# Interventional SHAP Values and Interaction Values for Piecewise Linear Regression Trees

**Artjom Zern**[1], **Klaus Broelemann**[1], **Gjergji Kasneci**[2]
[1]SCHUFA Holding AG, Germany
[2]University of Tuebingen, Germany
artjom.zern@schufa.de, klaus.broelemann@schufa.de, gjergji.kasneci@uni-tuebingen.de

**Abstract**

In recent years, game-theoretic Shapley values have gained increasing attention with respect to local model explanation by feature attributions. While the approach using Shapley values is model-independent, their (exact) computation is usually intractable, so efficient model-specific algorithms have been devised including approaches for decision trees or their ensembles in general. Our work goes further in this direction by extending the interventional TreeSHAP algorithm to piecewise linear regression trees, which gained more attention in the past few years. To this end, we introduce a decomposition of the contribution function based on decision paths, which allows a more comprehensible formulation of SHAP algorithms for tree-based models. Our algorithm can also be readily applied to computing SHAP interaction values for these models. In particular, as the main contribution of this paper, we provide a more efficient approach of interventional SHAP for tree-based models by precomputing statistics of the background data based on the tree structure.

## 1 Introduction

In recent years, predictive models have found their way into numerous applications. With the widespread dissemination and use, the wish to gain insights into the predictions made has also increased. One notable method for doing so are SHAP values (Lundberg and Lee 2017), which compute the contribution of each input feature on the prediction. These feature attributions can be used to analyze and explain model predictions. Additionally, SHAP interaction values were proposed to capture local interaction effects between features allowing a better interpretation of the model (Lundberg et al. 2020).

While SHAP values build on game-theoretic concepts for computing feature attributions, there exist variants as to what exactly is explained. Notably *observational SHAP* takes the underlying data distribution with its dependencies into account by using (observational) conditional expectations. Whereas *interventional SHAP* breaks up feature dependencies via interventions and therefore puts more emphasis on the model.

In case of independent features, observational and interventional SHAP coincide. Otherwise, they may differ significantly. Both have their benefits and drawbacks depending on the concrete application (Chen et al. 2020; Kumar et al. 2020; Miroshnikov, Kotsiopoulos, and Kannan 2021). Observational SHAP depends only on the predictions on the data manifold but it may assign contributions to features not used by the model due to feature dependencies. So it puts more emphasis on the data. The interventional SHAP, on the other hand, focuses on the model itself, but may depend on the predictions of the model outside the data distribution on which the model is trained. Janzing, Minorics, and Blöbaum (2020) discuss both variants from the causal perspective and argue that interventional SHAP is the preferable variant for feature attribution.

A major drawback of SHAP values in their original form is that their computational costs grow exponentially with the number of features. As a consequence model-agnostic exact SHAP values are intractable for most modern applications where the number of features can easily count hundreds or even thousands. For this reason multiple subsequent works employ model-specific computations and approximations for the sake of practical use.

A notable approach in this direction is the path-dependent TreeSHAP algorithm (Lundberg et al. 2020; Yang 2021), which is widely used due to its computational efficiency. It aims to approximate observational SHAP values of tree models by using precomputed node counts, but it implicitly assumes feature independence. As a result, the computed values depend not only on the prediction but also on the actual tree structure, which is an undesired property in terms of explaining model predictions. Lundberg et al. (2020) also present a variant for (exact) interventional SHAP. Both TreeSHAP variants have polynomial time complexity, while Van den Broeck et al. (2021) showed that there is in general no polynomial time algorithm for computing (exact) observational SHAP values of tree models. Under the assumption of feature independence, Arenas et al. (2021b,a) state that SHAP can be computed in polynomial time for a variety of models including decision trees.

Recently, piecewise linear regression trees and their ensembles have gained considerable attention (Guidotti et al. 2018). For instance, LightGBM added support for gradient boosting trees with linear models at the leaves.[1] Despite the

[1]https://github.com/microsoft/LightGBM/releases/tag/v3.2.0

interest, these models lack efficient model-specific SHAP implementations so far.

In this work, we address this problem and present an approach for computing SHAP values for piecewise linear trees. We concentrate on interventional SHAP, which puts more emphasis on the model than on the data and allows a more efficient computation compared to observational SHAP.

## 1.1 Contribution

Lundberg et al. (2020) presented an algorithm to compute interventional SHAP values for decision trees and decision tree ensembles. Inspired by their work, we provide the following contributions:

1. We introduce a decomposition of contribution functions, which facilitates the description of SHAP algorithms for tree-based models. Using this new notation, we formulate the interventional TreeSHAP algorithm in a more general and comprehensible way.
2. Based on the reformulation, we extend the interventional TreeSHAP to piecewise linear regression trees, for which, so far, no efficient implementation of SHAP has not been proposed yet. The extended approach that we propose has polynomial time complexity.
3. Our formulation of interventional SHAP algorithms also applies to interaction values resulting in more efficient algorithms for computing SHAP interaction values for tree-based models.
4. Eventually, we present an approach for aggregating background data for interventional SHAP computation, strongly mitigating the impact of the background data on the runtime.

The rest of the paper is organized as follows: After an overview of related work, with a focus on SHAP values and tree-based models, we review SHAP values and introduce the notations used in this work in Section 2. Section 3 introduces an additional notation which simplifies the formulation of SHAP algorithms for tree-based models. This notation is used to reformulate the existing interventional TreeSHAP algorithm and extend it to piecewise linear regression trees. Section 4 demonstrates the effectiveness and efficiency of these algorithms in a series of experiments,[2] before concluding in Section 5.

## 1.2 Related Work

**Tree-Based Prediction** Decision trees represent a popular tool for predictive modelling. While providing moderate predictive power, their effectiveness can be considerably improved by building ensembles, such as random forests (Breiman 2001) and recently gradient boosted trees (Chen and Guestrin 2016).

Model trees are an extension of decision trees that hold predictive models in the leaf nodes. Model trees use the tree-structure to map input samples to leaves and use the leaf models for prediction. Model trees can be used with different leaf models, such as linear regression (Quinlan et al. 1992; Wang and Witten 1997) or logistic regression (Landwehr, Hall, and Frank 2005). While classical model trees are trained like decision trees, there exist other approaches like gradient-based split criteria (Broelemann and Kasneci 2019; Haug, Broelemann, and Kasneci 2022) or fully trained models (Potts and Sammut 2005). Model trees with linear leaf models have also been combined into ensembles (de Vito 2017; Shi, Li, and Li 2019; Guryanov 2019).

In this work, we consider all these methods and present novel algorithms for efficiently computing interventional SHAP values.

**Explanation of Predictive Models** With an increased interest in complex predictive models, there is also an increased interest in explanation methods that can open the otherwise closed black-box nature of the complex models. There exists a plethora of methods for different models, applications and use cases. We refer to surveys to get an overview over this field (Du, Liu, and Hu 2019; Guidotti et al. 2018).

Explanation approaches may differ in the representation of explanations. For instance, logic-based representations are considered in (Ribeiro, Singh, and Guestrin 2018; Ignatiev 2020). In contrast, feature attribution methods assign to each feature a value describing its influence. In the following, we concentrate on the latter. Datta, Sen, and Zick (2016) presented Quantitative Input Influence (QII), which breaks correlations between inputs by interventions to allow causal reasoning and computes the marginal influence of inputs as Shapley values (Shapley 1953). Their framework covers the computation of interventional SHAP values. Ribeiro, Singh, and Guestrin (2016) presented LIME, a method to locally approximate models with linear models. The coefficients of the linear approximation are interpreted as local feature contributions.

