×10^{-5}	9.47×10^{-11}	1.40×10^{-3}	9.29×10^{-5}	6.12×10^{-5}	9.11×10^{-8}	4.71×10^{-6}	5.11×10^{-6}
	2nd Quartile	2.01×10^{-3}	8.87×10^{-5}	2.49×10^{-9}	2.17×10^{-3}	9.83×10^{-4}	8.67×10^{-5}	2.11×10^{-6}	6.67×10^{-5}	8.70×10^{-5}
	3rd Quartile	7.42×10^{-3}	1.42×10^{-4}	1.92×10^{-8}	5.37×10^{-3}	2.26×10^{-3}	2.03×10^{-4}	1.95×10^{-5}	1.34×10^{-4}	1.30×10^{-3}
LeverageSHAP										
	Mean	9.14×10^{-4}	5.88×10^{-5}	9.35×10^{-9}	1.84×10^{-3}	1.18×10^{-3}	9.29×10^{-5}	7.96×10^{-6}	7.11×10^{-5}	5.21×10^{-4}
	1st Quartile	4.42×10^{-4}	1.75×10^{-5}	4.88×10^{-11}	6.16×10^{-4}	1.04×10^{-4}	4.14×10^{-5}	8.39×10^{-8}	3.56×10^{-6}	4.31×10^{-6}
	2nd Quartile	7.69×10^{-4}	4.39×10^{-5}	1.01×10^{-9}	1.46×10^{-3}	1.03×10^{-3}	6.35×10^{-5}	2.66×10^{-6}	5.73×10^{-5}	7.13×10^{-5}
	3rd Quartile	9.71×10^{-4}	8.71×10^{-5}	1.92×10^{-9}	3.03×10^{-3}	1.32×10^{-3}	1.11×10^{-4}	1.27×10^{-5}	1.14×10^{-4}	5.06×10^{-4}

A.4 Experiment 3 extended results

Figures 2–6 show the full error and Spearman correlation grids for all probabilistic values and noise models. Tables 11–13 report the representative-value summaries used to construct Table 7, while tables 14–16 give the corresponding Spearman correlations.

Table 11: Normalized squared ℓ_2 error under noisy value queries. Averaged over all datasets with 10 explicands per set, $k = 10$ folds, and $m = 40n$. Each entry reports non-TreeMSR error/TreeMSR error approximating **Shapley values**. Gold, silver, bronze indicate 1st, 2nd, 3rd lowest error per column.

	Gaussian					Laplace					Heteroskedastic				
Estimator	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1
LinearMSR	2.03	2.50	1.29	2.26	1.93	1.92	2.07	2.43	1.47	1.96	1.81	2.07	1.56	1.99	2.79
TreeMSR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
LinearMC	538.06	103.41	3.17	7.13	3.06	179.91	102.94	6.38	2.46	8.36	160.01	157.78	3.10	3.57	1.85
TreeMC	1.86	1.86	1.70	1.91	1.21	1.82	1.82	2.69	1.13	1.53	1.69	2.46	0.76	1.40	0.84
WeightedMC	1054.62	447.90	6.56	3.08	1.92	893.10	407.90	10.30	1.89	2.48	872.27	385.22	3.49	1.60	1.02
OFA	157.60	78.61	2.18	3.28	1.84	152.23	70.65	3.66	1.07	2.13	139.67	60.39	1.88	1.02	0.61
WSL	32.16	49.29	8.20	12.65	5.66	130.67	256.59	9.83	7.12	6.45	1108.26	15.99	3.87	5.62	5.78
GELS (General)	877.49	390.34	5.87	1.93	0.95	894.95	335.15	8.54	1.12	1.03	808.28	274.99	3.18	1.65	0.75
ARM	867.21	315.66	3.58	2.85	1.74	569.04	221.01	7.52	1.37	1.18	649.86	162.96	2.54	1.03	0.89
WeightedSHAP	78.34	10.53	3.14	5.61	2.88	26.05	66.57	4.86	4.36	3.15	136.81	15.05	2.41	3.18	2.85
LeverageSHAP	1.84	2.34	2.38	2.97	2.39	2.01	3.06	2.03	2.18	1.64	1.62	2.15	1.46	1.85	1.29
KernelSHAP	16.59	11.19	2.58	3.04	0.99	8.14	4.78	3.65	1.73	1.55	9.21	2.69	1.88	1.22	1.29
PermutationSHAP	6.24	5.16	3.49	6.80	3.67	5.10	4.67	5.34	4.21	3.68	4.26	4.18	1.80	2.87	2.74
GELS (Shapley)	415.12	173.17	2.59	1.82	1.83	366.43	176.03	4.61	1.41	1.64	374.97	147.52	1.55	2.18	0.45

