SHAFF: Fast and consistent SHApley eFfect estimates via random Forests

Clément Bénard[1,2] **Gérard Biau**[2] **Sébastien Da Veiga**[1] **Erwan Scornet**[3]

[1] Safran Tech, Modeling & Simulation, 78114 Magny-Les-Hameaux, France
[2] Sorbonne Université, CNRS, LPSM, 75005 Paris, France
[3] Ecole Polytechnique, IP Paris, CMAP, 91128 Palaiseau, France
clement.benard@safrangroup.com

Abstract

Interpretability of learning algorithms is crucial for applications involving critical decisions, and variable importance is one of the main interpretation tools. Shapley effects are now widely used to interpret both tree ensembles and neural networks, as they can efficiently handle dependence and interactions in the data, as opposed to most other variable importance measures. However, estimating Shapley effects is a challenging task, because of the computational complexity and the conditional expectation estimates. Accordingly, existing Shapley algorithms have flaws: a costly running time, or a bias when input variables are dependent. Therefore, we introduce **SHAFF**, **SHA**pley e**F**fects via random **F**orests, a fast and accurate Shapley effect estimate, even when input variables are dependent. We show **SHAFF** efficiency through both a theoretical analysis of its consistency, and the practical performance improvements over competitors with extensive experiments. An implementation of **SHAFF** in C++ and R is available online.

1 Introduction

State-of-the-art learning algorithms are often qualified as black-boxes because of the high number of operations required to compute predictions. This complexity prevents to grasp how inputs are combined to generate the output, which is a strong limitation for many applications, especially those with critical decisions at stake—healthcare is a typical example. For this reason, interpretability of machine learning has become a topic of strong interest in the past few years. One of the main tools to interpret learning algorithms is variable importance, which enables to identify and rank the influential features of the problem. Recently, Shapley effects have been widely accepted as a very efficient variable importance measure since they can equitably handle interactions and dependence within input variables [Owen, 2014, Štrumbelj and Kononenko, 2014, Iooss and Prieur, 2017, Lundberg and Lee, 2017]. Shapley values were originally defined in economics and game theory [Shapley, 1953] to solve the problem of attributing the value produced by a joint team to its individual members. The main idea is to measure the difference of produced value between a subset of the team and the same subteam with an additional member. For a given member, this difference is averaged over all possible subteams and gives his Shapley value. Recently, Owen [2014] adapted Shapley values to the problem of variable importance in machine learning, where an input variable plays the role of a member of the team, and the produced value is the explained output variance. In this context, Shapley values are now called Shapley effects, and are extensively used to interpret both tree ensembles and neural networks. Next, Lundberg and Lee [2017] also introduced SHAP values to adapt Shapley effects to local importance measures, which break down the contribution of each variable for a given prediction. We focus on Shapley effects throughout the article, but our approach can be easily adpated to SHAP values as they share the same challenges.

Preprint. Under review.

The objective of variable importance is essentially to perform variable selection. More precisely, it is possible to identify two final aims [Genuer et al., 2010]: (i) find a small number of variables with a maximized accuracy, or (ii) detect and rank all influential variables to focus on for interpretation and further exploration with domain experts. The following example illustrates that different strategies should be used depending on the targeted objective: if two influential variables are strongly correlated, one must be discarded for objective (i), while the two must be kept in the second case. Indeed, if two variables convey the same statistical information, only one should be selected if the goal is to maximize the predictive accuracy with a small number of variables, i.e., objective (i). On the other hand, these two variables may be acquired differently and represent distinct physical quantities. Therefore, they may have different interpretations for domain experts, and both should be kept for objective (ii). Shapley effects are a relevant measure of variable importance for objective (ii), because they equitably allocate contributions due to interactions and dependence across all input variables.

The main obstacle to estimate Shapley effects is the computational complexity. The first step is to use a learning algorithm to generalize the relation between the inputs and the output. Most existing Shapley algorithms are agnostic to the learning model. Lundberg et al. [2018] open an interesting route by restricting their algorithm to tree ensembles, in order to develop fast greedy heuristics, specific to trees. Unfortunately, as mentioned by Aas et al. [2019], the algorithm is biased when input variables are dependent. In the present contribution, we focus our Shapley algorithm on random forests, well known for their good behavior on high-dimensional or noisy data, and their robustness. Using the specific structure of random forests, we develop **SHAFF**, a fast and accurate Shapley effect estimate.

Shapley effects. To formalize Shapley effects, we introduce a standard regression setting with an input vector $\mathbf{X} = (X^{(1)}, \dots, X^{(p)}) \in \mathbb{R}^p$, and an output $Y \in \mathbb{R}$. We denote by $\mathbf{X}^{(U)}$ the subvector with only the components in $U \subset \{1, \dots, p\}$. Formally, the Shapley effect of the j-th variable is defined by

$$Sh^{\star}(X^{(j)}) = \sum_{U \subset \{1,\dots,p\} \setminus \{j\}} \frac{1}{p} \binom{p-1}{|U|}^{-1} \frac{\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U \cup \{j\})}]] - \mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]}{\mathbb{V}[Y]}.$$

In other words, the Shapley effect of $X^{(j)}$ is the additional output explained variance when j is added to a subset $U \subset \{1, \dots, p\}$, averaged over all possible subsets. The variance difference is averaged for a given size of U through the combinatorial weight, and then the average is taken over all U sizes through the term $1/p$. Observe that the sum has 2^{p-1} terms, and each of them requires to estimate $\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]$, which is computationally costly. Overall, two obstacles arise to estimate Shapley effects:

1. the computational complexity is exponential with the dimension p;
2. $\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]$ requires a fast and accurate estimate for all variable subsets $U \subset \{1, \dots, p\}$.

In the literature, efficient strategies have been developed to handle these two issues. They all have drawbacks: they are either fast but with a limited accuracy, or accurate but computationally costly. We will see how **SHAFF** considerably improves this trade-off.

Related work. The computational issue of Shapley algorithms—1. above—is solved using Monte-Carlo methods in general [Song et al., 2016, Lundberg and Lee, 2017, Covert et al., 2020, Williamson and Feng, 2020, Covert and Lee, 2020]. In the case of tree ensembles, specific heuristics based on the tree structure enable to simplify the algorithm complexity [Lundberg et al., 2018].

For the second issue of conditional expectation estimates—2. above, two main approaches exist: train one model for each selected subset of variables (accurate but computationally costly) [Williamson and Feng, 2020], or train a single model once with all input variables and use greedy heuristics to derive the conditional expectations (fast but limited accuracy). In the latter case, existing algorithms estimate the conditional expectations with a quite strong bias when input variables are dependent. More precisely, Lundberg and Lee [2017, kernelSHAP], Covert et al. [2020, SAGE], and Covert and Lee [2020] simply replace the conditional expectations by the marginal distributions, Lundberg et al. [2018] use a greedy heuristic specific to tree ensembles, and Broto et al. [2020] leverage k-nearest neighbors to approximate sampling from the conditional distributions. Besides, efficient algorithms exist when it is possible to draw samples from the conditional distributions of the inputs [Song et al.,

Reference	Model	Local or global	Subset sampling	Conditional expectations
Song et al. [2016]	All	global	permutation	kown conditional distributions
Lundberg and Lee [2017]	All	local	Monte-Carlo	marginals
Lundberg et al. [2018]	tree ensembles	local	greedy heuristic	greedy heuristic
Aas et al. [2019]	All	local	Monte-Carlo	kown conditional distributions
Covert et al. [2020]	All	global	Monte-Carlo	marginals
Broto et al. [2020]	All	global	brute force	k-nearest neighbors
Williamson and Feng [2020]	All	global	Monte-Carlo	retrain model
Covert and Lee [2020]	All	local	Monte-Carlo	marginals

Table 1: State-of-the-art of Shapley algorithms.