Building on LIME, Lundberg and Lee (2017) combined the idea of local linear models with the game-theoretic concept of Shapley values naming their approach SHAP (SHapley Additive exPlanation). While the original work is model-agnostic, there exist extensions to specific models, such as decision trees and ensembles (Lundberg et al. 2020) or deep neural networks (Ancona, Oztireli, and Gross 2019).

To avoid exponential complexity, Lundberg and Lee (2017) proposed a randomized algorithm for the computation of SHAP values by sampling subsets of features. This approach is based on the observation that Shapley values arise as a solution to a linear least squares problem (Covert and Lee 2021). While Lundberg and Lee (2017) use this approach for interventional SHAP, Aas, Jullum, and Løland (2021) consider observational SHAP. They use a kernel-based approach to estimate the latent data distribution on a reference dataset. Also based on the least squares approach for Shapley values, Jethani et al. (2021) proposed to train a surrogate model which approximates the SHAP values function for a given model.

While SHAP values provide the feature contribution of the prediction for each individual sample, (Covert, Lundberg, and Lee 2020) proposed an approach, which, based

[2]The source code for the presented algorithms is available at https://github.com/schufa-innovationlab/pltreeshap

on Shapley values, computes the predictive contribution of each feature for the whole dataset. Instead of explaining the model output, they explain the loss of the model for a given dataset.

## 2 Preliminaries

In this section, we briefly review the theory of SHAP values as a foundation for our method in Section 3. As already mentioned, SHAP values are based on game-theoretic Shapley values, which will be introduced first.

**Notation** We denote the number of features by $n$ and the set of feature indices by $N = \{1, \dots, n\}$. The power set of $N$ is denoted by $2^N = \{S \subseteq N\}$ and the cardinality of a set $S \in 2^N$ is denoted by $|S|$.

### 2.1 Shapley Values and Interaction Indices

Considering a cooperative game with $n$ players and a contribution function $v \colon 2^N \to \mathbb{R}$ attributing a payoff for each coalition $S \subseteq N = \{1, \dots, n\}$, the Shapley values (Shapley 1953)

$$\phi_i(v) = \frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \binom{n-1}{|S|}^{-1} \Big( v(S \cup \{i\}) - v(S) \Big) \quad (1)$$

were introduced for distributing the total payoff $v(N)$ to individual players $i \in N$ depending on their contribution in each coalition. These values uniquely result from the following four axioms:

- **Efficiency:** $v(N) = v(\emptyset) + \sum_{i \in N} \phi_i(v)$,
- **Symmetry:** if players $i, j \in N$ have the same contribution, i.e. $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \subseteq N \setminus \{i, j\}$, then $\phi_i(v) = \phi_j(v)$,
- **Dummy player:** if player $i \in N$ has no contribution, i.e $v(S \cup \{i\}) = v(S)$ for all $S \subseteq N \setminus \{i\}$, then $\phi_i(v) = 0$,
- **Linearity:** $\phi_i(\alpha v + w) = \alpha \phi_i(v) + \phi_i(w)$ for any contribution functions $v$ and $w$ and a scalar $\alpha \in \mathbb{R}$.

Furthermore, Shapley interaction indices (Grabisch 1997) were introduced to assign a payoff for the interaction of players $U \subseteq N$ in addition to the payoff of each player $i \in U$. The Shapley interaction index of a coalition $U$ is given by

$$\phi_U(v) = \frac{1}{n - |U| + 1} \sum_{S \subseteq N \setminus U} \binom{n-|U|}{|S|}^{-1} \Delta_U v(S)$$
$$\text{with} \quad \Delta_U v(S) = \sum_{T \subseteq U} (-1)^{|U|-|T|} v(S \cup T). \quad (2)$$

In the case of singletons $U = \{i\}$, Shapley values and Shapley interaction indices coincide. For $U = \{i, j\}$, we will simply write $\phi_{ij}$ instead of $\phi_{\{i,j\}}$.

Using the Möbius transform (Grabisch 1997; Fujimoto, Kojadinovic, and Marichal 2006)

$$v(S) = \sum_{T \subseteq S} \mu_T, \qquad \mu_T = \sum_{S \subseteq T} (-1)^{|T|-|S|} v(S), \quad (3)$$

Shapley values and Shapley interaction indices take the simpler form

$$\phi_U(v) = \sum_{T \supseteq U} \frac{\mu_T}{|T| - |U| + 1}. \quad (4)$$

Especially for linear functions $v(S) = \mu_0 + \sum_{i \in S} \mu_i$, this leads to explicit Shapley values $\phi_i(v) = \mu_i = \mu_{\{i\}}$.

### 2.2 SHAP Values and SHAP Interaction Values

In regards of explaining a prediction $f(x)$ of a model $f \colon \mathbb{R}^n \to \mathbb{R}$, the prediction is considered as an cooperative game with the $n$ features as players. In order to decompose a prediction into feature contributions using Shapley values, the following two contribution functions were proposed (Lundberg and Lee 2017; Lundberg et al. 2020; Chen et al. 2020):

$$v_{\text{obsv}}(S) = \mathop{\mathbb{E}}_{X \sim \mathcal{D}} [f(X) | X_S = x_S], \quad (5a)$$
$$v_{\text{intv}}(S) = \mathop{\mathbb{E}}_{X \sim \mathcal{D}} [f(x_S, X_{N \setminus S})]. \quad (5b)$$

where $\mathcal{D}$ represents the latent data distribution.

The resulting Shapley values and Shapley interaction indices are termed SHAP values and SHAP interaction values respectively, where we distinguish between the observational SHAP using $v_{\text{obsv}}$ and the interventional SHAP using $v_{\text{intv}}$. Both contribution functions fulfill $v(\emptyset) = \mathbb{E}_{X \sim \mathcal{D}}[f(X)]$ and $v(N) = f(x)$ so that by the efficiency axiom the prediction is decomposed into

$$f(x) = \mathop{\mathbb{E}}_{X \sim \mathcal{D}} [f(X)] + \sum_{i \in N} \phi_i(v), \quad (6)$$

i.e. $\phi_i(v)$ can be interpreted as the deviation of $f(x)$ from the mean prediction caused by feature $x_i$.

In the following, we consider only the interventional SHAP.

## 3 Framework

The major disadvantage of SHAP values as presented in Section 2 are the computational costs. Iterating over all subsets of $n$ features, as Equation (1) implies, requires $2^n$ evaluations of the contribution function $v$.

On the other hand, Shapley values can be directly stated for special contribution functions, e.g. linear functions. If it is possible to decompose a contribution function $v$ into few such special contribution functions, the Shapley values of $v$ can be calculated efficiently. We propose such a decomposition for tree-based models like decision trees and piecewise linear regression trees. Due to the linearity of Shapley values, ensembles of trees are also covered.

Our algorithms are based on the decomposition of the power set $2^N$ of features into intervals of subsets. We are aiming especially for intervals on which SHAP values can be computed without iterating over all subsets in the interval. This effectively reduces the iteration of (1) into an iteration over intervals.

In Subsection 3.1, we introduce the generic decomposition of the power set $2^N$ into intervals of subsets and show

how this leads to a decomposition of Shapley values. In Subsections 3.2, we then present a concrete decomposition to compute interventional SHAP values for tree-based models. Based on this decomposition, we propose an approach for aggregating data to speed up SHAP value computation. Finally, we discuss the computational complexity of the proposed algorithms as well as the difference to other feature attribution methods in Subsection 3.4.