Table 12: Normalized squared ℓ_2 error under noisy value queries. Averaged over all datasets with 10 explicands per set, $k = 10$ folds, and $m = 40n$. Each entry reports non-TreeMSR error/TreeMSR error approximating **Banzhaf values**. Gold, silver, bronze indicate 1st, 2nd, 3rd lowest error per column.

	Gaussian					Laplace					Heteroskedastic				
Estimator	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1
LinearMSR	2.11	1.08	1.51	0.92	0.86	1.89	1.07	0.78	0.80	0.76	1.78	1.50	0.66	1.15	0.93
TreeMSR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
LinearMC	2078.02	1314.64	38.82	5.56	4.42	1521.60	1288.51	24.00	3.33	4.37	1539.12	1178.90	28.30	4.02	4.48
TreeMC	1.72	5.03	2.94	2.31	2.53	7.58	1.82	1.50	1.18	1.82	4.33	2.47	1.95	1.58	2.21
WeightedMC	1053.60	508.66	22.77	4.09	4.29	902.11	463.76	13.27	2.42	3.79	907.74	491.84	14.45	2.95	3.52
OFA	212.30	96.62	4.18	0.64	0.49	176.31	87.34	2.27	0.71	0.96	177.29	91.89	2.95	0.73	1.13
WSL	1270.63	419.06	84.35	35.07	29.18	786.48	453.10	39.67	30.30	40.14	1213.45	317.67	41.05	32.21	43.05
GELS	496.98	186.77	8.22	2.01	1.19	376.08	180.78	5.18	1.13	1.97	348.54	199.61	6.52	2.00	1.83
ARM	356.46	161.37	7.93	2.90	3.10	425.85	143.25	7.80	1.25	1.52	332.07	145.59	4.36	1.96	2.39
WeightedSHAP	741.01	214.67	30.40	16.01	14.20	426.59	354.92	24.28	13.09	19.21	590.88	428.58	22.20	17.37	19.90
AME	203.24	97.15	4.44	0.86	0.80	188.79	84.26	2.13	0.60	0.96	180.56	93.78	2.85	0.61	1.08
KernelBanzhaf	2.11	1.49	1.24	0.80	0.62	1.52	1.30	0.56	0.83	0.88	2.26	1.49	1.17	0.57	1.64

Table 13: Normalized squared ℓ_2 error under noisy value queries. Averaged over all datasets with 10 explicands per set, $k = 10$ folds, and $m = 40n$. Each entry reports non-TreeMSR error/TreeMSR error approximating **B(1,8) values**. Gold, silver, bronze indicate 1st, 2nd, 3rd lowest error per column.

	Gaussian					Laplace					Heteroskedastic				
Estimator	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1
LinearMSR	85.80	12.94	0.35	0.21	0.25	76.29	14.88	0.39	0.25	0.17	69.25	9.79	0.25	0.19	0.17
TreeMSR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
LinearMC	14.73	3.88	0.97	0.94	0.85	15.59	4.64	0.97	0.96	0.73	16.45	3.23	0.91	0.92	0.95
TreeMC	1.67	1.08	0.98	0.70	0.92	1.86	1.41	0.99	0.82	0.65	1.81	1.17	0.81	0.90	0.76
WeightedMC	1168.52	199.11	2.98	0.81	0.87	1181.15	244.73	2.77	0.98	0.56	1100.00	142.26	1.93	0.76	0.75
OFA	1954.17	400.10	12.57	5.96	5.13	2779.12	536.81	7.54	4.19	2.10	3903.40	441.22	7.85	5.62	5.01
WSL	662.28	123.45	13.89	9.64	10.77	782.84	132.00	10.15	10.69	7.49	756.54	86.54	11.54	12.51	11.56
GELS	433.46	74.62	1.95	0.72	1.26	490.39	88.03	1.71	1.33	0.49	450.14	59.05	2.49	0.55	0.81
ARM	367.19	67.89	1.70	0.78	0.76	350.51	71.63	1.56	0.83	0.51	329.52	47.52	0.92	0.81	0.69
WeightedSHAP	288.60	56.77	6.75	5.32	6.24	326.20	84.31	6.60	5.72	3.44	352.66	50.79	5.42	5.06	5.07

