TreePII: Efficient Computation of Higher-Order Probabilistic Interaction Indices in Tree Ensembles

Junho Choi
Kim Jaechul Graduate School of AI
KAIST
Daejeon, South Korea
junho.choi@kaist.ac.kr

Jaesik Choi
Kim Jaechul Graduate School of AI
KAIST
Daejeon, South Korea
jaesik.choi@kaist.ac.kr

Abstract

While higher-order interaction indices offer deep insights into the synergies and redundancies within tree ensembles, their exact computation is hindered by combinatorial bottlenecks. Recent advancements efficiently compute symmetric interactions, but struggle to scale for asymmetric Probabilistic Interaction Indices (PIIs), which are crucial for advanced tasks such as incorporating feature hierarchy structures. To bridge this gap, we introduce TreePII, a novel algorithm that computes exact, any-order path-induced PIIs by integrating the partial derivatives of multilinear extension using interpolatory quadrature. TreePII effectively bypasses traditional computational hurdles, demonstrating both theoretical and empirical improvements over existing baselines and opening new avenues for scalable analysis of tree ensembles. TreePII provides a foundation to make the exploration of asymmetric interactions computationally feasible for tree models.

1 Introduction

Tree ensembles, such as Random Forest [1] and Gradient Boosted Trees [2], are widely deployed in domains using tabular data due to their robust predictive performance [3]. With an increasing demand for transparency [4, 5] and the rise of related regulations [6], feature attribution methods based on cooperative game theory have become standard tools for interpreting these models [7]. However, explaining individual feature contributions is often insufficient to capture the underlying logic of complex models [8, 9]: features often gain new meaning through *interactions* in real-world tasks like natural language processing [10, 11]. Interaction indices (IIs) [12], which measure the synergies and redundancies between groups of features, can offer a much richer understanding. However, they are also computationally heavy since the number of possible feature subsets for higher-order interactions can grow combinatorially if performed naively.

Recent advancements, most notably TreeSHAP-IQ [9], have significantly improved the computation of Cardinal Interaction Indices (CIIs) by extending the methodology of Linear TreeSHAP [13] (a 1st order approach). However, existing methods are generally limited to *symmetric* IIs, which treat each feature with same weight during the attribution process. They cannot efficiently compute *asymmetric* Probabilistic Interaction Indices (PIIs) such as generalized Banzhaf [14] or weighted Shapley [15, 16] interactions, which give different priorities to different features. Asymmetric PIIs are critical for many real applications [17, 18, 19, 20, 21], making it imperative to be able to compute their interactions efficiently.

To bridge this gap, we introduce TreePII, a novel framework for computing exact, any-order PIIs in tree ensembles. TreePII operates by approaching PIIs as integrals of *marginal interactions* [12] - contributions made by a *subset* of features. TreePII calculates marginal interactions efficiently through depth-first search and combines it with interpolatory quadrature to effectively bypass tradi-

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

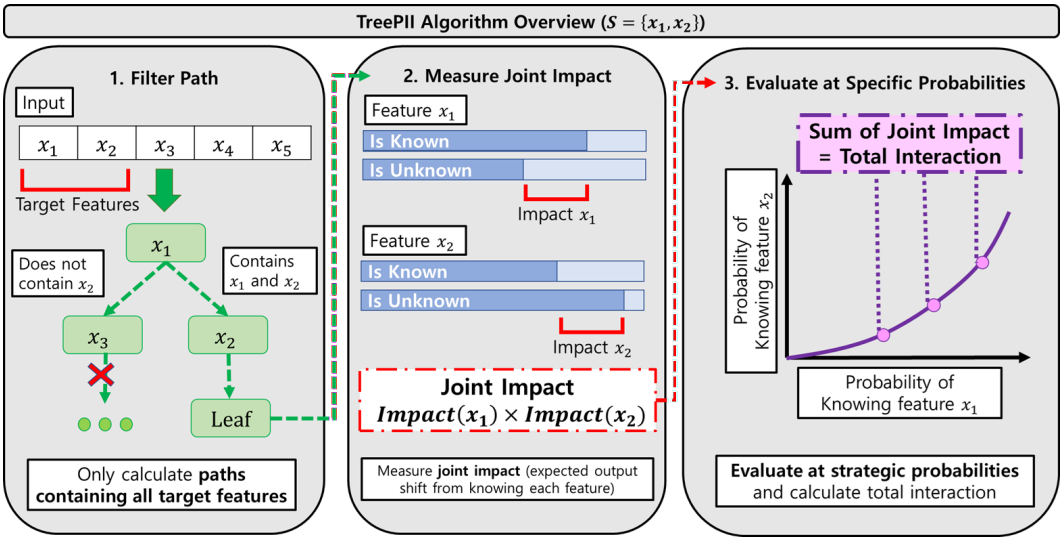

Figure 1: Illustration of using TreePII for interaction between x_1 and x_2 given an input of 5 features. (1) We only select paths in the tree that contain x_1 and x_2 through depth-first search. (2) For each selected path, we calculate the *marginal interaction*, which is the joint impact (shift in expected output) from knowing x_1 and x_2, as the product of the individual impacts. (3) PII is the sum of these joint impacts calculated at strategically chosen probabilities of knowing x_1 and x_2.

tional combinatorial bottlenecks (Figure 1). Ultimately, TreePII provides a foundation to make the exploration of asymmetric interactions computationally feasible for tree models.

Our main contributions are as follows:

- We introduce TreePII, an efficient algorithm for computing any-order Probabilistic Interaction Indices in tree ensembles.
- We derive the time and memory complexity of TreePII for both symmetric and asymmetric interaction indices, and demonstrate that TreePII shows significant runtime improvements over baseline approaches for both types of interactions.
- We show potential applications of TreePII, especially for asymmetric interactions.

2 Preliminaries

In this section, we review explanations based on cooperative game theory and Interaction Indices (IIs). We also discuss how these indices have been applied in explaining tree ensembles.

2.1 Notations

We use the following notation throughout this paper. A summary table is provided in Appendix A.

Inputs. Let $N := \{1, .., n\}$ be a set of features. The feature vector $\mathbf{x} \in \mathbb{R}^n$ is a single instance of these features, where x_i is the value of feature i in $\mathbf{x}$ and x_S is the values of features $i \in S \subseteq N$, $k = |S|$. A background dataset $\mathbf{B} \in \mathbb{R}^{r \times n}$ consists of r instances of dimension n.

Multilinear Extensions and Cooperative Games. Let $\mathbf{p} \in [0, 1]^n$ be the vector of inclusion probabilities (i.e., feature i joins a coalition with probability p_i) and $\mathbf{z} \in \{0, 1\}^n$ be the vector of binary indicator for inclusion (i.e., $z_i = 1$ if feature i joins a coalition and 0 otherwise). A multilinear extension of a cooperative game's value function g is defined as [22, 23]:

$$G(\mathbf{p}) = \sum_{S \subseteq N} g(\mathbf{z}) \prod_{i \in S} p_i \prod_{i \in N \setminus S} (1 - p_i) \tag{1}$$

$G(\mathbf{p})$ can be interpreted as the expected value of $g(\mathbf{z})$ over all coalitions. The expected marginal contribution of feature i is the partial derivative of $G(\mathbf{p})$ with respect to p_i. More generally, the k-th order expected marginal interaction $G_S(\mathbf{p})$ of subset S is the higher-order partial derivative with respect to $p_i, i \in S$, i.e.,:

$$G_S(\mathbf{p}) = \left(\prod_{i \in S} \frac{\partial}{\partial p_i} \right) G(\mathbf{p}) = \sum_{T \subseteq N \setminus S} \Delta_S(T) \prod_{i \in T} p_i \prod_{i \in N \setminus (S \cup T)} (1 - p_i) \tag{2}$$

where $\Delta_S(T) = \sum_{V \subseteq S} (-1)^{|V|-|S|} g(\mathbf{z}_{V \cup T})$ is referred to as the discrete derivative.

Trees. Let $\mathcal{T}$ be a binary tree with non-leaf nodes $v \in \mathcal{V}$, edges $e \in E$ and leaves $l \in \mathcal{L}$ based on single-variable splits: each v performs a split using feature i_v and threshold ν_v. Each leaf l has a value $f(l)$. We descend to left child a_v if $x_i \leq \nu_v$ and to right child b_v otherwise, and return $f(l)$ upon reaching a leaf l. Edge $e(v_p, v_c)$ connects parent node v_p to child v_c. The path taken to leaf l is represented as $\mathcal{P}(l) \subset E$ and unique features passed along the path is written as $F_l \subseteq N$. The *coverage* r_v of node v is the number of background instances arriving at v. $\mathcal{T}[v]$ is a subtree rooted at v. A tree ensemble consists of $n_\mathcal{T}$ trees. For notational simplicity, subsequent derivations focus on a single tree ($n_\mathcal{T} = 1$) since interactions are strictly additive across an ensemble.

2.2 Cooperative Game Theory and Interaction Indices as Explanations

In cooperative game theory, given players $i \in N$ and payoff function g, the goal is to allocate a fair payment to each player (referred to as the player's *value*). In machine learning, this framework translates to assigning the contribution of different entities (such as data instances or features) to a given task [7]. A player's value is defined as a weighted sum of their *marginal contribution*:

$$\phi_{g,i} = \sum_{S \subseteq N \setminus i} w(S)[g(\mathbf{z}_{S \cup i}) - g(\mathbf{z}_S)] \tag{3}$$

In terms of MLE, Equation 3 is an integral of the first partial derivative $G_i(\mathbf{p})$ with respect to an arbitrary measure Q.

The definition of $w(S)$ determines the *type* of value. The most well-known value is the Shapley value [24], which uses $w(S) = |S|!(n - |S| - 1)!/n!$. It is the unique solution that satisfies four axioms: linearity (values of a linear combination of games is equal to the linear combination of each game's values), symmetry (players with equal marginal contributions are allocated equal value), dummy (players with zero marginal contribution receive zero allocation), and efficiency (the sum of each player's value is equal to the total value $g(N) - g(\emptyset)$). Another solution is the generalized Banzhaf value [14, 25],[1] which is a more general form of the Banzhaf value [26]. The generalized Banzhaf value uses $w(S) = \prod_{j \in S} p_j \prod_{j \in N \setminus (S \cup i)} (1 - p_j)$. These values have been widely adapted to model explanations [27, 28, 29, 7, 15, 30, 31, 32, 33, 34, 25]. In terms of MLE, the generalized Banzhaf value is $G_i(\mathbf{p})$, while the Shapley value is its integral along the curve $p_i = t \forall i \in N, 0 \leq t \leq 1$.