## 3.1 Decomposing Shapley Values

Our framework is based on the observation that for a tree-based model not only the prediction function $f\colon \mathbb{R}^n \to \mathbb{R}$ is piecewisely defined but also the contribution function $v\colon 2^N \to \mathbb{R}$ as given in (5) can be represented piecewisely. To this end, we use indicator functions $\mathbf{1}_{\mathcal{U}}\colon 2^N \to \{0,1\}$ ($\mathcal{U} \subseteq 2^N$), which gives $\mathbf{1}_{\mathcal{U}}(S) = 1$ iff $S \in \mathcal{U}$, in combination with set intervals given by

$$[A,B] \coloneqq \{S\colon A \subseteq S \subseteq B\} \quad \text{for } A, B \subseteq N. \tag{7}$$

Note that the interval $[A,B]$ is non-empty if and only if $A \subseteq B$. Further, the Möbius transform (3) now takes the form

$$v(S) = \sum_{T \subseteq N} \mu_T \mathbf{1}_{[T,N]}(S). \tag{8}$$

The goal of our approach is to decompose the contribution functions (5) into the form

$$v(S) = \sum_{[A,B]} v_{[A,B]}(S) \cdot \mathbf{1}_{[A,B]}(S), \tag{9}$$

where the number of intervals $[A,B]$ is low (if possible) and the functions $v_{[A,B]}$ are as simple as possible, e.g., constant or linear. Thereby the contribution functions $v_{[A,B]}$ arise from the data distributions and the predictive models at the leaves. The computation of the intervals is based on

$$\mathbf{1}_{[A,B]} = \mathbf{1}_{[A\cup\{i\},B]} + \mathbf{1}_{[A,B\setminus\{i\}]} \quad \text{for} \quad i \in B \setminus A, \tag{10}$$

applied to splitting features $i \in N$ of the tree, i.e. the interval $[A,B]$ is divided into subsets $S \in [A \cup \{i\}, B]$ containing $i$ and subsets $S \in [A, B \setminus \{i\}]$ not containing $i$. Thus, the decomposition (9) results from the tree structure of the model. Based on the decomposition (9), the Shapley values and interaction indices are computed as follows. By linearity, the Shapley values can be computed summandwise, i.e.

$$\phi_U(v) = \sum_{[A,B]} \phi_U(v_{[A,B]} \cdot \mathbf{1}_{[A,B]}). \tag{11}$$

So the complexity of computing the Shapley value is the complexity of computing $\phi_U(v_{[A,B]} \cdot \mathbf{1}_{[A,B]})$ times the number of intervals. Assuming $v_{[A,B]}$ to be of the form (8) and by using the equation $\mathbf{1}_{[A,B]} \cdot \mathbf{1}_{[C,D]} = \mathbf{1}_{[A\cup C, B\cap D]}$, we get

$$\phi_U(v_{[A,B]}\mathbf{1}_{[A,B]}) = \sum_{T \subseteq B} \mu_T \phi_U(\mathbf{1}_{[A\cup T,B]}). \tag{12}$$

Finally, the right-hand side of (12) can be computed with the following proposition.

**Proposition 1.** *The Shapley values and interaction indices of $\mathbf{1}_{[A,B]}$ with $A \subseteq B \subseteq N$ are given by*

$$\phi_U(\mathbf{1}_{[A,B]}) = (-1)^{|U\cap(N\setminus B)|} \cdot \omega_{|A|-|B\cap U|,|N\setminus(B\cup U)|} \tag{13}$$

*if $U \subseteq A \cup (N \setminus B)$ and $\phi_U(\mathbf{1}_{[A,B]}) = 0$ otherwise. Here, the shortcut $\omega_{a,b} = \frac{1}{a+b+1}\binom{a+b}{a}^{-1}$ was used.*

A proof of this proposition can be found in the appendix. In this paper, $v_{[A,B]}$ in (11) will be constant or linear. Applying the proposition to this case results in the following corollary, whose proof can also be found in the appendix.

**Corollary 1.** *Let $A \subseteq B \subseteq N$ and let $v(S) = \mu_0 + \sum_{j\in S}\mu_j$ be a linear function. Setting $\sigma_A = \mu_0 + \sum_{j\in A}\mu_j$ and $\sigma_{B\setminus A} = \sum_{j\in B\setminus A}\mu_j$, the Shapley values and interaction indices of $v \cdot \mathbf{1}_{[A,B]}$ are given by*

$$\phi_j(v\cdot\mathbf{1}_{[A,B]}) = \begin{cases} \sigma_A \cdot \omega_{|A|-1,|N\setminus B|} \\ \quad + \sigma_{B\setminus A}\cdot\omega_{|A|,|N\setminus B|} & \text{if } j \in A, \\ -\sigma_A \cdot \omega_{|A|,|N\setminus B|-1} \\ \quad - \sigma_{B\setminus A}\cdot\omega_{|A|+1,|N\setminus B|-1} & \text{if } j \in N\setminus B, \\ \mu_j \cdot \omega_{|A|,|N\setminus B|} & \text{if } j \in B\setminus A, \end{cases} \tag{14a}$$

$$\phi_{ij}(v\cdot\mathbf{1}_{[A,B]}) = \begin{cases} \sigma_A \cdot \omega_{|A|-2,|N\setminus B|} \\ \quad + \sigma_{B\setminus A}\cdot\omega_{|A|-1,|N\setminus B|} & \text{if } i,j \in A, \\ \sigma_A \cdot \omega_{|A|,|N\setminus B|-2} \\ \quad + \sigma_{B\setminus A}\cdot\omega_{|A|+1,|N\setminus B|-2} & \text{if } i,j \in N\setminus B, \\ 0 & \text{if } i,j \in B\setminus A, \\ -\sigma_A \cdot \omega_{|A|-1,|N\setminus B|-1} \\ \quad - \sigma_{B\setminus A}\cdot\omega_{|A|,|N\setminus B|-1} & \text{if } i \in A,\, j \in N\setminus B, \\ \mu_j \cdot \omega_{|A|-1,|N\setminus B|} & \text{if } i \in A,\, j \in B\setminus A, \\ -\mu_j \cdot \omega_{|A|,|N\setminus B|-1} & \text{if } i \in N\setminus B,\, j \in B\setminus A. \end{cases} \tag{14b}$$

Now that we have outlined the basic idea of our approach, we will discuss the exact computation of the decomposition (9) for $v_{\text{intv}}$ in the following subsection.

## 3.2 Interventional SHAP

We follow the formulation in (Merrick and Taly 2020) and consider the interventional SHAP values as expectations of baseline Shapley values. More specifically, let

$$z(x,r,S) \in \mathbb{R}^n, \quad z(x,r,S)_i = \begin{cases} x_i & \text{if } i \in S, \\ r_i & \text{else,} \end{cases} \tag{15}$$

be the composite input of $x$ and a reference input $r \in \mathbb{R}^n$. Further let $v_{x,r}(S) = f\big(z(x,r,S)\big)$ be the single-reference game for $f$. Then the interventional SHAP values are calculated by

$$\phi_i(v_{\text{intv}}) = \mathop{\mathbb{E}}_{R\sim\mathcal{D}}[\phi_i(v_{x,R})]. \tag{16}$$

In case of an empirical distribution given by a set of sample points $D \subset \mathbb{R}^n$, we get

$$\phi_i(v_{\text{intv}}) = \frac{1}{|D|}\sum_{r\in D}\phi_i(v_{x,r}). \tag{17}$$