2016, Aas et al., 2019, Broto et al., 2020]. However, we only have access to a finite sample in practice, and the input dimension p can be large, which implies that estimating the conditional distributions of the inputs is a very difficult task. This last type of methods is therefore not really appropriate in our setting—see Table 1 for a summary of the existing Shapley algorithms.

As mentioned above, several of the presented methods provide local importance measures for specific prediction points, called SHAP values [Lundberg and Lee, 2017, Lundberg et al., 2018, Covert and Lee, 2020]. Their final objective differs from ours, since we are interested in global estimates. However, SHAP values share the same challenges as Shapley effects: the computational complexity and the conditional expectation estimates, and our approach can therefore be adapted to SHAP values. Let us also mention that several recent articles discuss Shapley values in the causality framework [Frye et al., 2020, Heskes et al., 2020, Janzing et al., 2020, Wang et al., 2021]. These works have a high potential since causality is quite often the ultimate goal when one is looking for interpretations. However, causality methods require strong prior knowledge and assumptions about the studied system, and can therefore be difficult to apply in some applications. In these cases, we argue that the best way to go is to use standard Shapley effects to detect and rank influential variables, as a starting point to deepen the analysis with domain experts.

Outline. We leverage random forests to develop **SHAFF**, a fast and accurate Shapley effect estimate. Such remarkable performance is reached by combining two new features. Firstly, we improve the Monte-Carlo approach by using importance sampling to focus on the most relevant subsets of variables identified by the forest. Secondly, we develop a projected random forest algorithm to compute fast and accurate estimates of the conditional expectations for any variable subset. The algorithm details are provided in Section 2. Next, we prove the consistency of **SHAFF** in Section 3. To our knowledge, **SHAFF** is the first Shapley effect estimate, which is both computationally fast and consistent in a general setting. In Section 4, several experiments show the practical improvement of our method over state-of-the-art algorithms.

2 SHAFF Algorithm

Existing approach. **SHAFF** builds on two Shapley algorithms: Lundberg and Lee [2017, kernelSHAP] and Williamson and Feng [2020]. From these approaches, we can deduce the following general three-step procedure to estimate Shapley effects. First, a set $\mathcal{U}_{n,K}$ of K variable subsets $U \subset \{1, \ldots, p\}$ is randomly drawn. Next, an estimate $\hat{v}_n(U)$ of $\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]$ is computed for all selected U from an available sample $\mathscr{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ of n independent random variables distributed as $(\mathbf{X}, Y)$. Finally, Shapley effects are defined as the least square solution of a weighted linear regression problem. If $I(U)$ is the binary vector of dimension p where the j-th component takes the value 1 if $j \in U$ and 0 otherwise, Shapley effect estimates are the minimum in β of the following cost function:

$$\ell_n(\beta) = \frac{1}{K} \sum_{U \in \mathcal{U}_{n,K}} w(U)(\hat{v}_n(U) - \beta^T I(U))^2,$$

where the weights $w(U)$ are given by

$$w(U) = \frac{p-1}{\binom{p}{|U|}|U|(p-|U|)},$$

and the coefficient vector β is constrained to have its components sum to $\hat{v}_n(\{1,\ldots,p\})$.

Algorithm overview. **SHAFF** introduces two new critical features to estimate Shapley effects efficiently, using an initial random forest model. Firstly, we apply importance sampling to select variable subsets $U \subset \{1,\ldots,p\}$, based on the variables frequently selected in the forest splits. This favors the sampling of subsets U containing influential and interacting variables. Secondly, for each selected subset U, the variance of the conditional expectation is estimated with the projected forest algorithm described below, which is both a fast and consistent approach. We will see that these features considerably reduce the computational cost and the estimate error. To summarize, once an initial random forest is fit, **SHAFF** proceeds in three steps:

1. sample many subsets U, typically a few hundreds, based on their occurrence frequency in the random forest (Subsection 2.1);
2. estimate $\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]$ with the projected forest algorithm for all selected U and their complementary sets $\{1,\ldots,p\} \setminus U$ (Subsection 2.2);
3. solve a weighted linear regression problem to recover Shapley effects (Subsection 2.3).

Initial random forest. Prior to **SHAFF**, a random forest is fit with the training sample $\mathscr{D}_n$ to generalize the relation between the inputs **X** and the output Y. A large number M of CART trees are averaged to form the final forest estimate $m_{M,n}(\mathbf{x}, \mathbf{\Theta}_M)$, where **x** is a new query point, and each tree is randomized by a component of $\mathbf{\Theta}_M = (\Theta_1,\ldots,\Theta_\ell,\ldots,\Theta_M)$. Each Θ_ℓ is used to bootstrap the data prior to the ℓ-th tree growing, and to randomly select `mtry` variables to optimize the split at each node. `mtry` is a parameter of the forest, and its efficient default value is $p/3$. In the sequel, we will need the forest parameter `min_node_size`, which is the minimum number of observations in a terminal cell of a tree, as well as the out-of-bag (OOB) sample of the ℓ-th tree: the observations which are left aside in the bootstrap sampling prior to the construction of tree ℓ. Given this initial random forest, we can now detail the main three steps of **SHAFF**.

2.1 Importance Sampling

The Shapley effect formula for a given variable $X^{(j)}$ sums terms over all subsets of variables $U \subset \{1,\ldots,p\} \setminus \{j\}$, which makes 2^{p-1} terms, an intractable problem in most cases. **SHAFF** uses importance sampling to draw a reasonable number of subsets U, typically a few hundreds, while preserving a high accuracy of the Shapley estimates. We take advantage of the initial random forest to define an importance measure for each variable subset U, used as weights for the importance sampling distribution.

Variable subset importance. In a tree construction, the best split is selected at each node among `mtry` input variables. Therefore, as highlighted by Proposition 1 in Scornet et al. [2015], the forest naturally splits on influential variables. **SHAFF** leverages this idea to define an importance measure for all variable subsets $U \subset \{1,\ldots,p\}$ as the probability that a given U occurs in a path of a tree of the forest. Empirically, this means that we count the occurrence frequency of U in the paths of the M trees of the forest, and denote it by $\hat{p}_{M,n}(U)$. Such approach is inspired by Basu et al. [2018] and Bénard et al. [2021a]. This principle is illustrated with the following simple example in dimension $p = 10$. Let us consider a tree, where the root node splits on variable $X^{(5)}$, the left child node splits on variable $X^{(3)}$, and the subsequent left child node at the third tree level, on variable $X^{(2)}$. Thus, the path that leads to the extreme left node at the fourth level uses the following index sequence of splitting variables: $\{5, 3, 2\}$. All in all, the following variable subsets are included in this tree path: $U = \{5\}$, $U = \{3, 5\}$, and $U = \{2, 3, 5\}$. Then, **SHAFF** runs through the forest to count the number of times each subset U occurs in the forest paths, and computes the associated frequency $\hat{p}_{M,n}(U)$. If a subset U does not occur in the forest, we obviously have $\hat{p}_{M,n}(U) = 0$. Notice that the computational complexity of this step is linear: $O(Mn)$.