Interaction indices (IIs) extend the concept of value to inter-player marginal contributions or *marginal interactions* [35, 12]. Specifically, II of a player subset S is defined as:

$$II_{g,S} = \sum_{T \subseteq N \setminus S} w(T) \Delta_S(T) \tag{4}$$

As with values, $w(S)$ determines the type of II. If $w(T)$ are probabilistic, i.e., $\sum_{T \subseteq N \setminus S} w(T) = 1, w(T) \geq 0$, the II is called a Probabilistic Interaction Index (PII). If $w(T)$ also depends only on the cardinality of T, i.e., $w(T) = f(|T|)$, it is called a Cardinal Interaction Index (CII). The Shapley Interaction index (SII) [35] and the generalized Banzhaf Interaction Index (g-BII) use Shapley and generalized Banzhaf weights, respectively. SII is a CII, while g-BII is a PII. Other variants of SII and BII [36, 37, 38, 39, 40] all fall under one of these categories. Like its first order counterpart, g-BII is equal to $G_S(\mathbf{p})$.

[1] While [14] refers to this definition as Weighted Banzhaf index, we call it generalized Banzhaf value to differentiate it from Weighted Banzhaf value in [25].

2.3 Game Theoretic Explanations for Trees

Cooperative game theory has been applied extensively in explaining trees, especially Shapley values [41, 42, 13] and Banzhaf values to a limited degree [33]. For IIs, most research focuses on model-agnostic applications [40]. The state-of-the-art solution for trees is TreeSHAP-IQ [9], which proposes an efficient algorithm for calculating any-order CIIs (including SII) for trees. While this method accelerates CII calculations significantly, they are still limited to *symmetric* IIs, which treat each feature to have the same inclusion probability $p_i = t$. However, real applications often require *asymmetric* treatment [17, 18, 19, 20, 21], which cannot be solved using CIIs.

We address the gap in interaction research in three steps: (a) presenting an algorithm for calculating the exact any-order g-BIIs for trees, (b) implementing an efficient extension of g-BII to asymmetric PIIs, and (c) showing its improvement over baseline approaches. It should be noted that TreePII evaluates interactions of a fixed trained tree under a missing-feature convention, explaining the model's learned logic rather than the effect of feature absence during training.

3 Generalized Banzhaf Interaction Index in Trees

In this section, we present the algorithm for efficient calculation of g-BII. Consider MLE in Equation 1. It is equivalent to the expected output of the model. For trees, this equivalence implies that:

$$G(\mathbf{p}) = \sum_{l \in \mathcal{L}} f(l) P(l)$$

Where $P(l)$ is the *expected* probability of reaching leaf l. $P(l)$ is the product of expected probability of passing all edges in $\mathcal{P}(l)$

$$P(l) = \prod_{e \in \mathcal{P}(l)} E(p_{e(v_p, v_c)})$$

$p_{e(v_p,v_c)}$ can take two values: probability when feature i used for split in v_p is known ($p_{i_{v_p},known}$), or unknown ($p_{i_{v_p},unknown}$). Typically, $p_{i_{v_p},known}$ is set to 1 and $p_{i_{v_p},unknown} = r_c/r_p$, where r_p and r_c are the coverages of parent and child nodes v_p and v_c. Since each feature i's inclusion in a coalition S is independent of other features, we can aggregate $p_{e(v_p,v_c)}$ by feature to get probability path weight of feature i for leaf l as $W_{l,i} = \prod_{v \in \mathcal{P}(l), i_v = i} p_{e(v,v_c)}$. Then:

$$P(l) = \prod_{i \in F_l} E(W_{l,i})$$

In g-BII, each feature i joins S with probability p_i. $W_{l,i}$ takes the value $PT_{l,i} = \prod_{v \in \mathcal{P}(l), i_v = i} p_{v,known}$ if i joins and $PB_{l,i} = \prod_{v \in \mathcal{P}(l), i_v = i} p_{v,unknown}$ otherwise (referred to as conditioned and background path weights, respectively). Therefore, $E(W_{l,i})$ can be written as:

$$E(W_{l,i}) = p_i PT_{l,i} + (1 - p_i) PB_{l,i} = p_i(PT_{l,i} - PB_{l,i}) + PB_{l,i}$$

Substituting back, we get the final form of $G(\mathbf{p})$ as:

$$G(\mathbf{p}) = \sum_{l \in \mathcal{L}} f(l) \prod_{i \in F_l} [p_i(PT_{l,i} - PB_{l,i}) + PB_{l,i}]$$

$G_S(\mathbf{p})$ then becomes:

$$G_S(\mathbf{p}) = \sum_{l \in \mathcal{L}} f(l) \left(\prod_{i \in S} \frac{\partial}{\partial p_i} \right) \prod_{i \in F_l} [p_i(PT_{l,i} - PB_{l,i}) + PB_{l,i}] \tag{5}$$

The key implication of Equation 5 is *path independence*: $G_S(\mathbf{p})$ can be calculated by calculating per-leaf marginal interaction ($G_{l,S}(\mathbf{p})$) using solely the values dependent on path $\mathcal{P}(l)$. Formally:

Theorem 1 (Path Independence of Marginal Interactions) *Let $\mathcal{T}$ be a decision tree and $S \subseteq N$ be a target subset of features. The total marginal interaction $G_S(p)$ decomposes additively across all leaves $l \in \mathcal{L}$:*

$$G_S(p) = \sum_{l \in \mathcal{L}} G_{l,S}(p) \tag{6}$$

The per-leaf marginal interaction of leaf l $G_{l,S}(p)$ is strictly independent of the remaining tree structure and can be computed exactly using only *information contained within path $\mathcal{P}(l)$: the leaf value $f(l)$, the inclusion probabilities p_i, and the conditioned and background path weights $PT_{l,i}$ and $PB_{l,i}$.*

Intuition of proof: each term in Equation 5 is a separate function of l. Furthermore, if we expand the partial derivative, each derivative is applied to a single linear term of of p_i. See Appendix B for the full proof.

Remark 1 *$G_{l,S}$ is nonzero only if $S \subseteq F_l$. In addition, since each path can have at most D features in F_l, the highest order for nonzero interaction in a decision tree is D.*

Theorem 1 is the formal generalization of the structural property reported in [33], which is defined only for 1st order Banzhaf values. By proving this decoupling extends to higher-order interactions with arbitrary $\mathbf{p}$, we can calculate any higher-order g-BII using depth-first search. Specifically, k-th order g-BII can be computed with the following time and memory complexity:

Proposition 1 (Complexity of k-th order g-BII) *The k-th order g-BII of decision tree $\mathcal{T}$ can be computed with time complexity $O\left(L\left(D + \binom{D}{k}\right)\right)$ and memory complexity $O\left(\binom{n}{k} + D\right)$, where D is the depth of the tree and L is the number of leaves $|\mathcal{L}|$.*

Intuition of proof: We perform a depth-first search to reach a leaf l, tracking $PT_{l,i}$ and $PB_{l,i}$ along $\mathcal{P}(l)$. Once we reach l, we calculate interactions for all possible coalitions in F_l. We repeat the process across all l. See Appendix B for the full proof.

4 Extending Generalized Banzhaf Interaction to Probabilistic Interactions

In this section, we extend our algorithm for calculating g-BII to PII.

4.1 Probabilistic Interactions as Integrals of Marginal Interactions

It is known that any PII can be represented as an integral in the unit hypercube with respect to some measure $Q(\mathbf{p})$ [35, 12], including many famous IIs [37, 38, 25]. In the worst case scenario, an II with arbitrary $w(T)$ can be solved by evaluating g at 2^{n-k} unique points that do not form a multilinear surface (such as a tensor grid) to find a linear combination that matches the distribution of $w(T)$, a process that corresponds to interpolatory quadrature.

4.2 Efficient Computation of Cardinal and Probabilistic Interaction Indices

The computation requirement decreases significantly for IIs with special properties: specifically (a) CIIs and (b) PIIs evaluated on a path. We handle each type separately in this section. The general pseudocode is provided in Algorithm 1 (Appendix C).

4.2.1 CII

For CIIs, since $w(T)$ is dependent on cardinality $0 \leq |T| \leq D - k$, there are at most $D - k + 1$ unique possible weights. The weights are also *symmetric*, so the interaction is calculated over the diagonal of the hypercube $p_i(t) = t$. Therefore, we can represent CII as:

$$CII(S) = \int_0^1 G_S(t) d\mu(t) \tag{7}$$

We can calculate any CII by evaluating at most $M = D - k + 1$ points along the diagonal of the unit hypercube and solving a system of linear equations with $w(T)$ as constraints. By using Chebyshev nodes for interpolatory quadrature [43], the time and memory complexity can be derived as follows:

Proposition 2 (Complexity of k-th Order CII) . *The k-th order CII of decision tree $\mathcal{T}$ can be computed with time complexity $O\left(D(Mn + M^3) + ML\left(D + \binom{D}{k}\right)\right)$ and memory complexity $O\left(\binom{n}{k} + M^2 + MD\right)$, where $M = D - k + 1$.*

Intuition of proof. The time complexity is a sum of finding integration nodes t_m and evaluating g-BII M times, while the memory complexity is from holding CII values and different quadrature points over depth D.

4.2.2 PII

For PII, consider the polynomial path $p_i(t) = \sum_{j \in Z} C_{ij} t^{c_{ij}}, 0 \leq p_i(t) \leq 1$. The integral becomes:

$$PII(S) = \int_0^1 G_S(\mathbf{p}(t))dt \tag{8}$$

Unlike CIIs, $w(T)$ are the *results* of integrating $B_{\mathbf{p}}(T) = \prod_{i \in T} p_i \prod_{i \in N \setminus (S \cup T)} (1 - p_i)$ rather than hard constraints. Consequently, for exact calculations, the number of quadrature points M is determined by the highest possible degree of the polynomial defining $B_T(\mathbf{p}(t))$, which is $(D - k) \max_{i \in N \setminus S} c_i$. Specifically, the integral can be calculated using Gauss-Legendre quadrature, which requires $M = \lceil (((D - k) \max_{i \in N \setminus S} c_i) + 1)/2 \rceil$ to converge. The time and memory complexity for calculating this integral is as follows.

