POLYSHAP: EXTENDING KERNELSHAP WITH INTERACTION-INFORMED POLYNOMIAL REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Shapley values have emerged as a central game-theoretic tool in explainable AI (XAI). However, computing Shapley values exactly requires 2^d game evaluations for a model with d features. Lundberg and Lee's KernelSHAP algorithm has emerged as a leading method for avoiding this exponential cost. KernelSHAP approximates Shapley values by approximating the game as a linear function, which is fit using a small number of game evaluations for random feature subsets.

In this work, we extend KernelSHAP by approximating the game via higher degree polynomials, which capture non-linear interactions between features. Our resulting PolySHAP method yields empirically better Shapley value estimates for various benchmark datasets, and we prove that these estimates are consistent.

Moreover, we connect our approach to *paired sampling* (antithetic sampling), a ubiquitous modification to KernelSHAP that improves empirical accuracy. We prove that paired sampling outputs *exactly* the same Shapley value approximations as second-order PolySHAP, *without ever fitting a degree 2 polynomial.* To the best of our knowledge, this finding provides the first strong theoretical justification for the excellent practical performance of the paired sampling heuristic.

1 INTRODUCTION

Understanding the contribution of individual features to a model's prediction is a central goal in explainable artificial intelligence (XAI) (Covert & Lee, 2021). Among the most influential approaches are those grounded in cooperative game theory, where the *Shapley value* (Shapley, 1953) provides a principled way to distribute a model's output to its d inputs.

The intuition behind the use of Shapley values is to attribute larger values to the players of a cooperative game with the most effect on the game's value. In XAI applications, players are typically features or training data points and the game value is typically a prediction or model loss.

Formally, we represent a cooperative game involving players $D = \{1, \dots, d\}$ via a value function $\nu : 2^D \to \mathbb{R}$ that maps subsets of players to values (2^D denotes the powerset of D). Shapley values are then defined[1] via the best linear approximation to the game ν. Concretely, for a subset $S \subseteq D$, $\nu(S)$ is approximated by a linear function in the binary features $\mathbb{1}[i \in S]$ for $i \in D$. The Shapley values are the coefficients of the linear approximation minimizing a specific weighted ℓ_2 loss:

$$\boldsymbol{\phi}^{\text{SV}}[\nu] := \underset{\boldsymbol{\phi} \in \mathbb{R}^d : \langle \boldsymbol{\phi}, \mathbf{1} \rangle = \nu(D)}{\arg\min} \sum_{S \subseteq D} \mu(S) \left(\nu(S) - \sum_{i=1}^{d} \phi_i \mathbb{1}[i \in S] \right)^2,$$

where the non-negative Shapley weights $\mu(S)$ are given in Equation (2). The constraint that the Shapley values sum to $\nu(D)$ enforces what is known as the "efficiency property", one of four axiomatic properties that motivate the original definition of Shapley values (see e.g, Molnar (2024)).

Since the sum above involves 2^d terms, exact minimization of the linear approximation to obtain $\boldsymbol{\phi}^{\text{SV}}[\nu]$ is infeasible for most practical games. Over the past several years, substantial research has focused on making the computation of Shapley values feasible in practice (Covert et al., 2020; Covert

[1] Without loss of generality, we assume $\nu(\emptyset) = 0$. Otherwise, we could consider the centered game $\nu(S) - \nu(\emptyset)$ which has the same Shapley values.

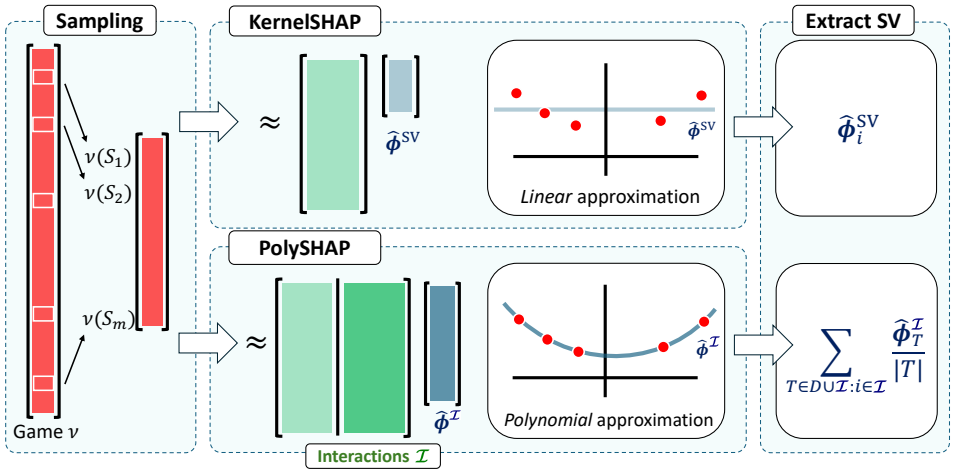


Figure 1: Both KernelSHAP and PolySHAP fit a function to approximate a sample of game evaluations. While KernelSHAP uses a linear approximation, PolySHAP uses a more expressive polynomial approximation. Finally, both algorithms return the Shapley values (SV) of their respective approximations (trivial for KernelSHAP, see Theorem 4.3 for PolySHAP).

& Lee, 2021; Mitchell et al., 2022; Musco & Witter, 2025; Witter et al., 2025), with *KernelSHAP* (Lundberg & Lee, 2017) emerging as one of the most widely used model-agnostic methods.

From the least squares definition of Shapley values, KernelSHAP can be viewed as a two step process: First, approximate the game ν with a linear function fit from a sample of game evaluations $\nu(S)$ on randomly selected subsets S. Second, return the Shapley values of the approximation, which, for linear functions, are simply the coefficients of each input.

A natural idea is to adapt this framework to fit ν with a richer function class that still admits fast Shapley value computation. One such class is tree-based models like XGBoost, which Witter et al. (2025) recently leveraged to approximate the game. When the tree-based approximation is accurate, their Regression MSR estimator produces more accurate Shapley value estimates than KernelSHAP. Butler et al. (2025) also use tree-based models to learn an approximation to the game, but extract Fourier coefficients which can be used to estimate more general attribution values. In the small sample regime with budgets less than $5n$, their Proxy SPEX estimator outperforms Kernel SHAP but achieves comparable performance to KernelSHAP for higher budgets.

In this work, we introduce an alterative approach called PolySHAP, where we approximate ν via a higher degree polynomial in the features $\mathbb{1}[i \in S]$ for $i \in D$, illustrated in Figure 1. For a degree k polynomial, let $d' = O(d^k)$ be the number of terms. We show that, after fitting an approximation with m samples, we can recover the Shapley values of the approximation in just $O(dd')$ time. Across various experiments, we find that higher degree PolySHAP approximations result in more accurate Shapley value estimates (see e.g., Figure 2). Moreover, we prove that the PolySHAP estimates are consistent, concretely that we obtain the Shapley values exactly as m goes to 2^d. This is in contrast to RegressionMSR, which needs an additional "regression adjustment" step to obtain a consistent estimator for tree-based approximations (Witter et al., 2025).

As a second main contribution of our work, we provide theoretical grounding for a seemingly unrelated sampling strategy called *paired sampling*, which is known to significantly improve the accuracy of KernelSHAP estimates (Covert & Lee, 2021; Mitchell et al., 2022; Olsen et al., 2024). In paired sampling, subsets are sampled in paired complements S and $D \setminus S$.While used in all state-of-the-art Shapley value estimators, the reason for paired sampling's superior performance is not well understood. Surprisingly, we prove that KernelSHAP with paired sampling outputs *exactly* the same Shapley value approximations as second-order PolySHAP *without ever fitting a degree 2 polynomial*. This theoretical finding generalizes a very recent result of Mayer & Wüthrich (2025), who showed that KernelSHAP with paired sampling exactly recovers Shapley values when the game has interactions of at most degree 2. Because the second-order PolySHAP will exactly fit a degree 2

game, their result follows immediately from a special case of ours. However, our finding is more general because it explains why paired sampling is effective for *all* games, not just those with at most degree 2 interactions.

Contributions. The main contributions of our work can be summarized as follows:

- We propose *PolySHAP*, an extension of KernelSHAP that models higher-order interaction terms to approximate ν, and prove it returns the Shapley values as the number of samples m goes to 2^d (Theorem 4.3). Moreover, we empirically show that PolySHAP results in more accurate Shapley value estimates than KernelSHAP and Permutation sampling.
- We establish a theoretical equivalence between paired KernelSHAP and second-order PolySHAP (Theorem 5.1), thereby explaining the practical benefits of paired sampling.

2 Related Work

KernelSHAP Sampling Strategies. Prior work on improving KernelSHAP has focused on refining the subset sampling procedure, aiming to reduce variance and improve computational efficiency (Kelodjou et al., 2024; Olsen & Jullum, 2024; Musco & Witter, 2025). Among these enhancements, paired sampling produces the largest improvement in accuracy Covert & Lee (2021), yet, until the present work, it was not understood why beyond limited special cases. Another notable enhancement is in the sampling distribution. While it is intuitive to sample subsets proportional to their Shapley weights (Equation 2), it turns out that sampling proportional to the *leverage scores* can be more effective (Musco & Witter, 2025). Paired sampling has also been observed to improve LeverageSHAP (KernelSHAP with leverage score sampling).

Other Shapley Value Estimators. Beyond the regression-based approach of KernelSHAP, prior Shapley value estimators are generally based on direct Monte Carlo approximation Kwon & Zou (2022a); Castro et al. (2009); Kwon & Zou (2022b); Kolpaczki et al. (2024); Li & Yu (2024). These methods estimate the ith Shapley value based on the following equivalent definition:

$$\phi_i^{\text{SV}}[\nu] = \frac{1}{d} \sum_{S \subseteq D \setminus \{i\}} \frac{\nu(S \cup \{i\}) - \nu(S)}{\binom{d-1}{|S|}}. \tag{1}$$

Permutation sampling, where subsets are sampled from a permutation, is a particularly effective approach (Castro et al., 2009; Mitchell et al., 2022). However, in direct Monte Carlo methods, each game evaluation is used to estimate at most two Shapley values. MSR methods reuse game evaluations in the estimate of every Shapley value, but at the cost of higher variance (Li & Yu, 2024; Witter et al., 2025). A recent benchmark finds that RegressionMSR with tree-based approximations, LeverageSHAP, KernelSHAP, and Permutation sampling are the most accurate (Witter et al., 2025).

Higher-Order Explanations. Another line of work seeks to improve approximations by explicitly modeling higher-order interactions. k_{ADD}-SHAP (Pelegrina et al., 2023) solves a least-squares problem over all interactions up to order k, and converges to the Shapley value for $k = 2$ and $k = 3$ (Pelegrina et al., 2025). With our results, we simplify k_{ADD}-SHAP and prove general convergence, where the practical differences are discussed in Appendix A.4. Relatedly, Mohammadi et al. (2025) propose a regularized least squares method based on the Möbius transform (Rota, 1964), which converges only when all higher-order interactions are included. By contrast, PolySHAP converges for any chosen set of interaction terms. Beyond approximation of the Shapley value, Kang et al. (2024) leverage the Fourier representation of games to detect and quantify higher-order interactions.

3 Preliminaries on Explainable AI and Cooperative Games

Notation. We use boldface letters to denote vectors, e.g., $\boldsymbol{x}$, with entries x_i, and the corresponding random variable $\tilde{\boldsymbol{x}}$. The all-one vector is denoted by $\mathbf{1}$, and $\langle \cdot, \cdot \rangle$ is the standard inner product.

Given the prediction of a machine learning model $f : \mathbb{R}^d \to \mathbb{R}$, post-hoc feature-based explanations aim to quantify the contribution of features D to the model output. Such explanations are defined by (i) the choice of an explanation game $\nu : 2^D \to \mathbb{R}$ and (ii) a game-theoretic attribution measure, such as the Shapley value (Covert et al., 2021). For a given instance $\boldsymbol{x} \in \mathbb{R}^d$, the *local explanation*

game $\nu_{\boldsymbol{x}}$ describes the model's prediction when restricted to subsets of features, with the remaining features replaced through perturbation. The perturbation is carried out using different imputation strategies, as summarized in Table 1.

Table 1: Local explanation games $\nu_{\boldsymbol{x}}$ for instance $\boldsymbol{x}$.

Method	**Game**	$\nu_{\boldsymbol{x}}(S)$	$\nu_{\boldsymbol{x}}(\emptyset)$
Baseline	$\nu_{\boldsymbol{x}}^{(b)}$	$f(\boldsymbol{x}_S, \boldsymbol{b}_{D\setminus S})$	$f(\boldsymbol{b})$
Marginal	$\nu_{\boldsymbol{x}}^{(m)}$	$\mathbb{E}[f(\boldsymbol{x}_S, \tilde{\boldsymbol{x}}_{D\setminus S})]$	$\mathbb{E}[f(\tilde{\boldsymbol{x}})]$
Conditional	$\nu_{\boldsymbol{x}}^{(c)}$	$\mathbb{E}[f(\tilde{\boldsymbol{x}}) \mid \tilde{\boldsymbol{x}}_S = \boldsymbol{x}_S]$	$\mathbb{E}[f(\tilde{\boldsymbol{x}})]$

Similarly, *global explanation games* are constructed from $\nu_{\boldsymbol{x}}$ by evaluating measures such as variance or risk (Fumagalli et al., 2025). Beyond analyzing features, other variants have been proposed, for instance to characterize properties of individual data points (Ghorbani & Zou, 2019).

Like most Shapley value estimators (except e.g., TreeSHAP (Lundberg et al., 2018)), PolySHAP is agnostic to how the game ν is defined.

KernelSHAP. Given a budget of m game evaluations, KernelSHAP solves the approximate least least squares problem:

$$\hat{\boldsymbol{\phi}}^{\text{SV}}[\nu] := \underset{\boldsymbol{\phi}\in\mathbb{R}^d:\langle\boldsymbol{\phi},\mathbf{1}\rangle=\nu(D)}{\arg\min} \sum_{\ell=1}^{m} \frac{\mu(S_\ell)}{p(S_\ell)} \left(\nu(S_\ell) - \sum_{i=1}^{d} \phi_i \mathbb{1}[i \in S_\ell]\right)^2 \quad \text{with } S_1, \ldots, S_m \sim p$$

where the Shapley weight, for subset $S \subseteq D$ is given by

$$\mu(S) := \frac{1}{\binom{d-2}{|S|-1}} \quad \text{if } 0 < |S| < d \quad \text{and 0 otherwise.} \tag{2}$$

While effective, KernelSHAP is inherently limited to a linear (additive) approximation of ν based on the sampled coalitions.

4 Interaction-Informed Approximation of Shapley Values

4.1 PolySHAP Interaction Representation

We introduce PolySHAP, a method for producing Shapley value estimates from a polynomial approximation of ν. Let the **interaction frontier** $\mathcal{I}$ be a subset of interaction terms

$$\mathcal{I} \subseteq \{T \subseteq D : |T| \geq 2\}.$$

We then extend the linear approximation of ν by defining an *interaction-based polynomial representation* restricted to interactions in $\mathcal{I}$.

Definition 4.1. *The **PolySHAP representation** $\boldsymbol{\phi}^{\mathcal{I}} \in \mathbb{R}^{d'}$ with $d' = d + |\mathcal{I}|$ is given by*

$$\boldsymbol{\phi}^{\mathcal{I}}[\nu] := \underset{\boldsymbol{\phi}\in\mathbb{R}^{d'}:\langle\boldsymbol{\phi},\mathbf{1}\rangle=\nu(D)}{\arg\min} \sum_{S\subseteq D} \mu(S) \left(\nu(S) - \sum_{T\in D\cup\mathcal{I}} \phi_T \prod_{j\in T} \mathbb{1}[j \in S]\right)^2.$$

Here, and in the following we abuse notation with $\phi_i := \phi_{\{i\}}$ and $\mathbb{1}[j \in i] := \mathbb{1}[j = i]$ for $i, j \in D$.

The PolySHAP representation generalizes the least squares formulation of the Shapley value to arbitrary interaction frontiers $\mathcal{I}$. For each interaction set $T \in \mathcal{I}$, the approximation contributes a coefficient ϕ_T only if all features in T are present in S.

Remark 4.2. *The PolySHAP representation directly extends the Faithful Shapley interaction index (Tsai et al., 2023) to arbitrary interaction frontiers.*

In the theorem below, we show how to recover Shapley values from the PolySHAP representation.

Theorem 4.3. *The Shapley values of ν are recovered from the PolySHAP representation as*

$$\phi_i^{SV}[\nu] = \phi_i^{\mathcal{I}} + \sum_{S\in\mathcal{I}:i\in S} \frac{\phi_S^{\mathcal{I}}}{|S|} \quad \text{for } i \in D. \tag{3}$$

In other words, consistent estimation of the PolySHAP representation directly implies consistent estimation of the Shapley value.

4.2 POLYSHAP ALGORITHM

A natural approximation strategy is to first estimate the PolySHAP representation and then map the result back to Shapley values using Theorem 4.3. Concretely, we approximate the PolySHAP representation by solving

$$\hat{\boldsymbol{\phi}}^{\mathcal{I}}[\nu] := \underset{\boldsymbol{\phi}\in\mathbb{R}^{d+|\mathcal{I}|}:\langle\boldsymbol{\phi},\mathbf{1}\rangle=\nu(D)}{\arg\min} \sum_{\ell=1}^{m} \frac{\mu(S_\ell)}{p(S_\ell)} \left(\nu(S_\ell) - \sum_{T\in D\cup\mathcal{I}} \phi_T \prod_{j\in T} \mathbb{1}[j\in S]\right)^2 \tag{4}$$

with m samples $S_1, \ldots, S_m$ drawn from some distribution p, where $d' < m \leq 2^d$. (When ν is clear from context, we write $\hat{\boldsymbol{\phi}}^{\mathcal{I}}$ for $\hat{\boldsymbol{\phi}}^{\mathcal{I}}[\nu]$.)