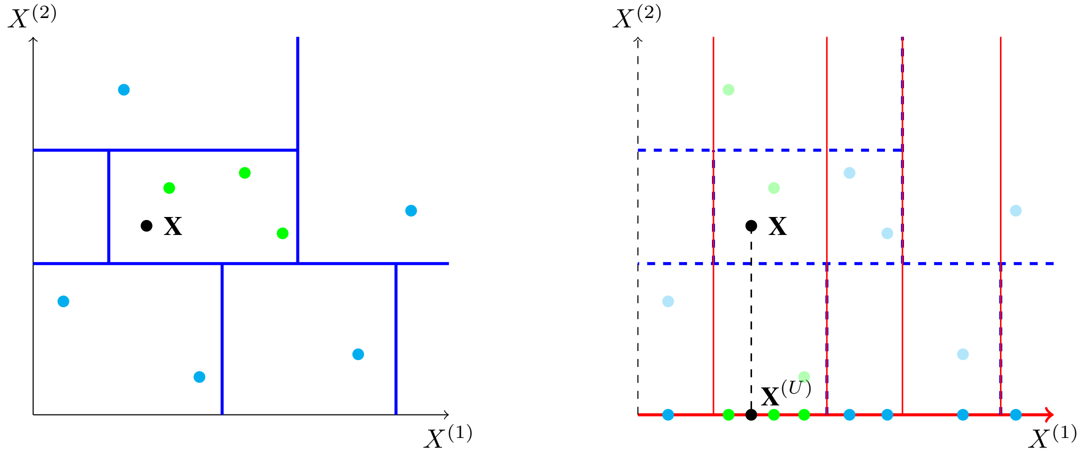

Figure 1: Example of the partition of $[0, 1]^2$ by a random CART tree (left side) projected on the subspace spanned by $\mathbf{X}^{(U)} = X^{(1)}$ (right side). Here, $p = 2$ and $U = \{1\}$.

Paired importance sampling. The occurrence frequencies $\hat{p}_{M,n}(U)$ defined above are scaled to sum to 1, and then define a discrete distribution for the set of all subsets of variables $U \subset \{1, \ldots, p\}$, excluding the full and empty sets. By construction, this distribution is skewed towards the subsets U containing influential variables and interactions, and is used for the importance sampling. Finally, **SHAFF** draws a number K of subsets U with respect to this discrete distribution, where K is a hyperparameter of the algorithm. We define $\mathcal{U}_{n,K}$ the random set of the selected variable subsets U. For all $U \in \mathcal{U}_{n,K}$, **SHAFF** also includes the complementary set $\{1, \ldots, p\} \setminus U$ in $\mathcal{U}_{n,K}$, as Covert and Lee [2020] show that this "paired sampling" improves the final Shapley estimate accuracy. Clearly, the computational complexity and the accuracy of the algorithm increase with K. The next step of **SHAFF** is to efficiently estimate $\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]$ for all drawn $U \in \mathcal{U}_{n,K}$.

2.2 Projected Random Forests

In order to estimate $\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]$ for the selected variable subsets $U \in \mathcal{U}_{n,K}$, most existing methods use greedy algorithms. However, such estimates are not accurate in moderate or large dimensions when input variables are dependent [Aas et al., 2019, Sundararajan and Najmi, 2020]. Another approach is to train a new model for each subset U, but this is computationally costly [Williamson and Feng, 2020]. To solve this issue, we design the projected random forest algorithm (PRF), to obtain a fast and accurate estimate of $\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]/\mathbb{V}[Y]$ for any variable subset $U \subset \{1, \ldots, p\}$.

PRF principle. PRF takes as inputs the initial forest and a given subset U. The general principle is to project the partition of each tree of the forest on the subspace spanned by the variables in U, as illustrated in Figure 1. Then the training data is spread across this new tree partitions, and the cell outputs are recomputed by averaging the output Y_i of the observations falling in each new cell, as in the original forest. The projection enables to eliminate the variables not contained in U from the tree predictions, and thus to estimate $\mathbb{E}[Y|\mathbf{X}^{(U)}]$ instead of $\mathbb{E}[Y|\mathbf{X}]$. Finally, the predictions for the out-of-bag samples are computed with the projected tree estimates, and averaged across all trees. The obtained predictions are used to estimate the targeted normalized variance $\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]/\mathbb{V}[Y]$, denoted by $\hat{\nu}_{M,n}(U)$. More formally, we let $m_{M,n}^{(U,OOB)}(\mathbf{X}_i^{(U)}, \mathbf{\Theta}_M)$ be the out-of-bag PRF estimate for observation i and subset U, and take

$$\hat{\nu}_{M,n}(U) = 1 - \frac{1}{n\hat{\sigma}_Y} \sum_{i=1}^{n} \big(Y_i - m_{M,n}^{(U,OOB)}(\mathbf{X}_i^{(U)}, \mathbf{\Theta}_M)\big)^2,$$

where $\hat{\sigma}_Y$ is the standard estimate of $\mathbb{V}[Y]$.

PRF algorithm. The critical feature of PRF is the algorithmic trick to compute the projected partition efficiently, leaving the initial tree structures untouched. Indeed, a naive computation of the projected partitions from the cell edges is computationally very costly, as soon as the dimension increases. Instead, we simply drop observations down the initial trees, ignoring splits which use a variable outside of U. This enables to recover the projected partitions with an efficient computational complexity. To explain this mechanism in details, we focus on a given tree of the initial forest. Thus, the training observations are dropped down the tree, and when a split involving a variable outside of U is met, data points are sent both to the left and right children nodes. Consequently, each observation falls in multiple terminal leaves of the tree. We drop the new query point $\mathbf{X}^{(U)}$ down the tree, following the same procedure, and retrieve the set of terminal leaves where $\mathbf{X}^{(U)}$ falls. Next, we collect the training observations which belong to every terminal leaf of this collection, in other words, we intersect the collection of leaves where $\mathbf{X}^{(U)}$ falls. Finally, we average the outputs Y_i of the selected training points to generate the tree prediction for $\mathbf{X}^{(U)}$. Notice that such set of selected observations can be empty if $\mathbf{X}^{(U)}$ belongs to a large collection of terminal leaves. To avoid this issue, PRF uses the following strategy. Recall that a partition of the input space is associated to each tree level, and consequently, a projected tree partition can also be defined at each tree level. Thus, when $\mathbf{X}^{(U)}$ is dropped down the tree, it is stopped before reaching a tree level where it falls in an empty cell of the associated projected partition. Overall, this mechanism is equivalent to the projection of the tree partition on the subspace span by $X^{(U)}$, because all splits on variables $X^{(j)}$ with $j \notin U$ are ignored, and the resulting overlapping cells are intersected—see Figure 1.