Proposition 3 (Complexity of k-th Order PII) . *The k-th order PII of decision tree $\mathcal{T}$ with depth D and leaves $L = |\mathcal{L}|$ can be computed with time complexity with $O\left(Mn + M^2 + ML\left(D + \binom{D}{k}\right)\right)$ and memory complexity $O\left(\binom{n}{k} + Mn + D\right)$, where $M = \lceil ((D - k) \max_{i \in N \setminus S} c_i) + 1)/2 \rceil$.*

Intuition of proof. The time complexity is a sum of finding t_m and evaluating g-BII M times, while the memory complexity is from holding PII values and depth traversal. In particular, unlike generic CII which require calculating the quadrature points per depth, the quadrature points in PII can be shared across all depths.

Remark 2 *For non-polynomial paths, we cannot use Gauss-Legendre quadrature for exact values. However, it still maintains standard Gauss-Legendre error bounds.*

In practice, non-polynomial paths are rarely used in standard interpretability literature. Furthermore, for non-integer rational weights (i.e., $c_i = k_i/B$, $k_i, B \in \mathcal{Z}$), a change of variables $u = t^{1/B}$ converts the path into an integer polynomial, and therefore can be calculated exactly using Gauss-Legendre quadrature.

Remark 3 *If $p_i(t) = t^{c_i}$, the PII reduces to* Weighted *SII with weights c_i. If all $c_i = 1$, it further reduces to SII. Consequently, SII can also be computed as PII rather than CII.*

Intuitively, CIIs require Chebyshev nodes to stabilize the inversion of hard cardinality constraints, whereas PIIs evaluate continuous 1D integrals via numerical quadrature. Consequently, indices that can be represented as CII or PII can be computed via either method, though the PII formulation requires roughly half the evaluation points.

4.3 Comparison against TreeSHAP-IQ

TreeSHAP-IQ [9] is the state-of-the-art method for calculating exact CII for tree ensembles. It extends the summary polynomial method of Linear TreeSHAP [13] to higher-order interactions. TreePII has two major strengths over TreeSHAP-IQ: lower time and memory complexity for calculating generic CIIs, and the capacity to calculate asymmetric PIIs.

Table 1: Comparison of theoretical complexity of TreeSHAP-IQ and TreePII for computing CII.

Complexity Type	TreeSHAP-IQ	TreePII
Time	$O\left(LD\binom{n-1}{k-1}\right)$	$O\left(D(Mn+M^3)+ML\left(D+\binom{D}{k}\right)\right)$
Memory	$O\left(D^2\binom{n}{k}\right)$	$O\left(\binom{n}{k}+M^2+MD\right)$

Firstly, as shown in Table 1, the time complexity of TreePII does not scale combinatorially with number of features n; instead, it scales with depth D. In typical decision trees, $D << n$, so $\binom{D}{k} << \binom{n-1}{k-1}$. Since $M = D - k + 1$, the impact of $O(D(Mn + M^3))$ is negligible compared $\binom{D}{k}$, so the time complexity is generally lower. For memory complexity, while both methods require $\mathcal{O}\left(\binom{n}{k}\right)$ to store the final output, TreePII reduces the working memory during tree traversal to $\mathcal{O}(M^2 + MD)$, decoupling the combinatorial cost from the traversal state. Consequently, as we show in quantitative experiments (Section 5), TreePII outperforms TreeSHAP-IQ in runtime in both synthetic and real datasets across different n, D, L, and k.

The second strength of TreePII is that it can calculate asymmetric PIIs. CIIs are symmetric by nature, so TreeSHAP-IQ cannot calculate asymmetric PIIs. In contrast, TreePII can calculate along any path $p_i(t)$ - exactly for polynomial paths and approximately for non-polynomial paths - which allows asymmetric PIIs. We demonstrate this versatility through application of Weighted SII in Section 6.

5 Experiments

In this section, we demonstrate the performance of TreePII against TreeSHAP-IQ [9], the state-of-the-art method for calculating CIIs in decision trees, as well as a naive baseline for asymmetric PIIs. We focus on single decision trees since all methods scale linearly with the number of trees in an ensemble. We perform two tests: (a) scalability synthetic benchmark tests over different number of features n, depth D, leaves L, and interaction order k, and (b) runtime tests on real datasets. The complete experimental details for both experiments (including hardware, software, model hyperparameters and real dataset descriptions) are provided in Appendix D.

5.1 Scalability Benchmarks

We perform scalability benchmark test by comparing SII computation of TreePII against TreeSHAP-IQ on a tree trained on a synthetic dataset across different tree complexity parameters. TreePII outperforms TreeSHAP-IQ under different complexity parameters, including n and k (Figure 2). We find similar performance gains for other parameters such as D and L, and also find exact convergence often occur before theoretical maximum M (Appendix E).

5.2 Comparison on Real Datasets

We test the average runtime over 30 instances for calculating SII and Weighted SII on real datasets. SII is compared against TreeSHAP-IQ, while Weighted SII is compared against naive baseline (brute-force approach that calculates over all possible coalitions) since TreeSHAP-IQ cannot calculate asymmetric PIIs. The latter experiment is performed on datasets with smaller n due to the exponential memory requirements of the naive baseline. TreePII itself scales to asymmetric interactions on larger datasets within standard memory limits (e.g., peak memory < 2GB for the Covertype dataset). The results for 2nd-order SII and Weighted SII in decision trees are reported in Tables 2 and 3. We only report averages in the main text for clarity as the standard deviations are small. The complete results including standard deviations are presented in Appendix F. For SII, we find a consistent speedup in TreePII over TreeSHAP-IQ across all datasets, ranging from around $\sim 2\times$ in Digits dataset to $\sim 90\times$ in California Housing dataset, which is consistent with the generally lower theoretical time complexity of TreePII over TreeSHAP-IQ. We find similar performance for asymmetric SII as TreePII easily outperforms the naive method.

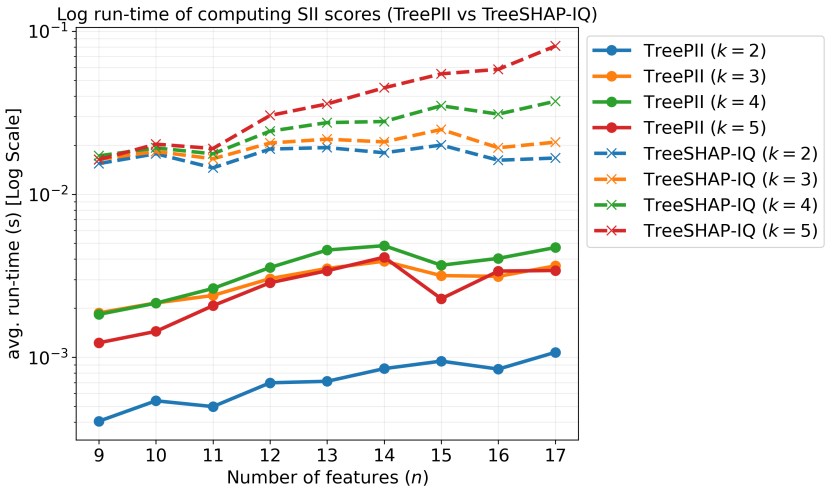

Figure 2: Runtime across different number of input features n for different order of interactions k. TreePII consistently outperforms TreeSHAP-IQ in all scenarios.

Table 2: Runtime comparison across different datasets for 2nd-order SIIs in decision trees. TreePII consistently outperforms TreeSHAP-IQ.

Dataset	TreeSHAP-IQ (ms)	TreePII (ms)	Speedup
Adult Income [44]	214.96	7.32	29.36
Bank Marketing [45]	45.13	1.94	23.24
Bike Sharing [46]	168.35	2.09	80.44
Breast Cancer [47]	1.92	0.17	11.47
California Housing [48]	819.43	9.14	89.62
Communities & Crime [49]	2181.68	575.24	3.79
COMPAS [50]	5.57	1.28	4.37
Covertype [51]	949.57	282.58	3.36
Diabetes [52]	3.44	0.17	20.00
Digits [53]	19.78	8.44	2.34
Parkinsons [54]	1.02	0.17	5.85
Titanic [55]	2.99	0.12	25.40
TUANDROMD [56]	304.04	77.96	3.90
Wine [57]	0.57	0.10	5.59

Table 3: Runtime comparison across different datasets for 2nd-order Weighted SIIs in decision trees. TreePII consistently outperforms the naive brute-force baseline.

Dataset	Naive (ms)	TreePII (ms)	Speedup
California Housing	45501.29	22.20	2050.06
Diabetes	6.88	0.22	31.51
Titanic	6.94	0.10	67.62

6 Applications

In this section, we show an application of asymmetric PII in hierarchical analysis using PII with $p_i(t) = t^{c_i}$, which is equivalent to Weighted SII with weights c_i. Since we integrate on $0 \leq t \leq 1$, $p_i(t) = t^{c_i} > t^{c_j} = p_j(t)$ if $c_i < c_j$. Consequently, any feature i with low c_i enters a coalition more frequently than feature j with high c_j. The resulting PII sets feature i as a 'contextual baseline' and j as a 'modifier', where feature j tends to take effect after the effect of i has been established. The interactions become *directional*: a negative interaction indicates feature i alone can make sufficient contribution to the prediction (hence lower gains from adding feature j), while positive interaction indicates feature j makes a true additive contribution on top of feature i. This attribution structure is useful for reflecting hierarchy by giving upstream features lower c_i than downstream features. For example, we may align c_i with temporal orderings (where earlier-occurring features are given lower c_i than later-occurring features to set as context); fairness (setting $c_i \simeq 0$ for protected attributes to see if any unprotected attribute is highly redundant and acts as a proxy); or controllability (setting c_i low for fixed features so that we can see the relevance of changeable features).

To demonstrate the plausibility of asymmetric PIIs being more reasonable than CIIs, we consider data with a root context (X_0) that drives two downstream symptoms (X_1, X_2). The target model is $Y = X_0 \wedge X_1 \wedge X_2$. We compare standard SII with weighted SII that puts coefficient of 0.05 on X_0 and 15 for X_1 and X_2. The comparison of interactions are as follows:

Table 4: Change in interaction before and after applying weights.

Pair	SII	Weighted SII
X_0,X_1	3.6671	2.6125
X_0,X_2	3.5655	2.5108
X_1,X_2	3.7924	4.8836

As seen in Table 4, standard SII fails to capture directionality, splitting the 3-way interaction approximately equally among all pairs ($\sim$ 3.6 each). In contrast, by assigning a low coefficient to X_0, weighted SII reduces spurious interactions involving X_0 ($\sim$ 2.4) and attributes the vast majority of the synergy to X_1, X_2 pair ($\sim$ 4.88). This experiment demonstrates that weighted SII with appropriate weights can extract more hierarchy-aware interactions.