Table 14: Rank stability under noisy value queries. Averaged over all datasets with 10 explicands per set, $k = 10$ folds, and $m = 40n$. Each entry reports Spearman correlation in approximating **Shapley values**. Gold, silver, bronze indicate 1st, 2nd, 3rd highest Spearman correlation per column.

	Gaussian					Laplace					Heteroskedastic				
Estimator	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1
LinearMSR	0.83	0.79	0.66	0.53	0.40	0.83	0.79	0.66	0.53	0.46	0.83	0.78	0.65	0.48	0.42
TreeMSR	0.85	0.82	0.66	0.56	0.48	0.85	0.82	0.69	0.52	0.46	0.85	0.81	0.67	0.52	0.44
LinearMC	0.54	0.53	0.48	0.35	0.31	0.52	0.51	0.46	0.38	0.28	0.53	0.54	0.44	0.33	0.30
TreeMC	0.85	0.80	0.64	0.48	0.38	0.85	0.81	0.63	0.49	0.37	0.85	0.79	0.62	0.45	0.40
WeightedMC	0.83	0.75	0.58	0.41	0.33	0.83	0.76	0.59	0.41	0.34	0.83	0.74	0.55	0.40	0.33
OFA	0.49	0.51	0.46	0.41	0.29	0.48	0.50	0.48	0.38	0.33	0.49	0.45	0.44	0.36	0.29
WSL	0.83	0.77	0.57	0.40	0.35	0.84	0.74	0.57	0.39	0.32	0.83	0.76	0.54	0.41	0.32
GELS (General)	0.36	0.37	0.36	0.29	0.24	0.37	0.37	0.36	0.28	0.24	0.38	0.38	0.35	0.27	0.22
ARM	0.40	0.40	0.38	0.35	0.31	0.39	0.42	0.37	0.31	0.30	0.39	0.42	0.39	0.31	0.27
WeightedSHAP	0.84	0.78	0.61	0.41	0.36	0.84	0.78	0.63	0.43	0.35	0.85	0.78	0.59	0.42	0.38
LeverageSHAP	0.82	0.81	0.65	0.50	0.45	0.83	0.80	0.64	0.49	0.43	0.83	0.78	0.63	0.50	0.38
KernelSHAP	0.82	0.79	0.66	0.48	0.42	0.81	0.79	0.66	0.51	0.43	0.82	0.78	0.64	0.50	0.42
PermutationSHAP	0.85	0.79	0.62	0.44	0.40	0.85	0.78	0.60	0.46	0.37	0.85	0.78	0.61	0.43	0.37
GELS (Shapley)	0.42	0.47	0.41	0.37	0.26	0.43	0.42	0.40	0.36	0.32	0.44	0.42	0.40	0.35	0.29

Table 15: Rank stability under noisy value queries. Averaged over all datasets with 10 explicands per set, $k = 10$ folds, and $m = 40n$. Each entry reports Spearman correlation in approximating **Banzhaf values**. Gold, silver, bronze indicate 1st, 2nd, 3rd highest Spearman correlation per column.