Algorithm 1: Interventional SHAP

```
Input: sample $x \in \mathbb{R}^n$, background data $D \subset \mathbb{R}^n$
Output: SHAP (interaction) values $\hat{\phi}$ of $x$
$\hat{\phi} \leftarrow$ vector of zeros
for $r \in D$ do
    {initialize stack with root node $\nu_0$}
    put $(\nu_0, [\emptyset, N])$ on stack
    while stack is not empty do
        $(\nu, [A, B]) \leftarrow$ top element of the stack (removed)
        if $\nu$ is inner node then
            $i \leftarrow$ splitting feature for node $\nu$
            $\nu_x \leftarrow$ child node of $\nu$ according to $x_i$
            $\nu_r \leftarrow$ child node of $\nu$ according to $r_i$
            if $\nu_x = \nu_r$ then
                put $(\nu_x, [A, B])$ on stack
            else
                put $\big(\nu_x, [A \cup \{i\}, B]\big)$ on stack {only if $i \in B$}
                put $\big(\nu_r, [A, B \setminus \{i\}]\big)$ on stack {only if $i \notin A$}
            end if
        else
            {update SHAP values using Corollary 1}
            $\hat{\phi} \leftarrow \hat{\phi} + \frac{1}{|D|}\phi(v_{\nu,x,r}\mathbf{1}_{[A,B]})$
        end if
    end while
end for
```

For a model tree, the contribution function $v_{x,r}$ takes the form

$$v_{x,r}(S) = \sum_{\nu \in \text{leaf}} v_{\nu,x,r}(S) \cdot \mathbf{1}_{[A_{\nu,x,r}, B_{\nu,x,r}]}(S), \tag{18}$$

where $v_{\nu,x,r}(S) = f_\nu\big(z(x,r,S)\big)$ is the single-reference game of the predictive model $f_\nu$ at leaf $\nu$. The interval $[A_{\nu,x,r}, B_{\nu,x,r}]$ collects all subsets $S$ for which $z(x,r,S)$ lands in leaf $\nu$ and so it characterizes the features $i \in N$ for which the decisions for $x_i$ and $r_i$ differ on the path from the root to the leaf $\nu$. These intervals are given by

$$\begin{aligned} A_{\nu,x,r} &= \{i \in N\colon \text{if an edge according to } x_i \text{ was taken}\}, \\ B_{\nu,x,r} &= N \setminus \{i \in N\colon \text{if edge according to } r_i \text{ was taken}\}, \end{aligned} \tag{19}$$

where features $i \in B_{\nu,x,r} \setminus A_{\nu,x,r}$ either do not occur on the path or the decisions for $x_i$ and $r_i$ were the same. The decomposition (18) is computed by traversing the tree from the root to the leaves, so the decomposition (9) for $v_{\text{intv}}$ results by $|D|$ tree traversals. We have summarized the computation of the Shapley values of $v_{\text{intv}}$ in Algorithm 1.

The algorithm is illustrated in Figure 1 by an example. The interval passed to a node represents the feature subsets $S$ for which the composite input $z(x,r,S)$ traverses that node. In that example, node $\nu_1$ is visited if and only if $1 \in S$. This is expressed by the interval $\big[\{1\}, \{1,2,3\}\big]$.

We note that if $f_\nu$ is constant, then $v_{\nu,x,r}$ is also constant and $\phi_i(v_{\nu,x,r}\mathbf{1}_{[A,B]})$ vanishes for $i \notin A \cup (N \setminus B)$. That means, that only SHAP values of features on the path from the root to the leaf node $\nu$ have to be updated. This is actually

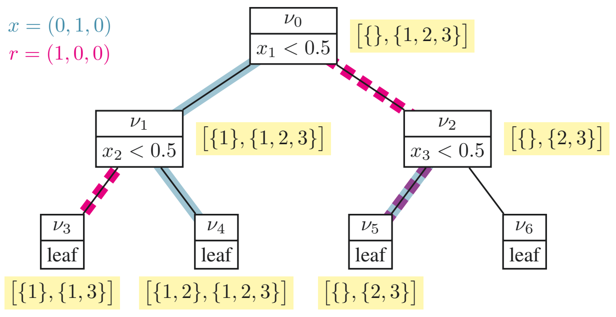

Figure 1: Illustrative example for Algorithm 1. For the instance $x = (0,1,0)$ and the background point $r = (1,0,0)$, the traversal of a tree is visualized. Paths taken by $x$ are highlighted with cyan solid lines. Paths taken by $r$ are highlighted with magenta dashed lines. For each passed node $\nu$, the corresponding set intervals $[A_\nu, B_\nu]$ are also shown. By taking the path for $x$ (cyan line), the splitting feature is added to the set $A$. By taking the path for $r$ (magenta dashed line), the splitting feature is removed from $B$. If both paths overlap as from $\nu_2$ to $\nu_5$, then the interval is passed on unchanged.

used by the interventional TreeSHAP algorithm (Lundberg et al. 2020, Algorithm 3), which traverse the tree from the leaf nodes back to the root to update SHAP values. For a linear model $f_\nu(x) = \langle w, x\rangle + b$ the function $v_{\nu,x,r}$ is also linear, i.e. $v_{\nu,x,r}(S) = \langle w, r\rangle + b + \sum_{j\in S} w_j(x_j - r_j)$. In that case it is not sufficient to consider only splitting features.

Algorithm 1 iterates over the background dataset $D$. This might be already computationally expensive for medium sized datasets. A remedy would be to use a subsample $D_0 \subset D$ for the SHAP computation. Another approach is presented in the following section.

### 3.3 Aggregating Background Data

Different background data points $r \in D$ may take partly the same decision paths resulting in the same intervals $[A_{\nu,x,r}, B_{\nu,x,r}]$ at some leaf nodes. For instance, replacing the reference point $r = (1,0,0)$ in Figure 1 by the point $\tilde{r} = (1,0,1)$ would result in the same intervals on the left side of the tree. So we may aggregate data points according to the decision paths they take. To this end, combining eqs. (17), (18) and using the linearity of $\phi_U$ gives

$$\begin{aligned} \phi_U(v_{\text{intv}}) &= \frac{1}{|D|}\sum_{r\in D}\sum_{\nu\in\text{leaf}} \phi_U(v_{\nu,x,r}\cdot \mathbf{1}_{[A_{\nu,x,r},B_{\nu,x,r}]}) \\ &= \sum_{\nu\in\text{leaf}}\sum_{[A,B]\in\mathcal{I}_{\nu,x}} \phi_U(v_{\nu,[A,B]}\cdot\mathbf{1}_{[A,B]}), \end{aligned} \tag{20}$$

where $\mathcal{I}_{\nu,x}$ contains all intervals which may reach node $\nu$ and $v_{\nu,[A,B]}$ summarizes the contribution functions at $\nu$ for all data points $r \in D$ producing the interval $[A,B]$, i.e.

$$v_{\nu,[A,B]} = \frac{|D_{\nu,[A,B]}|}{|D|} \operatorname*{mean}_{r\in D_{\nu,[A,B]}} v_{\nu,x,r}, \tag{21a}$$

$$D_{\nu,[A,B]} = \big\{r \in D\colon [A_{\nu,x,r}, B_{\nu,x,r}] = [A,B]\big\}. \tag{21b}$$