In Figure 3, we show a more realistic demonstration by applying Weighted SII to adult income prediction task based on temporal ordering. We give features age, gender, race, and country the lowest c_i (features that are determined at birth); followed by education and marital status (features that occur after birth); and the remainder (the 'consequence' features). We then compare the top 20 interactions by magnitude for standard and Weighted SII. We find that some signals are strengthened while others are weakened. For example, education loses its positive interaction with race, which indicates that the positive standard SII was driven by the upstream feature's contribution rather than that of education. In contrast, age gains positive interaction with work class, which means that even if age is known, the target's work class is still relevant to predicting income level. Many bottom-level interactions are amplified, indicating their synergies and redundancies are clearer after accounting for upstream features.

While Weighted SII reveals valuable information regarding the relations captured by the model, calculating the dependencies in Figure 3 requires evaluating asymmetric subset weights. Using the naive method, resolving these weights for even a moderate-size tree would take a significant amount of time. TreePII can calculate the entire Weighted SII quickly, establishing TreePII as a highly scalable enabler for incorporating hierarchy into tree-based model explanations.

7 Conclusion

In this paper, we introduce TreePII, a novel algorithm that advances the theoretical foundations of game theory-based tree ensemble explainability through path-independent tree traversal and interpolatory quadrature. By approaching the calculation of Probabilistic Interaction Index (PII) as an integral over marginal interactions evaluated using interpolatory quadrature, TreePII effectively bypasses the combinatorial bottlenecks of existing methods, resulting in significant improvements in

Hierarchy	Variable	c_i
Top	Age, gender, race, country	1
Middle	Education; marital status	2
Bottom	Remaining features	3

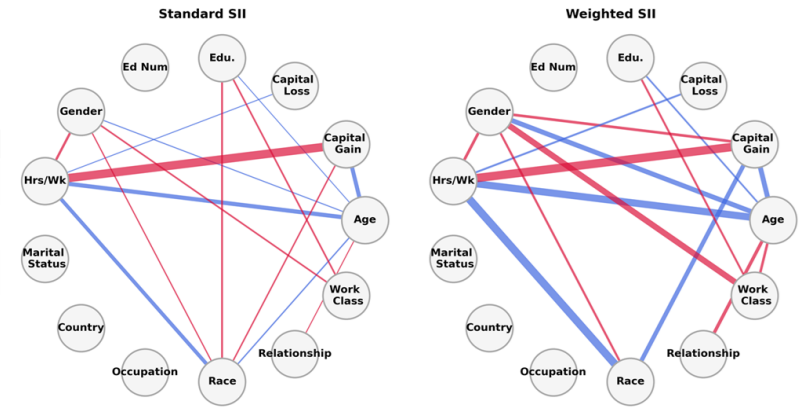

Figure 3: Comparison of top 20 interactions from standard vs. Weighted SII calculated using TreePII. Applying a temporal hierarchy (left table) changes the interactions. Negative upstream-downstream interactions indicate that the upstream features are true drivers of the interactions. Positive interaction implies the downstream feature contributes to the prediction even after accounting for upstream features. This effect can be masked in standard SII since it treats all features equally.

both time and memory complexity. Furthermore, its unique capability to compute asymmetric PIIs enables the integration of complex domain knowledge, such as feature hierarchy. By providing a fast, exact, and flexible framework for higher-order interaction calculations, TreePII opens the door for a scalable analysis of complex tree ensemble models in real-world applications.

Limitations

While TreePII achieves significant speedups scaling with tree depth D rather than total number of features n, this advantage can diminish in edge cases: if $D \approx n$, the subset combination $\binom{D}{k} \approx \binom{n}{k}$. In addition, for PIIs, applying quadrature for highly complex polynomial paths requiring large M may encounter floating-point issues, though this issue should rarely occur in practice. Lastly, while TreePII computes asymmetric interactions efficiently, it relies on user-specified feature priorities. Selecting and validating these hierarchies is a crucial direction for future work that TreePII now makes computationally feasible.

Broader Impact

TreePII contributes positively to societal outcomes by providing a tractable method to audit tree ensembles for complex, higher-order biases and hierarchical relationships. By enabling exact asymmetric interaction calculations, practitioners can better align model behavior with domain knowledge. A potential negative societal impact that is common to many explanation methods is automation bias, where a user over-relies on the explanations to justify inherently flawed models. However, by providing exact rather than approximate values, TreePII mitigates the risk of providing misleading approximations.

Acknowledgments

This work was supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2019-II190075, Artificial Intelligence Graduate School Support Program (KAIST); No. RS-2022-II220984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation) and by the Korea Planning & Evaluation Institute of Industrial Technology (KEIT) grant funded by the Ministry of Trade, Industry and Energy (No. RS-2025-25458052, Development of Core Technologies for Manufacturing Foundation Models).

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[3] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. In *8th ICML Workshop on Automated Machine Learning (AutoML)*.

[4] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[5] Pierre Dewitte and Nathalie Smuha. Ai meets the gdpr. navigating the impact of data protection on ai systems. 2025.

[6] Regulation 2016/679 of the european parliament and of the council of april 27, 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *OJ*, L 119:1–88, 2016-05-04.

[7] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. The shapley value in machine learning. In *The 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence*, pages 5572–5579. International Joint Conferences on Artificial Intelligence Organization, 2022.

[8] Ian Covert, Scott M Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures. *Advances in neural information processing systems*, 33:17212–17223, 2020.

[9] Maximilian Muschalik, Fabian Fumagalli, Barbara Hammer, and Eyke Hüllermeier. Beyond treeshap: Efficient computation of any-order shapley interactions for tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14388–14396, 2024.

[10] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating hierarchical explanations on text classification via feature interaction detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5578–5593, 2020.

[11] Divyansh Singhvi, Diganta Misra, Andrej Erkelens, Raghav Jain, Isabel Papadimitriou, and Naomi Saphra. Using shapley interactions to understand how models use structure. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20727–20737, 2025.

[12] Katsushige Fujimoto, Ivan Kojadinovic, and Jean-Luc Marichal. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior*, 55(1):72–99, 2006.

[13] Albert Bifet, Jesse Read, Chao Xu, et al. Linear tree shap. *Advances in Neural Information Processing Systems*, 35:25818–25828, 2022.

[14] Jean-Luc Marichal and Pierre Mathonet. Weighted banzhaf power and interaction indexes through weighted approximations of games. *European journal of operational research*, 211(2):352–358, 2011.

[15] Ehud Kalai and Dov Samet. On weighted shapley values. *International journal of game theory*, 16(3):205–222, 1987.

[16] Pierre Dehez. Allocation of fixed costs: characterization of the (dual) weighted shapley value. *International Game Theory Review*, 13(02):141–157, 2011.

[17] Prakash P Shenoy. The banzhaf power index for political games. *Mathematical Social Sciences*, 2(3):299–315, 1982.

[18] Andrew Heard and Tim Swartz. Empirical banzhaf indices. *Public Choice*, 97(4):701–707, 1998.

[19] Tobias Hiller. Weighted shapley values and allocation of portfolio risk: one approach to solve the low-risk puzzle? *Theory and Decision*, pages 1–16, 2025.

[20] S Sivanandham, S Srivatsa Srinivas, and Ananya Jain. Developing a tourism circuit around a brownfield destination: a framework based on analytic hierarchy process and weighted shapley value. *Applied Operations and Analytics*, 1(1):1–14, 2025.

[21] Meng Li, Wenzhong Li, and Genjiu Xu. A weighted method for sharing costs in highways. *Operations Research Letters*, page 107415, 2026.

[22] Guillermo Owen. Multilinear extensions and the banzhaf value. *Naval research logistics quarterly*, 22(4):741–750, 1975.

[23] Guillermo Owen. Multilinear extensions of games. *The Shapley Value. Essays in Honor of Lloyd S. Shapley*, pages 139–151, 1988.

[24] Lloyd S Shapley et al. A value for n-person games. 1953.

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

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

[27] Amirata Ghorbani and James Y Zou. Neuron shapley: Discovering the responsible neurons. *Advances in neural information processing systems*, 33:5922–5932, 2020.

[28] Hugh Chen, Scott Lundberg, and Su-In Lee. Explaining models by propagating shapley values of local components. *Explainable AI in Healthcare and Medicine: Building a Culture of Transparency and Accountability*, pages 261–270, 2021.

[29] Edoardo Mosca, Ferenc Szigeti, Stella Tragianni, Daniel Gallagher, and Georg Groh. Shap-based explanation methods: a review for nlp interpretability. In *Proceedings of the 29th international conference on computational linguistics*, pages 4593–4603, 2022.

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

[31] Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 8780–8802. PMLR, 2022.

[32] Roma Patel, Marta Garnelo, Ian Gemp, Chris Dyer, and Yoram Bachrach. Game-theoretic vocabulary selection via the shapley value and banzhaf index. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2789–2798, 2021.

[33] Adam Karczmarz, Anish Mukherjee, Piotr Sankowski, and Piotr Wygocki. Improved feature importance computations for tree models: Shapley vs. banzhaf. *arXiv preprint arXiv:2108.04126*, 2021.

[34] Jiachen T Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 6388–6421. PMLR, 2023.

[35] 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.

[36] Jean-Luc Marichal, Ivan Kojadinovic, and Katsushige Fujimoto. Axiomatic characterizations of generalized values. *Discrete Applied Mathematics*, 155(1):26–43, 2007.

[37] Mukund Sundararajan, Kedar Dhamdhere, and Ashish Agarwal. The shapley taylor interaction index. In *International conference on machine learning*, pages 9259–9268. PMLR, 2020.

[38] 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.

[39] Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Hammer. Shap-iq: Unified approximation of any-order shapley interactions. *Advances in Neural Information Processing Systems*, 36:11515–11551, 2023.

[40] Maximilian Muschalik, Hubert Baniecki, Fabian Fumagalli, Patrick Kolpaczki, Barbara Hammer, and Eyke Hüllermeier. shapiq: Shapley interactions for machine learning. *Advances in Neural Information Processing Systems*, 37:130324–130357, 2024.

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

[42] Jilei Yang. Fast treeshap: Accelerating shap value computation for trees. *arXiv preprint arXiv:2109.09847*, 2021.