	Gaussian					Laplace					Heteroskedastic				
Estimator	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1
LinearMSR	0.83	0.82	0.70	0.59	0.49	0.83	0.82	0.71	0.56	0.52	0.83	0.82	0.70	0.57	0.52
TreeMSR	0.85	0.83	0.70	0.56	0.51	0.85	0.83	0.67	0.56	0.45	0.85	0.83	0.71	0.58	0.48
LinearMC	0.58	0.57	0.49	0.36	0.31	0.55	0.58	0.48	0.36	0.32	0.57	0.60	0.46	0.36	0.31
TreeMC	0.85	0.81	0.66	0.49	0.44	0.84	0.82	0.66	0.51	0.43	0.84	0.81	0.64	0.50	0.44
WeightedMC	0.84	0.75	0.55	0.41	0.36	0.83	0.76	0.57	0.38	0.35	0.83	0.76	0.59	0.40	0.36
OFA	0.49	0.45	0.44	0.40	0.34	0.45	0.44	0.47	0.45	0.37	0.46	0.47	0.43	0.40	0.34
WSL	0.80	0.69	0.50	0.33	0.30	0.80	0.69	0.50	0.30	0.29	0.79	0.69	0.49	0.33	0.30
GELS	0.40	0.42	0.40	0.37	0.35	0.42	0.41	0.40	0.35	0.29	0.43	0.41	0.42	0.34	0.33
ARM	0.48	0.50	0.43	0.37	0.31	0.46	0.47	0.46	0.37	0.36	0.47	0.47	0.44	0.38	0.34
WeightedSHAP	0.83	0.74	0.54	0.36	0.30	0.83	0.75	0.55	0.39	0.36	0.83	0.75	0.55	0.38	0.29
AME	0.46	0.48	0.44	0.40	0.38	0.47	0.46	0.47	0.41	0.35	0.45	0.47	0.47	0.40	0.35
KernelBanzhaf	0.83	0.82	0.71	0.57	0.47	0.83	0.83	0.67	0.58	0.52	0.83	0.82	0.69	0.58	0.52

Table 16: Rank stability under noisy value queries. Averaged over all datasets with 10 explicands per set, $k = 10$ folds, and $m = 40n$. Each entry reports Spearman correlation in approximating **B(1,8) values**. Gold, silver, bronze indicate 1st, 2nd, 3rd highest Spearman correlation per column.

	Gaussian					Laplace					Heteroskedastic				
Estimator	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1	10^{-3}	10^{-2}	0.1	0.5	1
LinearMSR	0.72	0.71	0.61	0.54	0.41	0.70	0.71	0.60	0.50	0.43	0.71	0.71	0.61	0.50	0.41
TreeMSR	0.84	0.78	0.63	0.45	0.38	0.84	0.78	0.61	0.45	0.42	0.83	0.77	0.61	0.44	0.37
LinearMC	0.76	0.73	0.58	0.46	0.38	0.76	0.73	0.58	0.46	0.39	0.74	0.71	0.56	0.43	0.34
TreeMC	0.84	0.77	0.60	0.46	0.36	0.83	0.77	0.63	0.47	0.40	0.84	0.77	0.59	0.44	0.37
WeightedMC	0.82	0.74	0.56	0.40	0.34	0.82	0.73	0.57	0.39	0.31	0.82	0.71	0.55	0.39	0.32
OFA	0.46	0.46	0.38	0.30	0.25	0.45	0.45	0.37	0.30	0.24	0.45	0.44	0.35	0.24	0.25
WSL	0.79	0.67	0.48	0.35	0.28	0.79	0.68	0.47	0.28	0.28	0.78	0.67	0.45	0.31	0.26
GELS	0.43	0.46	0.42	0.34	0.29	0.40	0.45	0.38	0.31	0.26	0.43	0.44	0.37	0.29	0.30
ARM	0.42	0.43	0.39	0.36	0.26	0.44	0.44	0.42	0.32	0.23	0.43	0.42	0.37	0.35	0.27
WeightedSHAP	0.82	0.70	0.52	0.38	0.33	0.81	0.71	0.52	0.38	0.33	0.80	0.71	0.49	0.34	0.32