We then convert $\hat{\boldsymbol{\phi}}^{\mathcal{I}}$ into Shapley value estimates via Theorem 4.3. The rationale behind this is approach is that the more expressive PolySHAP representation more accurately represents ν, which in turn yields more accurate Shapley value estimates. We refer to this interaction-aware extension of KernelSHAP as *PolySHAP*.

In order to produce the PolySHAP solution in practice, we use the matrix representation of the regression problem. Define the sampled design matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{m\times d'}$ and the sampled target vector $\tilde{\mathbf{y}} \in \mathbb{R}^m$. The rows are indexed by $\ell \in [m]$, and the columns of $\tilde{\mathbf{X}}$ are indexed by interactions $T \in D \cup \mathcal{I}$. The entries of the sampled design matrix and sampled target vector are given by

$$[\tilde{\mathbf{X}}]_{\ell,T} = \sqrt{\frac{\mu(S_\ell)}{p(S_\ell)}} \cdot \mathbb{1}[T \subseteq S_\ell] \quad \text{and} \quad [\tilde{\mathbf{y}}]_\ell = \sqrt{\frac{\mu(S_\ell)}{p(S_\ell)}} \cdot \nu(S_\ell). \tag{5}$$

In this notation, we may write

$$\hat{\boldsymbol{\phi}}^{\mathcal{I}} = \underset{\boldsymbol{\phi}\in\mathbb{R}^{d'}:\langle\mathbf{1},\boldsymbol{\phi}\rangle=\nu(D)}{\arg\min} \|\tilde{\mathbf{X}}\boldsymbol{\phi} - \tilde{\mathbf{y}}\|_2^2. \tag{6}$$

We would like to apply standard regression tools when solving the problem, so we convert from the constrained problem to an unconstrained reformulation. Let $\mathbf{P}_{d'}$ be the matrix that projects *off* the all ones vector in d' dimensions i.e., $\mathbf{P}_{d'} = \mathbf{I} - \frac{1}{d'}\mathbf{1}_{d'}\mathbf{1}_{d'}^\top$. We have

$$\begin{aligned}\underset{\boldsymbol{\phi}\in\mathbb{R}^{d'}:\langle\mathbf{1},\boldsymbol{\phi}\rangle=\nu(D)}{\arg\min} \|\tilde{\mathbf{X}}\boldsymbol{\phi} - \tilde{\mathbf{y}}\|_2^2 &= \underset{\boldsymbol{\phi}\in\mathbb{R}^{d'}:\langle\boldsymbol{\phi},\mathbf{1}\rangle=0}{\arg\min} \left\|\tilde{\mathbf{X}}\boldsymbol{\phi} + \tilde{\mathbf{X}}\mathbf{1}\frac{\nu(D)}{d'} - \tilde{\mathbf{y}}\right\|_2^2 + \mathbf{1}\frac{\nu(D)}{d'} \\ &= \mathbf{P}_{d'} \underset{\boldsymbol{\phi}\in\mathbb{R}^{d'}}{\arg\min} \left\|\tilde{\mathbf{X}}\mathbf{P}_{d'}\boldsymbol{\phi} + \tilde{\mathbf{X}}\mathbf{1}\frac{\nu(D)}{d'} - \tilde{\mathbf{y}}\right\|_2^2 + \mathbf{1}\frac{\nu(D)}{d'}. \end{aligned} \tag{7}$$

PolySHAP is described in pseudocode in Algorithm 1.

Algorithm 1 PolySHAP

Require: game $\nu_{\mathbf{x}}$, interaction frontier $\mathcal{I}$, sampling distribution p, sampling budget $m > d'$.

1: Define $\nu(S) := \nu_{\mathbf{x}}(S) - \nu_{\mathbf{x}}(\emptyset)$ ▷ Center for notational simplicity
2: $\{S_\ell\}_{\ell=1}^m \leftarrow \text{SAMPLE}(m, p)$
3: Construct $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{y}}$ ▷ Equation (5)
4: $\hat{\boldsymbol{\phi}}^{\mathcal{I}} \leftarrow \text{SOLVELEASTSQUARES}(\tilde{\mathbf{X}}\mathbf{P}_{d'}, \tilde{\mathbf{y}} - \tilde{\mathbf{X}}\mathbf{1}\frac{\nu(D)}{d'}) + \mathbf{1}\frac{\nu(D)}{d'}$
5: $\hat{\boldsymbol{\phi}}^{\text{SV}} \leftarrow \text{POLYSHAPTOSV}(\hat{\phi}^{\mathcal{I}})$ ▷ Equation (3)
6: **return** $\nu_{\boldsymbol{x}}(\emptyset), \hat{\boldsymbol{\phi}}^{\text{SV}}$

Computational Complexity. The computational complexity of PolySHAP can be divided into two components: evaluating the game for the sampled coalitions, and solving the regression problem followed by extraction of the Shapley values. Evaluating the game requires at least one model call for local explanation games, and highly depends on the application setting. Solving the regression problem scales with $\mathcal{O}(m \cdot d'^2 + d'3)$, whereas transforming the PolySHAP representation to Shapley values is of order $\mathcal{O}(d \cdot d')$. Importantly, this complexity scales *linearly* with the budget m, and *quadratically* with the number of regression variables d'. In practice, the dominant factor in computational cost is usually the game evaluations, i.e., the model predictions. However, for smaller model architectures, the runtime can be influenced by the number of regression variables.

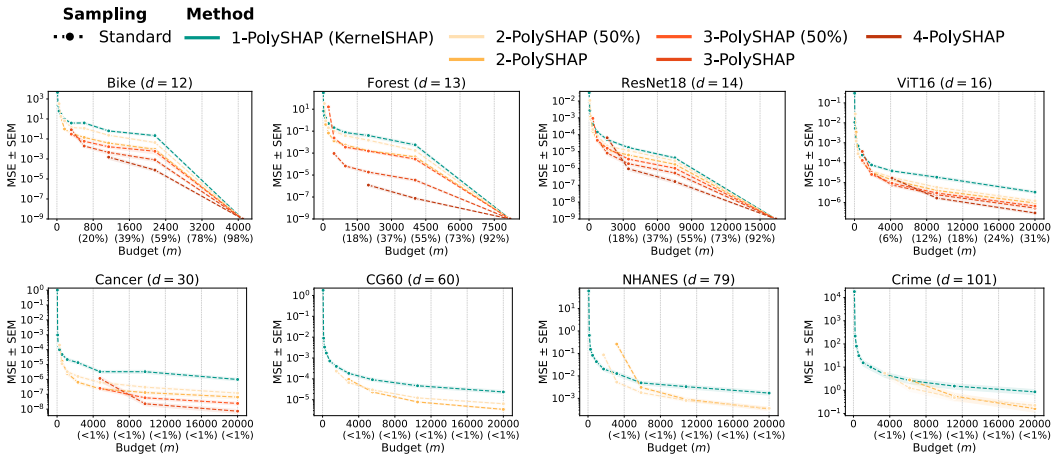


Figure 2: Approximation quality measured by MSE ($\pm$ SEM) for varyious sampling budgets m on different games. Adding any number of interactions in PolySHAP improves approximation quality.

4.3 Sampling Strategies for PolySHAP

PolySHAP uses a distribution p to sample m game evaluations for approximating the least squares objective. Previous work (Lundberg & Lee, 2017; Covert & Lee, 2021) chose p proportional to $\mu(S)$, which cancels the multiplicative correction term in Equation (4).

However, sampling proportionally to *leverage scores* offers improves estimation quality, and is supported by theoretical guarantees (Musco & Witter, 2025). Let $\mathbf{X} \in \mathbb{R}^{2^d \times d'}$ be the full deterministic matrix (each subset is sampled exactly once with probability 1). The leverage score for the row corresponding to subset S is given by

$$\ell_S = [\mathbf{X}\mathbf{P}_D]_S^\top \left(\mathbf{P}_D \mathbf{X}^\top \mathbf{X} \mathbf{P}_D\right)^\dagger [\mathbf{X}\mathbf{P}_D]_S \tag{8}$$

where $(\cdot)^\dagger$ denotes the pseudoinverse, and $[\mathbf{X}\mathbf{P}_D]_S$ is the Sth row of $\mathbf{X}\mathbf{P}_D$.

Theorem 4.4 (Leverage Score Sampling Guarantee (Musco & Witter, 2025))**.** *Let $\epsilon, \delta > 0$. When $m = O(d' \log \frac{d'}{\delta} + d' \frac{1}{\epsilon\delta})$ subsets are sampled proportionally to their leverage scores (with or without replacement and with or without paired sampling), the approximation $\hat{\phi}^{\mathcal{I}}$ satisfies, with probability $1 - \delta$,*

$$\sum_{S \subseteq D} \mu(S) \left(\nu(S) - \sum_{T \in D \cup \mathcal{I}} \tilde{\phi}_T^{\mathcal{I}} \prod_{j \in T} \mathbb{1}[j \in S] \right)^2 \leq \sum_{S \subseteq D} \mu(S) \left(\nu(S) - \sum_{T \in D \cup \mathcal{I}} \phi_T^{\mathcal{I}} \prod_{j \in T} \mathbb{1}[j \in S] \right)^2 .$$

Musco & Witter (2025) show that $\ell_S = 1/\binom{n}{|S|}$ for KernelSHAP, i.e., leverage score sampling is equivalent to sampling subsets uniformly by their size. For the k-additive interaction frontier, we can directly compute the leverage scores using symmetry and Equation (8), although a closed-form solution remains unknown. In practice, we observed little variation between leverage scores of order 1 and those of higher orders, which is why we recommend using order-1 leverage scores.

4.4 Construction of Interaction Frontiers $\mathcal{I}$

The interaction frontier $\mathcal{I}$ determines the number of additional variables (columns) in the linear regression problem. Its size must be balanced against the budget m (rows). Since lower-order interaction terms occur more frequently and are thus less sensitive to noise, it is natural to expand these terms first. To this end, we define the **k-additive interaction frontier** for $k = 2, \ldots, d$ as

$$\mathcal{I}_{\leq k} := \{S \subseteq D : 2 \leq |S| \leq k\} \quad \text{with } |\mathcal{I}_{\leq k}| = \sum_{i=2}^{k} \binom{d}{i}.$$

The k-additive interaction frontier includes all interactions up to order k by sequentially extending the $(k-1)$-additive interaction frontier with $\binom{n}{k}$ sets. It is widely used in Shapley-based interaction indices (Sundararajan et al., 2020; Tsai et al., 2023; Bordt & von Luxburg, 2023). In the following, we refer to PolySHAP using $\mathcal{I}_{\leq k}$ as **k-PolySHAP**.

Corollary 4.5. *The k-PolySHAP representation is equal to order-k Faith-SHAP (Tsai et al., 2023).*

A notable special case of k-PolySHAP is the interaction frontier without interactions: 1-PolySHAP, i.e., without interactions ($\mathcal{I} = \emptyset$), is equivalent to KernelSHAP.

We further show convergence for k_{ADD}-SHAP, extending Theorem 4.2 in (Pelegrina et al., 2025).

Proposition 4.6. *k_{ADD}-SHAP converges to the Shapley value for $k = 1, \ldots, d$.*

k_{ADD}-SHAP is linked to k-PolySHAP, but we recommend PolySHAP in practice, see Appendix A.4.

Partial Interaction Frontiers. In high dimensions, the k-additive interaction frontier grows combinatorially with $\binom{n}{k}$. With a limited evaluation budget m, including all interaction terms of a given order may yield an underdetermined least-squares system. To address this, we introduce the **partial interaction frontier** $\mathcal{I}_\ell$ with exactly ℓ elements:

$$\mathcal{I}_\ell := \mathcal{I}_{\leq k_\ell} \cup \mathcal{R}, \quad \text{with } |\mathcal{I}_\ell| = \ell,$$

where k_ℓ is the largest order such that $|\mathcal{I}_{\leq k_\ell}| \leq \ell$, and $\mathcal{R} \subseteq \mathcal{I}_{\leq k_\ell + 1} \setminus \mathcal{I}_{\leq k_\ell}$ denotes a set of $\ell - |\mathcal{I}_{\leq k_\ell}|$ interaction terms of order $k_\ell + 1$. In words, $\mathcal{I}_\ell$ sequentially covers the k-additive interaction frontier up to k_ℓ, and supplements them with a selected subset of the subsequent higher-order interactions. In our experiments, we demonstrate that partially including higher-order interactions improves approximation quality, whereas using the full k-additive interaction frontier provides the largest gains.

5 Paired KernelSHAP is Paired 2-PolySHAP

A common heuristic when estimating Shapley values is to sample subsets in pairs S and $D \setminus S$. A kind of antithetic sampling (Glasserman, 2004), paired sampling substantially improves the approximation of estimators (Covert & Lee, 2021; Mitchell et al., 2022; Olsen & Jullum, 2024). Adding higher order interactions to PolySHAP improves Shapley value estimates, provided we have enough samples (see Figure 2): 3-PolySHAP outperforms 2-PolySHAP, which outperforms KernelSHAP (1-PolySHAP). Surprisingly, paired sampling partially collapses this hierarchy (see Figure 3).

Theorem 5.1 (Paired KernelSHAP is Paired 2-PolySHAP)**.** *Suppose that subsets are sampled in pairs i.e., if S is sampled then so is its complement $D \setminus S$, and, the matrix $\tilde{\mathbf{X}}$ has full column rank for interaction frontier D and $\mathcal{I}_{\leq 2}$. Then*

$$\hat{\phi}^{SV} = \text{PolySHAPtoSV}(\hat{\phi}^{\mathcal{I}_{\leq 2}})$$

In words, Shapley values approximated by 2-PolySHAP are precisely the KernelSHAP estimates.

We prove Theorem 5.1 by explicitly building the approximate solutions of KernelSHAP and 2-PolySHAP. Of particular help is a new technical projection lemma that we also use in the proof of Theorem 4.3. See Appendix A for the details.

Generalizing Prior Work. Mayer & Wüthrich (2025) recently showed that paired KernelSHAP exactly recovers the Shapley values of games with interactions of at most size 2. This follows immediately from Theorem 5.1, because 2-PolySHAP will precisely a game with order-2 interactions and paired Kernel SHAP will return the same solution. However, Theorem 5.1 is far more generally because it explains why paired sampling performs so well for *all* games, not just a restricted class.

Higher Dimensional Extensions. A natural question is whether similar results hold for higher order interactions. Suppose k is an odd number, we find empirically that paired $(k+1)$-PolySHAP returns the same approximate Shapley values as paired k-PolySHAP. We conjecture that this pattern holds for all odd k such that $1 \leq k < n$. However, it is not obvious how to adapt our proof of Theorem 5.1, since we would need the explicit mapping of $k+1$-PolySHAP representations to k-PolySHAP representations (this is clear when $k = 1$, but not so for higher dimensions).

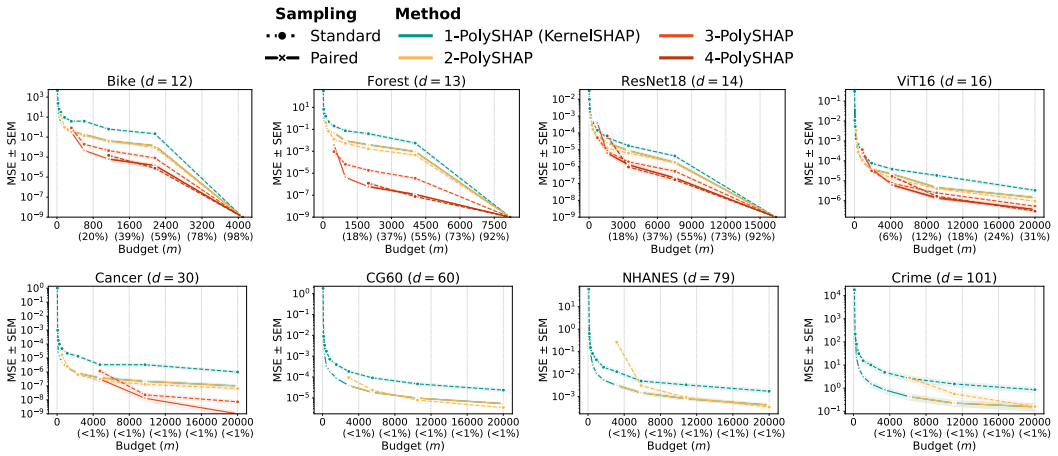


Figure 3: Approximation quality measured by MSE ($\pm$ SEM) for standard (dotted) and paired (solid) sampling. With paired sampling, KernelSHAP achieves the same performance as 2-PolySHAP.

6 Experiments

We empirically validate PolySHAP and approximate Shapley values on 15 local explanation games across 30 randomly selected instances, see Table 2. We evaluate all methods with m samples ranging from $d+1$ to $\min(2^d, 20000)$, and compare PolySHAP against *Permutation Sampling* (Castro et al., 2009), *SVARM* (Kolpaczki et al., 2024), *MSR* (Fumagalli et al., 2023; Wang & Jia, 2023), *Unbiased KernelSHAP* (Covert & Lee, 2021), *RegressionMSR* with XGBoost (Witter et al., 2025), and *KernelSHAP* (1-PolySHAP) with leverage score sampling (Lundberg & Lee, 2017; Musco & Witter, 2025).

Table 2: Explanation games.