[43] Sven Weisbrich, Georgios Malissiovas, and Frank Neitzel. On the fast approximation of point clouds using chebyshev polynomials. *Journal of Applied Geodesy*, 15(4):305–317, 2021.

[44] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[45] Rita P. Moro, S. and P. Cortez. Bank Marketing. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5K306.

[46] Hadi Fanaee-T. Bike Sharing. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5W894.

[47] Mangasarian Olvi Street Nick Wolberg, William and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.

[48] R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics Probability Letters*, 33(3):291–297, 1997.

[49] Michael Redmond. Communities and Crime Unnormalized. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C5PC8X.

[50] J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. *ProPublica*, 2016.

[51] Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

[52] BRADLEY EFRON, TREVOR HASTIE, IAIN JOHNSTONE, and ROBERT TIBSHIRANI. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[53] E. Alpaydin and C. Kaynak. Optical Recognition of Handwritten Digits. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50P49.

[54] Max Little. Parkinsons. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C59C74.

[55] Will Cukierski. Titanic - machine learning from disaster. `https://kaggle.com/competitions/titanic`, 2012. Kaggle.

[56] Parthajit Borah and Dhruba K. Bhattacharyya. TUANDROMD (Tezpur University Android Malware Dataset). UCI Machine Learning Repository, 2020. DOI: https://doi.org/10.24432/C5560H.

[57] Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1992. DOI: https://doi.org/10.24432/C5PC7J.

A Table of Notations

Table 5: Summary of key notations.

Notation	Description
$N := \{1, \ldots, n\}$	Set of all features
$S \subseteq N, k = \lvert S \rvert$	Subset of features and its cardinality
$\mathbf{x} \in \mathbb{R}^n, x_i$	Feature vector and the value of feature i in $\mathbf{x}$
$\mathbf{B} \in \mathbb{R}^{r \times n}$	Background dataset consisting of r instances
$\mathbf{p} \in [0,1]^n, p_i$	Inclusion probability vector, and inclusion probability of feature i
$\mathbf{z} \in \{0,1\}^n, z_i$	Binary indicator vector, and indicator of feature i joining a coalition
$g(\mathbf{z})$	Value function of the cooperative game
$G(\mathbf{p})$	Multilinear extension (MLE) of $g(\mathbf{z})$
$G_S(\mathbf{p}), G_{l,S}(\mathbf{p})$	k-th order partial derivative of $G(\mathbf{p})$ and its additive decomposition by leaves in trees
$\Delta_S(T)$	Discrete derivative of subset S with respect to subset T
$\phi_{g,i}, II_{g,S}$	Value of feature i, and Interaction Index for subset S
$\mathcal{V}, E, \mathcal{L}$	Sets of non-leaf nodes, edges, and leaves in tree $\mathcal{T}$
v, e, l	A non-leaf node, edge, and leaf in tree $\mathcal{T}$
i_v, ν_v	Feature and threshold used for the split at node v
r_v, r_p, r_c	Coverage (number of background instances) arriving at node v, parent node p, and child node c
D	Maximum depth of the tree
$f(l)$	Predicted value upon reaching leaf l
$\mathcal{P}(l), F_l$	Path of edges to reach leaf l, and unique features along that path
$w(T)$	Weight assigned to $\Delta_S(T)$ in an Interaction Index
$P(l)$	Expected probability of reaching leaf l
$W_{l,i}$	Aggregated probability path weight for feature i along path to l
$PT_{l,i}, PB_{l,i}$	Conditioned vs background probability path weight when i is known vs. unknown. Realized values of $W_{l,i}$
t_m, c_m	Evaluation points in unit hypercube/path and their linear coefficients
$p_i(t) = \sum_{i,j} C_{ij} t^{c_{ij}}$	Polynomial path parameterization with coefficient C_i and exponent c_i

B Full Proofs

B.1 Proof of Theorem 1

Consider the form of $G_{l,S}(\mathbf{p})$:

$$G_{l,S}(\mathbf{p}) = f(l) \left(\prod_{i \in S} \frac{\partial}{\partial p_i} \right) \prod_{i \in F_l} E(W_{l,i})$$

Since $E(W_{l,i}) = p_i(PT_{l,i} - PB_{l,i}) + PB_{l,i}$, if we differentiate with respect to p_i, one of the two cases can happen:

Case 1. $i \notin F_l$

There is no term with p_i in the product, so the derivative becomes 0. This result implies that for any subset S, if any of $i \notin F_l$, the marginal interaction of the leaf becomes 0 since we can always differentiate first with i.

Case 2. $i \in F_l$

$$\frac{\partial}{\partial p_i} \prod_{j \in F_l} E(W_{l,j}) = \frac{\partial}{\partial p_i} [p_i(PT_{l,i} - PB_{l,i})] \prod_{j \in F_l \setminus i} E(W_{l,j}) = (PT_{l,i} - PB_{l,i}) \prod_{j \in F_l \setminus i} E(W_{l,j})$$

If we iterate over all $i \in S$, we have:

$$\left(\prod_{i\in S}\frac{\partial}{\partial p_i}\right)\prod_{i\in F_l}E(W_{l,i})=\prod_{i\in S}(PT_{l,i}-PB_{l,i})\prod_{i\in F_l\setminus S}E(W_{l,i})$$

If $E(W_{l,i})\neq 0 \quad \forall i\in S$, we may also multiply and divide by $\prod_{i\in S}E(W_{l,i})$ to get:

$$\prod_{i\in S}(PT_{l,i}-PB_{l,i})\prod_{i\in F_l\setminus S}E(W_{l,i})=\prod_{i\in S}\frac{PT_{l,i}-PB_{l,i}}{E(W_{l,i})}\prod_{i\in F_l}E(W_{l,i})=Mod(l,S)BW(l)$$

where $Mod(l,S)=\prod_{i\in S}\frac{PT_{l,i}-PB_{l,i}}{E(W_{l,i})}$ and $BW(l)=\prod_{i\in F_l}E(W_{l,i})$. Since $f(l)$, $Mod(l,S)$, and $BW(l)$ are all determined solely by its path, $G_{l,S}(\mathbf{p})$ can be computed independent of the remaining tree.

B.2 Proof of Proposition 1

From Theorem 1, we can calculate the marginal interaction by each leaf through depth-first search, which traditionally requires D operations. Once we reach a leaf, we need to perform $\binom{|F_l|}{k}$ multiplications to find all possible S of order k in the l, which is bounded by $\binom{D}{k}$. Lastly, we repeat this operation across each leaf, resulting in total time complexity of $O\left(L\left(D+\binom{D}{k}\right)\right)$.

For memory complexity, since we are performing depth-first search, we need $O(D)$ to hold $PT_{l,i}$ and $PB_{l,i}$ for each search. We also need $O\left(\binom{n}{k}\right)$ for holding all interaction terms, resulting in total memory complexity of $O\left(\binom{n}{k}+D\right)$.

B.3 Proof of Propositions 2 and 3

B.3.1 Time Complexity.

For both CII and PII, g-BII is evaluated M times, resulting in $O\left(ML\left(D+\binom{D}{k}\right)\right)$. However, there is a slight difference in the precomputation of t_m.

CII. For CII, we have to perform three different calculations for each unique $M_l=|F_l|\geq k$: computing the integration points, marginalizing the weights, and solving the linear system. First, we calculate the integration points, which takes $O(M^2)$ to calculate the Chebyshev nodes and populate the Vandermonde matrix. The second calculation is marginalizing the weights. Conventionally, the relation between $W(T)$ and $B_{\mathbf{p}}(T)$ is:

$$W(|T|)=\int_0^1 t^{|T|}(1-t)^{n-k-|T|}dt$$

However, our integral at a given leaf works on at most M_l features, i.e.,

$$W_{local}(|T|)=\int_0^1 t^{|T|}(1-t)^{M_l-k-|T|}dt$$

To find the relation between the local and raw weights, we can multiply $((t+(1-t))^{n-M_l}$ to the integrand of W_{local}

$$t^{|T|}(1-t)^{M_l-k-|T|}((t+(1-t))^{n-M_l}=t^{|T|}(1-t)^{M_l-k-|T|}\left[\sum_{j=0}^{n-M_l}\binom{n-M_l}{j}t^j(1-t)^{n-M_l-j}\right]$$

$$=\sum_{j=0}^{n-M_l}\binom{n-M_l}{j}t^{|T|+j}(1-t)^{n-k-(|T|+j)}$$

Here, $t^{|T|+j}(1-t)^{n-k-(|T|+j)}$ is the integrand of the global weight with $|T|+j$ as the input. Taking integral of both sides, we get:

$$W_{local}(|T|) = \sum_{j=0}^{n-M_l} \binom{n-M_l}{j} W(|T|+j)$$

A single marginalization takes $O(n)$ at worst, which must be done for each $|T|$ up to $M_l - k + 1$. The time complexity for whole marginalization is $O(M_l n)$.

Once we have both weights and integration points selected, we simply solve the system:

$$\mathbf{c} = \mathbf{T}^{-1}\mathbf{w}$$

where $\mathbf{T}$ is the matrix of $B_{\mathbf{p}}(T)$ calculated for the leaf, $\mathbf{w}$ are the localized weights, and $\mathbf{c}$ is the vector of weights for each integration point. The matrix inversion takes $O(M_l^3)$.

The total time complexity for the setup for a single leaf is therefore $O(M_l^2 + M_l n + M_l^3) = O(M_l n + M_l^3)$. Since this complexity is bounded by the worse case when $M_l = M = D - k + 1$, the total time complexity is $O(Mn + M^3)$. Most importantly, we do not repeat the calculation if we reach a leaf with the same depth - we reuse the integration points and weights. Therefore, this part scales with D rather than L, resulting in $O(D(Mn + M^3))$.

PII. For PII, because we are integrating along a straight path on t rather than solving for constraints, we do not need to calculate t_m by leaf. Instead, we can share the same points as the maximum depth D. The time for calculating t_m is $O(M^2)$ for Gauss-Legendre quadrature and it takes $O(Mn)$ to calculate all p_i at each t_m, for a total of $O(Mn + M^2)$.

Therefore, the total time complexities for CII and PII are $O\left(D(Mn + M^3) + ML\left(D + \binom{D}{k}\right)\right)$ and $O\left(Mn + M^2 + ML\left(D + \binom{D}{k}\right)\right)$, respectively.

B.3.2 Memory Complexity.