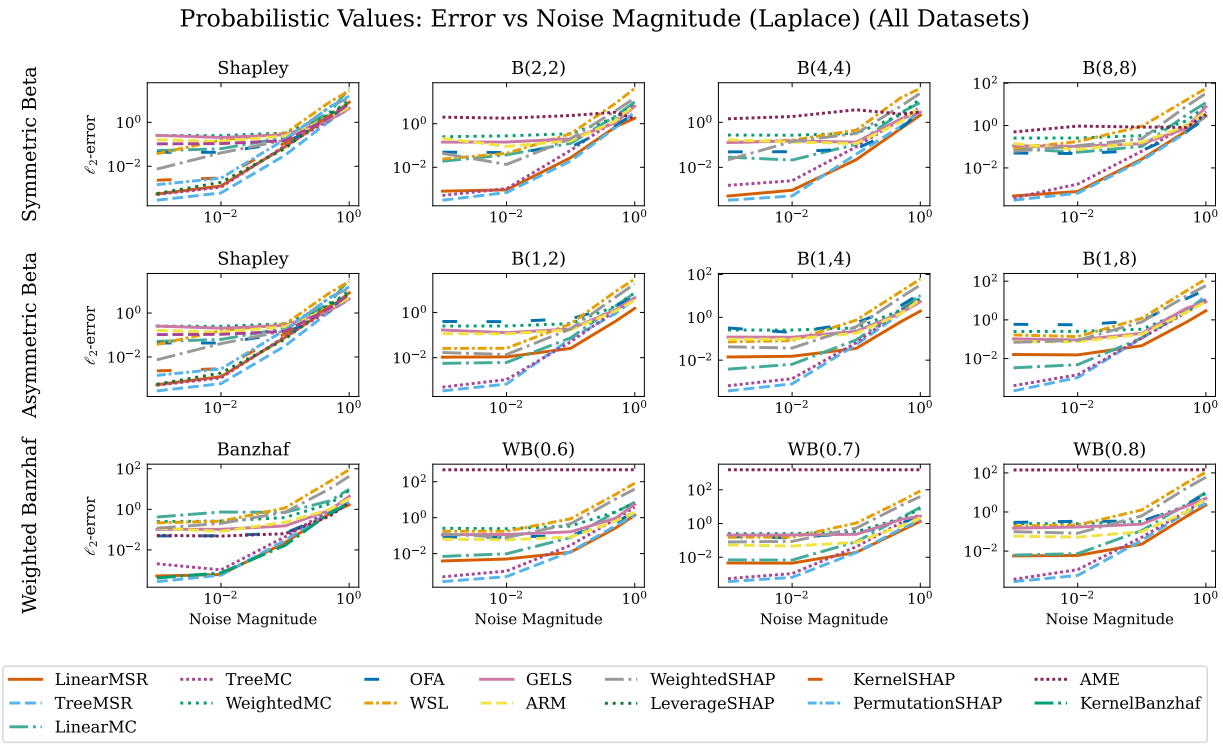

Figure 2: Error versus Laplacian noise magnitude for probabilistic-value estimation. $m = 40n$. $k = 10$ folds. Each panel corresponds to one probabilistic value, and curves are averaged over 10 explicands per dataset. Lower is better.