PRF computational complexity. An efficient implementation of the PRF algorithm is detailed in Algorithm 1 in the Supplementary Material. The computational complexity of PRF for all $U \in \mathcal{U}_{n,K}$ does not depend on the dimension p, is linear with M, K, and quasi-linear with n: $O(MKn\log(n))$. PRF is therefore faster than growing K random forests from scratch, one for each subset U, which has an averaged complexity of $O(MKpn\log^2(n))$ [Louppe, 2014]. The computational gain of **SHAFF** can be considerable in high dimension, since the complexity of all competitors depends on p—see the Supplementary Material for a detailed computational complexity analysis. Notice that the PRF algorithm is close in spirit to a component of the Sobol-MDA [Bénard et al., 2021b], used to measure the loss of output explained variance when an input variable j is removed from a random forest. In particular, a naive adaptation leads to a quadratic complexity with respect to the sample size n, whereas our PRF algorithm has a quasi-linear complexity, which makes it operational. Finally, the last step of **SHAFF** is to take advantage of the estimated $\hat{v}_{M,n}(U)$ for $U \in \mathcal{U}_{n,K}$ to recover Shapley effects.

2.3 Shapley Effect Estimates

The importance sampling introduces the corrective terms $\hat{p}_{M,n}(U)$ in the final loss function. Thus, **SHAFF** estimates $\hat{\mathbf{Sh}}_{M,n} = (\hat{Sh}_{M,n}(X^{(1)}), \dots, \hat{Sh}_{M,n}(X^{(p)}))$ as the minimum in β of the following cost function

$$\ell_{M,n}(\beta) = \frac{1}{K} \sum_{U \in \mathcal{U}_{n,K}} \frac{w(U)}{\hat{p}_{M,n}(U)} (\hat{v}_{M,n}(U) - \beta^T I(U))^2,$$

where the sum of the components of β is constrained to be the proportion of output explained variance of the initial forest, fit with all input variables. Finally, this can be written in the following compact form:

$$\hat{\mathbf{Sh}}_{M,n} = \underset{\beta \in [0,1]^p}{\text{argmin}} \quad \ell_{M,n}(\beta)$$
$$\text{s.t. } ||\beta||_1 = \hat{v}_{M,n}(\{1, \dots, p\}).$$

3 SHAFF Consistency

We prove in this section that **SHAFF** is consistent, in the sense that the estimated value can be arbitrarily close to the ground truth theoretical Shapley effect, provided that the sample size is large enough. To our knowledge, we provide the first Shapley algorithm which requires to fit only a single initial model and is consistent in the general case. We insist that our result is valid even when input variables exhibit strong dependences. The consistency of **SHAFF** holds under the following mild and standard assumption on the data distribution:

(A1). *The response $Y \in \mathbb{R}$ follows*

$$Y = m(\boldsymbol{X}) + \varepsilon,$$

where $\boldsymbol{X} = (X^{(1)}, \dots, X^{(p)}) \in [0,1]^p$ admits a density over $[0,1]^p$ bounded from above and below by strictly positive constants, m is continuous, and the noise ε is sub-Gaussian, independent of $\boldsymbol{X}$, and centered.

To alleviate the mathematical analysis, we slightly modify the standard Breiman random forests: the bootstrap sampling is replaced by a subsampling without replacement of a_n observations, as it is usually done in the theoretical analysis of random forests [Scornet et al., 2015, Mentch and Hooker, 2016]. Additionally, we follow Wager and Athey [2018] with an additional small modification of the forest algorithm, which is sufficient to ensure its consistency. Firstly, a node split is constrained to generate child nodes with at least a small fraction $\gamma > 0$ of the parent node observations. Secondly, the split selection is slightly randomized: at each tree node, the number `mtry` of candidate variables drawn to optimize the split is set to $\texttt{mtry} = 1$ with a small probability $\delta > 0$. Otherwise, with probability $1 - \delta$, the default value of `mtry` is used. It is stressed that these last modifications are mild, since γ and δ can be chosen arbitrarily small.

Finally, we introduce the following two assumptions on the asymptotic regime of the algorithm parameters. Assumption (A2) enforces that the tree partitions are not too complex with respect to the sample size n. On the other hand, Assumption (A3) states that the number of trees and the number of sampled variable subsets U grow with n. This ensures that all possible variable subsets have a positive probability to be drawn, which is required for the convergence of our algorithm based on importance sampling.

(A2). *The asymptotic regime of a_n, the size of the subsampling without replacement, and the number of terminal leaves t_n are such that $a_n \leq n - 2$, $a_n/n < 1 - \kappa$ for a fixed $\kappa > 0$, $\lim\limits_{n \to \infty} a_n = \infty$, $\lim\limits_{n \to \infty} t_n = \infty$, and $\lim\limits_{n \to \infty} 2^{t_n} \frac{(\log(a_n))^9}{a_n} = 0$.*

(A3). *The number of Monte-Carlo sampling K_n and the number of trees M_n grow with n, such that $M_n \longrightarrow \infty$ and $n.M_n/K_n \longrightarrow 0$.*

We also let the theoretical Shapley effect vector be $\mathbf{Sh}^\star = (Sh^\star(X^{(1)}), \dots, Sh^\star(X^{(p)}))$ to formalize our main result.

Theorem 1. *If Assumptions (A1), (A2), and (A3) are satisfied, then **SHAFF** is consistent, that is*

$$\hat{\mathbf{Sh}}_{M_n,n} \xrightarrow{p} \mathbf{Sh}^\star.$$

Sketch of proof of Theorem 1. Firstly, we need three lemmas to prove Theorem 1, gathered in the Supplementary Material. Lemma 1 states that all variable subsets U have a positive probability to be drawn asymptotically, which ensures that the importance sampling approach can converge. Lemma 2 states the consistency of the projected forest estimate, and the proof uses arguments from Györfi et al. [2006] to control both the approximation and estimation errors. Lemma 3 applies the two previous lemmas to state the convergence of the loss function of the weigthed regression problem solved to recover Shapley effect estimates.

Secondly, we apply Theorem 2 from Lundberg and Lee [2017] to show that the minimum of the theoretical loss function are the theoretical Shapley effects. Finally, using Lemma 3 and Theorem 5.7 from Van der Vaart [2000, page 45], we show that the minimum of the empirical loss function converges towards the minimum of the theoretical loss function, which gives the consistency of **SHAFF**. □

4 Experiments

We run two batches of experiments to show the improvements of **SHAFF** over the main competitors Broto et al. [2020], Williamson and Feng [2020], and Covert et al. [2020, SAGE]. Experiment 1 is a simple linear case with a redundant variable, while Experiment 2 is a non-linear example with high order interactions. In both cases, existing Shapley algorithms exhibit a bias which significantly modifies the accurate variable ranking, as opposed to **SHAFF**.