ID	d	Domain
Housing	8	tabular
ViT9	9	image
Bike	12	tabular
Forest	13	tabular
Adult	14	tabular
ResNet18	14	image
DistilBERT	14	language
Estate	15	tabular
ViT16	16	image
CIFAR10	16	image
Cancer	30	tabular
CG60	60	synthetic
IL60	60	synthetic
NHANES	79	tabular
Crime	101	tabular

For tabular datasets, we trained random forests, while for image classification we used a `ResNet18` (He et al., 2016) with 14 superpixels and vision transformers with 3x3 (ViT9) and 4x4 (ViT16) super-patches on ImageNet (Deng et al., 2009), and CIFAR-10 (Krizhevsky et al., 2009). For language modeling, we used a fine-tuned `DistilBert` (Sanh et al., 2019) to predict sentiment on the IMDB dataset (Maas et al., 2011) with review excerpts of length 14. For tabular datasets, the games were defined via path-dependent feature perturbation, allowing ground-truth Shapley values to be obtained from TreeSHAP (Lundberg et al., 2020). For all other datasets, we used baseline imputation and exhaustive Shapley value computation. As evaluation metrics, we report *mean-squared error (MSE)*, top-5 precision (*Precision@5*), and *Spearman correlation* with *standard error of the mean (SEM)*. Code is available in the supplementary material, and additional details and results, including a runtime analysis, are provided in Appendix B.

PolySHAP Variants. For comparability across methods, we sample subsets using order-1 leverage scores, i.e., uniformly over subset sizes. We further adopt sampling without replacement and distinguish between standard and paired subset sampling. We apply the *border trick* (Fumagalli et al., 2023), replacing random sampling with exhaustive enumeration of sizes when the expected samples exceed the number of subsets. We use *k-PolySHAP* with $k \in \{1, 2, 3, 4\}$, and additionally the partial interaction frontiers that cover 50% of all k-order interactions, denoted by *k-PolySHAP (50%)*. For high-dimensional settings, we introduce PolySHAP (log) that adds $d \log(\binom{d}{3})$ order-3 interactions.

Higher-order Interactions Improve Approximation. Figure 2 reports the MSE with SEM for selected explanation games and standard sampling. Across different games, we observe that incorporating higher-order interactions in PolySHAP consistently improves approximation quality.

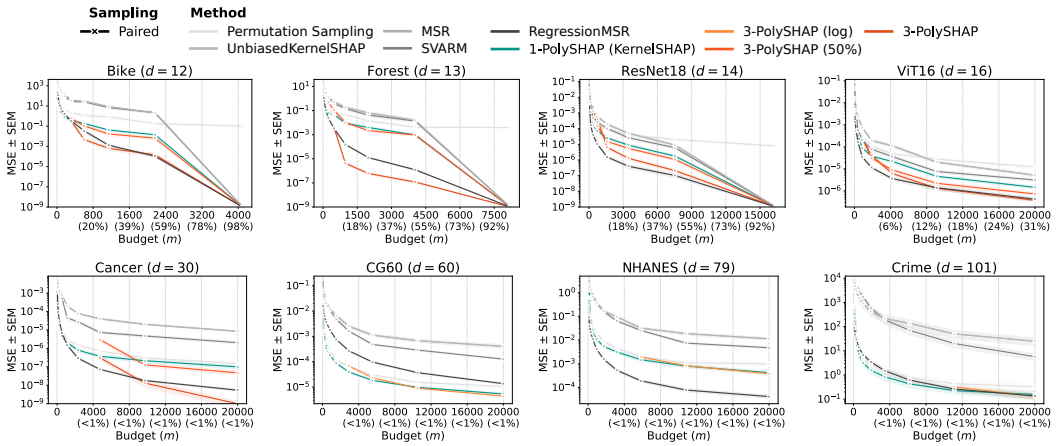


Figure 4: Approximation quality of PolySHAP variants and baseline methods measured by MSE ($\pm$ SEM) using paired sampling. With paired sampling, PolySHAP consistently improves upon KernelSHAP when order-3 interactions are included. In higher dimensions ($d \geq 60$), only a few of these can be modeled, yielding smaller improvements.

However, higher-order PolySHAP requires a larger sampling budget, and hence performance is only plotted for $m \geq d'$. Nevertheless, 2-PolySHAP, and even partial interaction inclusion (e.g., 2-PolySHAP at 50%), still yield notable improvements in approximation accuracy.

Paired KernelSHAP is 2-PolySHAP. As shown in Theorem 5.1, under paired sampling, KernelSHAP and 2-PolySHAP are equivalent indicated by the overlapping lines. We confirm this empirically in Figure 3. However, there is an important distinction: 2-PolySHAP requires more budget, whereas KernelSHAP can be computed already with $d + 1$ samples. Lastly, we observe a similar pattern for 3-PolySHAP: Under paired sampling 3-PolySHAP substantially improves its approximation quality and is equivalent to 4-PolySHAP.

Practical Benefits of PolySHAP. In practice, we adopt paired sampling and benchmark PolySHAP against all baselines in Figure 4 and Figure 7 in Appendix B.2. Because of our paired sampling result, the practical benefits of PolySHAP become apparent only when order-3 interactions are included. In low-dimensional settings, the 3-PolySHAP yields the best performance on Housing, Adult, Estate, Forest, and Cancer datasets (see e.g., Figure 4 and Figure 7). In budget-restricted cases, partially incorporating order-3 interactions already provides substantial gains, cf. 3-PolySHAP (50%) and 3-PolySHAP (log). In high-dimensional settings ($d \geq 60$), however, only a small number of order-3 interactions can be added, resulting in more modest improvements. Among all baselines, only RegressionMSR achieves comparable performance, although its performance depends strongly on XGBoost, as indicated by its poor results on CG60. Moreover, RegressionMSR has an inherent advantage since all tabular games rely on tree-based models.

7 Conclusion & Future Work

By reformulating the computation of the Shapley value as a polynomial regression problem with selected interaction terms, PolySHAP extends beyond the linear regression framework of KernelSHAP. We demonstrate that PolySHAP provides consistent estimates of the Shapley value (Theorem 4.3), and produces more accurate Shapley value estimates (see Figure 2 and Figure 3). Moreover, we show that paired subset sampling in KernelSHAP (Covert & Lee, 2021) implicitly captures all second-order interactions at no extra cost (Theorem 5.1), explaining why paired sampling improves estimator accuracy on games with arbitrary interaction structures.

Future work could explore more structured variants of interaction frontier, for example by detecting important interactions (Tsang et al., 2020) or leveraging inherent interaction structures in graph-structured inputs (Muschalik et al., 2025). In addition, we empirically find that paired k-PolySHAP produce the same estimates as $(k+1)$-PolySHAP for odd $k > 1$, but leave the proof for future work.

Ethics Statement

This work introduces a framework for efficient approximation of Shapley values, which are primarily used for explainable AI (XAI). We do not see any ethical concerns associated with this work.

Reproducibility Statement

We provide our code to reproduce our experimental results in a repository. The code repository can be used to (i) compute the ground-truth and approximated Shapley values across the local explanation games (with separate scripts for runtime), (ii) evaluating the approximation quality via various metrics, and (iii) plotting the results. Our implementation is based on the `shapiq` (Muschalik et al., 2024) library, and implements the `PolySHAP` and `RegressionMSR` approximator class, including changes for the optimized sampling strategies in the `CoalitionSampler` class.

For submission, this code is submitted in the supplementary materials, and, upon acceptance, will be made publicly available in a GitHub repository.

References

Sebastian Bordt and Ulrike von Luxburg. From Shapley Values to Generalized Additive Models and back. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 709–745, 2023.

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. *CoRR*, abs/2505.17495, 2025.

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

Javier Castro, Daniel Gómez, Elisenda Molina, and Juan Tejada. Improving polynomial estimation of the Shapley value by stratified random sampling with optimum allocation. *Computers & Operations Research*, 82:180–188, 2017. doi: 10.1016/j.cor.2017.01.019.

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 (KDD)*, pp. 785–794. ACM, 2016. doi: 10.1145/2939672.2939785.

Paulo Cortez and Aníbal de Jesus Raimundo Morais. A data mining approach to predict forest fires using meteorological data. 2007.

Ian Covert and Su-In Lee. Improving KernelSHAP: Practical Shapley Value Estimation Using Linear Regression. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 3457–3465, 2021.

Ian Covert, Scott M. Lundberg, and Su-In Lee. Understanding Global Feature Contributions With Additive Importance Measures. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ian Covert, Scott M. Lundberg, and Su-In Lee. Explaining by Removing: A Unified Framework for Model Explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021. doi: 10.5555/3546258.3546467.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

An Dinh, Stacey Miertschin, Amber Young, and Somya D. Mohanty. A data-driven approach to predicting diabetes and cardiovascular disease with machine learning. *BMC Medical Informatics Decisision Making*, 19(1):211:1–211:15, 2019. doi: 10.1186/S12911-019-0918-5.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

H. Fanaee-T and J. Gama. Event Labeling Combining Ensemble Detectors and Background Knowledge. *Progress in Artificial Intelligence*, 2(2):113–127, 2014. doi: 10.1007/s13748-013-0040-3.

Matthias Feurer, Jan N. van Rijn, Arlind Kadra, Pieter Gijsbers, Neeratyoy Mallik, Sahithya Ravi, Andreas Mueller, Joaquin Vanschoren, and Frank Hutter. OpenML-Python: an extensible Python API for OpenML. *CoRR*, abs/1911.02490, 2020.

Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Hammer. SHAP-IQ: Unified Approximation of any-order Shapley Interactions. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Fabian Fumagalli, Maximilian Muschalik, Eyke Hüllermeier, Barbara Hammer, and Julia Herbinger. Unifying Feature-Based Explanations with Functional ANOVA and Cooperative Game Theory. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 5140–5148, 2025.

Amirata Ghorbani and James Y. Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 2242–2251, 2019.

Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Stochastic Modelling and Applied Probability. Springer, 2004. doi: 10.1007/978-0-387-21617-1.

Michel Grabisch and Marc Roubens. An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of Game Theory*, 28(4):547–565, 1999. doi: 10.1007/s001820050125.

Michel Grabisch, Jean-Luc Marichal, and Marc Roubens. Equivalent representations of set functions. *Mathematics of Operations Research*, 25(2):157–178, 2000. doi: 10.1287/moor.25.2.157.12225.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Justin Singh Kang, Yigit Efe Erginbas, Landon Butler, Ramtin Pedarsani, and Kannan Ramchandran. Learning to understand: Identifying interactions via the möbius transform. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997. doi: 10.1016/S0167-7152(96)00140-X.

Gwladys Kelodjou, Laurence Rozé, Véronique Masson, Luis Galárraga, Romaric Gaudel, Maurice Tchuenté, and Alexandre Termier. Shaping up SHAP: enhancing stability through layer-wise neighbor selection. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 13094–13103, 2024. doi: 10.1609/AAAI.V38I12.29208.

R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 202–207, 1996.

Patrick Kolpaczki, Viktor Bengs, Maximilian Muschalik, and Eyke Hüllermeier. Approximating the shapley value without marginal contributions. In *Proceeedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 13246–13255, 2024.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 8780–8802, 2022a.

Yongchan Kwon and James Y. Zou. Weightedshap: analyzing and improving shapley based feature attributions. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2022b.

Weida Li and Yaoliang Yu. Faster approximation of probabilistic and distributional values via least squares. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Scott M. Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4765–4774, 2017.

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

Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex J. 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. doi: 10.1038/s42256-019-0138-9.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT)*, pp. 142–150, 2011.

Michael Mayer and Mario V Wüthrich. Shapley values: Paired-sampling approximations. *CoRR*, abs/2508.12947, 2025.

Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling Permutations for Shapley Value Estimation. *Journal of Machine Learning Research*, 23(43):1–46, 2022.

Majid Mohammadi, Ilaria Tiddi, and Annette ten Teije. Unlocking the game: Estimating games in möbius representation for explanation and high-order interaction detection. In *Proceeedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 19512–19519, 2025. doi: 10.1609/AAAI.V39I18.34148.

Christoph Molnar. *Interpretable machine learning: A guide for making black box models explainable*. Leanpub, 2024.

Maximilian Muschalik, Hubert Baniecki, Fabian Fumagalli, Patrick Kolpaczki, Barbara Hammer, and Eyke Hüllermeier. shapiq: Shapley Interactions for Machine Learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pp. 130324–130357, 2024.

Maximilian Muschalik, Fabian Fumagalli, Paolo Frazzetto, Janine Strotherm, Luca Hermes, Alessandro Sperduti, Eyke Hüllermeier, and Barbara Hammer. Exact Computation of Any-Order Shapley Interactions for Graph Neural Networks. In *Proceedings of the Conference on Learning Representations (ICLR)*, 2025.

Christopher Musco and R. Teal Witter. Provably Accurate Shapley Value Estimation via Leverage Score Sampling. In *Proccedings of the International Conference on Learning Representations (ICLR)*, 2025.

Lars Henry Berge Olsen and Martin Jullum. Improving the sampling strategy in kernelshap, 2024.

Lars Henry Berge Olsen, Ingrid Kristine Glad, Martin Jullum, and Kjersti Aas. A comparative study of methods for estimating model-agnostic Shapley value explanations. *Data Mining and Knowledge Discovery*, pp. 1–48, 2024.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011. doi: 10.5555/1953048.2078195.

Guilherme Dean Pelegrina, Leonardo Tomazeli Duarte, and Michel Grabisch. A k-additive choquet integral-based approach to approximate the SHAP values for local interpretability in machine learning. *Artificial Intelligence*, 325:104014, 2023. doi: 10.1016/J.ARTINT.2023.104014.

Guilherme Dean Pelegrina, Patrick Kolpaczki, and Eyke Hüllermeier. Shapley value approximation based on k-additive games. *CoRR*, abs/2502.04763, 2025.

Michael Redmond. Communities and crime unnormalized, 2011.

Gian-Carlo Rota. On the foundations of combinatorial theory: I. theory of möbius functions. In *Classic Papers in Combinatorics*, pp. 332–360. Springer, 1964.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

L. S. Shapley. A Value for n-Person Games. In *Contributions to the Theory of Games (AM-28), Volume II*, pp. 307–318. Princeton University Press, 1953.

W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pp. 861–870, 1993.

Mukund Sundararajan, Kedar Dhamdhere, and Ashish Agarwal. The Shapley Taylor Interaction Index. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 9259–9268, 2020.

Che-Ping Tsai, Chih-Kuan Yeh, and Pradeep Ravikumar. Faith-Shap: The Faithful Shapley Interaction Index. *Journal of Machine Learning Research*, 24(94):1–42, 2023.

Michael Tsang, Sirisha Rambhatla, and Yan Liu. How does This Interaction Affect Me? Interpretable Attribution for Feature Interactions. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6147–6159, 2020.

Jiachen T. Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In Francisco J. R. Ruiz, Jennifer G. Dy, and Jan-Willem van de Meent (eds.), *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 6388–6421, 2023.

R. Teal Witter, Yurong Liu, and Christopher Musco. Regression-adjusted monte carlo estimators for shapley values and probabilistic values. *CoRR*, abs/2506.11849, 2025.

I-Cheng Yeh and Tzu-Kuang Hsu. Building real estate valuation models with comparative approach through case-based reasoning. *Appied Soft Computing*, 65:260–271, 2018. doi: 10.1016/J.ASOC.2018.01.029.

CONTENTS OF THE SUPPLEMENTARY MATERIAL

A PROOFS

A.1 PROJECTION LEMMA

We introduce the following technical lemma that will be useful in the proofs of Theorem 4.3 and Theorem 5.1.

Lemma A.1 (Projection Lemma). *Let $n \geq d_+ > d$. Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with full column rank, a vector $\mathbf{y} \in \mathbb{R}^n$, and a real number $c \in \mathbb{R}$. Let $\mathbf{X}_+ \in \mathbb{R}^{n \times d_+}$ be a matrix where the first d columns are equal to $\mathbf{X}$. Define*

$$\boldsymbol{\beta}_+^* = \underset{\boldsymbol{\beta} \in \mathbb{R}^{d_+} : \langle \boldsymbol{\beta}, \mathbf{1}_{d_+} \rangle = c}{\arg\min} \|\mathbf{X}_+ \boldsymbol{\beta} - \mathbf{y}\|_2^2.$$

Then

$$\underset{\boldsymbol{\beta} \in \mathbb{R}^d : \langle \boldsymbol{\beta}, \mathbf{1} \rangle = c}{\arg\min} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 = \underset{\boldsymbol{\beta} \in \mathbb{R}^d : \langle \boldsymbol{\beta}, \mathbf{1}_d \rangle = c}{\arg\min} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{X}_+ \boldsymbol{\beta}_+^*\|_2^2. \tag{9}$$

Proof of Lemma A.1. We will first reformulate the constrained least squares problem as an unconstrained problem. Let $\mathbf{P}_d$ be the matrix that projects *off* the all ones vector in d dimensions i.e., $\mathbf{P}_{\mathbf{1}_d} = \mathbf{I} - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top$. Similarly, let $\mathbf{P}_{d_+} = \mathbf{I} - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top$. In general, we will drop the subscript d when the dimension is clear from context. We have

$$\begin{aligned}
\underset{\boldsymbol{\beta} \in \mathbb{R}^d : \langle \boldsymbol{\beta}, \mathbf{1} \rangle = c}{\arg\min} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 &= \underset{\boldsymbol{\beta} \in \mathbb{R}^d : \langle \boldsymbol{\beta}, \mathbf{1} \rangle = 0}{\arg\min} \left\|\mathbf{X}\boldsymbol{\beta} + \mathbf{X}\mathbf{1}\frac{c}{d} - \mathbf{y}\right\|_2^2 + \mathbf{1}\frac{c}{d} \\
&= \mathbf{P}_d \underset{\boldsymbol{\beta} \in \mathbb{R}^d}{\arg\min} \left\|\mathbf{X}\mathbf{P}_d\boldsymbol{\beta} + \mathbf{X}\mathbf{1}\frac{c}{d} - \mathbf{y}\right\|_2^2 + \mathbf{1}\frac{c}{d} \\
&= \mathbf{P}_d(\mathbf{X}\mathbf{P}_d)^\dagger \left(\mathbf{y} - \mathbf{X}\mathbf{1}\frac{c}{d}\right) + \mathbf{1}\frac{c}{d},
\end{aligned} \tag{10}$$

where the $(\cdot)^\dagger$ denotes the pseudoinverse, and the last equality follows by the standard solution to an unconstrained least squares problem. Similarly,

$$\boldsymbol{\beta}_+^* = \underset{\boldsymbol{\beta} \in \mathbb{R}^{d_+} : \langle \boldsymbol{\beta}, \mathbf{1}_{d_+} \rangle = c}{\arg\min} \|\mathbf{X}_+\boldsymbol{\beta} - \mathbf{y}\|_2^2 = \mathbf{P}_{d_+}(\mathbf{X}_+\mathbf{P}_{d_+})^\dagger \left(\mathbf{y} - \mathbf{X}_+\mathbf{1}\frac{c}{d_+}\right) + \mathbf{1}\frac{c}{d_+}. \tag{11}$$