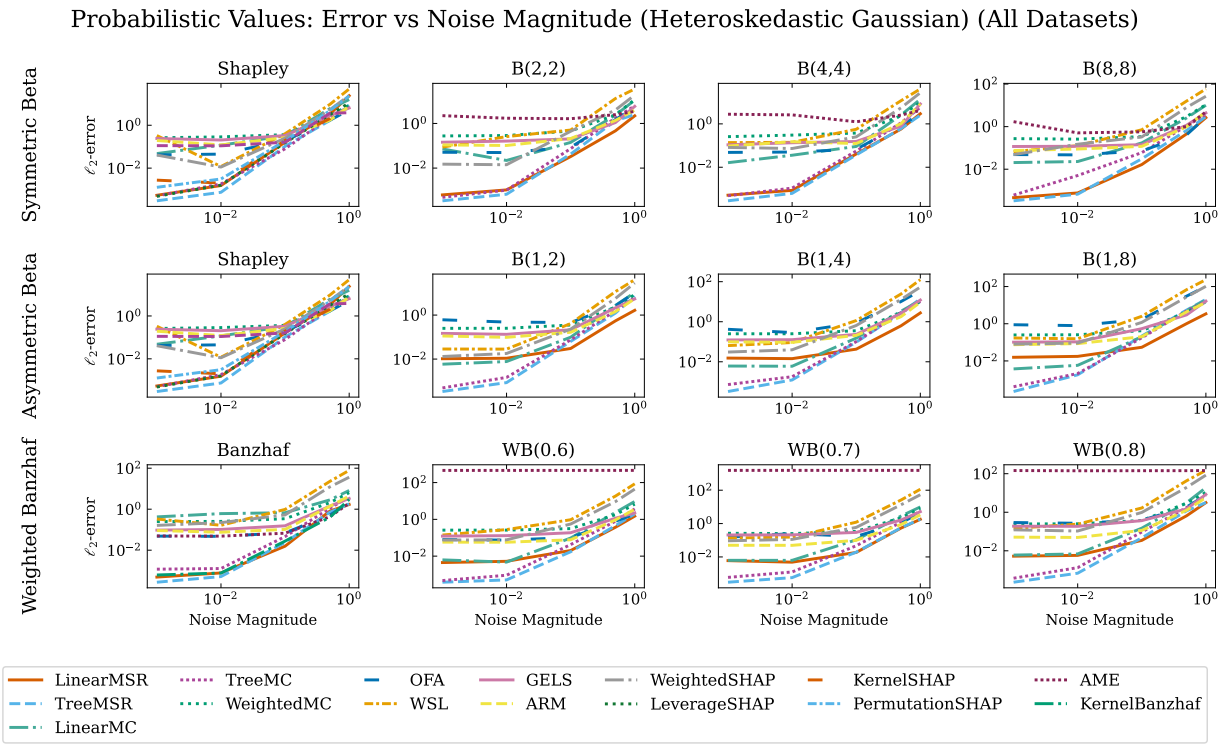

Figure 3: Error versus coalition-size-dependent heteroskedastic Gaussian noise magnitude for probabilistic-value estimation. $m = 40n$. $k = 10$ folds. Each panel corresponds to one probabilistic value, and curves are averaged over 10 explicands per dataset. Lower is better.