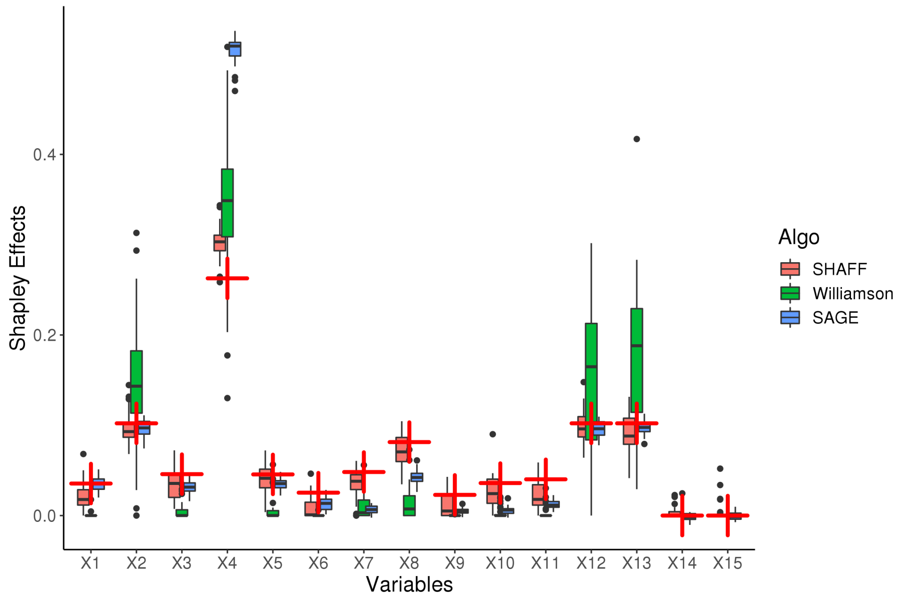

Figure 2: Shapley effects for Experiment 1. Red crosses are the theoretical Shapley effects.

Experiment settings. Our implementation of **SHAFF** is based on `ranger`, a fast random forest software written in `C++` and `R` from Wright and Ziegler [2017]. We implemented Williamson and Feng [2020] from scratch, as it only requires to sample variable subsets U, fit a random forest for each U, and recover Shapley effects by solving the linear regression problem defined in Section 2. Notice that we limit tree depth to 6 when $|U| \leq 2$ to avoid overfitting. We implemented SAGE following Algorithm 1 from Covert et al. [2020], and setting $m = 30$. The original implementation of Broto et al. [2020] in the R package `sensitivity` has an exponential complexity with p. Even for $p = 10$, we could not have the experiments done within 24 hours when parallelized on 16 cores. Therefore, we do not display the results for Broto et al. [2020], which seem to have a high bias on toy examples. In all procedures, the number K of sampled subsets U is set to 500, and we use 500 trees for the forest growing. Each run is repeated 30 times to estimate the standard deviations. For both experiments, we analytically derive the theoretical Shapley effects, and display this ground truth with red crosses in Figures 2 and 3—see the Supplementary Material for the formulas.

Experiment 1. In the first experiment, we consider a linear model and a correlated centered Gaussian input vector of dimension 11. The output Y follows

$$Y = \beta^T \mathbf{X} + \varepsilon,$$

where $\beta \in [0, 1]^{11}$, and the noise ε is centered, independent, and such that $\mathbb{V}[\varepsilon] = 0.05 \times \mathbb{V}[Y]$. Finally, two copies of $X^{(2)}$ are appended to the data as $X^{(12)}$ and $X^{(13)}$, and two dummy Gaussian variables $X^{(14)}$ and $X^{(15)}$ are also added. We draw a sample $\mathscr{D}_n$ of size $n = 3000$.

Figure 2 shows that **SHAFF** is more accurate than its competitors. Covert et al. [2020, SAGE] has a strong bias for several variables, in particular $X^{(4)}$, $X^{(7)}$, $X^{(8)}$, and $X^{(10)}$. The algorithm from Williamson and Feng [2020] has a lower performance since its variance is higher than for the other methods. Notice that Williamson and Feng [2020] recommend to set $K = 2n$ ($= 6000$ here). Since we use $K = 500$ to compare all algorithms, this high variance is quite expected and show the improvement due to the importance sampling of our method. Besides, the computational complexity of Williamson and Feng [2020] is $O(n^2)$ whereas **SHAFF** is quasi-linear. Finally, in this experiment, the random forest has a proportion of explained variance of about 86%, and the noise variance is 5%, which explains the small negative bias of many estimated values.

Experiment 2. In the second experiment, we consider two independent blocks of 5 interacting variables. The input vector is Gaussian, centered, and of dimension 10. All variables have unit

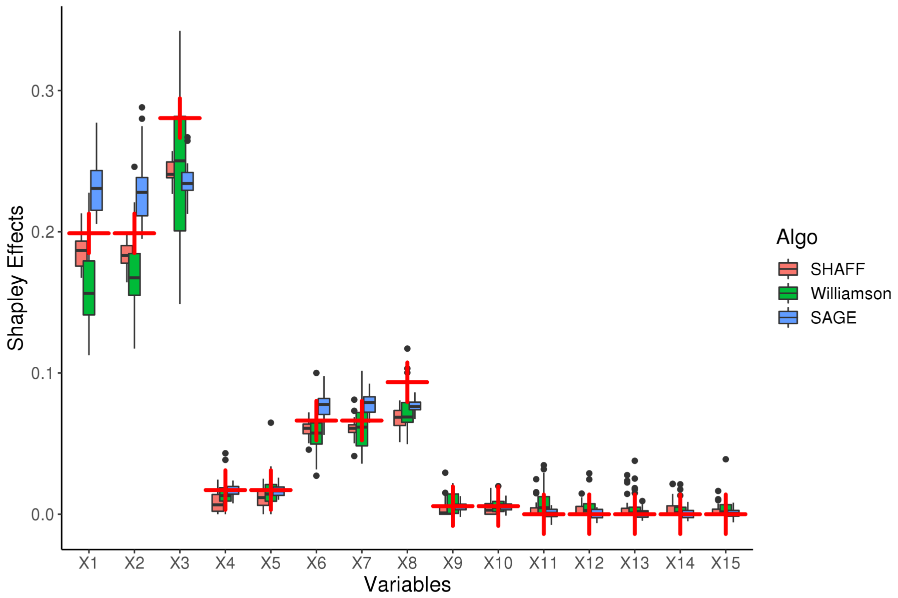

Figure 3: Shapley effects for Experiment 2. Red crosses are the theoretical Shapley effects.

variance, and all covariances are null, except $\text{Cov}(X^{(1)}, X^{(2)}) = \text{Cov}(X^{(6)}, X^{(7)}) = 0.9$, and $\text{Cov}(X^{(4)}, X^{(5)}) = \text{Cov}(X^{(9)}, X^{(10)}) = 0.5$. The output Y follows

$$Y = 3\sqrt{3} \times X^{(1)} X^{(2)} \mathbb{1}_{X^{(3)}>0} + \sqrt{3} \times X^{(4)} X^{(5)} \mathbb{1}_{X^{(3)}<0} \\ + 3 \times X^{(6)} X^{(7)} \mathbb{1}_{X^{(8)}>0} + X^{(9)} X^{(10)} \mathbb{1}_{X^{(8)}<0} + \varepsilon,$$

where the noise ε is centered, independent, and such that $\mathbb{V}[\varepsilon] = 0.05 \times \mathbb{V}[Y]$. We add 5 dummy Gaussian variables $X^{(11)}$, $X^{(12)}$, $X^{(13)}$, $X^{(14)}$, and $X^{(15)}$, and draw a sample $\mathscr{D}_n$ of size $n = 10000$.