Let $\mathbf{proj}_{\mathbf{X}\mathbf{P}_d} = (\mathbf{X}\mathbf{P}_d)(\mathbf{X}\mathbf{P}_d)^\dagger$ be the projection *onto* $\mathbf{X}\mathbf{P}_d$. We have

$$\begin{aligned}
&\mathbf{X} \underset{\boldsymbol{\beta} \in \mathbb{R}^d : \langle \boldsymbol{\beta}, \mathbf{1} \rangle = c}{\arg\min} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{X}_+\boldsymbol{\beta}_+^*\|_2^2 = \mathbf{X}\mathbf{P}_d(\mathbf{X}\mathbf{P}_d)^\dagger \left(\mathbf{X}_+\boldsymbol{\beta}_+^* - \mathbf{X}\mathbf{1}\frac{c}{d}\right) + \mathbf{X}\mathbf{1}\frac{c}{d} \\
&= \mathbf{proj}_{\mathbf{X}\mathbf{P}_d} \left(\mathbf{X}_+ \left[\mathbf{P}_{d_+}(\mathbf{X}\mathbf{P}_{d_+})^\dagger \left(\mathbf{y} - \mathbf{X}_+\mathbf{1}\frac{c}{d_+}\right) + \mathbf{1}\frac{c}{d_+}\right] - \mathbf{X}\mathbf{1}\frac{c}{d}\right) + \mathbf{X}\mathbf{1}\frac{c}{d} \\
&= \mathbf{proj}_{\mathbf{X}\mathbf{P}_d}\mathbf{proj}_{\mathbf{X}_+\mathbf{P}_{d_+}}\mathbf{y} - \mathbf{proj}_{\mathbf{X}\mathbf{P}_d}\mathbf{proj}_{\mathbf{X}_+\mathbf{P}_{d_+}}\mathbf{X}_+\mathbf{1}\frac{c}{d_+} + \mathbf{proj}_{\mathbf{X}\mathbf{P}_d}\mathbf{X}_+\mathbf{1}\frac{c}{d_+} - \mathbf{proj}_{\mathbf{X}\mathbf{P}_d}\mathbf{X}\mathbf{1}\frac{c}{d} + \mathbf{X}\mathbf{1}\frac{c}{d}.
\end{aligned} \tag{12}$$

Since the column space of $\mathbf{X}\mathbf{P}_d$ is contained in the column space of $\mathbf{X}_+\mathbf{P}_{d_+}$, observe that $\mathbf{proj}_{\mathbf{X}\mathbf{P}_d}\mathbf{proj}_{\mathbf{X}_+\mathbf{P}_{d_+}} = \mathbf{proj}_{\mathbf{X}\mathbf{P}_d}$. Then

$$\begin{aligned}
(12) &= \mathbf{proj}_{\mathbf{X}\mathbf{P}_d}\mathbf{y} - \mathbf{proj}_{\mathbf{X}\mathbf{P}_d}\mathbf{X}\mathbf{1}\frac{c}{d} + \mathbf{X}\mathbf{1}\frac{c}{d} \\
&= \mathbf{X}\mathbf{P}_d(\mathbf{X}\mathbf{P}_d)^\dagger \left(\mathbf{y} - \mathbf{X}\mathbf{1}\frac{c}{d}\right) + \mathbf{X}\mathbf{1}\frac{c}{d} \\
&= \mathbf{X} \underset{\boldsymbol{\beta} \in \mathbb{R}^d : \langle \boldsymbol{\beta}, \mathbf{1} \rangle = c}{\arg\min} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2.
\end{aligned} \tag{13}$$

Since $\mathbf{X}$ has full column rank, we have $\mathbf{X}^\dagger\mathbf{X} = \mathbf{I}$, so multiplying on the left by $\mathbf{X}^\dagger$ yields the statement.

□

A.2 PolySHAP is Consistent

In this section, we will prove Theorem 4.3.

Theorem 4.3. *The Shapley values of ν are recovered from the PolySHAP representation as*

$$\phi_i^{SV}[\nu] = \phi_i^{\mathcal{I}} + \sum_{S \in \mathcal{I}: i \in S} \frac{\phi_S^{\mathcal{I}}}{|S|} \quad \textit{for } i \in D. \tag{3}$$

Proof of Theorem 4.3. Recall $d' = d + |\mathcal{I}|$. Define the target vector $\mathbf{y} \in \mathbb{R}^{2^d}$ so that $[\tilde{\mathbf{y}}]_S = \sqrt{\mu(S)} \cdot \nu(S_\ell)$. Define the design matrix $\mathbf{X} \in \mathbb{R}^{2^d \times d}$ so that

$$[\mathbf{X}]_{S,i} = \sqrt{\mu(S)} \cdot \mathbb{1}[i \subseteq S], \tag{14}$$

and the extended design matrix $\mathbf{X}_+ \in \mathbb{R}^{2^d \times d'}$ so that

$$[\mathbf{X}_+]_{S,T} = \sqrt{\mu(S)} \cdot \mathbb{1}[T \subseteq S], \tag{15}$$

for $T \in D \cup \mathcal{I}$.

In this notation, we may write

$$\boldsymbol{\phi}^{\text{SV}}[\nu] = \underset{\boldsymbol{\phi} \in \mathbb{R}^d : \langle \mathbf{1}, \boldsymbol{\phi} \rangle = \nu(D)}{\arg\min} \|\mathbf{X}\boldsymbol{\phi} - \mathbf{y}\|_2^2, \tag{16}$$

and

$$\boldsymbol{\phi}^{\mathcal{I}}[\nu] = \underset{\boldsymbol{\phi} \in \mathbb{R}^{d'} : \langle \mathbf{1}, \boldsymbol{\phi} \rangle = \nu(D)}{\arg\min} \|\mathbf{X}_+\boldsymbol{\phi} - \mathbf{y}\|_2^2. \tag{17}$$

Consider the game $\hat{\nu} : 2^D \to \mathbb{R}$ where

$$\hat{\nu}(S) = \sum_{T \in D \cup \mathcal{I} : T \subseteq S} \phi_T^{\mathcal{I}}[\nu]. \tag{18}$$

For this game, the target vector is given by $\hat{\mathbf{y}} = \mathbf{X}_+\boldsymbol{\phi}^{\mathcal{I}}[\nu]$. Then its Shapley values are given by

$$\begin{aligned}
\boldsymbol{\phi}^{\text{SV}}[\hat{\nu}] &= \underset{\boldsymbol{\phi} \in \mathbb{R}^d : \langle \mathbf{1}, \boldsymbol{\phi} \rangle = \nu(D)}{\arg\min} \|\mathbf{X}\boldsymbol{\phi} - \hat{\mathbf{y}}\|_2^2 \\
&= \underset{\boldsymbol{\phi} \in \mathbb{R}^d : \langle \mathbf{1}, \boldsymbol{\phi} \rangle = \nu(D)}{\arg\min} \|\mathbf{X}\boldsymbol{\phi} - \mathbf{X}_+\boldsymbol{\phi}^{\mathcal{I}}\|_2^2 \\
&= \underset{\boldsymbol{\phi} \in \mathbb{R}^d : \langle \mathbf{1}, \boldsymbol{\phi} \rangle = \nu(D)}{\arg\min} \|\mathbf{X}\boldsymbol{\phi} - \mathbf{y}\|_2^2 \\
&= \boldsymbol{\phi}^{\text{SV}}[\nu],
\end{aligned} \tag{19}$$

where the penultimate equality follows by Lemma A.1. All that remains to compute the Shapley values $\hat{\nu}$. Since we have an explicit representation of $\hat{\nu}$ in terms of its Möbius transform in Equation (18), we know its Shapley values are

$$\phi_i^{\text{SV}}[\hat{\nu}] = \sum_{T \in D \cup \mathcal{I} : i \in T} \frac{\phi_T^{\mathcal{I}}[\nu]}{|T|}. \tag{20}$$

by e.g., Table 3 in Grabisch et al. (2000). The statement follows.

□

A.3 PAIRED KERNELSHAP IS PAIRED 2-POLYSHAP

We introduce some helpful notation, and then use it to restate Theorem 5.1 more formally below.

Define $d_k = \sum_{\ell=1}^{k} \binom{d}{\ell}$. Let $\tilde{\mathbf{X}}_k \in \mathbb{R}^{m \times \binom{d}{k}}$ be the matrix where the ℓ, T entry is given by

$$[\tilde{\mathbf{X}}_k]_{\ell,T} = \frac{\sqrt{\mu(S_\ell)}}{\sqrt{p(S_\ell)}} \mathbb{1}[T \subseteq S_\ell] \tag{21}$$

where $S_\ell \subseteq D$ is the ℓth sampled subset, and $T \subseteq D$ such that $|T| = k$. Then the matrix $\tilde{\mathbf{X}}_{\leq k} \in \mathbb{R}^{m \times d_k}$ is given by

$$\tilde{\mathbf{X}}_{\leq k} = \begin{bmatrix} \tilde{\mathbf{X}}_1 & \dots & \tilde{\mathbf{X}}_k \end{bmatrix}. \tag{22}$$

Let $\mathbf{M}_{2\to 1} \in \mathbb{R}^{d \times d_2}$ be the matrix that projects a 2-PolySHAP to a 1-PolySHAP. The entry corresponding to $i \in D$, and $S \subseteq D$ such that $|S| \leq 2$ is given by

$$[\mathbf{M}_{2\to 1}]_{i,S} = \frac{\mathbb{1}[i \in S]}{|S|}. \tag{23}$$

Theorem A.2 (Paired KernelSHAP is Paired 2-PolySHAP)**.** *Suppose $\tilde{\mathbf{X}}_{\leq 2}$ has full column rank. Further, suppose that both 1-PolySHAP and 2-PolySHAP are computed with the* same *paired samples i.e., if S is sampled then so is its complement $D \setminus S$. Then*

$$\underset{\boldsymbol{\phi} \in \mathbb{R}^d : \langle \mathbf{1}_d, \boldsymbol{\phi} \rangle = \nu(D)}{\arg\min} \|\tilde{\mathbf{X}}_1 \boldsymbol{\phi} - \tilde{\mathbf{y}}\|_2^2 = \mathbf{M}_{2\to 1} \underset{\boldsymbol{\phi} \in \mathbb{R}^{d_2} : \langle \mathbf{1}_{d_2}, \boldsymbol{\phi} \rangle = \nu(D)}{\arg\min} \|\tilde{\mathbf{X}}_{\leq 2} \boldsymbol{\phi} - \tilde{\mathbf{y}}\|_2^2. \tag{24}$$

In words, the Shapley values of the approximate 1-PolySHAP are exactly the same as those of the approximate 2-PolySHAP.

Proof of Theorem A.2. Define

$$\tilde{\mathbf{z}} + \mathbf{1}_{d_2} \frac{\nu(D)}{d_2} = \underset{\boldsymbol{\phi} \in \mathbb{R}^{d_2} : \langle \boldsymbol{\phi}, \mathbf{1}_{d_2} \rangle = \nu(D)}{\arg\min} \|\tilde{\mathbf{X}}_{\leq 2} \boldsymbol{\phi} - \tilde{\mathbf{y}}\|_2^2, \tag{25}$$

where $\tilde{\mathbf{z}}$ is orthogonal to the all ones vector. By Lemma A.1 and the structure $\mathbf{A}_{\leq 2} = [\mathbf{A}_1 \quad \mathbf{A}_2]$, we have

$$\underset{\boldsymbol{\phi} \in \mathbb{R}^d : \langle \mathbf{1}_d, \boldsymbol{\phi} \rangle = \nu(D)}{\arg\min} \|\tilde{\mathbf{X}}_1 \boldsymbol{\phi} - \tilde{\mathbf{y}}\|_2^2 = \underset{\boldsymbol{\phi} \in \mathbb{R}^d : \langle \mathbf{1}_d, \boldsymbol{\phi} \rangle = \nu(D)}{\arg\min} \left\| \tilde{\mathbf{X}}_1 \boldsymbol{\phi} - \tilde{\mathbf{X}}_{\leq 2} \left(\tilde{\mathbf{z}} + \mathbf{1}_{d_2} \frac{\nu(D)}{d_2} \right) \right\|_2^2. \tag{26}$$

Using Equation 10, we can write Equation 26 explicitly as

$$(26) = \mathbf{P}_d (\mathbf{P}_d \tilde{\mathbf{X}}_1^\top \tilde{\mathbf{X}}_1 \mathbf{P}_d)^\dagger \mathbf{P}_d \tilde{\mathbf{X}}_1^\top \left[\tilde{\mathbf{X}}_{\leq 2} \left(\tilde{\mathbf{z}} + \mathbf{1}_{d_2} \frac{\nu(D)}{d_2} \right) - \tilde{\mathbf{X}}_1 \mathbf{1}_d \frac{\nu(D)}{d} \right] + \mathbf{1}_d \frac{\nu(D)}{d} \tag{27}$$

where $\mathbf{P}_d = \mathbf{1} - \frac{1}{d} \mathbf{1}\mathbf{1}^\top$ is the matrix that projects off the all ones direction in d dimensions.

Our goal is to show that

$$(27) = \mathbf{M}_{2\to 1} \left(\tilde{\mathbf{z}} + \mathbf{1}_{d_2} \frac{\nu(D)}{d_2} \right). \tag{28}$$

We'll begin with the all ones component. Observe that

$$\frac{\nu(D)}{d} [\mathbf{M}_{2\to 1} \mathbf{1}]_i = \frac{\nu(D)}{d_2} \left(\sum_{j=1}^{d} \mathbb{1}[i = j] + \sum_{T \subseteq D : |T| = 2} \frac{\mathbb{1}[i \in T]}{2} \right) = \nu(D) \frac{1 + \frac{d-1}{2}}{d + \binom{d}{2}} = \frac{\nu(D)}{d},$$

so $\mathbf{M}_{2\to 1} \mathbf{1}_{d_2} \frac{\nu(D)}{d_2} = \mathbf{1}_d \frac{\nu(D)}{d}$.

Now it remains to show the equality for the component orthogonal to the all ones direction. Since $\tilde{\mathbf{X}}_{\leq 2} = \begin{bmatrix} \tilde{\mathbf{X}}_1 & \tilde{\mathbf{X}}_2 \end{bmatrix}$ has full column rank by assumption, $\tilde{\mathbf{X}}_1$ must have full column rank as well. It

follows that $(\mathbf{P}_d\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_1\mathbf{P}_d)(\mathbf{P}_d\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_1\mathbf{P}_d)^\dagger = \mathbf{P}_d$. Then, after multiplying Equations 27 and 28 by $(\mathbf{P}_d\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_1\mathbf{P}_d)$, it suffices to show that

$$\begin{aligned}\mathbf{P}_d\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_1\mathbf{P}_d\mathbf{M}_{2\to1}\tilde{\mathbf{z}} &= \mathbf{P}_d\tilde{\mathbf{X}}_1^\top\left[\tilde{\mathbf{X}}_{\le2}\left(\tilde{\mathbf{z}}+\mathbf{1}_{d_2}\frac{\nu(D)}{d_2}\right)-\tilde{\mathbf{X}}_1\mathbf{1}_d\frac{\nu(D)}{d}\right]\\ &= \mathbf{P}_d\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_{\le2}\tilde{\mathbf{z}}+\mathbf{P}_d\left[\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_{\le2}\mathbf{1}_{d_2}\frac{\nu(D)}{d_2}-\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_{\le1}\mathbf{1}_d\frac{\nu(D)}{d}\right]. \end{aligned}\tag{29}$$

We will first show that the second term on the right hand side is 0. First, notice that

$$[\tilde{\mathbf{X}}_{\le k}\mathbf{1}_{d_k}]_S = \sum_{T\subseteq D:|T|\le k}\frac{\sqrt{\mu_S}}{\sqrt{p_S}}\mathbb{1}[T\subseteq S] = \frac{\sqrt{\mu_S}}{\sqrt{p_S}}|S|_k \tag{30}$$

where $|S|_k = \sum_{\ell=1}^k\binom{|S|}{\ell}$. Then

$$\frac{1}{d_k}[\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_{\le}\mathbf{1}_{d_k}]_i = \sum_{S:i\in S}\frac{\mu_S}{p_S}\frac{|S|_k}{d_k}. \tag{31}$$

We have $\frac{|S|_2}{d_2} = \frac{|S|+\binom{|S|}{2}}{d+\binom{d}{k}} = \frac{|S|}{d}\frac{1+(|S|-1)/2}{1+(d-1)/2} = \frac{|S|}{d}\cdot\frac{|S|+1}{d+1}$. Together,

$$\begin{aligned}\left[\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_{\le2}\mathbf{1}_{d_2}\frac{1}{d_2}-\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_1\mathbf{1}_d\frac{1}{d}\right]_i &= \sum_{S:i\in S}\frac{\mu_S}{p_S}\frac{|S|}{d}\left(\frac{|S|+1}{d+1}-\frac{d+1}{d+1}\right)\\ &= \frac{1}{d(d+1)}\sum_{S:i\in S}\frac{\mu_S}{p_S}|S|(|S|-d)\\ &= \frac{1}{2d(d+1)}\sum_{S}\frac{\mu_S}{p_S}|S|(|S|-d)\end{aligned}\tag{32}$$

where the last equality follows because the subsets are sampled in paired complements. In particular, for a given pair S and $D\setminus S$, the item i is in exactly one of them, and the coefficient $\frac{\mu_S}{p_S}|S|(|S|-d)$ is the same for both. We have shown that every entry is the same, i.e., a scaling of $\mathbf{1}$, so $\mathbf{P}_d$ projects off the entire vector.