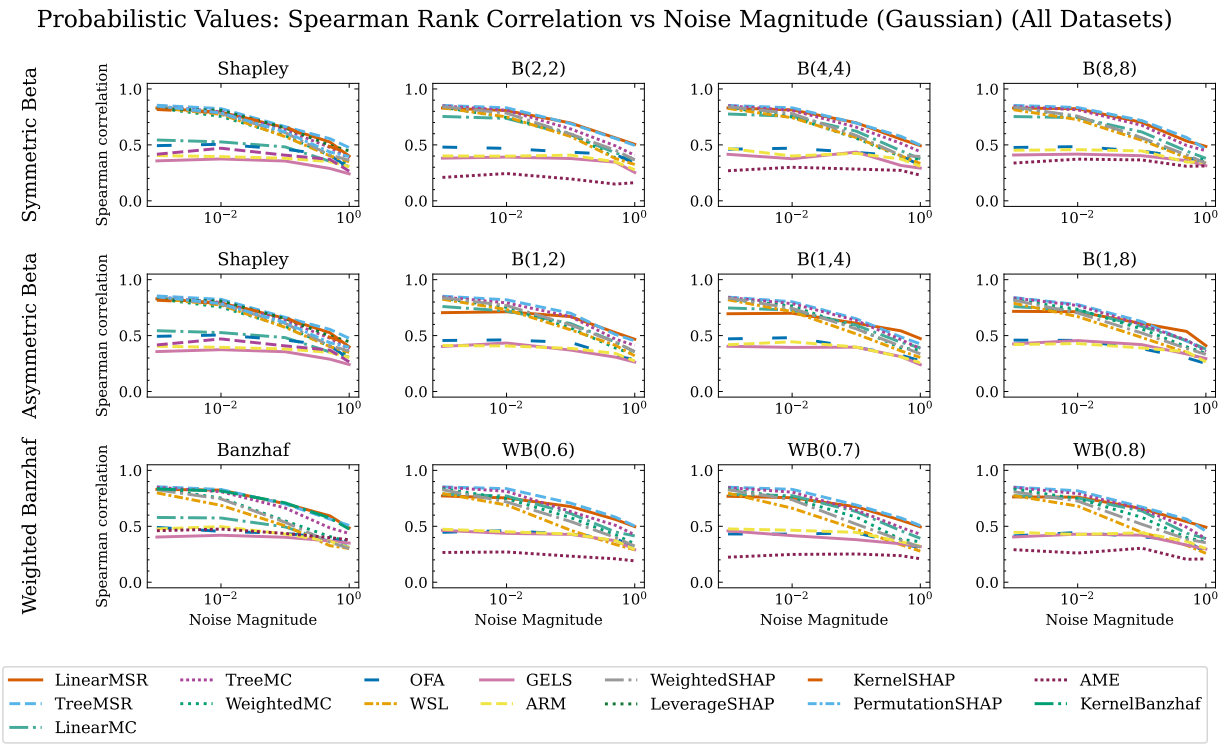

Figure 4: Spearman rank correlation versus Gaussian noise magnitude for probabilistic-value estimation. $m = 40n$. $k = 10$ folds. Each panel corresponds to one probabilistic value, and curves are averaged over 10 explicands per dataset. Higher is better.

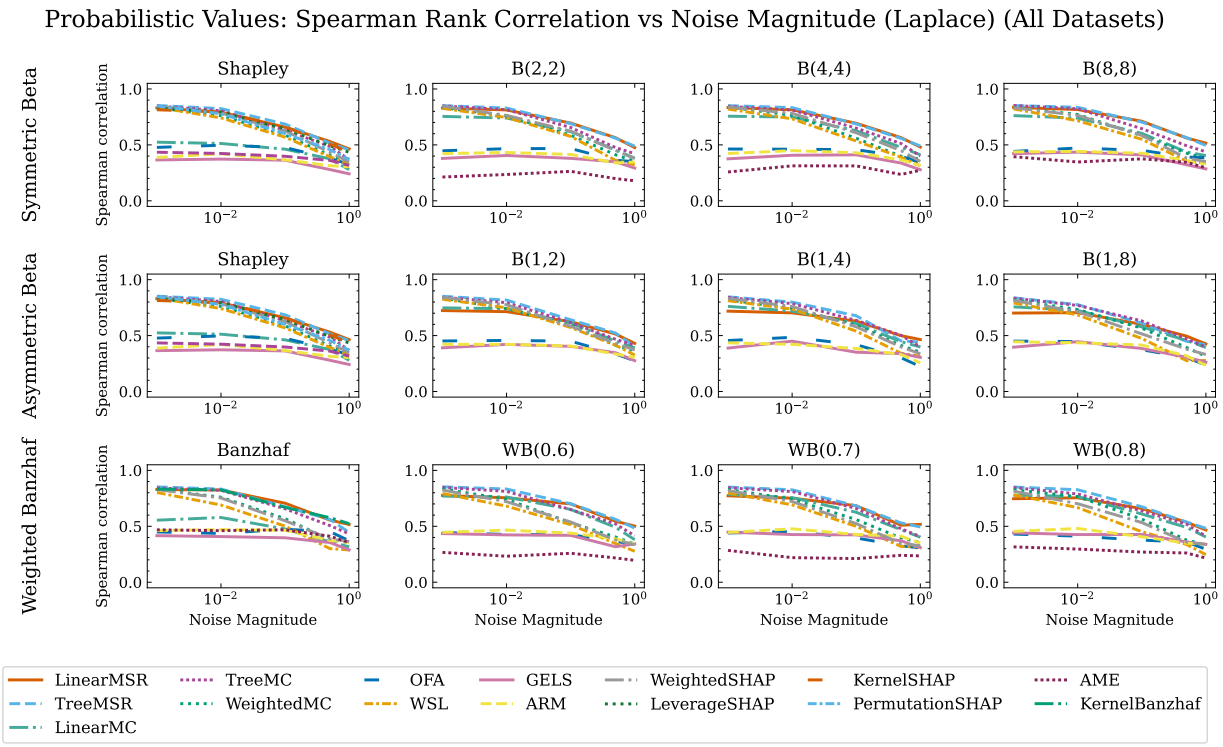

Figure 5: Spearman rank correlation versus Laplacian noise magnitude for probabilistic-value estimation. $m = 40n$. $k = 10$ folds. Each panel corresponds to one probabilistic value, and curves are averaged over 10 explicands per dataset. Higher is better.