In this context of strong interactions and correlations, we observe that all competitors have a strong bias for most variables, as opposed to **SHAFF**, which is also the only algorithm providing the accurate variable ranking given by the theoretical Shapley effects. In particular, **SHAFF** properly identifies variable $X^{(3)}$ as the most important one, whereas SAGE considerably overestimates the Shapley effects of variables $X^{(1)}$ and $X^{(2)}$. **SHAFF** also clearly ranks variable $X^{(8)}$ as more important than $X^{(6)}$ and $X^{(7)}$, as opposed to its competitors. Besides, the proportion of explained variance of the forest is about 84% in this setting, which explains the negative bias observed for several estimates.

5 Conclusion

We introduced **SHAFF**, **SHA**pley e**F**fects via random **F**orests, an algorithm to estimate Shapley effects based on random forests, which has an implementation in `C++` and `R` available online. The challenges in Shapley estimation are the exponential computational complexity, and the estimates of conditional expectations. **SHAFF** addresses the first point by using importance sampling to favor the subsets of influential variables, which often occur along the forest paths. For the second point, **SHAFF** uses the projected forest algorithm, a fast procedure to eliminate variables from the forest prediction mechanism. Thanks to this approach, **SHAFF** only needs to fit a random forests once, as opposed to other methods which retrain many models and are computationally costly. Importantly, we prove that **SHAFF** is consistent. To our knowledge, we propose the first Shapley algorithm which do not retrain several models and is proved to be consistent under mild assumptions. Furthermore, we conducted several experiments to show the practical performance improvements over state-of-the-art Shapley algorithms. Notice that the adaptation of **SHAFF** to SHAP values is straightforward, since the projected random forests provides predictions of the output conditional on any variable subset. Finally, in specific settings, it is obviously possible that other learning algorithms outperform random forests. Then, we can use such efficient model to generate a new large sample of simulated observations, which can then feeds **SHAFF** and improves its accuracy.

References

K. Aas, M. Jullum, and A. Løland. Explaining individual predictions when features are dependent: more accurate approximations to shapley values. *arXiv preprint arXiv:1903.10464*, 2019.

S. Basu, K. Kumbier, J.B. Brown, and B. Yu. Iterative random forests to discover predictive and stable high-order interactions. *Proceedings of the National Academy of Sciences*, 115:1943–1948, 2018.

C. Bénard, G. Biau, S. Da Veiga, and E. Scornet. Sirus: Stable and interpretable rule set for classification. *Electronic Journal of Statistics*, 15:427–505, 2021a.

C. Bénard, S. Da Veiga, and E. Scornet. Mda for random forests: inconsistency, and a practical solution via the sobol-mda. *arXiv preprint arXiv:2102.13347*, 2021b.

B. Broto, F. Bachoc, and M. Depecker. Variance reduction for estimation of shapley effects and adaptation to unknown input distribution. *SIAM/ASA Journal on Uncertainty Quantification*, 8: 693–716, 2020.

I. Covert, S. Lundberg, and S.-I. Lee. Understanding global feature contributions through additive importance measures. *arXiv preprint arXiv:2004.00668*, 2020.

Ian Covert and Su-In Lee. Improving kernelshap: Practical shapley value estimation via linear regression. *arXiv preprint arXiv:2012.01536*, 2020.

C. Frye, C. Rowat, and I. Feige. Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability. In *Advances in Neural Information Processing Systems*, volume 33, pages 1229–1239. Curran Associates, Inc., 2020.

R. Genuer, J.-M. Poggi, and C. Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31:2225–2236, 2010.

L. Györfi, M. Kohler, A. Krzyzak, and H. Walk. *A distribution-free theory of nonparametric regression*. Springer, New York, 2006.

T. Heskes, E. Sijben, I.G. Bucur, and T. Claassen. Causal shapley values: Exploiting causal knowledge to explain individual predictions of complex models. In *Advances in Neural Information Processing Systems*, volume 33, pages 4778–4789. Curran Associates, Inc., 2020.

B. Iooss and C. Prieur. Shapley effects for sensitivity analysis with correlated inputs: Comparisons with sobol'indices, numerical estimation and applications. *arXiv:1707.01334*, 2017.

D. Janzing, L. Minorics, and P. Blöbaum. Feature relevance quantification in explainable ai: A causal problem. In *International Conference on Artificial Intelligence and Statistics*, pages 2907–2916. PMLR, 2020.

G. Louppe. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.

S.M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, New York, 2017.

S.M. Lundberg, G.G. Erion, and S.-I. Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

L. Mentch and G. Hooker. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *Journal of Machine Learning Research*, 17:841–881, 2016.

A.B. Owen. Sobol'indices and shapley value. *SIAM/ASA Journal on Uncertainty Quantification*, 2: 245–251, 2014.

A.B. Owen and C. Prieur. On shapley value for measuring importance of dependent inputs. *SIAM/ASA Journal on Uncertainty Quantification*, 5:986–1002, 2017.

E. Scornet, G. Biau, and J.-P. Vert. Consistency of random forests. *The Annals of Statistics*, 43: 1716–1741, 2015.

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

E. Song, B.L. Nelson, and J. Staum. Shapley effects for global sensitivity analysis: theory and computation. *SIAM/ASA Journal on Uncertainty Quantification*, 4:1060–1083, 2016.

E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41:647–665, 2014.

M. Sundararajan and A. Najmi. The many shapley values for model explanation. In *International Conference on Machine Learning*, pages 9269–9278. PMLR, 2020.

A.W. Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

S. Wager and S. Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113:1228–1242, 2018.

J. Wang, J. Wiens, and S. Lundberg. Shapley flow: A graph-based approach to interpreting model predictions. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 721–729. PMLR, 13–15 Apr 2021.

B. Williamson and J. Feng. Efficient nonparametric statistical inference on population feature importance using shapley values. In *International Conference on Machine Learning*, pages 10282–10291. PMLR, 2020.

M.N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77:1–17, 2017.

A Computational Complexity

We provide the average computational complexity of **SHAFF**, as well as its competitors Broto et al. [2020], Williamson and Feng [2020], and Covert et al. [2020, SAGE]. For these last two algorithms, random forests are used as the required black-box model. Only **SHAFF** is quasi-linear with the sample size n and independent of the dimension p.

A.1 SHAFF

We derive the computational complexity of each step of **SHAFF**. Overall, the computational complexity is $O(MKn\log(n))$.

Importance sampling. In order to compute the variable subset importance, **SHAFF** counts the occurence of variable subsets U in the tree paths of the forest, which has a complexity of $O(Mn)$, since each tree has about $O(n)$ nodes. The sampling of K subsets U has a complexity of $O(K)$.