Finally, it remains to show that

$$\mathbf{P}_d\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_1\mathbf{P}_d\mathbf{M}_{2\to1}\tilde{\mathbf{z}} = \mathbf{P}_d\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_{\le2}\tilde{\mathbf{z}}. \tag{33}$$

It is easy to verify that $\langle\mathbf{1}_d,\mathbf{M}\tilde{\mathbf{z}}\rangle = \langle\mathbf{1}_d,\tilde{\mathbf{z}}\rangle = 0$, so $\mathbf{P}_d\mathbf{M}_{2\to1}\tilde{\mathbf{z}} = \mathbf{M}_{2\to1}\tilde{\mathbf{z}}$. Therefore, it suffices to prove that $\mathbf{P}_d\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_1\mathbf{M}_{2\to1} = \mathbf{P}_d\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_{\le2}$.

Notice that $[\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_1]_{i,j} = \sum_{S:i\in S,j\in S}\frac{\mu_S}{p_S}$ where $i,j\in D$. Then

$$\begin{aligned}[\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_1\mathbf{M}_{2\to1}]_{i,R} &= \sum_{j=1}^d\frac{\mathbb{1}[j\in R]}{|R|}\sum_{S:i\in S,j\in S}\frac{\mu_S}{p_S}\\ &= \sum_{S:i\in S}\frac{\mu_S}{p_S}\sum_{j=1:j\in S,j\in R}^d\frac{1}{|R|}\\ &= \sum_{S:i\in S}\frac{\mu_S}{p_S}\frac{|R\cap S|}{|R|}.\end{aligned}\tag{34}$$

Meanwhile,

$$[\tilde{\mathbf{X}}_1^\top\tilde{\mathbf{X}}_{\le2}]_{i,R} = \sum_{S:i\in S,R\subseteq[S]}\frac{\mu_S}{p_S}. \tag{35}$$

Clearly, Equations 35 and 34 are equal when $|R|=1$. Now consider the case when $|R|=2$; we have

$$(34) = \sum_{S:i\in S,|R\cap S|=1}\frac{\mu_S}{p_S}\frac{1}{2}+\sum_{S:i\in S,|R\cap S|=2}\frac{\mu_S}{p_S} = \frac{1}{4}\sum_{S:|R\cap S|=1}\frac{\mu_S}{p_S}+\sum_{S:i\in S,R\subseteq S}\frac{\mu_S}{p_S}. \tag{36}$$

where the last equality follows by sampling in paired complements. In particular, exactly one of the paired samples S and $D \subseteq S$ will contain item i, and the coefficient μ_S/p_S is the same for both. Finally, because its the same for all i, the projection $\mathbf{P}_d$ eliminates the first term. The statement follows.

□

A.4 k_{ADD}-SHAP CONVERGES TO THE SHAPLEY VALUE

In this section, we prove Proposition 4.6 and discuss the differences between PolySHAP and k_{ADD}-SHAP, and its practical implications. We generally recommend to prefer PolySHAP over k_{ADD}-SHAP.

Proposition 4.6. *k_{ADD}-SHAP converges to the Shapley value for $k = 1, \dots, d$.*

Proof. The k_{ADD} approximation algorithm (Pelegrina et al., 2023) is based on the interaction representation (Grabisch et al., 2000) of ν given by

$$\nu(S) = \sum_{T \subseteq D} \gamma_{|S \cap T|}^{|T|} I_{\text{Sh}}(T) \quad \text{with } \gamma_r^t := \sum_{\ell=0}^{r} \binom{r}{\ell} B_{t-\ell},$$

where B_t are the Bernoulli numbers and I_{Sh} the Shapley interaction index (Grabisch & Roubens, 1999) with

$$I_{\text{Sh}}(S) := \sum_{T \subseteq D \setminus S} \frac{1}{(d - |S| + 1)\binom{d-|S|}{|T|}} \sum_{L \subseteq S} (-1)^{|S|-|L|} \nu(T \cup L).$$

The Shapley interaction index generalizes the Shapley value to arbitrary subsets, and it holds $\phi_i^{\text{SV}}[\nu] = I_{\text{Sh}}(i)$ for all $i \in D$. The k_{ADD}-SHAP approximation algorithm then restricts this representation to interactions up to order k.

Definition A.3 (k_{ADD}-SHAP (Pelegrina et al., 2025))**.** *The k_{ADD}-SHAP algorithms solves the constrained weighted least-squares problem*

$$I^{k_{ADD}} := \underset{I \in \mathbb{R}^{\sum_{\ell=0}^{k} \binom{d}{\ell}}}{\arg\min} \sum_{S \subseteq D} \mu(S) \left(\nu(S) - \sum_{T \subseteq D : |T| \leq k} \gamma_{|S \cap T|}^{|T|} I_T \right)^2$$
$$\text{s.t. } \nu(D) - \nu(\emptyset) = \sum_{T \subseteq D : |T| \leq k} \left(\gamma_{|T|}^{|T|} - \gamma_0^{|T|} \right) I_T.$$

In practice, the least-squares objective is approximated and solved similar to KernelSHAP (Lundberg & Lee, 2017), and the Shapley value estimates that are output are $I_i^{k_{ADD}}$ for $i \in D$ from the approximated least-squares system.

Our first observation is that the output I_i is the Shapley value of the approximated game, i.e.

$$\phi_i^{\text{SV}}[\sum_{T \subseteq D : |T| \leq k} \gamma_{\mathbb{1}[i \in T]}^{|T|} I_T] = I_i.$$

We will show that the Shapley values of this approximation are the Shapley values of the PolySHAP representation $\phi^{\mathcal{I}_{\leq k}}$, which then are equal to the Shapley values of ν by Theorem 4.3.

In contrast to PolySHAP, k_{ADD}-SHAP fits a coefficient for the empty set $\phi_\emptyset$. However, we may rewrite

$$\left(\nu(S) - \sum_{T \subseteq D : |T| \leq k} \gamma_{|S \cap T|}^{|T|} I_T \right)^2 = \left(\nu(S) - \gamma_0^0 I_\emptyset - \sum_{T \subseteq D : 0 < |T| \leq k} \gamma_{|S \cap T|}^{|T|} I_T \right)^2,$$

and thus $I_\emptyset$ is an additive shift of ν, which does not affect the Shapley values of the approximation, i.e.

$$\phi_i^{\text{SV}}[\sum_{T \subseteq D : |T| \leq k} \gamma_{|S \cap T|}^{|T|} I_T] = \phi_i^{\text{SV}}[\sum_{T \subseteq D : 0 < |T| \leq k} \gamma_{|S \cap T|}^{|T|} I_T].$$

Moreover, we can compute $\gamma_0^t = B_t$ and

$$\gamma_s^s = \sum_{\ell=0}^{s} \binom{s}{\ell} B_{s-\ell} = \sum_{\ell=0}^{s} \binom{s}{\ell} B_\ell = \sum_{\ell=0}^{s-1} \binom{s}{\ell} B_\ell + B_s = \mathbb{1}[s = 1] + B_s,$$

by the recursion of Bernoulli numbers, and thus

$$\sum_{S\subseteq D:|S|\leq k}\left(\gamma_{|S|}^{|S|}-\gamma_0^{|S|}\right)I_S=\sum_{i\in D}I_i,$$

which is already mentioned by Pelegrina et al. (2025)[Proof of Theorem 4.2]. Now, without loss of generality, we can assume that $\nu(\emptyset)=0$, since it does not affect the Shapley values of ν, and thus the class of approximations is given by

$$\mathcal{F}^{k_{\text{ADD}}}:=\left\{S\mapsto\sum_{T\subseteq D:0<|T|\leq k}\gamma_{|S\cap T|}^{|T|}I_T:\phi\in\mathbb{R}^{d+|\mathcal{I}_{\leq k}|}\text{ and }\sum_{i\in D}I_i=\nu(D)\right\}.$$

Lemma A.4. *There is an equivalence between the function class $\mathcal{F}^{k_{\text{ADD}}}$ and the class of functions of PolySHAP representation with interaction frontier $\mathcal{I}_{\leq k}$, i.e.*

$$\mathcal{F}^{k_{\text{ADD}}}=\left\{S\mapsto\sum_{T\in D\cup\mathcal{I}_{\leq k}}\phi_T\prod_{j\in T}\mathbb{1}[j\in S]:\phi\in\mathbb{R}^{d+|\mathcal{I}_{\leq k}|}\text{ and }\langle\phi,1\rangle=\nu(D)\right\}$$

Proof. For the game ν there exist the two equivalent representations (Grabisch et al., 2000)[Table 3 and 4]

$$\nu(S)=\sum_{T\subseteq D}\gamma_{|S\cap T|}^{|T|}I_{\text{Sh}}(T)\quad\text{ with }\gamma_r^t:=\sum_{\ell=0}^{r}\binom{r}{\ell}B_{t-\ell},$$

where I_{Sh} is the Shapley interaction index (Grabisch & Roubens, 1999), and the Möbius representation

$$\nu(S)=\sum_{T\subseteq D}m(S)\prod_{j\in S}\mathbb{1}[j\in S]\quad\text{with }m(S):=\sum_{L\subseteq S}(-1)^{|S|-|L|}\nu(L).$$

Moreover, there exist the two conversion formulas (Grabisch et al., 2000)[Table 3 and 4]

$$I_{\text{Sh}}(S)=\sum_{T\subseteq D:T\supseteq S}\frac{1}{|T|-|S|+1}m(T)\text{ and }m(S)=\sum_{T\subseteq D:T\supseteq S}B_{|T|-|S|}I_{\text{Sh}}(T).$$

From the conversion formulas it is obvious that

$$I_{\text{Sh}}(S)=0,\quad\forall S\subseteq D:|S|>k\quad\Leftrightarrow\quad m(S),\quad\forall S\subseteq D:|S|>k.$$

Hence, restricting the interaction representation to order k yields the same function class as restricting the Möbius representation to order k. Moreover, the constraints are similarly converted, which proves the equivalence. □

Utilizing Lemma A.4, we obtain that for

$$I^{k_{\text{ADD}}^+}:=\underset{I\in\mathbb{R}^{\sum_{\ell=1}^k\binom{d}{\ell}}}{\arg\min}\sum_{S\subseteq D}\mu(S)\left(\nu(S)-\sum_{T\subseteq D:|T|\leq k}\gamma_{|S\cap T|}^{|T|}I_T\right)^2\quad\text{s.t. }\nu(D)=\sum_{i\in D}I_i$$

we have equivalence between the approximations

$$\sum_{T\subseteq D:|T|\leq k}\gamma_{|S\cap T|}^{|T|}I_T^{k_{\text{ADD}}^+}=\sum_{T\in D\cup\mathcal{I}_{\leq k}}\phi_T^{\mathcal{I}_{\leq k}}\prod_{j\in T}\mathbb{1}[j\in S],$$

where $\phi_T^{\mathcal{I}_{\leq k}}$ is the PolySHAP representation, due to the equivalent function classes parametrized by the vectors $I^{k_{\text{ADD}}^+}$ and $\phi^{\mathcal{I}_{\leq k}}$. By Theorem 4.3, we know that the Shapley values of this approximation are equal to the Shapley values of ν, and hence, we have

$$I_i^{k^{\text{ADD}}}=\phi_i^{\text{SV}}[\sum_{T\subseteq D:|T|\leq k}\gamma_{|S\cap T|}^{|T|}I_T^{k_{\text{ADD}}}]=\phi_i^{\text{SV}}[\sum_{T\subseteq D:|T|\leq k}\gamma_{|S\cap T|}^{|T|}I_T^{k_{\text{ADD}}^+}]=\phi_i^{\text{SV}}[\nu],$$

which concludes the proof and show convergence of k_{ADD}-SHAP to the Shapley value.

Practical difference between k_{ADD}-SHAP and PolySHAP. In contrast to PolySHAP, k_{ADD}-SHAP was proposed for k-additive interaction frontiers. Moreover, the design matrix of k_{ADD}-SHAP is less intuitive, making the PolySHAP formulation a simpler and more transparent alternative. More importantly, ta key practical difference arises from our use of the modified representation $I^{k^+_{\text{ADD}}}$ as an intermediate step in the proof. While $I^{k^+_{\text{ADD}}}$ and $I^{k_{\text{ADD}}}$ yield the same Shapley values when all subsets are evaluated, they diverge under approximation. In particular, unlike PolySHAP, k_{ADD}-SHAP is affected by the value of $\nu(\emptyset)$, and its least-squares fit includes an additional variable. For these reasons, we recommend PolySHAP in practice over k_{ADD}-SHAP.

□

Table 3: Datasets used for tabular explanation games

Name (ID in bold)	Reference	License	Source
California **Housing**	(Kelley Pace & Barry, 1997)	Public Domain	`sklearn`
Bike Regression	(Fanaee-T & Gama, 2014)	CC-BY 4.0	OpenML
Forest Fires	(Cortez & Morais, 2007)	CC-BY 4.0	UCI Repo
Adult Census	(Kohavi, 1996)	CC-BY 4.0	OpenML
Real **Estate**	(Yeh & Hsu, 2018)	CC-BY 4.0	UCI Repo
Breast **Cancer**	(Street et al., 1993)	CC-BY 4.0	`shap`
Correlated Groups (**CG60**)	synthetic	MIT	`shap`
Independent Linear (**IL60**)	synthetic	MIT	`shap`
NHANES I	(Dinh et al., 2019)	Public Domain	`shap`
Communities and **Crime**	(Redmond, 2011)	CC-BY 4.0	`shap`

B Experimental Details and Additional Results

In this section, we provide additional details regarding our experiments and the local explanation game setup (Appendix B.1) with additional results on the remaining games using MSE (Appendix B.2), Precision@5 (Appendix B.3), and Spearman correlation (Appendix B.4). Lastly, we report results of the runtime analysis (Appendix B.5).

B.1 Experimental Details

All experiments were conducted on a consumer-grade laptop with an 11th Gen Intel Core i7-11850H CPU and 30GB of RAM, where we used `cuda`[2] on a NVIDIA RTX A2000 GPU for inference of the CIFAR10 game.

Non-tabular Datasets. We used the 30 pre-computed games provided by the `shapiq` benchmark for the ResNET18 (He et al., 2016), and the vision transformers pre-trained on ImageNet (Deng et al., 2009). We used the pre-computed language game using a DistilBERT (Sanh et al., 2019) model and sentiment analysis on the IMDB dataset (Maas et al., 2011) from the `shapiq` benchmark. Lastly for the CIFAR10 game, we used a vision transformer (vit-base-patch16-224-in21k) (Dosovitskiy et al., 2021) fine-tuned on CIFAR10 (Krizhevsky et al., 2009), which is publicly available[3].

Datasets. The datasets and their source used for the tabular explanation games are described in Table 3. The *Forest Fires*[4] and *Real Estate*[5] were sourced from UCI Machine Learning Repository (UCI Repo), whereas *Bike Regression* was taken from OpenML (Feurer et al., 2020). The *California Housing* dataset was sourced from scikit-learn (Pedregosa et al., 2011)[`sklearn`], and the remaining datasets were sourced from the `shap`[6] library.

Random forest configuration. We use the standard implementation for `RandomForestRegressor` and `RandomForestClassifier` from *scikit-learn* (Pedregosa et al., 2011)[`sklearn`] with 10 tree instances of maximum depth 10 and fit the training data using accuracy (classification) and F1-score (regression). For all datasets, a $80/20$ percent train-test-split was executed.

RegressionMSR. For the RegressionMSR approach, we use `XGBoost` (Chen & Guestrin, 2016) with its default configuration as a tree-based backbone combined with the `MonteCarlo` approximator (equivalent to MSR (Witter et al., 2025)) from the `shapiq` package.

[2] https://developer.nvidia.com/cuda-toolkit
[3] https://huggingface.co/aaraki/vit-base-patch16-224-in21k-finetuned-cifar10
[4] https://archive.ics.uci.edu/ml/datasets/forest+fires
[5] https://archive.ics.uci.edu/dataset/477/real+estate+valuation+data+set
[6] https://shap.readthedocs.io/en/latest/

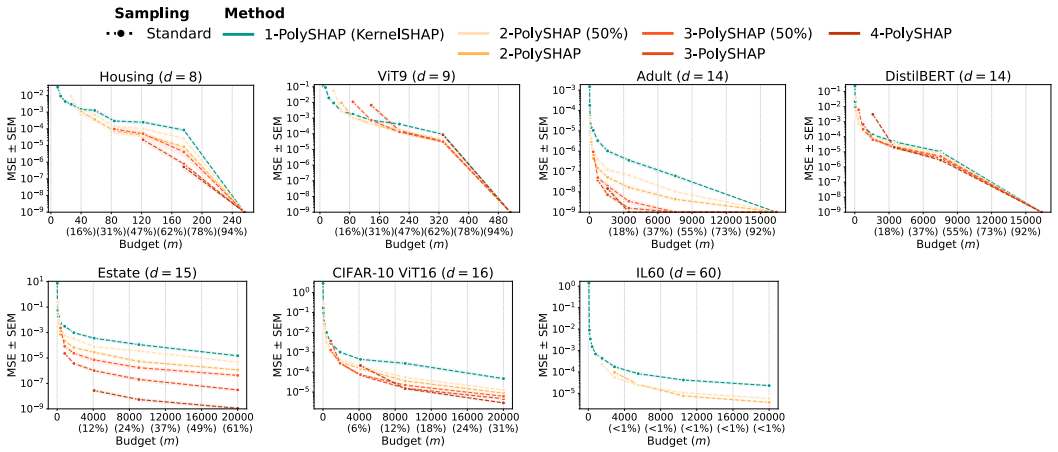


Figure 5: Approximation quality measured by MSE ($\pm$ SEM) for varying budget (m) on remaining explanation games. Adding interactions in PolySHAP can substantially improve approximation quality

MSR and Unbiased KernelSHAP. It was shown (Fumagalli et al., 2023)[Theorem 4.5] that MSR (Wang & Jia, 2023) is equivalent to Unbiased KernelSHAP (Covert & Lee, 2021). We use the implementation of Unbiased KernelSHAP provided in the `shapiq` package.