Algorithm 2: Aggregate Data

```
**Input:** tree $\overline{\mathcal{T}} = (\overline{V}, \overline{E})$ and background data $D \subset \mathbb{R}^n$
**Output:** covers $c\colon \overline{V} \to \mathbb{R}_{\geq 0}$, mean vectors $\overline{r}\colon \overline{V} \to \mathbb{R}^n$
$c(\overline{\nu}) \leftarrow 0$ for all $\overline{\nu} \in \overline{V}$
$\overline{r}(\overline{\nu}) \leftarrow 0$ for all $\overline{\nu} \in \overline{V}$
for $r \in D$ do
  put $\overline{\nu}_0$ on stack    $\{\overline{\nu}_0 = \overline{\nu}_{\nu_0,()}$ is the root node of $\overline{\mathcal{T}}\}$
  while stack is not empty do
    $\overline{\nu}_{\nu,\alpha} \leftarrow$ top element of the stack (removed)
    $c(\overline{\nu}_{\nu,\alpha}) \leftarrow c(\overline{\nu}_{\nu,\alpha}) + \frac{1}{|D|}$
    $\overline{r}(\overline{\nu}_{\nu,\alpha}) \leftarrow \overline{r}(\overline{\nu}_{\nu,\alpha}) + \frac{1}{|D| \cdot c(\overline{\nu}_{\nu,\alpha})}\big(r - \overline{r}(\overline{\nu}_{\nu,\alpha})\big)$
    if $\nu$ is inner node then
      $i \leftarrow$ splitting feature for node $\nu$
      $\nu_1 \leftarrow$ child node of $\nu$ according to $r_i$
      $\nu_2 \leftarrow$ the other child node of $\nu$
      put $\overline{\nu}_{\nu_1,(\alpha,1)}$ on stack
      put $\overline{\nu}_{\nu_2,(\alpha,0)}$ on stack
    end if
  end while
end for
```

Since we regard linear leaf models, we further have

$$\operatorname*{mean}_{r \in D_{\nu,[A,B]}} v_{\nu,x,r} = v_{\nu,x,\overline{r}} \quad \text{with} \quad \overline{r} = \overline{r}_{\nu,[A,B]} = \operatorname{mean}(D_{\nu,[A,B]}). \tag{22}$$

The set $D_{\nu,[A,B]} \subset D$ consists of all data points $r \in D$ such that $A$ is the set of all splitting features on the path from the root to the node $\nu$ for which the corresponding edge is not chosen by $r$. Thus, we may identify intervals with binary vectors $\alpha \in \{0,1\}^{\operatorname{depth}(\nu)}$ indicating if the edge at depth $k$ was chosen by $r$ ($\alpha_k = 1$) or not ($\alpha_k = 0$). In order to precompute the covers $c_{\nu,[A,B]} = \frac{|D_{\nu,[A,B]}|}{|D|}$ and the mean vectors $\overline{r}_{\nu,[A,B]}$, we construct an additional tree structure using these binary vectors. Denoting the original tree by $\mathcal{T} = (V, E)$, the new tree $\overline{\mathcal{T}} = (\overline{V}, \overline{E})$ takes the form

$$\overline{V} = \big\{\overline{\nu}_{\nu,\alpha} \colon \nu \in V,\ \alpha \in \{0,1\}^{\operatorname{depth}(\nu)}\big\}, \tag{23a}$$

$$\overline{E} = \big\{(\overline{\nu}_{\nu_1,\alpha_1}, \overline{\nu}_{\nu_2,\alpha_2}) \colon (\nu_1,\nu_2) \in E,\ \alpha_2 \in \{(\alpha_1,0),(\alpha_1,1)\}\big\}, \tag{23b}$$

i.e. $\overline{\mathcal{T}}$ is a quadtree with each inner node having two copies of corresponding child nodes from the original tree $\mathcal{T}$. Algorithm 2 describes the computation of covers $c\colon \overline{V} \to \mathbb{R}_{\geq 0}$ and mean vectors $\overline{r}\colon \overline{V} \to \mathbb{R}^n$, whose values at the leaf nodes of $\overline{\mathcal{T}}$ represent the covers $c_{\nu,[A,B]}$ and mean vectors $\overline{r}_{\nu,[A,B]}$, respectively. Adjusting Algorithm 1 to use the aggregated data instead of iterating over the background dataset results in Algorithm 3.

Algorithm 3: Interventional SHAP with Aggregated Data

```
**Input:** tree $\overline{\mathcal{T}} = (\overline{V}, \overline{E})$, cover $c\colon \overline{V} \to \mathbb{R}_{\geq 0}$, mean vectors $\overline{r}\colon \overline{V} \to \mathbb{R}^n$, and sample $x \in \mathbb{R}^n$
**Output:** SHAP (interaction) values $\hat{\phi}$ of $x$
$\hat{\phi} \leftarrow$ vector of zeros
put $\big(\overline{\nu}_0, [\emptyset, N]\big)$ on stack    $\{\overline{\nu}_0$ is the root node of $\overline{\mathcal{T}}\}$
while stack is not empty do
  $\big(\overline{\nu}_{\nu,\alpha}, [A,B]\big) \leftarrow$ top element of the stack (removed)
  if $\nu$ is inner node then
    $i \leftarrow$ splitting feature for node $\nu$
    $\nu_1 \leftarrow$ child node of $\nu$ according to $x_i$
    $\nu_2 \leftarrow$ the other child node of $\nu$
    if $c(\overline{\nu}_{\nu_1,(\alpha,1)}) > 0$ then
      {path taken by $x$ and by background data}
      put $\big(\overline{\nu}_{\nu_1,(\alpha,1)}, [A,B]\big)$ on stack
    end if
    if $i \in B$ and $c(\overline{\nu}_{\nu_1,(\alpha,0)}) > 0$ then
      {path taken by $x$ and not by background data}
      put $\big(\overline{\nu}_{\nu_1,(\alpha,0)}, [A \cup \{i\}, B]\big)$ on stack
    end if
    if $i \notin A$ and $c(\overline{\nu}_{\nu_2,(\alpha,1)}) > 0$ then
      {path taken by background data and not by $x$}
      put $\big(\overline{\nu}_{\nu_2,(\alpha,1)}, [A, B \setminus \{i\}]\big)$ on stack
    end if
  else
    $r \leftarrow \overline{r}(\overline{\nu}_{\nu,\alpha})$
    $\hat{\phi} \leftarrow \hat{\phi} + c(\overline{\nu}_{\nu,\alpha}) \cdot \phi(v_{\nu,x,r} \cdot \mathbf{1}_{[A,B]})$
  end if
end while
```

## 3.4 Discussion

**Features with Contribution** For sets $A_\nu$ and $B_\nu$ resulting from Algorithm 1, their difference $B_\nu \setminus A_\nu$ includes all features not present on the path to the node $\nu$. In case of constant leaf models, we have $\phi_i(v_{\nu,x,r}\mathbf{1}_{[A_\nu,B_\nu]}) = 0$ for all $i \in B_\nu \setminus A_\nu$, i.e. only features used by the tree can have nonzero SHAP values. For SHAP interaction values, we get $\phi_{ij}(v_{\nu,x,r}\mathbf{1}_{[A_\nu,B_\nu]}) = 0$ if $i$ or $j$ is in $B_\nu \setminus A_\nu$, i.e. only features along the path interact. In case of linear leaf models, features used by the leaf model also receive a contribution and they interact with features on the path to that leaf node.

**Computational Complexity** Algorithm 1 traverses the tree for every $r \in D$ and updates the Shapley values at the leaf nodes, resulting in $O(|D| \cdot |V|)$ Shapley updates. For piecewise constant trees, only features on the path to the leaf have to be considered. So the complexity of updating SHAP values is $O(d)$ with $d$ being the depth of the tree. For SHAP interaction values, the update is done in $O(d^2)$. In case of linear leaf models, features used by the linear model have also be considered leading to the complexity $O(n)$ for updating SHAP values. Updating the interaction values can be done in $O(dn)$, since the interaction values vanish if none of the two features is on the path to the leaf node.

Aggregating the background data using Algorithm 2 takes $|D|$ traversals of the original tree. So computing the covers takes $O(|D| \cdot |V|)$ time, while computing the mean vectors is done in $O(|D| \cdot |V| \cdot n)$. Note that for piecewise constant trees only the covers are needed.