Projected random forests. An efficient implementation of the PRF algorithm is detailed in Algorithm 1. For the sake of clarity, we provide a version of PRF for a single variable subset U and one query point $\mathbf{X}^{(U)}$. Let us consider a given tree. The new observation $\mathbf{X}^{(U)}$ is dropped down the tree, eventually applying multiple splits at each level, because data points are sent on both sides of splits involving a variable outside of U. At the same time, the PRF computes which training observations fall in the same projected cell as $\mathbf{X}^{(U)}$, and stops going down the tree just before the size of this projected cell becomes lower than the parameter `min_node_size`. Such procedure has a complexity of $O(n)$ since we sequentially apply splits to reduce the number of training observations from about n to `min_node_size` to reach the terminal projected cell. Therefore, the computational complexity to compute the PRF prediction for a given U and $\mathbf{X}^{(U)}$ is $O(Mn)$.

In **SHAFF**, the PRF is run for all subsets $U \in \mathcal{U}_{n,K}$ and the full OOB sample for each tree. In practice, we do not naively run Algorithm 1 for all U and OOB observations, i.e., $O(Kn)$ times, since it would lead to a quadratic complexity with n. Instead, for a given tree, all OOB and training observations are dropped down the tree simultaneously. Even if multiple splits are applied at each tree level, we are still partitioning two samples of size $O(n)$ by sequentially applying splits: splitting one time all cells of a given partition takes $O(n)$ operations, and this has to be repeated $O(\log(n))$ times so that each cell reaches a size of `min_node_size`. Therefore, the global complexity of running PRF for the full OOB samples and the K subsets U is $O(MKn\log(n))$.

Shapley effect estimates. The complexity to solve a least square problem with p columns and K rows is $O(p^3K)$. However in practice, K is always fixed to default value, and when $p > K$, only at most $O(K)$ input variables are selected in the subsets U. For the non-selected inputs, the Shapley effect is null, and they can be removed from the least square problem, leading to a complexity of $O(K^4)$.

A.2 Competitors

Broto et al. [2020] The conditional expectations are estimated for all $U \in \{1,\dots,p\}$, which makes 2^p estimates. Efficient k-nearest neighbor algorithms have a complexity of $O(pn\log(n))$. Overall the complexity is $O(n\log(n)p2^p)$, which is exponential with respect to the dimension p.

Williamson and Feng [2020] Growing K random forests from scratch, one for each subset U, has an averaged complexity of $O(MKpn\log^2(n))$ [Louppe, 2014]. Williamson and Feng [2020] recommend to use $K = O(n)$, which makes a global complexity of $O(Mpn^2\log^2(n))$, and is quadratic with respect to the sample size n and depends on the dimension p.

Covert et al. [2020, SAGE] Running a prediction for random forests takes $O(M\log(n))$ operations. Since SAGE computes np predictions, the global complexity is $O(Mpn\log(n))$ and depends on the dimension p.

Algorithm 1 Projected Random Forest

```
Inputs: A random forest fit with $\mathscr{D}_n$, a variable subset $U \subset \{1, \dots, p\}$, and a query point $\mathbf{X}^{(U)}$.
for all trees in the forest:
    # Step 1: initialize variables
    initialize $nodes\_level$ as a list of nodes containing only the root node;
    initialize $nodes\_child$ as an empty list of child nodes;
    initialize $samples$ as the list of observation indices of the full training data of the tree;
    for all levels in the tree:
        # Step 2: drop $\mathbf{X}^{(U)}$ to the next tree level with the relevant training observations
        for all nodes in $nodes\_level$:
            if the node splits on a variable in $U$:
                compute whether $\mathbf{X}^{(U)}$ falls in the left or right child node;
                append the child node to $nodes\_child$;
                set $samples\_child$ as the observations in $samples$ which satisfy the split
            else:
                append both the left and right children nodes to $nodes\_child$;
                set $samples\_child = samples$;
            if the size of $samples\_child$ is lower then $min\_node\_size$:
                break the loop through the tree levels;
            else:
                set $samples = samples\_child$;
        set $nodes\_level = nodes\_child$;
        # Step 3: compute prediction
        compute the tree prediction as the average of $Y_i$ for all $i$ in $samples$;
average predictions of all trees;
return final prediction;
```

B Proof of Theorem 1

We need the following three lemmas to prove Theorem 1. Lemma 1 gives the convergence of the importance sampling, because all variable subsets U have a positive probability to be drawn asymptotically. Lemma 2 states the consistency of the projected forest estimate, and the proof follows arguments from Scornet et al. [2015]. Lemma 3 uses the two previous lemmas to state the convergence of the loss function of the weighted regression problem solved to recover Shapley effect estimates.

Lemma 1. *If Assumption (A3) is satisfied, we have*

$$\mathbb{P}\big(\hat{p}_{M_n,n}(U) > 0\big) \longrightarrow 1.$$

Lemma 2. *If Assumptions (A1) and (A2) are satisfied, the PRF is consistent, that is, for all $M \in \mathbb{N}^\star$ and $U \subset \{1, \dots, p\}$,*

$$\hat{v}_{M,n}(U) \xrightarrow{p} \mathbb{V}[\mathbb{E}[Y|\boldsymbol{X}^{(U)}]]/\mathbb{V}[Y] \overset{\text{def}}{=} v^\star(U).$$

We let Z be a discrete random variable taking values in the set of all subsets of $\{1, \dots, p\}$, excluding the full and empty sets. The discrete distribution of Z is given by the weights $w(U)$ (the weights are scaled to sum to 1).

Lemma 3. *If Assumptions (A1), (A2), and (A3) are satisfied, we have*

$$\ell_{M,n}(\beta) \xrightarrow{p} \mathbb{E}[(v^\star(Z) - \beta^T I(Z))^2] \overset{\text{def}}{=} \ell^\star(\beta).$$

Proof of Theorem 1. We assume that Assumptions (A1), (A2), and (A3) are satisfied. Since $\ell^\star$ is convex and β belongs to the compact set $[0, 1]^p$, the pointwise convergence of Lemma 3 gives the uniform convergence

$$\sup_{\beta \in [0,1]^p} |\ell_{M,n}(\beta) - \ell^\star(\beta)| \xrightarrow{p} 0.$$

Additionally, since $\ell^\star$ is a quadratic convex function and the constraint domain $[0,1]^p$ is convex, $\ell^\star$ has a unique minimum. According to Theorem 2 from Lundberg and Lee [2017], this unique minimum is $\mathbf{Sh}^\star$. Finally, since the minimum of $\ell^\star$ is unique and $\ell_{M,n}$ uniformly converges to $\ell^\star$, we apply Theorem 5.7 from Van der Vaart [2000, page 45] to conclude that

$$\hat{\mathbf{Sh}}_{M,n} \xrightarrow{p} \mathbf{Sh}^\star.$$

□

We prove Lemmas 1, 2, and 3 involved in the proof of Theorem 1.