SVARM Shapley Value Approximation without Requesting Marginals (SVARM) was proposed by (Kolpaczki et al., 2024) and uses stratification of MSR (Castro et al., 2017). We use the implementation of SVARM provided in the `shapiq` package.

B.2 Additional Results on Approximation Quality using MSE

In this section, we report approximation quality measured by MSE for the remaining explanation games.

Figure 5 reports the MSE for the *Housing*, *ViT9*, *Adult*, *DistilBERT*, *Estate* , and *IL60* explanation games. Similar to Figure 2, we observe that PolySHAP's approximation quality substantially improves with higher-order interactions. Again, this comes at the cost of larger budget requirements, indicated by the delay of the line plots. The Permutation Sampling and KernelSHAP (1-PolySHAP) baseline are consistently outperformed by higher-order PolySHAP, while RegressionMSR yields comparable results.

Figure 6 shows the approximation quality of PolySHAP with and without (standard) paired subset sampling. Similar to Figure 3, we observe a strong improvement of 1-PolySHAP due to the equivalence to 2-PolySHAP. The same observation holds for 3-PolySHAP.

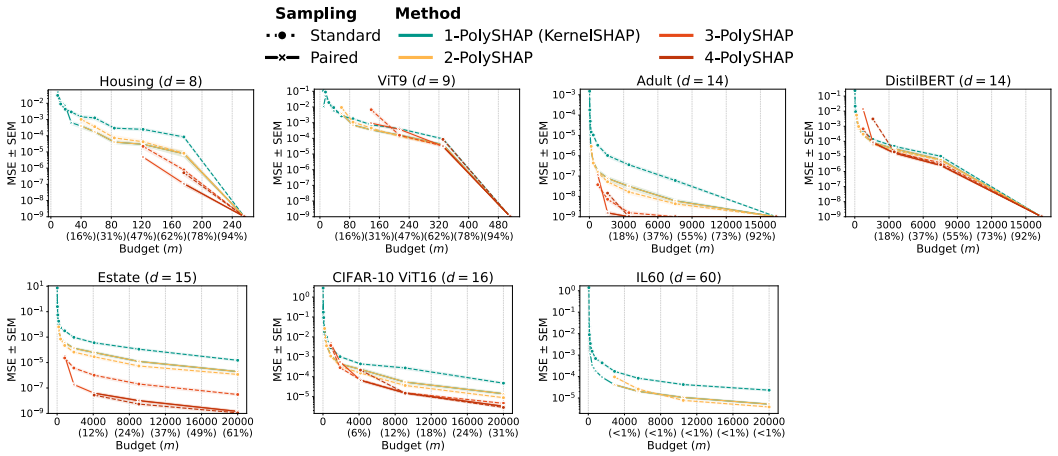


Figure 6: Approximation quality measured by MSE (± SEM) for standard (dotted) and paired (solid) sampling for remaining local explanation games. Under paired sampling, 2-PolySHAP marginally improves, whereas KernelSHAP substantially improves due to its equivalence to 2-PolySHAP.

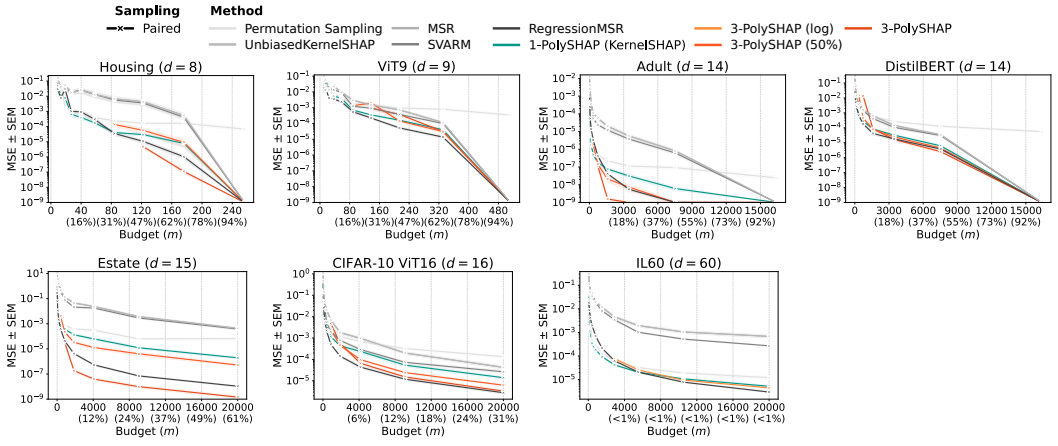


Figure 7: Approximation quality of PolySHAP variants and baselines measured by MSE (± SEM) for paired sampling for remaining local explanation games.

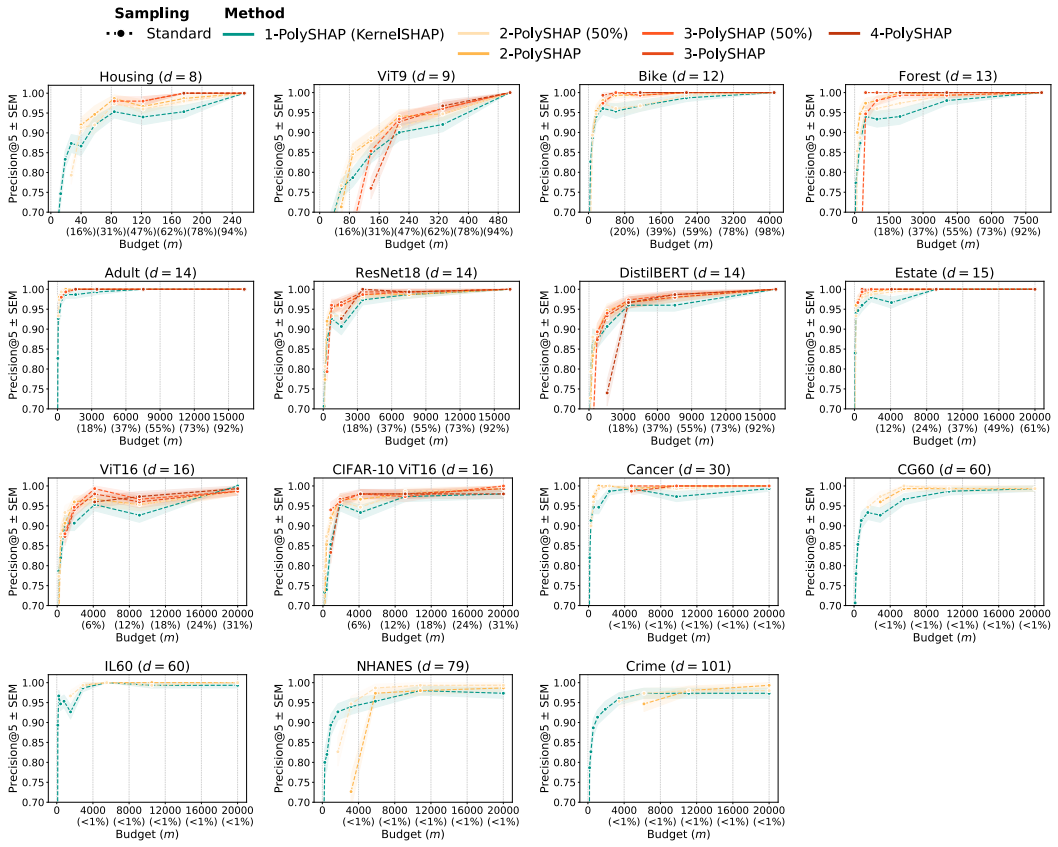


Figure 8: Approximation quality measured by Precision@5 ($\pm$ SEM) for varying budget (m) on different games. Adding interactions in PolySHAP can substantially improve approximation quality

B.3 Approximation Quality using Precision@5

In this section, we report approximation quality with respect to top-5 precision (*Precision@5*) for all explanation games from Table 2.

In Figure 8 that higher-order interactions also improve the approximation quality regarding the Precision@5 metric. However, the distinction is not as clear as for MSE, since ranking is not considered in the optimization objective. In general, the approximation quality varies across different games, where the low-dimensional tabular explanation games show very good results, in contrast to the more challenging non-tabular games (ViT9, DistilBERT, ResNet18 and ViT16), and high-dimensional games (CG60, IL60, NHANES, and Crime), which require more budget for similar results.

In Figure 8 Precision@5 is compared for standard sampling and paired sampling. Again, we observe improvements for 1-PolySHAP and 3-PolySHAP when using paired sampling due to its equivalence to 2-PolySHAP and 4-PolySHAP, respectively.

In Figure 10, we report the Precision@5 metric for the PolySHAP variants and the baselines for paired sampling. Again, we observe state-of-the-art performance for PolySHAP and Regression-MSR. PolySHAP's performance substantially improves, if the budget allows to capture order-3 interactions.

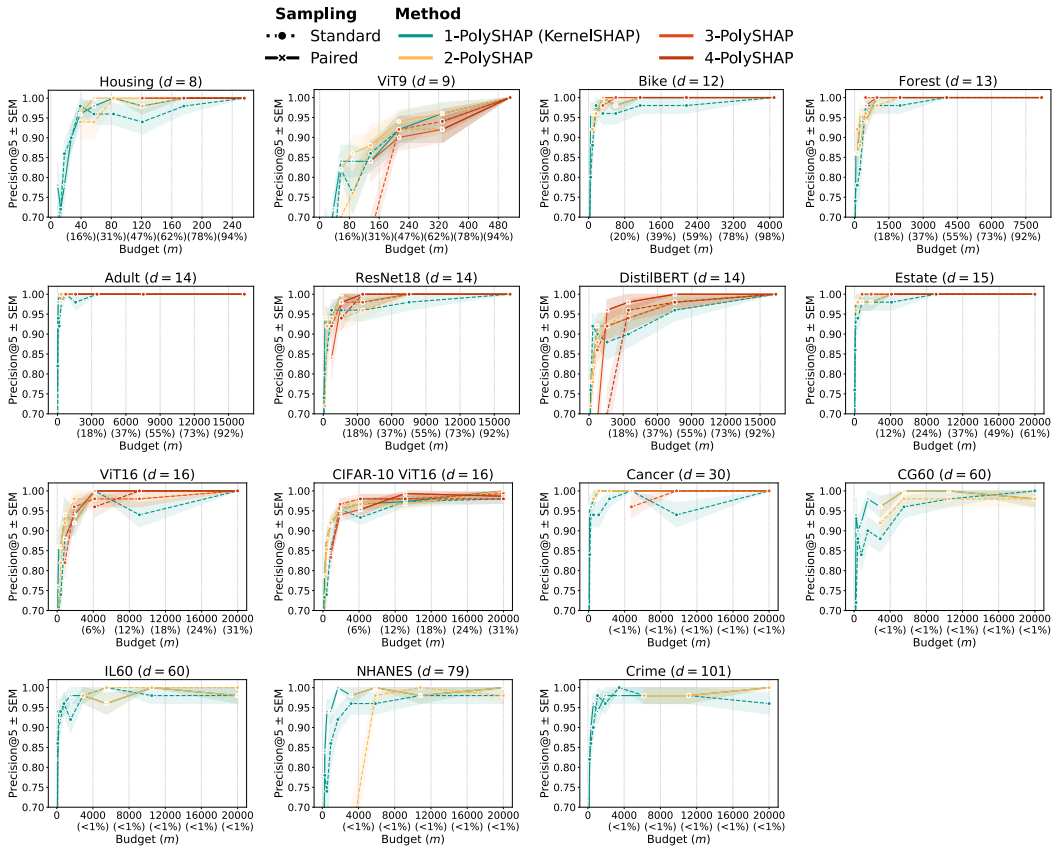


Figure 9: Approximation quality measured by Precision@5 ($\pm$ SEM) for standard (dotted) and paired (solid) sampling. Under paired sampling, 2-PolySHAP marginally improves, whereas KernelSHAP substantially improves due to its equivalence to 2-PolySHAP

B.4 Approximation Quality using Spearman Correlation

In this section, we report approximation quality with respect to Spearman correlation (*SpearmanCorrelation*) for all explanation games from Table 2.

Figure 11 reports Spearman correlation of PolySHAP and the baseline methods. Again, we observe consistent improvements of higher-order interactions in this metric. For high-dimensional settings (≥ 60), we further observe that the baselines clearly outperform PolySHAP in this metric. Since we have seen that PolySHAP performs very well in the Precision@5 metric, we conjecture that this difference is mainly due features with lower absolute Shapley values.

In Figure 12, we observe a similar pattern as with MSE and Precision@5. Using paired sampling drastically improves the approximation quality of 1-PolySHAP, due to its equivalence to 2-PolySHAP. Since 3-PolySHAP often performs very well in this metric, we do not observe strong differences between 3-PolySHAP and 4-PolySHAP in both sampling settings.

In Figure 13, we report SpearmanCorrelation for the PolySHAP variants and the baseline methods under paired sampling. Again, we observe state-of-the-art performance for PolySHAP and the RegressionMSR baseline. PolySHAP substantially improves, if the budget allows to capture order-3 interactions.

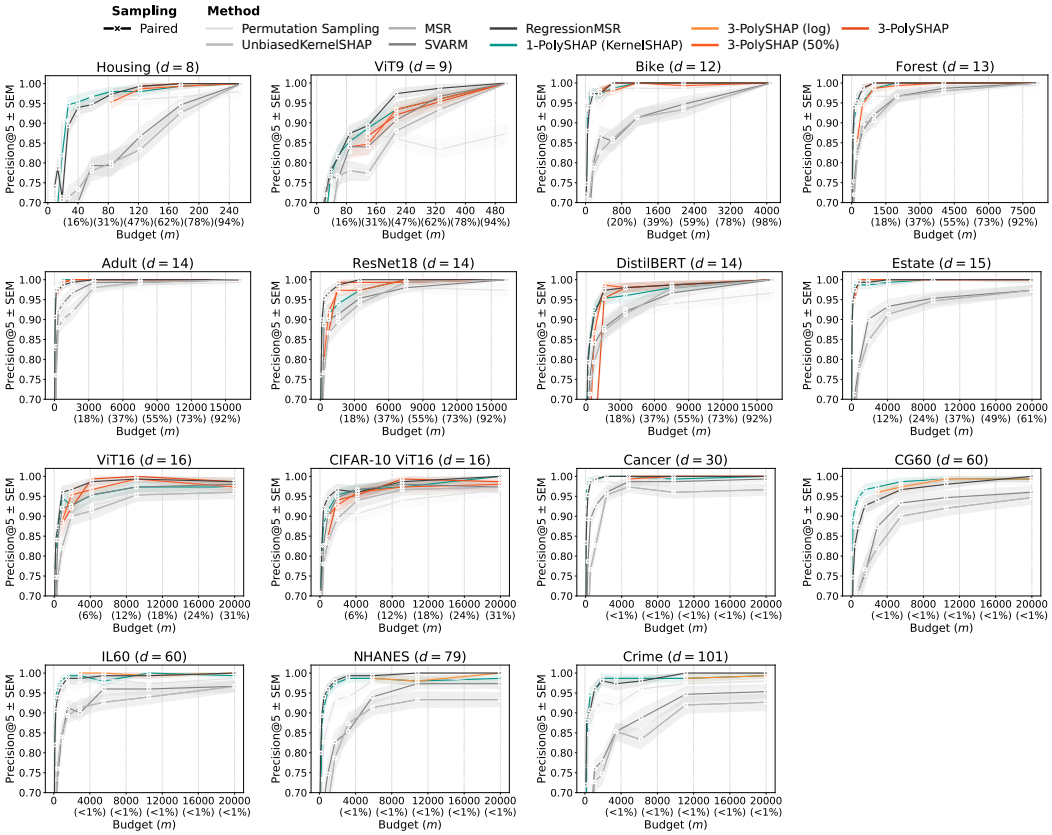


Figure 10: Approximation quality of PolySHAP variants and baselines measured by Precision@5 ($\pm$ SEM) for paired sampling

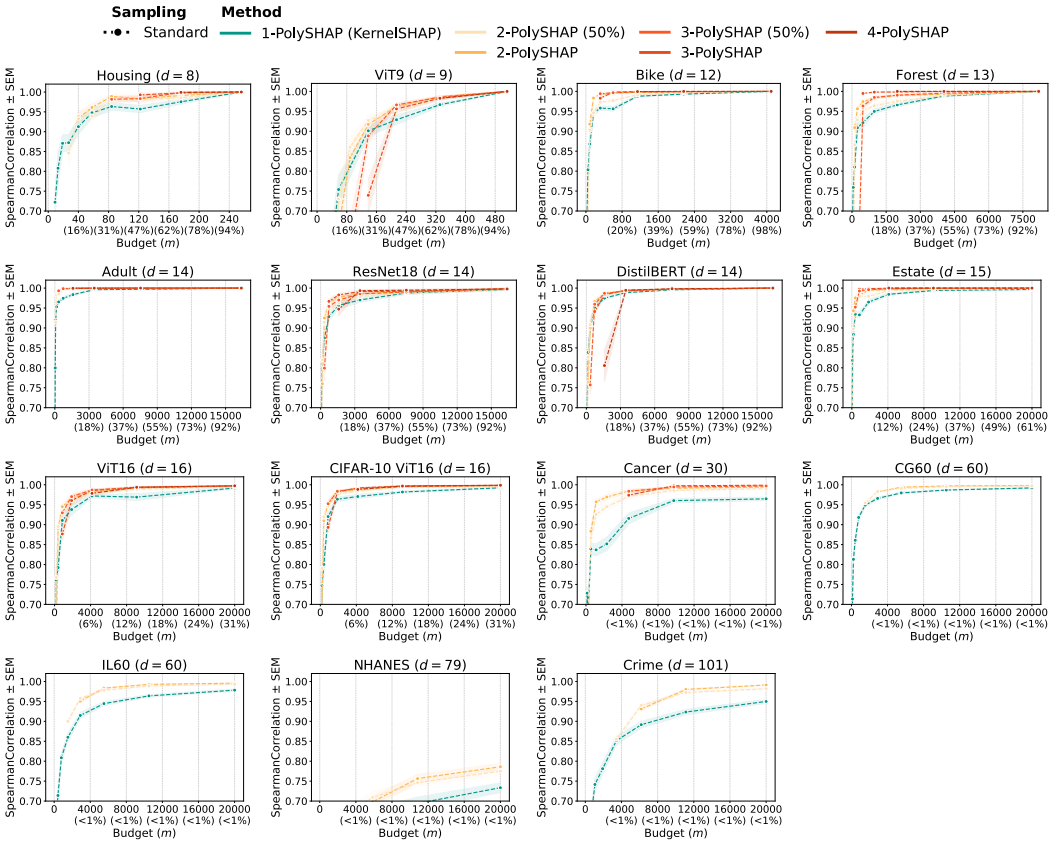


Figure 11: Approximation quality measured by SpearmanCorrelation ($\pm$ SEM) for varying budget (m) on different games. Adding interactions in PolySHAP can substantially improve approximation quality