Algorithm 3 traverses the quadtree $\overline{\mathcal{T}}$, where at most three paths at each inner node are taken. This leads to $O(3^d)$ up-

| Method | SHAP values | SHAP interaction |
|---|---|---|
| Algorithm 1 | $O(bldm)$ | $O(bld^2m)$ |
| Algorithms 2+3 | $O(bl + 3^d dm)$ | $O(bl + 3^d d^2m)$ |
| TreeSHAP (intv.) | $O(bldm)$ | $O(bldnm)$ |
| TreeSHAP (path) | $O(ld^2m)$ | $O(ld^2nm)$ |
| Algorithm 1 | $O(blnm)$ | $O(bldnm)$ |
| Algorithms 2+3 | $O(bln + 3^d nm)$ | $O(bln + 3^d dnm)$ |

**Symbols**

| | |
|---|---|
| $m$ : | number of explained instances $x$ |
| $n$ : | number of features |
| $b$ : | number of background data points |
| $l$ : | number of nodes in the tree |
| $d$ : | depth of the tree |

Table 1: Overview of computational complexities. The upper half of the table shows the complexities of SHAP algorithms for piecewise constant trees including TreeSHAP (Lundberg et al. 2020). The lower part shows the complexities of SHAP algorithms for piecewise linear trees.

dates of Shapley values. In total, computing SHAP values of piecewise linear trees using Algorithm 3 is done in $O(3^d n)$, while the interaction values are computed in $O(3^d dn)$. We note that this complexity is per sample $x \in \mathbb{R}^n$. The advantage of Algorithm 3 is that we do not have to iterate over the whole background dataset for each sample $x$. For instance, computing the SHAP values of a piecewise linear tree for $m$ samples takes $O(|D| \cdot |V| \cdot n \cdot m)$ using Algorithm 1, while it takes $O(|D| \cdot |V| \cdot n + 3^d nm)$ using Algorithms 2 and 3. The discussed complexeties are summarized in Table 1.

**Feature Attributions** Regarding piecewise linear regression trees, one possible feature attribution approach would be using the coefficients of the linear leaf model used for the prediction. This approach would also be similar to LIME (Ribeiro, Singh, and Guestrin 2016), which locally approximates the model by a linear function and uses the coefficients for feature attributions. However, this approach would ignore the contribution of features used in the decision paths. The interventional SHAP on the other hand considers the feature contributions for both decision paths and leaf models, as can be seen by the Shapley updates $\phi(v_{\nu,x,r}\mathbf{1}_{[A,B]})$.

## 4 Evaluation

In this section, we conduct a runtime evaluation of the presented algorithms using various datasets and instances of (piecewise linear) gradient boosting trees.

All experiments were run on a dedicated workstation with the following parameters: Intel Core i7-8700K CPU @ 3.7 GHz, 64 GB main memory.

**Models and Parameters** We evaluate the runtime of the proposed algorithms on Stochastic Gradient Boosting with Trees (Friedman 2002) as implemented through LightGBM (Ke et al. 2017). This includes piecewise constant trees as well as piecewise linear trees. For each evaluated dataset, we select the parameters of the model by a grid search with `n_estimators` $\in \{20, 50, 100, 250, 500, 1000\}$ and `max_depth` $\in \{2, 4, 6, 8, 10\}$.

| Name | ID | # instances | # features |
|---|---|---|---|
| cpu_act | 44132 | 8192 | 21 |
| isolet | 44135 | 7797 | 613 |
| Ailerons | 44137 | 13750 | 33 |
| house_16H | 44139 | 22784 | 16 |
| sulfur | 44145 | 10081 | 6 |
| superconduct | 44148 | 21263 | 79 |
| year | 44027 | 515345 | 90 |

Table 2: Overview of the datasets. The column 'ID' contains the OpenML IDs of the datasets.

**Datasets** Since we focus on runtime evaluation, we simply use publicly available datasets for which piecewise linear regression trees can be readily applied. For this, we use the numerical regression datasets provided by Grinsztajn, Oyallon, and Varoquaux (2022, Sections A.1.2). These datasets are tabular data, which were preprocessed for neural networks and are publicly available on OpenML.[3] We have selected a subset of these datasets for presentation based on the number of instances and number of features. Table 2 provides an overview of the selected datasets.

### 4.1 Runtime Experiments

We evaluate the runtimes of our SHAP implementations on gradient boosting trees trained on different datasets. We consider piecewise linear trees as well as piecewise constant trees. For the latter, we compare the runtime with the path-dependent TreeSHAP (Lundberg et al. 2020, Algorithm 2), which uses aggregated data in form of node counts. We also compare with interventional TreeSHAP (Lundberg et al. 2020, Algorithm 3). We used 75% of the data for the training and as background dataset for SHAP computation. SHAP (interaction) values were computed for 100 instances from the remaining 25% of the data.

The resulting runtimes are presented in Table 3. In most cases the approach with aggregated data (Algorithms 2+3) is more efficient than iterating over 1000 subsampled data points. For our approach, the computation of interaction values has almost the same runtime as for computing SHAP values and the computation time is of similar order as the interaction values computation by TreeSHAP.

### 4.2 Accuracy with Subsampled Background Data

We also evaluate the accuracy of the computed SHAP values for piecewise linear trees when using subsampled background data compared to using the whole background dataset. For each feature, we measured the accuracy of the SHAP values using the $R^2$ score. We averaged these scores to get a single $R^2$ score for the computation. Table 4 shows the results. As these suggest, subsampling the background data might be an inappropriate approach for some datasets, e.g. cpu_act. So the approach using the whole background dataset by aggregation (Algorithms 2+3) is clearly favorable as it also takes less computation time.

[3]www.openml.org

| | Dataset | # trees | depth | TreeSHAP (path) | intv. TreeSHAP (1000) | intv. SHAP (1000) *Algorithm 1* | data aggregation *Algorithm 2* | intv. SHAP (agg.) *Algorithm 3* | TreeSHAP (path) interaction | intv. SHAP (1000) interaction | intv. SHAP (agg) interaction |
|---|---|---|---|---|---|---|---|---|---|---|---|
| p.w. constant | cpu_act | 500 | 4 | 0.009 | 8.38 | 4.84 | 0.38 | 0.09 | 0.72 | 6.32 | 0.15 |
| | isolet | 1000 | 8 | 0.108 | 34.54 | 23.97 | 5.24 | 3.41 | 30.08 | 34.28 | 5.86 |
| | Ailerons | 100 | 10 | 0.009 | 3.25 | 1.93 | 0.40 | 0.24 | 1.10 | 2.51 | 0.34 |
| | house_16H | 50 | 6 | 0.004 | 1.67 | 1.12 | 0.28 | 0.08 | 0.44 | 1.52 | 0.13 |
| | sulfur | 1000 | 8 | 0.054 | 28.65 | 13.60 | 2.26 | 0.96 | 3.00 | 16.37 | 1.29 |
| | superconduct | 500 | 10 | 0.050 | 13.77 | 8.21 | 3.36 | 1.28 | 11.19 | 12.31 | 2.03 |
| | year | 1000 | 10 | 0.129 | 19.88 | 13.86 | 207.00 | 7.57 | 30.90 | 20.08 | 13.95 |
| p.w. linear | cpu_act | 100 | 8 | - | - | 14.47 | 2.41 | 1.19 | - | 23.02 | 1.98 |
| | isolet | 1000 | 8 | - | - | 1590.40 | 608.99 | 216.69 | - | 3927.03 | 618.28 |
| | Ailerons | 250 | 4 | - | - | 18.55 | 5.94 | 0.50 | - | 27.63 | 0.85 |
| | house_16H | 50 | 8 | - | - | 7.04 | 2.68 | 0.62 | - | 10.39 | 0.96 |
| | sulfur | 1000 | 6 | - | - | 58.08 | 6.85 | 2.49 | - | 67.89 | 3.05 |
| | superconduct | 500 | 4 | - | - | 66.25 | 46.86 | 1.63 | - | 126.34 | 3.53 |
| | year | 1000 | 6 | - | - | 189.58 | 6116.39 | 19.70 | - | 431.37 | 54.17 |