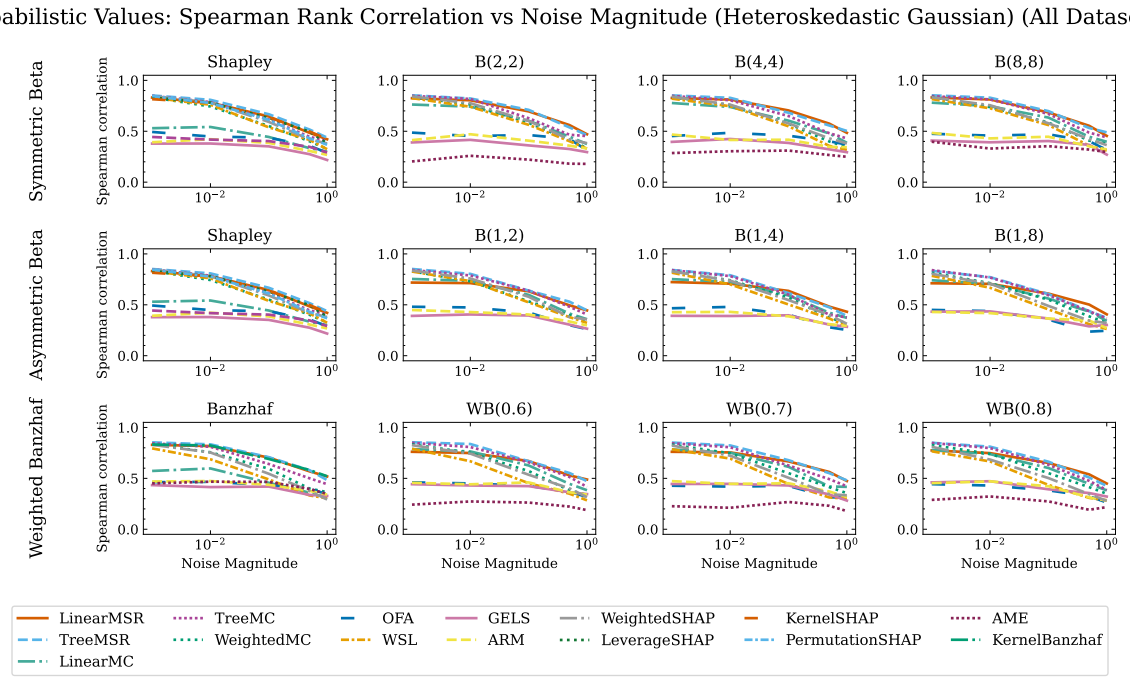

Figure 6: Spearman rank correlation versus coalition-size-dependent heteroskedastic Gaussian noise magnitude for probabilistic-value estimation. $m = 40n$. $k = 10$ folds. Each panel corresponds to one probabilistic value, and curves are averaged over 10 explicands per dataset. Higher is better.

A.5 Datasets

For completeness, we report the datasets used in our reproduction. Since we follow the original experimental protocol, this table is adapted from Appendix G of Witter et al. (2025).

Table 17: Datasets used in our experiments. We use the same eight datasets as Witter et al. (2025). Source, access-method, and license information is reproduced from their Appendix G.

Dataset	n	Access method	License
Adult	12	`shap.datasets`	CC-BY 4.0
Forest Fires	13	UCI ML Repository	CC-BY 4.0
Real Estate	15	UCI ML Repository	CC-BY 4.0
Bike Sharing	16	OpenML	Public Domain
Breast Cancer	30	`sklearn.datasets`	CC-BY 4.0
Independent	60	`shap.datasets`	MIT
NHANES	79	`shap.datasets`	Public Domain
Communities	101	`shap.datasets`	CC-BY 4.0