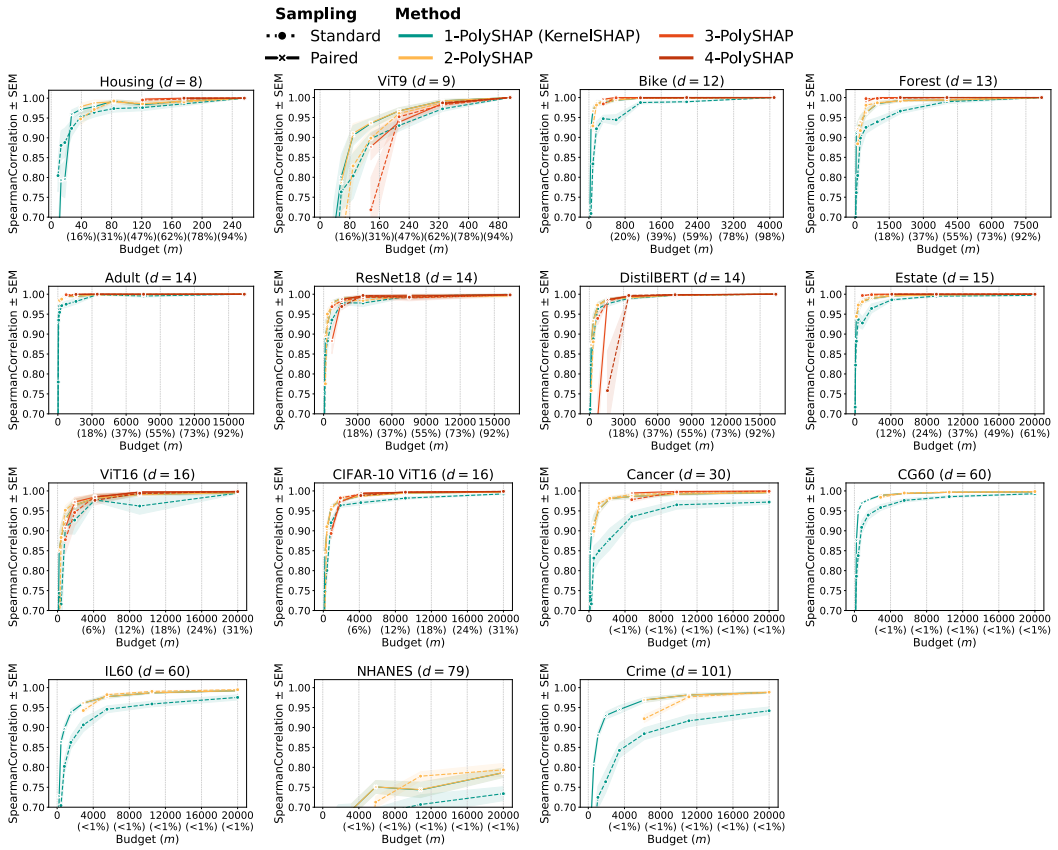


Figure 12: Approximation quality measured by SpearmanCorrelation (± SEM) for standard (dotted) and paired (solid) sampling. Under paired sampling, 2-PolySHAP marginally improves, whereas KernelSHAP substantially improves due to its equivalence to 2-PolySHAP

B.5 Runtime Analysis

In this section, we analyze the runtime of PolySHAP and the RegressionMSR baseline, since both methods approximate the game values, and subsequently extract Shapley value estimates.

Figure 14 reports the runtime in seconds (log-scale) of PolySHAP and RegressionMSR for the spent budget on different explanation games. As expected, we observe a *linear* relationship between the budget m and the computation time in PolySHAP, indicated by the overlapping linear fits (dashed lines). Overall, the computational overhead of the computations executed in RegressionMSR and PolySHAP variants after game evaluations will be negligible in most application settings.

Complexity of Evaluations. In realistic application settings, the runtime for game evaluations should be considered a main driver of computational complexity of PolySHAP and RegressionMSR. This is verified by the CIFAR10 ViT16 game in Figure 14, a), which requires one model call of the ViT16 for each game evaluation. The computational difference between RegressionMSR and all PolySHAP variants are thereby negligible.

For the runtime of the path-dependent tree games, reported in Figure 14, b) the game evaluations require only a single pass through the random forests, which becomes negligible with increasing dimensionality.

Complexity of Computation. The computational overhead of RegressionMSR and PolySHAP variants besides the game evaluations is negligible in many application settings. However, there is an impact on runtime for the higher-order k-PolySHAP variants, due to the increasing number

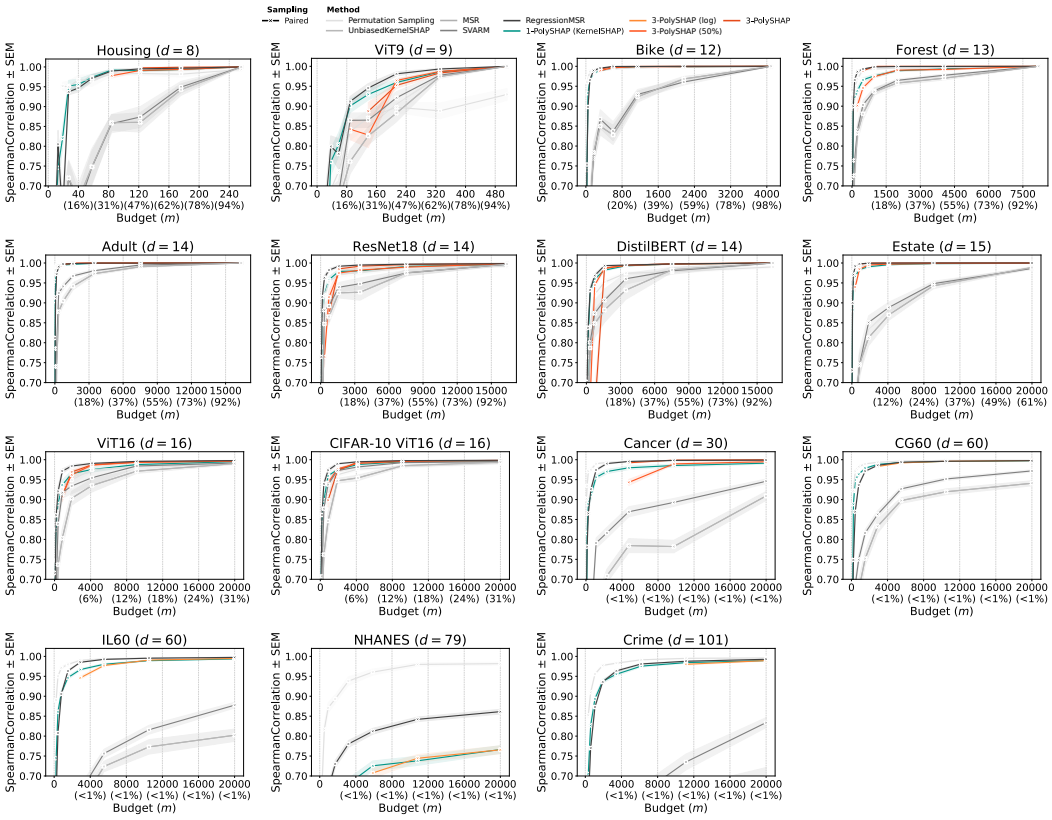
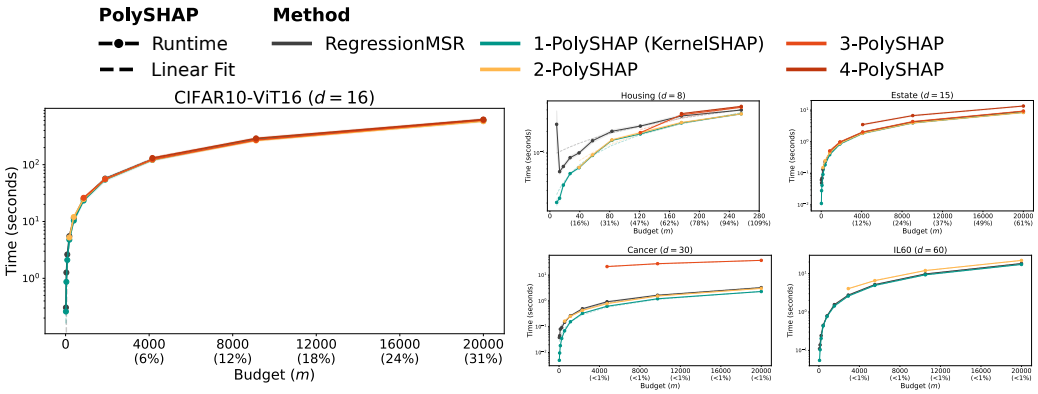


Figure 13: Approximation quality of PolySHAP variants and baselines measured by SpearmanCorrelation (± SEM) for paired sampling



a) Real-World ViT Inference Game **b) Path-dependent Tree Games**

Figure 14: Runtime in seconds (log-scale) of PolySHAP and RegressionMSR for varying budgets (m) of **a)** a real-world ViT inference game on CIFAR10, and **b)** selected path-dependent tree games of varying dimensionality. The runtime increases linearly with the budget m, indicated by the linear fit (dashed line). As expected in practice, the runtime of the real-world ViT inference game on CIFAR10 is dominated by the model calls required for each budget. The computational difference of the approximators are negligible. For path-dependent tree games that require only a single tree traversal per game evaluation, the runtime increases for higher-order PolySHAP due to the increasing number of regression variables with stronger effects in high-dimensional games.

of regression variables that yield a *polynomial* increase of computation time. For the tree-path dependent games in Figure 14, b) this effect is visible due to the very efficient computation of game values.

The RegressionMSR method utilizes the XGBoost library (Chen & Guestrin, 2016), which scales well to high-dimensional problems, indicated by the low runtime observed in Figure 14. The runtime of these computations is generally higher than 1- and 2-PolySHAP, but less than 3- and 4-PolySHAP for high-dimensional problems.

B.6 Additional Tables

Table 4: Summary statistics of the MSE error for ALL Shapley value estimators we consider with paired sampling. Increasing the degree of PolySHAP improves its performance, but k-PolySHAP requires budget $m \geq d_k = \mathcal{O}(k)$. RegressionMSR with XGBoost performs very well, except on games like CG60 or Crime where the decision tree struggles to approximate ν.

	Housing ($d=8$)	ViT9 ($d=9$)	Bike ($d=12$)	Forest ($d=13$)	Adult ($d=14$)	ResNet18 ($d=14$)	DistilBERT ($d=14$)	Estate ($d=15$)	ViT16 ($d=16$)	Cancer ($d=30$)	IL60 ($d=60$)	CG60 ($d=60$)	NHANES ($d=79$)	Crime ($d=101$)
m			1140	1988		1590			4156	4749		2900	3174	6188
Permutation Sampling														
Mean	1.7×10^{-4}	7.6×10^{-4}	7.4×10^{-1}	1.6×10^{-2}	3.3×10^{-7}	1.0×10^{-4}	5.7×10^{-4}	2.0×10^{-4}	3.7×10^{-5}	7.2×10^{-7}	7.5×10^{-5}	6.0×10^{-5}	3.1×10^{-3}	6.4×10^{-1}
1st Quartile	3.2×10^{-5}	2.9×10^{-4}	2.1×10^{-1}	4.6×10^{-3}	2.0×10^{-7}	2.1×10^{-5}	2.3×10^{-4}	3.5×10^{-5}	1.9×10^{-5}	4.2×10^{-7}	5.1×10^{-5}	5.0×10^{-5}	1.1×10^{-3}	1.8×10^{-1}
2nd Quartile	1.5×10^{-4}	5.7×10^{-4}	5.1×10^{-1}	1.3×10^{-2}	2.4×10^{-7}	3.6×10^{-5}	5.8×10^{-4}	8.6×10^{-5}	3.2×10^{-5}	7.0×10^{-7}	7.1×10^{-5}	5.7×10^{-5}	4.0×10^{-3}	3.4×10^{-1}
3rd Quartile	2.2×10^{-4}	1.0×10^{-3}	7.7×10^{-1}	1.6×10^{-2}	4.1×10^{-7}	2.1×10^{-4}	9.5×10^{-4}	1.9×10^{-4}	3.5×10^{-5}	8.5×10^{-7}	9.8×10^{-5}	7.6×10^{-5}	4.4×10^{-3}	6.7×10^{-1}
1-PolySHAP (KernelSHAP)														
Mean	5.5×10^{-6}	3.3×10^{-5}	4.0×10^{-2}	4.9×10^{-3}	1.2×10^{-7}	1.9×10^{-5}	1.0×10^{-4}	5.2×10^{-5}	1.5×10^{-5}	5.5×10^{-7}	4.7×10^{-5}	4.3×10^{-5}	2.6×10^{-3}	5.7×10^{-1}
1st Quartile	1.3×10^{-6}	6.7×10^{-6}	1.7×10^{-2}	4.5×10^{-4}	4.6×10^{-8}	5.5×10^{-6}	3.8×10^{-5}	2.2×10^{-5}	7.5×10^{-6}	9.0×10^{-8}	3.6×10^{-5}	3.0×10^{-5}	1.1×10^{-3}	1.1×10^{-1}
2nd Quartile	4.3×10^{-6}	1.1×10^{-5}	2.5×10^{-2}	1.2×10^{-3}	9.3×10^{-8}	1.3×10^{-5}	1.1×10^{-4}	4.0×10^{-5}	1.5×10^{-5}	2.1×10^{-7}	4.0×10^{-5}	3.7×10^{-5}	2.3×10^{-3}	1.9×10^{-1}
3rd Quartile	5.8×10^{-6}	6.6×10^{-5}	4.8×10^{-2}	8.1×10^{-3}	1.1×10^{-7}	2.9×10^{-5}	1.3×10^{-4}	5.8×10^{-5}	2.2×10^{-5}	7.6×10^{-7}	5.3×10^{-5}	4.2×10^{-5}	4.2×10^{-3}	6.1×10^{-1}
2-PolySHAP (50%)														
Mean	5.5×10^{-6}	3.3×10^{-5}	4.0×10^{-2}	4.9×10^{-3}	1.2×10^{-7}	1.9×10^{-5}	1.0×10^{-4}	5.2×10^{-5}	1.5×10^{-5}	5.4×10^{-7}	4.7×10^{-5}	4.3×10^{-5}	2.6×10^{-3}	5.7×10^{-1}
1st Quartile	1.3×10^{-6}	6.7×10^{-6}	1.7×10^{-2}	4.5×10^{-4}	4.6×10^{-8}	5.5×10^{-6}	3.8×10^{-5}	2.2×10^{-5}	7.5×10^{-6}	9.2×10^{-8}	3.6×10^{-5}	3.0×10^{-5}	1.1×10^{-3}	1.1×10^{-1}
2nd Quartile	4.3×10^{-6}	1.1×10^{-5}	2.5×10^{-2}	1.2×10^{-3}	9.3×10^{-8}	1.3×10^{-5}	1.1×10^{-4}	4.0×10^{-5}	1.5×10^{-5}	2.1×10^{-7}	4.0×10^{-5}	3.7×10^{-5}	2.3×10^{-3}	1.9×10^{-1}
3rd Quartile	5.8×10^{-6}	6.6×10^{-5}	4.8×10^{-2}	8.1×10^{-3}	1.1×10^{-7}	2.9×10^{-5}	1.3×10^{-4}	5.8×10^{-5}	2.2×10^{-5}	7.6×10^{-7}	5.3×10^{-5}	4.2×10^{-5}	4.2×10^{-3}	6.1×10^{-1}
2-PolySHAP														
Mean	5.5×10^{-6}	3.3×10^{-5}	4.0×10^{-2}	4.9×10^{-3}	1.2×10^{-7}	1.9×10^{-5}	1.0×10^{-4}	5.2×10^{-5}	1.5×10^{-5}	5.4×10^{-7}	4.7×10^{-5}	4.3×10^{-5}	2.6×10^{-3}	5.7×10^{-1}
1st Quartile	1.3×10^{-6}	6.7×10^{-6}	1.7×10^{-2}	4.5×10^{-4}	4.6×10^{-8}	5.5×10^{-6}	3.8×10^{-5}	2.2×10^{-5}	7.5×10^{-6}	9.2×10^{-8}	3.6×10^{-5}	3.0×10^{-5}	1.1×10^{-3}	1.1×10^{-1}
2nd Quartile	4.3×10^{-6}	1.1×10^{-5}	2.5×10^{-2}	1.2×10^{-3}	9.3×10^{-8}	1.3×10^{-5}	1.1×10^{-4}	4.0×10^{-5}	1.5×10^{-5}	2.1×10^{-7}	4.0×10^{-5}	3.7×10^{-5}	2.3×10^{-3}	1.9×10^{-1}
3rd Quartile	5.8×10^{-6}	6.6×10^{-5}	4.8×10^{-2}	8.1×10^{-3}	1.1×10^{-7}	2.9×10^{-5}	1.3×10^{-4}	5.8×10^{-5}	2.2×10^{-5}	7.6×10^{-7}	5.3×10^{-5}	4.2×10^{-5}	4.2×10^{-3}	6.1×10^{-1}
3-PolySHAP (50%)														
Mean	4.1×10^{-6}	2.7×10^{-5}	3.5×10^{-2}	1.7×10^{-3}	6.5×10^{-8}	9.1×10^{-6}	5.3×10^{-5}	1.0×10^{-5}	7.9×10^{-6}	2.0×10^{-6}				
1st Quartile	6.5×10^{-7}	8.2×10^{-6}	7.6×10^{-3}	9.8×10^{-5}	2.5×10^{-8}	2.4×10^{-6}	2.9×10^{-5}	5.1×10^{-6}	3.3×10^{-6}	4.8×10^{-7}				
2nd Quartile	1.4×10^{-6}	1.9×10^{-5}	1.4×10^{-2}	2.1×10^{-4}	3.9×10^{-8}	4.0×10^{-6}	4.2×10^{-5}	8.5×10^{-6}	4.9×10^{-6}	8.5×10^{-7}				
3rd Quartile	3.4×10^{-6}	3.6×10^{-5}	3.5×10^{-2}	7.2×10^{-4}	8.8×10^{-8}	1.5×10^{-5}	6.8×10^{-5}	1.1×10^{-5}	9.6×10^{-6}	1.9×10^{-6}				
3-PolySHAP														
Mean	4.0×10^{-8}	2.7×10^{-5}	8.0×10^{-4}	4.3×10^{-7}	1.9×10^{-9}	6.9×10^{-6}	7.5×10^{-5}	2.7×10^{-8}	5.7×10^{-6}	3.2×10^{-7}				
1st Quartile	5.5×10^{-9}	5.6×10^{-6}	1.2×10^{-4}	1.7×10^{-7}	4.2×10^{-10}	1.1×10^{-6}	4.2×10^{-5}	1.4×10^{-8}	2.1×10^{-6}	1.2×10^{-7}				
2nd Quartile	1.5×10^{-8}	1.8×10^{-5}	1.8×10^{-4}	3.7×10^{-7}	1.3×10^{-9}	2.5×10^{-6}	6.6×10^{-5}	2.1×10^{-8}	3.0×10^{-6}	1.6×10^{-7}				
3rd Quartile	6.1×10^{-8}	4.0×10^{-5}	8.5×10^{-4}	7.1×10^{-7}	2.2×10^{-9}	8.2×10^{-6}	1.1×10^{-4}	3.0×10^{-8}	6.5×10^{-6}	2.1×10^{-7}				
4-PolySHAP														
Mean	4.0×10^{-8}	2.7×10^{-5}	8.0×10^{-4}	4.3×10^{-7}	1.9×10^{-9}	6.9×10^{-6}	7.5×10^{-5}	2.7×10^{-8}	5.7×10^{-6}					
1st Quartile	5.5×10^{-9}	5.6×10^{-6}	1.2×10^{-4}	1.7×10^{-7}	4.2×10^{-10}	1.1×10^{-6}	4.2×10^{-5}	1.4×10^{-8}	2.1×10^{-6}					
2nd Quartile	1.5×10^{-8}	1.8×10^{-5}	1.8×10^{-4}	3.7×10^{-7}	1.3×10^{-9}	2.5×10^{-6}	6.6×10^{-5}	2.1×10^{-8}	3.0×10^{-6}					
3rd Quartile	6.1×10^{-8}	4.0×10^{-5}	8.5×10^{-4}	7.1×10^{-7}	2.2×10^{-9}	8.2×10^{-6}	1.1×10^{-4}	3.0×10^{-8}	6.5×10^{-6}					
RegressionMSR														
Mean	4.3×10^{-7}	1.2×10^{-5}	2.0×10^{-3}	1.3×10^{-5}	6.0×10^{-8}	2.1×10^{-6}	4.1×10^{-5}	4.5×10^{-7}	3.4×10^{-6}	9.0×10^{-8}	5.8×10^{-5}	2.5×10^{-4}	5.3×10^{-4}	6.0×10^{-1}
1st Quartile	9.3×10^{-8}	4.2×10^{-6}	6.4×10^{-4}	9.4×10^{-6}	3.6×10^{-8}	2.2×10^{-7}	1.8×10^{-5}	1.9×10^{-7}	6.8×10^{-7}	4.8×10^{-8}	4.8×10^{-5}	2.0×10^{-4}	2.0×10^{-4}	2.4×10^{-1}
2nd Quartile	1.4×10^{-7}	8.8×10^{-6}	1.2×10^{-3}	1.1×10^{-5}	5.2×10^{-8}	1.9×10^{-6}	3.3×10^{-5}	4.7×10^{-7}	1.9×10^{-6}	8.7×10^{-8}	6.5×10^{-5}	2.5×10^{-4}	4.2×10^{-4}	3.3×10^{-1}
3rd Quartile	3.2×10^{-7}	1.8×10^{-5}	1.4×10^{-3}	1.5×10^{-5}	7.9×10^{-8}	3.0×10^{-6}	4.2×10^{-5}	6.6×10^{-7}	3.2×10^{-6}	1.2×10^{-7}	6.8×10^{-5}	2.6×10^{-4}	5.9×10^{-4}	6.7×10^{-1}