For CII, we need standard $O(D)$ points for each depth-first search, and $O\left(\binom{n}{k}\right)$ to hold the interactions. We also need $O(M)$ and $O(M^2)$ to hold the marginalized weights and matrix $\mathbf{T}$ at each leaf during computation, and $O(MD)$ to hold the entire set of integration rules once all unique leaves have been traversed. The total memory complexity is $O\left(\binom{n}{k} + D + M + M^2 + MD\right)$ = $O\left(\binom{n}{k} + M^2 + MD\right)$.

For PII, we also need standard $O(D)$ points for each depth-first search and $O\left(\binom{n}{k}\right)$ to hold the interactions. We also hold the exponentiated probabilities for $O(Mn)$ and the corresponding integration weights for $O(M)$. The total memory complexity is $O\left(\binom{n}{k} + D + Mn + M\right) = \left(\binom{n}{k} + Mn + D\right)$.

C Pseudo-Code for Main Algorithm

Algorithm 1 TreePII

Require: Tree $\mathcal{T}$, instance x, interaction order k, quadrature rules $\mathcal{Q}$, path functions $p_j(t)$

Ensure: Interaction indices $II_S \in \mathbb{R}^{\binom{n}{k}}$

Initialize $II_S \leftarrow 0$ for all subsets S where $|S| = k$
for each leaf $l \in \mathcal{L}$ in $\mathcal{T}$ **do**
  $F_l \leftarrow$ active features on path to l
  $d \leftarrow |F_l|$
  if $d < k$ **then**
    continue ▷ Order k requires at least k splits
  end if
  $PT_l \leftarrow$ conditioned probability path weights for F_l ▷ 1 if follows split, else 0
  $PB_l \leftarrow$ background probability path weights for F_l ▷ e.g., r_c/r_p
  $\mathbf{t}, \mathbf{c} \leftarrow \mathcal{Q}(d)$ ▷ Retrieve M quadrature points and weights
  $P =$ EVALUATE_PROBABILITIES$(\mathbf{t}, F_l, p_j(t))$ ▷ Convert $\mathbf{t}$ to probability matrix P
  for each subset $S \subseteq F_l$ with $|S| = k$ **do**
    $R \leftarrow F_l \setminus S$ ▷ Remaining path features
    $II_{l,S} \leftarrow 0$
    for $m = 1$ to M **do**
      $E(W_{l\setminus S}) \leftarrow \prod_{j \in R}[P_{m,j} PT_{l,j} + (1 - P_{m,j}) PB_{l,j}]$
      $G_{l,S}(t_m) \leftarrow \big(f(l) \prod_{i \in S}(PT_{l,i} - PB_{l,i})\big) \times E(W_{l\setminus S})$ ▷ Leaf marginal interaction
      $II_{l,S} \leftarrow II_{l,S} + c_m \times G_{l,S}(t_m)$ ▷ Integrate using quadrature
    end for
    $II_S \leftarrow II_S + II_{l,S}$ ▷ Add leaf contribution to subset S
  end for
end for
return II_S ▷ Return II

Algorithm 2 EVALUATE_PROBABILITIES

Require: Quadrature points $\mathbf{t}$ of size M, path features F_l, path function $p_i(t)$

Ensure: Probability matrix $P \in \mathbb{R}^{M \times |F_l|}$

for $m = 1$ to M **do**
  for each feature $j \in F_l$ **do**
    $P_{m,j} \leftarrow p_j(t_m)$ ▷ make path, e.g. $P_{m,j} \leftarrow t_m^{c_j}$
  end for
end forreturn P

D Experimental Settings

To evaluate the scalability, convergence by M, and real runtime of TreePII against TreeSHAP-IQ, we conduct multiple experiments using both synthetic and real-world datasets. The experimental settings are provided in this section.

Hardware and Software. All runtime benchmarks are executed on a machine equipped with an 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz processor and 32GB of RAM running Windows 11 Education. For core library dependencies, we use `scikit-learn` (v1.4.2), `numpy` (v1.26.4), and `scipy` (v1.13.1), alongside the official open-source implementation of TreeSHAP-IQ.

Real Data. For real data experiments, we use the following datasets: Adult Income [44] (predicts whether a person's annual income exceeds \$50,000 based on demographic data); Bank Marketing, [45] (predicts whether a client will subscribe to a term deposit based on data from direct marketing campaigns); Bike Sharing [46] (estimates the daily count of rental bikes in a capital bikeshare system); Breast Cancer [47] (classifies breast mass tumors as malignant or benign based on cell features); California Housing [48] (estimates the median house value for California districts using demographic and geographical data); Communities & Crime [49] (predicts the per capita violent crime rate of communities across the United States); COMPAS [50] (forecasts criminal recidivism using demographics and criminal history); Covertype [51] (predicts the dominant forest cover type for 30×30 meter geographic cells using cartographic variables); Diabetes [52] (predicts disease progression one year later using clinical variables); Digits [53] (classifies preprocessed 8×8 pixel images of handwritten digits); Parkinsons [54] (classifies Parkinson's disease using biomedical voice and speech measurements); Titanic [55] (predicts passenger survival on the Titanic based on demographic and ticket-class information); TUANDROMD [56] (detects Android malware applications by analyzing requested permissions and system API calls); and Wine [57] (classifies wines into three different cultivars using chemical analysis data).

Real Data Processing. The post-processed data descriptions are provided in Table 6. Prior to model training, all datasets are split into 80% training and 20% testing sets. During data preprocessing, target variables with more than 20 unique numerical values are formulated as regression tasks, while the remainder are treated as classification tasks. Classification targets are 0-indexed using label encoding. Categorical features are processed using Ordinal Encoding (missing values imputed via the most frequent category, and unknown categories mapped to -1) to prevent dimensionality explosion, while numerical features are scaled using Min-Max scaling (missing values imputed via the median).

Table 6: Post-processed dataset statistics used for experiments.

Dataset	Features	Test Samples (20% of Total Samples)
Adult Income [44]	14	9769
Bank Marketing [45]	16	2233
Bike Sharing [46]	16	3476
Breast Cancer [47]	30	114
California Housing [48]	8	4128
Communities & Crime [49]	146	399
COMPAS [50]	51	1443
Covertype [51]	54	116203
Diabetes [52]	10	89
Digits [53]	64	360
Parkinsons [54]	23	39
Titanic [55]	10	179
TUANDROMD [56]	241	893
Wine [57]	13	36

Model Configurations. The configurations for generating the synthetic data and the hyperparameters for the evaluated tree models are presented in Tables 7 and 8, respectively. The trained model performance is presented in Table 9. All experiments utilize a fixed seed of 42 to ensure reproducibility.

All tree-based models (Decision Tree (DT), Random Forest (RF), and Gradient Boost (GB)) in the experiments are trained using the `scikit-learn` library. Numerical operations and interpolatory quadrature rules (e.g., Gauss-Legendre) are computed using `numpy` and `scipy` library. For the

real-world dataset benchmarks, model hyperparameters are dynamically allocated based on dataset size: *Small* (≤ 20 features and ≤ 5000 samples), *Medium* (> 20 features or > 5000 samples), and *Large* (> 100 features or > 20000 samples).

We use small ensembles (1-5 trees) since evaluating higher-order interactions for standard sized ensembles is computationally prohibitive for both methods on the experiment hardware. However, since both explainers scale with the number of trees, these relative speedups extrapolate directly to larger ensembles.

Baseline Methods. For our baseline comparisons for CIIs, we utilize the official open-source implementation of TreeSHAP-IQ provided by the original authors [9] and compare the runtime for calculating SII. For asymmetric PII, we implement a naive brute-force approach (calculating the multilinear extension over all possible coalitions) for comparison and compare the runtime for computing Weighted SII. We set exponents c_i (or feature weights) linearly spaced from 1.0 to 3.0 across the feature dimension n for the asymmetric evaluation. Due to the exponential scaling of the naive baseline, we limit the asymmetric experiments to datasets with 12 or fewer features (California Housing, Diabetes, and Titanic datasets).

Evaluation. To evaluate the runtime of the explainers in our benchmark tables, we measure the execution time averaged over 30 instances randomly sampled from the test set of each respective dataset. We also calculate the standard deviation of the execution time across the 30 instances. We evaluate $k = 1, 2, 3$ for all datasets other than Communities & Crime and TUANDROMD datasets, which are evaluated at $k = 1, 2$ due to experiment hardware memory limitations.

Table 7: Synthetic dataset generation parameters (`sklearn.datasets.make_classification`)

Experiment Objective	Samples	Features	Informative	Redundant	Noise (`flip_y`)
Scalability (n, k)	5000	9–17	n	0	0.0
Tree Complexity	20000	14	10	2	0.2
Convergence by M	1000	9–11	n	0	0.0

Table 8: Tree model and explainer configurations

Experiment	Model Type	Max Depth	Estimators	Interaction Order
Scalability (n and k)	Decision Tree	8	1	2, 3, 4, 5
Scalability (D and L)	Decision Tree	2, 5, 10, …, 30	1	2
Scalability (k)	Decision Tree	$\min(20, n)$	1	1, 2, 3, 4, 5, 6
Convergence by M	Decision Tree	10	1	1, 2, 3
Real Datasets (Small)	DT / RF / GB	5 / 5 / 4	1 / 1 / 5	1, 2, 3
Real Datasets (Medium)	DT / RF / GB	10 / 10 / 6	1 / 1 / 5	1, 2, 3
Real Datasets (Large)	DT / RF / GB	15 / 15 / 8	1 / 1 / 5	1, 2, 3

Table 9: Trained model performances.