Table 3: Average runtime (in seconds) for computing SHAP (interaction) values of 100 instances. The average value was calculated for 10 computations. The upper half of the table contains the evaluation for piecewise constant trees. The lower half is for piecewise linear trees. Evaluated methods are (from left to right) the path-dependent TreeSHAP, interventional TreeSHAP and Algorithm 1 with a subsample of 1000 background data points, Algorithm 2 for data aggregation, and Algorithm 3 for SHAP with aggregated data. The last three columns contain the runtime for computing interaction values.

| Dataset | time (agg.) | time (100) | avr $R^2$ (100) | min $R^2$ (100) | avr $R^2$ (1000) |
|---|---|---|---|---|---|
| cpu_act | 28.3 | 30.4 | 0.79 | 0.69 | 0.983 |
| house_16H | 38.2 | 41.6 | 0.95 | 0.87 | 0.980 |
| sulfur | 68.6 | 148.2 | 0.98 | 0.94 | 0.998 |

Table 4: $R^2$ scores of SHAP values computed with subsampled background data of sizes 100 and 1000. Average and minimum were computed over 10 computations. The computation time (in seconds) is also included.

## 5 Conclusion

In this work, we have proposed a novel decomposition of contribution functions that enables a more comprehensible formulation of SHAP algorithms for tree-based models. Using this decomposition, we derived an efficient method for computing interventional SHAP values and interaction values of piecewise linear regression trees. In addition, we have presented an approach to aggregate data to speed up the computation of SHAP values, which enables the tractable calculation of SHAP values for larger datasets without the need for subsampling. Compared to the path-dependent TreeSHAP algorithm for piecewise constant trees, TreeSHAP is still more efficient in computing SHAP values, while our algorithm computes exact interventional SHAP values and therefore does not suffer from inconsistencies that may arise from TreeSHAP approximations. Moreover, the run time of our method for computing SHAP interaction values is in the same order of magnitude as TreeSHAP.

## A Proofs

In this section, we prove Proposition 1 and Corollary 1 stated in the paper. For the proof of Proposition 1, we will need to prove two lemmata first. The first lemma states the Möbius transform of $\mathbf{1}_{[A,B]}$.

**Lemma 1.** *For $A \subseteq B \subseteq N$, we have*

$$\mathbf{1}_{[A,B]} = \sum_{T\subseteq N\setminus B} (-1)^{|T|}\mathbf{1}_{[A\cup T,N]}. \tag{24}$$

*Proof.* Let $v_{\text{rhs}}$ be the right-hand side of (24). If $S \in [A,B]$, then

$$v_{\text{rhs}}(S) = \underbrace{\mathbf{1}_{[A,N]}(S)}_{=1} + \sum_{\substack{T\subseteq N\setminus B\\ T\neq\emptyset}} (-1)^{|T|}\underbrace{\mathbf{1}_{[A\cup T,N]}(S)}_{=0} = 1.$$

For $S \notin [A,B]$ we have either $A \not\subseteq S$ or $S\cap(N\setminus B)\neq\emptyset$. If $A\not\subseteq S$, then

$$v_{\text{rhs}}(S) = \sum_{T\subseteq N\setminus B} (-1)^{|T|}\underbrace{\mathbf{1}_{[A\cup T,N]}(S)}_{=0} = 0.$$

Finally, if some $i \in S_0 := S\cap(N\setminus B)$ exists, then

$$v_{\text{rhs}}(S) = \sum_{T\subseteq S_0\setminus\{i\}} (-1)^{|T|}\underbrace{\left(\mathbf{1}_{[A\cup T,N]}(S) - \mathbf{1}_{[A\cup T\cup\{i\},N]}(S)\right)}_{=0}.$$

Hence, the values of $v_{\text{rhs}}$ coincide with the values of $\mathbf{1}_{[A,B]}$. □

**Lemma 2.** *For $n\in\mathbb{N}_0$ and $k_0\in\mathbb{N}$, we have*

$$\sum_{k=0}^{n}\binom{n}{k}\frac{(-1)^k}{k+k_0} = \frac{1}{n+k_0}\binom{n+k_0-1}{n}^{-1}. \tag{25}$$

*Proof.* We give a proof by induction on $n \in \mathbb{N}_0$. The case $n = 0$ is obvious and the induction step reads

$$\begin{aligned}
&\sum_{k=0}^{n+1} \binom{n+1}{k} \frac{(-1)^k}{k+k_0} \\
&= \frac{1}{k_0} + \frac{(-1)^{n+1}}{n+1+k_0} + \sum_{k=1}^{n} \underbrace{\binom{n+1}{k}}_{=\binom{n}{k}+\binom{n}{k-1}} \frac{(-1)^k}{k+k_0} \\
&= \sum_{k=0}^{n} \binom{n}{k} \frac{(-1)^k}{k+k_0} + \sum_{k=1}^{n+1} \binom{n}{k-1} \frac{(-1)^k}{k+k_0} \\
&= \sum_{k=0}^{n} \binom{n}{k} \frac{(-1)^k}{k+k_0} - \sum_{k=0}^{n} \binom{n}{k} \frac{(-1)^k}{k+1+k_0} \\
&\overset{(25)}{=} \frac{1}{n+k_0} \binom{n+k_0-1}{n}^{-1} - \frac{1}{n+1+k_0} \binom{n+k_0}{n}^{-1} \\
&= \frac{1}{n+1+k_0} \binom{n+k_0}{n+1}^{-1}.
\end{aligned}$$

□

*Proof of Proposition 1.* Equation (4) implies $\phi_U(\mathbf{1}_{[T,N]}) = \frac{1}{|T|-|U|+1}$ for any $T$ with $U \subseteq T \subseteq N$. Thus we get

$$\begin{aligned}
\phi_U(\mathbf{1}_{[A,B]}) &\overset{(24)}{=} \phi_U\Bigg(\sum_{T \subseteq N \setminus B} (-1)^{|T|} \mathbf{1}_{[A \cup T, N]}\Bigg) \\
&= \sum_{\substack{T \subseteq N \setminus B \\ T \cup A \supseteq U}} \tfrac{(-1)^{|T|}}{|T|+|A|-|U|+1}.
\end{aligned}$$

We assume $U \subseteq A \cup (N \setminus B)$, since otherwise the condition of the sum cannot hold and we get $\phi_U(\mathbf{1}_{[A,B]}) = 0$. Let $U_0 := U \cap (N \setminus B)$. The condition $T \cup A \supseteq U$ (with $T \subseteq N \setminus B$) can be stated as $T = U_0 \,\dot\cup\, L$ with $L \subseteq (N \setminus B) \setminus U_0$. So we get

$$\begin{aligned}
\phi_U(\mathbf{1}_{[A,B]}) &= \sum_{L \subseteq (N \setminus B) \setminus U_0} \tfrac{(-1)^{|L|+|U_0|}}{|L|+|U_0|+|A|-|U|+1} \\
&= \sum_{k=0}^{n-|B|-|U_0|} \tbinom{n-|B|-|U_0|}{k} \tfrac{(-1)^{k+|U_0|}}{k+|U_0|+|A|-|U|+1} \\
&\overset{(25)}{=} \tfrac{(-1)^{|U_0|}}{n-|B|+|A|-|U|+1} \tbinom{n-|B|+|A|-|U|}{n-|B|-|U_0|}^{-1} \\
&= (-1)^{|U_0|} \cdot \omega_{|A|-|U|+|U_0|, n-|B|-|U_0|}.
\end{aligned}$$