Table 5: Summary statistics of the MSE error for ALL Shapley value estimators we consider with standard (not paired) sampling. Increasing the degree of PolySHAP improves its performance, but k-PolySHAP requires budget $m \geq d_k = \mathcal{O}(k)$. RegressionMSR with XGBoost performs very well, except on games like CG60 or Crime where the decision tree struggles to approximate ν.

	Housing ($d=8$)	ViT9 ($d=9$)	Bike ($d=12$)	Forest ($d=13$)	Adult ($d=14$)	ResNet18 ($d=14$)	DistilBERT ($d=14$)	Estate ($d=15$)	ViT16 ($d=16$)	Cancer ($d=30$)	IL60 ($d=60$)	CG60 ($d=60$)	NHANES ($d=79$)	Crime ($d=101$)
m			1140	1988		1590			4156	4749		2900	3174	6188
Permutation Sampling														
Mean	1.2×10^{-3}	1.2×10^{-3}	5.6×10^{0}	2.7×10^{-2}	2.6×10^{-6}	1.0×10^{-4}	6.3×10^{-4}	5.5×10^{-4}	6.0×10^{-5}	5.5×10^{-6}	1.0×10^{-4}	1.0×10^{-4}	7.6×10^{-3}	3.9×10^{0}
1st Quartile	3.8×10^{-4}	3.1×10^{-4}	1.3×10^{0}	6.3×10^{-3}	6.9×10^{-7}	4.7×10^{-5}	3.1×10^{-4}	1.0×10^{-4}	1.6×10^{-5}	9.5×10^{-7}	7.7×10^{-5}	8.0×10^{-5}	2.6×10^{-3}	8.8×10^{-1}
2nd Quartile	9.0×10^{-4}	9.9×10^{-4}	1.4×10^{0}	3.0×10^{-2}	1.5×10^{-6}	8.5×10^{-5}	5.4×10^{-4}	2.3×10^{-4}	4.7×10^{-5}	2.3×10^{-6}	9.7×10^{-5}	1.0×10^{-4}	5.2×10^{-3}	1.2×10^{0}
3rd Quartile	1.4×10^{-3}	1.4×10^{-3}	3.2×10^{0}	3.9×10^{-2}	3.2×10^{-6}	1.6×10^{-4}	7.2×10^{-4}	1.1×10^{-4}	8.6×10^{-5}	3.7×10^{-6}	1.3×10^{-4}	1.2×10^{-4}	1.0×10^{-2}	1.4×10^{0}
1-PolySHAP (KernelSHAP)														
Mean	4.2×10^{-5}	9.3×10^{-5}	8.4×10^{-1}	4.4×10^{-2}	1.3×10^{-6}	4.6×10^{-5}	1.5×10^{-4}	2.7×10^{-4}	2.9×10^{-5}	4.0×10^{-6}	1.8×10^{-4}	2.3×10^{-4}	1.2×10^{-2}	3.5×10^{0}
1st Quartile	1.9×10^{-5}	2.7×10^{-5}	3.4×10^{-1}	8.1×10^{-3}	1.9×10^{-7}	1.0×10^{-5}	6.4×10^{-5}	1.4×10^{-4}	1.3×10^{-5}	1.4×10^{-6}	9.6×10^{-5}	1.1×10^{-4}	3.5×10^{-3}	8.6×10^{-1}
2nd Quartile	3.9×10^{-5}	5.1×10^{-5}	5.6×10^{-1}	2.5×10^{-2}	7.1×10^{-7}	2.5×10^{-5}	1.2×10^{-4}	2.0×10^{-4}	2.7×10^{-5}	2.0×10^{-6}	1.4×10^{-4}	1.4×10^{-4}	7.6×10^{-3}	1.4×10^{0}
3rd Quartile	6.2×10^{-5}	1.5×10^{-4}	9.7×10^{-1}	3.5×10^{-2}	1.6×10^{-6}	8.8×10^{-5}	1.8×10^{-4}	2.7×10^{-4}	4.8×10^{-5}	4.1×10^{-6}	2.3×10^{-4}	1.8×10^{-4}	2.2×10^{-2}	1.8×10^{0}
2-PolySHAP (50%)														
Mean	2.2×10^{-5}	4.0×10^{-5}	2.1×10^{-1}	5.9×10^{-3}	3.0×10^{-7}	1.8×10^{-5}	8.0×10^{-5}	7.9×10^{-5}	1.2×10^{-5}	7.0×10^{-7}	7.3×10^{-5}	5.8×10^{-5}	5.0×10^{-3}	1.2×10^{0}
1st Quartile	7.5×10^{-6}	1.0×10^{-5}	7.5×10^{-2}	1.4×10^{-3}	2.0×10^{-7}	6.7×10^{-6}	2.3×10^{-5}	1.6×10^{-5}	6.0×10^{-6}	2.3×10^{-7}	5.1×10^{-5}	4.1×10^{-5}	1.7×10^{-3}	2.5×10^{-1}
2nd Quartile	1.6×10^{-5}	2.1×10^{-5}	1.7×10^{-1}	3.7×10^{-3}	3.1×10^{-7}	1.3×10^{-5}	4.9×10^{-5}	3.5×10^{-5}	1.0×10^{-5}	3.2×10^{-7}	6.7×10^{-5}	6.3×10^{-5}	3.9×10^{-3}	3.8×10^{-1}
3rd Quartile	2.4×10^{-5}	3.2×10^{-5}	3.5×10^{-1}	7.0×10^{-3}	3.8×10^{-7}	2.6×10^{-5}	9.0×10^{-5}	1.1×10^{-4}	1.7×10^{-5}	1.0×10^{-6}	8.4×10^{-5}	7.4×10^{-5}	9.0×10^{-3}	9.1×10^{-1}
2-PolySHAP														
Mean	1.9×10^{-5}	3.4×10^{-5}	3.6×10^{-2}	1.5×10^{-3}	7.6×10^{-8}	1.4×10^{-5}	5.3×10^{-5}	2.7×10^{-5}	9.7×10^{-6}	2.9×10^{-7}	1.1×10^{-4}	9.5×10^{-5}	2.4×10^{-1}	3.5×10^{0}
1st Quartile	2.3×10^{-6}	1.3×10^{-5}	1.7×10^{-2}	4.0×10^{-4}	4.0×10^{-8}	2.5×10^{-6}	1.5×10^{-5}	1.3×10^{-5}	4.6×10^{-6}	6.8×10^{-8}	7.8×10^{-5}	6.1×10^{-5}	9.7×10^{-2}	5.7×10^{-1}
2nd Quartile	7.6×10^{-6}	2.0×10^{-5}	3.9×10^{-2}	1.2×10^{-3}	5.9×10^{-8}	1.0×10^{-5}	3.8×10^{-5}	2.6×10^{-5}	8.7×10^{-6}	1.5×10^{-7}	9.8×10^{-5}	9.2×10^{-5}	2.0×10^{-1}	1.2×10^{0}
3rd Quartile	1.6×10^{-5}	4.0×10^{-5}	4.8×10^{-2}	1.6×10^{-3}	1.2×10^{-7}	2.0×10^{-5}	9.1×10^{-5}	3.2×10^{-5}	1.4×10^{-5}	3.9×10^{-7}	1.5×10^{-4}	1.1×10^{-4}	2.5×10^{-1}	3.9×10^{0}
3-PolySHAP (50%)														
Mean	2.7×10^{-5}	3.9×10^{-5}	1.9×10^{-2}	6.6×10^{-4}	4.4×10^{-8}	1.3×10^{-5}	5.5×10^{-5}	5.8×10^{-6}	7.9×10^{-6}	2.3×10^{-7}				
1st Quartile	1.1×10^{-6}	5.9×10^{-6}	2.7×10^{-3}	6.6×10^{-5}	2.2×10^{-8}	1.4×10^{-6}	1.4×10^{-5}	1.1×10^{-6}	2.7×10^{-6}	6.6×10^{-8}				
2nd Quartile	2.3×10^{-6}	2.1×10^{-5}	8.1×10^{-3}	2.9×10^{-4}	3.6×10^{-8}	3.2×10^{-6}	3.8×10^{-5}	1.9×10^{-6}	4.5×10^{-6}	1.1×10^{-7}				
3rd Quartile	2.5×10^{-5}	4.1×10^{-5}	3.8×10^{-2}	6.6×10^{-4}	6.4×10^{-8}	1.7×10^{-5}	6.3×10^{-5}	8.5×10^{-6}	1.4×10^{-5}	2.4×10^{-7}				
3-PolySHAP														
Mean	9.7×10^{-7}	2.3×10^{-5}	4.0×10^{-3}	1.8×10^{-5}	9.1×10^{-9}	1.3×10^{-5}	6.5×10^{-5}	1.1×10^{-6}	6.5×10^{-6}	2.0×10^{-6}				
1st Quartile	1.1×10^{-7}	1.1×10^{-5}	1.0×10^{-3}	3.2×10^{-6}	3.4×10^{-9}	1.2×10^{-6}	3.3×10^{-5}	4.7×10^{-7}	2.0×10^{-6}	2.1×10^{-7}				
2nd Quartile	2.4×10^{-7}	1.2×10^{-5}	1.4×10^{-3}	6.9×10^{-6}	6.0×10^{-9}	4.8×10^{-6}	5.2×10^{-5}	6.0×10^{-7}	3.6×10^{-6}	3.6×10^{-7}				
3rd Quartile	8.0×10^{-7}	2.9×10^{-5}	3.8×10^{-3}	1.7×10^{-5}	1.3×10^{-8}	1.4×10^{-5}	8.6×10^{-5}	1.3×10^{-6}	1.0×10^{-5}	6.4×10^{-7}				
4-PolySHAP														
Mean	8.3×10^{-7}	6.7×10^{-5}	1.6×10^{-3}	1.2×10^{-6}	2.3×10^{-8}	6.1×10^{-5}	3.5×10^{-3}	2.0×10^{-8}	1.2×10^{-5}					
1st Quartile	2.3×10^{-8}	8.4×10^{-6}	4.7×10^{-4}	1.6×10^{-7}	3.8×10^{-9}	9.1×10^{-6}	1.4×10^{-3}	8.8×10^{-9}	5.0×10^{-6}					
2nd Quartile	8.4×10^{-8}	3.2×10^{-5}	6.0×10^{-4}	5.4×10^{-7}	1.6×10^{-8}	2.7×10^{-5}	2.4×10^{-3}	1.3×10^{-8}	9.3×10^{-6}					
3rd Quartile	5.8×10^{-7}	7.2×10^{-5}	9.8×10^{-4}	9.1×10^{-7}	3.0×10^{-8}	7.7×10^{-5}	3.9×10^{-3}	2.3×10^{-8}	1.6×10^{-5}					
RegressionMSR														
Mean	8.6×10^{-7}	1.3×10^{-5}	2.6×10^{-3}	1.5×10^{-5}	7.0×10^{-8}	3.5×10^{-6}	6.5×10^{-5}	4.6×10^{-7}	1.2×10^{-5}	7.3×10^{-8}	6.5×10^{-5}	2.4×10^{-4}	4.5×10^{-4}	5.8×10^{-1}
1st Quartile	4.0×10^{-7}	4.0×10^{-6}	8.7×10^{-4}	6.9×10^{-6}	5.6×10^{-8}	2.7×10^{-7}	2.5×10^{-5}	2.7×10^{-7}	1.6×10^{-6}	3.6×10^{-8}	5.3×10^{-5}	1.8×10^{-4}	1.9×10^{-4}	2.8×10^{-1}
2nd Quartile	8.7×10^{-7}	1.1×10^{-5}	1.2×10^{-3}	1.2×10^{-5}	6.2×10^{-8}	1.6×10^{-6}	6.3×10^{-5}	3.9×10^{-7}	2.4×10^{-6}	6.7×10^{-8}	6.6×10^{-5}	2.4×10^{-4}	4.4×10^{-4}	3.4×10^{-1}
3rd Quartile	1.1×10^{-6}	2.0×10^{-5}	2.8×10^{-3}	2.3×10^{-5}	9.2×10^{-8}	3.7×10^{-6}	9.3×10^{-5}	6.7×10^{-7}	7.9×10^{-6}	1.1×10^{-7}	7.5×10^{-5}	2.5×10^{-4}	5.7×10^{-4}	6.0×10^{-1}

C USAGE OF LARGE LANGUAGE MODELS (LLMS)

In this work, we used large language models (LLMs) for suggestions regarding refinement of writing, e.g. grammar, clarity and conciseness.