Dataset	Task	DT	RF	GB
Adult Income	Classification (Acc)	0.8462	0.8400	0.8264
Bank Marketing	Classification (Acc)	0.8092	0.7922	0.8155
Bike Sharing	Regression (R^2)	0.9989	0.9987	0.6466
Breast Cancer	Classification (Acc)	0.9474	0.9386	0.9386
California Housing	Regression (R^2)	0.6473	0.5995	0.4757
Communities & Crime	Regression (R^2)	0.9740	0.9686	0.6376
COMPAS	Classification (Acc)	0.9653	0.9688	0.9924
Covertype	Classification (Acc)	0.8505	0.7801	0.7956
Diabetes	Regression (R^2)	0.3345	0.2102	0.2808
Digits	Classification (Acc)	0.8500	0.7889	0.9056
Parkinsons	Classification (Acc)	0.9231	0.8974	0.9231
Titanic	Classification (Acc)	0.7263	0.7821	0.5978
TUANDROMD	Classification (Acc)	0.9899	0.9832	0.9731
Wine	Classification (Acc)	0.9444	0.8333	0.8611

E Additional Scalability Results

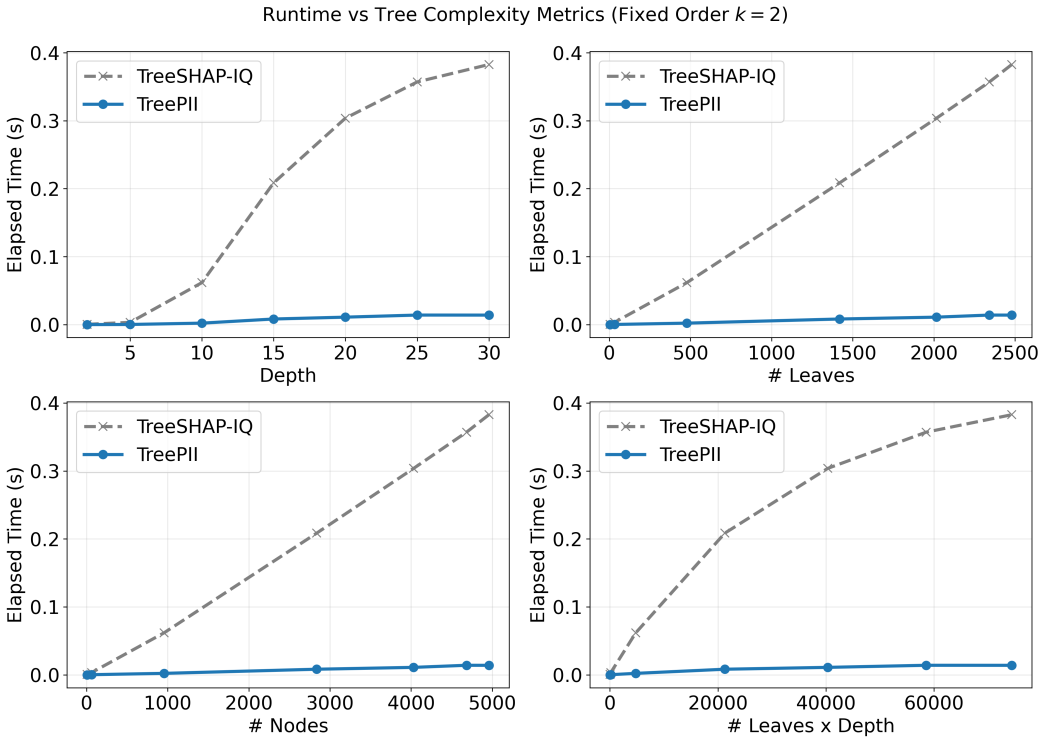

Figure 4: Runtime across different tree complexity parameters. TreePII consistently outperforms TreeSHAP-IQ in all scenarios.

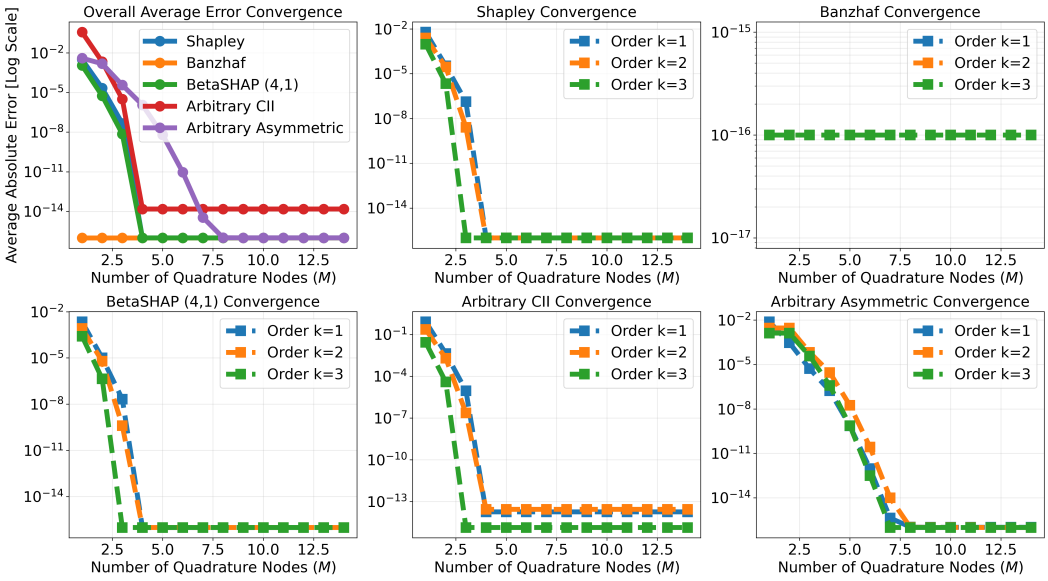

Figure 5: Convergence by M. TreePII becomes exact after reaching a specific threshold M, which depends on the depth of the tree and the type of interaction index being calculated.

F Complete Real Dataset Experiment Results

Table 10: Runtime comparisons for SII computation in Decision Trees

Dataset	Order	TreeSHAP-IQ (ms)	TreePII (ms)	Speedup
Adult Income	1	190.38 ± 7.95	1.30 ± 0.59	146.06
	2	214.96 ± 6.37	7.32 ± 0.66	29.36
	3	284.07 ± 25.71	86.16 ± 6.50	3.30
Bank Marketing	1	41.81 ± 1.34	0.47 ± 0.50	89.79
	2	45.13 ± 1.01	1.94 ± 0.46	23.24
	3	57.71 ± 1.75	6.87 ± 0.63	8.40
Bike Sharing	1	148.42 ± 7.21	0.84 ± 0.40	177.52
	2	168.35 ± 10.97	2.09 ± 0.62	80.44
	3	139.43 ± 8.66	0.91 ± 0.37	152.90
Breast Cancer	1	1.38 ± 0.52	0.10 ± 0.32	13.23
	2	1.92 ± 0.42	0.17 ± 0.38	11.47
	3	6.70 ± 0.52	0.31 ± 0.47	21.26
COMPAS	1	2.91 ± 0.52	0.34 ± 0.51	8.43
	2	5.57 ± 0.45	1.28 ± 0.47	4.37
	3	76.57 ± 1.38	1.17 ± 0.40	65.55
California Housing	1	765.25 ± 24.26	1.95 ± 0.46	391.89
	2	819.43 ± 16.06	9.14 ± 0.85	89.62
	3	864.69 ± 14.57	126.42 ± 5.33	6.84
Communities & Crime	1	150.87 ± 5.36	11.56 ± 0.96	13.05
	2	2181.68 ± 118.38	575.24 ± 17.10	3.79
Covertype	1	609.26 ± 13.74	12.34 ± 0.51	49.38
	2	949.57 ± 17.42	282.58 ± 7.68	3.36
	3	16630.75 ± 213.29	323.64 ± 12.64	51.39
Diabetes	1	3.02 ± 0.40	0.15 ± 0.40	20.13
	2	3.44 ± 0.43	0.17 ± 0.39	20.00
	3	3.33 ± 0.49	0.37 ± 0.50	8.95
Digits	1	10.07 ± 0.53	0.41 ± 0.52	24.29
	2	19.78 ± 0.82	8.44 ± 0.48	2.34
	3	452.16 ± 14.09	5.36 ± 0.49	84.41
Parkinsons	1	0.71 ± 0.52	0.07 ± 0.26	10.24
	2	1.02 ± 0.21	0.17 ± 0.39	5.85
	3	2.39 ± 0.50	0.32 ± 0.50	7.53
TUANDROMD	1	6.27 ± 0.78	0.87 ± 0.50	7.18
	2	304.04 ± 30.74	77.96 ± 7.77	3.90
Titanic	1	2.72 ± 0.50	0.07 ± 0.26	40.45
	2	2.99 ± 0.43	0.12 ± 0.32	25.40
	3	2.96 ± 0.41	0.27 ± 0.45	10.99
Wine	1	0.46 ± 0.54	0.08 ± 0.29	5.99
	2	0.57 ± 0.55	0.10 ± 0.31	5.59
	3	0.80 ± 0.50	0.22 ± 0.44	3.68

Table 11: Runtime comparisons for SII computation in GradientBoost

Dataset	Order	TreeSHAP-IQ (ms)	TreePII (ms)	Speedup
Adult Income	1	100.71 ± 11.12	0.78 ± 0.43	129.48
	2	94.85 ± 4.03	2.62 ± 0.48	36.24
	3	113.45 ± 3.33	8.81 ± 0.63	12.88
Bank Marketing	1	29.72 ± 0.63	0.38 ± 0.51	78.40
	2	36.23 ± 0.89	1.03 ± 0.18	35.08
	3	42.03 ± 1.90	1.35 ± 0.47	31.08
Bike Sharing	1	45.66 ± 1.41	0.04 ± 0.20	1272.39
	2	44.77 ± 1.39	0.49 ± 0.48	91.59
	3	40.04 ± 1.67	0.07 ± 0.26	595.37
Breast Cancer	1	7.17 ± 0.48	0.21 ± 0.44	33.79
	2	10.48 ± 0.45	1.04 ± 0.28	10.11
	3	33.19 ± 1.38	0.84 ± 0.37	39.57
COMPAS	1	9.69 ± 0.47	0.31 ± 0.47	31.78
	2	17.89 ± 0.64	3.26 ± 0.49	5.49
	3	139.15 ± 3.12	0.83 ± 0.31	168.49
California Housing	1	138.47 ± 2.47	0.66 ± 0.29	208.72
	2	152.88 ± 4.17	1.61 ± 0.33	95.01
	3	153.51 ± 3.35	6.70 ± 0.62	22.93
Communities & Crime	1	114.95 ± 3.17	4.54 ± 0.57	25.33
	2	395.62 ± 19.08	196.98 ± 7.04	2.01
Covertype	1	138.52 ± 3.69	1.96 ± 0.60	70.68
	2	197.06 ± 5.52	46.37 ± 1.50	4.25
	3	2560.20 ± 51.16	15.82 ± 1.02	161.84
Diabetes	1	7.26 ± 0.49	0.03 ± 0.18	216.12
	2	7.79 ± 0.47	0.21 ± 0.45	37.35
	3	7.56 ± 0.51	0.32 ± 0.50	23.77
Digits	1	16.36 ± 0.52	0.46 ± 0.54	35.90
	2	29.86 ± 1.02	8.31 ± 0.54	3.59
	3	572.40 ± 9.26	2.47 ± 0.40	232.02
Parkinsons	1	4.65 ± 0.51	0.37 ± 0.50	12.49
	2	6.39 ± 0.54	0.52 ± 0.53	12.37
	3	12.85 ± 0.64	0.54 ± 0.53	23.59
TUANDROMD	1	25.51 ± 1.38	3.02 ± 0.50	8.46
	2	705.02 ± 32.56	279.09 ± 16.88	2.53
Titanic	1	6.07 ± 0.53	0.17 ± 0.39	35.72
	2	7.96 ± 0.60	0.17 ± 0.39	46.61
	3	7.90 ± 0.52	0.46 ± 0.51	17.13
Wine	1	4.97 ± 0.34	0.13 ± 0.35	36.93
	2	5.87 ± 0.50	0.17 ± 0.38	34.82
	3	6.32 ± 0.49	0.17 ± 0.40	36.29