Using $U \setminus U_0 = B \cap U$ and $(N \setminus B) \setminus U_0 = N \setminus (B \cup U)$ completes the proof. □

*Proof of Corollary 1.* We have

$$v = \mu_0 \mathbf{1}_{[\emptyset,N]} + \sum_{l \in N} \mu_l \mathbf{1}_{[\{l\},N]}.$$

Using $\mathbf{1}_{[A,B]} \cdot \mathbf{1}_{[C,D]} = \mathbf{1}_{[A \cup C, B \cap D]}$ and the linearity of $\phi_U$ gives

$$\begin{aligned}
\phi_U(v \cdot \mathbf{1}_{[A,B]}) &= \phi_U\Big(\mu_0 \mathbf{1}_{[A,B]} + \sum_{l \in N} \mu_l \mathbf{1}_{[A \cup \{l\}, B]}\Big) \\
&= \Big(\mu_0 + \sum_{l \in A} \mu_l\Big) \phi_U(\mathbf{1}_{[A,B]}) \\
&\quad + \sum_{l \in B \setminus A} \mu_l \phi_U(\mathbf{1}_{[A \cup \{l\}, B]}).
\end{aligned} \tag{26}$$

Note, that we also used $\mathbf{1}_{[A \cup \{l\}, B]} \equiv 0$ if $l \in N \setminus B$. From here on, the statement of the corollary results from simple application of Proposition 1 on the equation (26) for every case in (14). For example, let $U = \{j\} \subseteq B \setminus A$. This implies $U \not\subseteq A \cup (N \setminus B)$ and therefore $\phi_U(\mathbf{1}_{[A,B]}) = 0$. It also implies $U \subseteq (A \cup \{l\}) \cup (N \setminus B)$ if and only if $l = j$. Hence, we get

$$\phi_U(\mathbf{1}_{[A \cup \{l\}, B]}) = \begin{cases} \omega_{|A|, |N \setminus B|} & \text{if } l = j, \\ 0 & \text{if } l \neq j, \end{cases}$$

by Proposition 1. So in total, $\phi_j(v \cdot \mathbf{1}_{[A,B]}) = \mu_j \cdot \omega_{|A|,|N \setminus B|}$ for $j \in B \setminus A$. The other cases are handled analogously. □

## References

Aas, K.; Jullum, M.; and Løland, A. 2021. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *Artificial Intelligence*, 298: 103502.

Ancona, M.; Oztireli, C.; and Gross, M. 2019. Explaining deep neural networks with a polynomial time algorithm for Shapley value approximation. In *International Conference on Machine Learning*, 272–281. PMLR.

Arenas, M.; Barceló, P.; Bertossi, L.; and Monet, M. 2021a. On the Complexity of SHAP-Score-Based Explanations: Tractability via Knowledge Compilation and Non-Approximability Results. arXiv:2104.08015.

Arenas, M.; Barceló, P.; Bertossi, L.; and Monet, M. 2021b. The Tractability of SHAP-Score-Based Explanations for Classification over Deterministic and Decomposable Boolean Circuits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8): 6670–6678.

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

Broelemann, K.; and Kasneci, G. 2019. A Gradient-Based Split Criterion for Highly Accurate and Transparent Model Trees. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2030–2037. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-4-1.

Chen, H.; Janizek, J. D.; Lundberg, S.; and Lee, S.-I. 2020. True to the Model or True to the Data? arXiv:2006.16234.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Covert, I.; and Lee, S.-I. 2021. Improving KernelSHAP: Practical Shapley value estimation using linear regression. In *International Conference on Artificial Intelligence and Statistics*, 3457–3465. PMLR.

Covert, I.; Lundberg, S. M.; and Lee, S.-I. 2020. Understanding global feature contributions with additive importance measures. *Advances in Neural Information Processing Systems*, 33.

Datta, A.; Sen, S.; and Zick, Y. 2016. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, 598–617. IEEE.

de Vito, L. 2017. LinXGBoost: Extension of XGBoost to generalized local linear models. arXiv:1710.03634.

Du, M.; Liu, N.; and Hu, X. 2019. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1): 68–77.

Friedman, J. H. 2002. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4): 367–378.

Fujimoto, K.; Kojadinovic, I.; and Marichal, J.-L. 2006. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior*, 55(1): 72–99.

Grabisch, M. 1997. K-order additive discrete fuzzy measures and their representation. *Fuzzy sets and systems*, 92(2): 167–189.

Grinsztajn, L.; Oyallon, E.; and Varoquaux, G. 2022. Why do tree-based models still outperform deep learning on tabular data? arXiv:2207.08815.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2018. A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*, 51(5): 1–42.

Guryanov, A. 2019. Histogram-Based Algorithm for Building Gradient Boosting Ensembles of Piecewise Linear Decision Trees. In *International Conference on Analysis of Images, Social Networks and Texts*, 39–50. Springer.

Haug, J.; Broelemann, K.; and Kasneci, G. 2022. Dynamic Model Tree for Interpretable Data Stream Learning. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*, 2562–2574. IEEE.

Ignatiev, A. 2020. Towards Trustable Explainable AI. In *IJCAI*, 5154–5158.

Janzing, D.; Minorics, L.; and Blöbaum, P. 2020. Feature relevance quantification in explainable AI: A causal problem. In *International Conference on artificial intelligence and statistics*, 2907–2916. PMLR.

Jethani, N.; Sudarshan, M.; Covert, I. C.; Lee, S.-I.; and Ranganath, R. 2021. FastSHAP: Real-Time Shapley Value Estimation. In *International Conference on Learning Representations*.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Kumar, I.; Venkatasubramanian, S.; Scheidegger, C.; and Friedler, S. 2020. Problems with Shapley-value-based explanations as feature importance measures. In *Proceedings of the International Conference on Machine Learning*, 8083–8092.

Landwehr, N.; Hall, M.; and Frank, E. 2005. Logistic model trees. *Machine learning*, 59(1-2): 161–205.

Lundberg, S. M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J. M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; and Lee, S.-I. 2020. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1): 56–67.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 4768–4777. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964.

Merrick, L.; and Taly, A. 2020. The Explanation Game: Explaining Machine Learning Models Using Shapley Values. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 17–38. Springer.

Miroshnikov, A.; Kotsiopoulos, K.; and Kannan, A. R. 2021. Mutual information-based group explainers with coalition structure for machine learning model explanations. arXiv:2102.10878.

Potts, D.; and Sammut, C. 2005. Incremental learning of linear model trees. *Machine Learning*, 61(1-3): 5–48.

Quinlan, J. R.; et al. 1992. Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, 343–348. World Scientific.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-Precision Model-Agnostic Explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

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

Shi, Y.; Li, J.; and Li, Z. 2019. Gradient Boosting with Piece-Wise Linear Regression Trees. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 3432–3438. International Joint Conferences on Artificial Intelligence Organization.

Van den Broeck, G.; Lykov, A.; Schleich, M.; and Suciu, D. 2021. On the Tractability of SHAP Explanations. In *Proceedings of the 35th Conference on Artificial Intelligence (AAAI)*.

Wang, Y.; and Witten, I. H. 1997. Induction of model trees for predicting continuous classes. In *Poster papers of the 9th European Conference on Machine Learning*. Springer.

Yang, J. 2021. Fast TreeSHAP: Accelerating SHAP Value Computation for Trees. arXiv:2109.09847.