Table 12: Runtime comparisons for SII computation in Random Forest

Dataset	Order	TreeSHAP-IQ (ms)	TreePII (ms)	Speedup
Adult Income	1	210.82 ± 38.97	1.17 ± 0.70	180.11
	2	217.79 ± 5.20	8.41 ± 0.59	25.89
	3	303.57 ± 24.25	146.32 ± 16.92	2.07
Bank Marketing	1	30.37 ± 0.85	0.40 ± 0.53	75.05
	2	35.30 ± 1.32	1.70 ± 0.47	20.71
	3	45.21 ± 1.38	8.71 ± 0.58	5.19
Bike Sharing	1	131.18 ± 4.39	0.67 ± 0.46	194.98
	2	140.24 ± 4.03	1.73 ± 0.48	81.29
	3	123.23 ± 3.23	0.90 ± 0.34	137.14
Breast Cancer	1	1.25 ± 0.40	0.14 ± 0.37	8.78
	2	1.90 ± 0.47	0.28 ± 0.48	6.74
	3	6.17 ± 0.52	0.50 ± 0.51	12.29
COMPAS	1	3.43 ± 0.48	0.14 ± 0.38	23.80
	2	5.40 ± 0.54	1.37 ± 0.49	3.93
	3	70.41 ± 2.21	1.02 ± 0.32	68.90
California Housing	1	585.52 ± 11.06	1.45 ± 0.56	403.28
	2	631.78 ± 10.77	6.83 ± 0.65	92.48
	3	660.48 ± 14.02	94.58 ± 3.81	6.98
Communities & Crime	1	92.85 ± 2.45	6.62 ± 0.66	14.03
	2	1315.49 ± 34.97	352.16 ± 8.80	3.74
Covertype	1	743.69 ± 15.86	19.12 ± 1.54	38.89
	2	1202.02 ± 21.24	438.63 ± 11.65	2.74
	3	20172.64 ± 183.09	862.25 ± 46.06	23.40
Diabetes	1	2.32 ± 0.44	0.21 ± 0.42	11.27
	2	2.64 ± 0.51	0.10 ± 0.31	26.24
	3	2.80 ± 0.46	0.31 ± 0.48	9.06
Digits	1	13.72 ± 0.46	0.54 ± 0.56	25.39
	2	25.57 ± 0.96	11.11 ± 0.56	2.30
	3	584.88 ± 15.09	6.97 ± 0.64	83.91
Parkinsons	1	1.42 ± 0.50	0.24 ± 0.44	5.96
	2	1.90 ± 0.53	0.35 ± 0.48	5.43
	3	3.35 ± 0.46	0.62 ± 0.52	5.42
TUANDROMD	1	8.55 ± 0.53	1.14 ± 0.33	7.47
	2	382.56 ± 12.06	126.04 ± 4.87	3.04
Titanic	1	2.64 ± 0.51	0.07 ± 0.26	39.34
	2	2.59 ± 0.57	0.11 ± 0.34	23.51
	3	2.59 ± 0.52	0.44 ± 0.51	5.92
Wine	1	0.71 ± 0.51	0.11 ± 0.32	6.71
	2	0.99 ± 0.31	0.11 ± 0.32	9.29
	3	1.12 ± 0.30	0.37 ± 0.50	3.00

Table 13: Runtime comparisons for Weighted SII computation in Decision Trees

Dataset	Order (k)	Naive (ms)	TreePII (ms)	Speedup
California Housing	1	37534.77 ± 280.18	5.56 ± 0.78	6756.75
	2	45501.29 ± 376.47	22.20 ± 1.27	2050.06
Diabetes	1	8.71 ± 0.60	0.07 ± 0.26	127.78
	2	6.88 ± 0.46	0.22 ± 0.41	31.51
Titanic	1	9.04 ± 1.46	0.11 ± 0.35	79.12
	2	6.94 ± 0.44	0.10 ± 0.31	67.62

Table 14: Runtime comparisons for Weighted SII computation in GradientBoost

Dataset	Order (k)	Naive (ms)	TreePII (ms)	Speedup
California Housing	1	1048.62 ± 20.41	0.54 ± 0.35	1927.97
	2	925.46 ± 26.96	3.32 ± 0.61	278.60
Diabetes	1	12.79 ± 0.63	0.24 ± 0.44	53.73
	2	8.91 ± 0.52	0.13 ± 0.35	66.35
Titanic	1	15.86 ± 0.61	0.17 ± 0.38	94.76
	2	11.97 ± 0.53	0.20 ± 0.41	59.15

Table 15: Runtime comparisons for Weighted SII computation in Random Forest

Dataset	Order (k)	Naive (ms)	TreePII (ms)	Speedup
California Housing	1	31387.45 ± 360.97	4.62 ± 0.73	6799.65
	2	36520.00 ± 516.83	18.03 ± 1.39	2025.00
Diabetes	1	12.70 ± 0.51	0.20 ± 0.42	62.17
	2	10.08 ± 0.51	0.13 ± 0.35	75.55
Titanic	1	11.56 ± 0.66	0.17 ± 0.38	69.12
	2	8.69 ± 0.55	0.13 ± 0.35	64.72

G Setup and Query Runtime and Memory Decomposition

To address potential issues with small ensembles, we conduct a scalability experiment using larger ensembles (100 trees) on several real datasets for 2nd-order SII that compares TreePII against TreeSHAP-IQ. In particular, to minimize potential differences caused by implementation differences, we implement a recursive version of TreePII. The results show that TreePII has lower setup and query runtime and memory usage, as implied by theoretical time and memory complexities (Note: Due to being recursive rather than vectorized, these TreePII runtimes are higher than the main implementation).

Dataset	Setup (s)		Setup Memory (MB)	
	TreePII	**TreeSHAP-IQ**	**TreePII**	**TreeSHAP-IQ**
Titanic	0.0064 ± 0.0004	2.2601	0.1712 ± 0.0038	6.87
Parkinsons	0.0059 ± 0.0005	4.5574	0.1511 ± 0.0005	11.27
Breast_cancer	0.0066 ± 0.0004	10.7731	0.1605 ± 0.0008	22.61
Diabetes	0.0041 ± 0.0004	2.5217	0.1765 ± 0.0031	7.73
Wine	0.0063 ± 0.0005	1.3938	0.1473 ± 0.0003	4.48

Dataset	Query (ms)		Query Memory (MB)	
	TreePII	**TreeSHAP-IQ**	**TreePII**	**TreeSHAP-IQ**
Titanic	953.1871 ± 16.6141	2493.0660 ± 38.7303	0.02	0.09
Parkinsons	481.3314 ± 7.1385	1135.7119 ± 17.3667	0.03	0.59
Breast_cancer	864.9298 ± 21.6688	1786.5606 ± 31.2465	0.05	1.13
Diabetes	1041.8875 ± 18.4460	3221.8550 ± 37.5532	0.01	0.09
Wine	299.9525 ± 7.2973	888.6816 ± 16.0820	0.02	0.13

H Comparison against Monte Carlo-based Approximations

We implement a Monte Carlo (MC) sampling approximations of asymmetric PIIs for both integral (sampling t) and subset sampling (sampling subsets) on the Diabetes dataset over different number of samples. The comparisons are as follows:

Method	Runtime (ms)	Mean Abs Error (MAE)
TreePII	6.15	0.0000
Subset MC ($N = 100$)	398.03 ± 21.69	0.0894 ± 0.0384
TreePII + MC on t ($N = 100$)	2.74 ± 0.51	0.2121 ± 0.1362
Subset MC ($N = 500$)	1999.40 ± 44.16	0.0341 ± 0.0089
TreePII + MC on t ($N = 500$)	13.33 ± 0.62	0.1051 ± 0.0527
Subset MC ($N = 1000$)	3965.39 ± 63.69	0.0264 ± 0.0083
TreePII + MC on t ($N = 1000$)	26.32 ± 0.89	0.0724 ± 0.0322
Subset MC ($N = 5000$)	7933.68 ± 91.93	0.0195 ± 0.0067
TreePII + MC on t ($N = 5000$)	53.51 ± 1.76	0.0515 ± 0.0336

The runtime for Monte Carlo-based method is generally longer and increases linearly with the number of samples for both Monte Carlo-based methods. Furthermore, the mean absolute error does not reach zero even at large N. For t-sampling with $N = 100$ (which is the only one with faster runtime), the mean absolute error is the highest (0.2121). This result implies that reaching sufficient level of accuracy within short time can be difficult.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We state that we introduce TreePII, a novel algorithm for calculating any-order probabilistic interactions in tree ensembles. We also state that we show both theoretical and empirical improvements over baseline methods, as well as applications of asymmetric interactions.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have an explicit limitation section discussing the limitations of this paper.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: We provide the full proof of each theorem and proposition in the appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide the entire experimental details in the appendix, including the hardware and software details, dataset details, model hyperparameters, and experimental settings.

   Guidelines:

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

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

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

Answer: [No]

Justification: The code will be released upon acceptance.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: We provide the entire experimental details in the appendix, including the hardware and software details, dataset details, preprocessing details, model hyperparameters, and experimental settings.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: We provide standard deviations on runtimes for real datasets in the appendix. We state that we calculate standard deviations in the experimental settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Hardware descriptions are provided in the experimental settings.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We include an explicit section for Broader impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We use public tabular datasets and Python tree ensemble libraries.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All citations are provided for the public datasets used in the experiments, as well as the open-source code of the baseline method.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: We do not provide new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: All experiments are performed on public tabular datasets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: All experiments are performed on public tabular datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development does not involve LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.