Proof of Lemma 1. We assume that Assumption (A3) is satisfied, and denote by $T_{n,\ell}$ the random set of all variable subsets of $\{1,\dots,p\}$ belonging to a path of the ℓ-th tree. To prove the result, we derive an upper bound for $\mathbb{P}(\hat{p}_{M,n}(U)=0)$. First, we write

$$\mathbb{P}(\hat{p}_{M,n}(U)=0|\mathscr{D}_n) = \mathbb{P}\big(\bigcap_{\ell=1}^{M_n} U \notin T_{n,\ell}|\mathscr{D}_n\big),$$

and since the trees are independent conditional on $\mathscr{D}_n$

$$\mathbb{P}(\hat{p}_{M,n}(U)=0|\mathscr{D}_n) = \mathbb{P}(U \notin T_{n,1}|\mathscr{D}_n)^{M_n}.$$

For $n > s\log_2(p)$, where s is the minimum number of observations in a terminal leaf, there is at least one path in each tree that has at least p splits. Additionally, recall that the random forest algorithm is slightly modified such that `mtry` is randomly set to 1 with a small probability δ. Thus, if we define the random event A_n as `mtry` is set to 1 and a new variable of U is selected at each node of a path of length at least $|U|$, then A_n is included in $\{U \in T_{n,1}\}$. This event A_n is of probability lower bounded by $(\delta/p)^p$, and thus for $n > s\log_2(p)$

$$\mathbb{P}(U \in T_{n,1}|\mathscr{D}_n) \geq P(A_n) \geq (\delta/p)^p,$$

and then

$$\mathbb{P}(\hat{p}_{M,n}(U)=0|\mathscr{D}_n) \leq (1-(\delta/p)^p)^{M_n}.$$

Finally, Assumption (A3) gives that the number of trees increases with n, and we obtain

$$\mathbb{P}\big(\hat{p}_{M,n}(U)=0\big) \longrightarrow 0,$$

which is the desired result. □

Proof of Lemma 2. We assume that Assumptions (A1) and (A2) are satisfied and consider $M \in \mathbb{N}^\star$ and $U \subset \{1,\dots,p\}$. Recall that

$$\hat{v}_{M,n}(U) = 1 - \frac{1}{n\hat{\sigma}_Y}\sum_{i=1}^{n}\big(Y_i - m_{M,n}^{(U,OOB)}(\mathbf{X}_i^{(U)},\mathbf{\Theta}_M)\big)^2.$$

The right hand side is expanded as follows:

$$\begin{aligned}\hat{v}_{M,n}(U) &= 1 - \frac{1}{n\hat{\sigma}_Y}\sum_{i=1}^{n}\big(m(\mathbf{X}_i) + \varepsilon_i - m_{M,n}^{(U,OOB)}(\mathbf{X}_i^{(U)},\mathbf{\Theta}_M)\big)^2 \\ &= 1 - \frac{1}{n\hat{\sigma}_Y}\sum_{i=1}^{n}\big(m(\mathbf{X}_i) - \mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}] + \varepsilon_i \\ &\qquad - [m_{M,n}^{(U,OOB)}(\mathbf{X}_i^{(U)},\mathbf{\Theta}_M) - \mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}]]\big)^2.\end{aligned}$$

Therefore,

$$\begin{aligned}\hat{v}_{M,n}(U) = 1 - &\frac{1}{n\hat{\sigma}_Y}\sum_{i=1}^{n}(m(\mathbf{X}_i) - \mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}])^2 \\ &+ \varepsilon_i^2 + 2\varepsilon_i \times (m(\mathbf{X}_i) - \mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}]) \\ &- 2\varepsilon_i \times \big(m_{M,n}^{(U,OOB)}(\mathbf{X}_i^{(U)},\mathbf{\Theta}_M) - \mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}]\big) \\ &- 2(m(\mathbf{X}_i) - \mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}]) \times \big(m_{M,n}^{(U,OOB)}(\mathbf{X}_i^{(U)},\mathbf{\Theta}_M) - \mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}]\big) \\ &+ \big(m_{M,n}^{(U,OOB)}(\mathbf{X}_i^{(U)},\mathbf{\Theta}_M) - \mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}]\big)^2. \qquad \text{(B.1)}\end{aligned}$$

Now, using the law of large numbers, we obtain

$$\frac{1}{n}\sum_{i=1}^{n}(m(\mathbf{X}_i)-\mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}])^2+\varepsilon_i^2 \\ +2\varepsilon_i\times(m(\mathbf{X}_i)-\mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}])\xrightarrow{p}\mathbb{E}[\mathbb{V}[m(\mathbf{X})|\mathbf{X}^{(U)}]]+\mathbb{V}[\varepsilon],$$

and also $\hat{\sigma}_Y\xrightarrow{p}\mathbb{V}[Y]$. Combining these two limits, we have

$$1-\frac{1}{n\hat{\sigma}_Y}\sum_{i=1}^{n}(m(\mathbf{X}_i)-\mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}])^2+\varepsilon_i^2 \\ +2\varepsilon_i\times(m(\mathbf{X}_i)-\mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}])\xrightarrow{p}1-(\mathbb{E}[\mathbb{V}[m(\mathbf{X})|\mathbf{X}^{(U)}]]+\mathbb{V}[\varepsilon])/\mathbb{V}[Y].$$

Rewriting this limit using the law of total variance, we are led to

$$\begin{aligned}1-(\mathbb{E}[\mathbb{V}[m(\mathbf{X})|\mathbf{X}^{(U)}]]+\mathbb{V}[\varepsilon])/\mathbb{V}[Y]\\ &=(\mathbb{V}[Y]-\mathbb{E}[\mathbb{V}[m(\mathbf{X})|\mathbf{X}^{(U)}]]+\mathbb{V}[\varepsilon])/\mathbb{V}[Y]\\ &=(\mathbb{V}[m(\mathbf{X})]+\mathbb{V}[\varepsilon]-\mathbb{E}[\mathbb{V}[m(\mathbf{X})|\mathbf{X}^{(U)}]]-\mathbb{V}[\varepsilon])/\mathbb{V}[Y]\\ &=\mathbb{V}[\mathbb{E}[m(\mathbf{X})|\mathbf{X}^{(U)}]]/\mathbb{V}[Y]\\ &=\mathbb{V}[\mathbb{E}[Y|\mathbf{X}^{(U)}]]/\mathbb{V}[Y]\\ &=v^\star(U).\end{aligned}$$

Overall, the result of the lemma holds if the last three terms of the decomposition (B.1) converge towards 0 in probability. This is clearly true if the OOB PRF estimate is $\mathbb{L}^2$-consistent, that is for $i\in\{1,\dots,n\}$,

$$\mathbb{E}\big[\big(m_{M,n}^{(U,OOB)}(\mathbf{X}_i^{(U)},\mathbf{\Theta}_M)-\mathbb{E}[m(\mathbf{X}_i)|\mathbf{X}_i^{(U)}]\big)^2\big]\longrightarrow 0.$$

According to Lemma 2 from Bénard et al. [2021b], the $\mathbb{L}^2$-convergence of the OOB forest estimate follows from the convergence of the standard forest estimate. Therefore, we only need to show the $\mathbb{L}^2$-convergence of the PRF estimate to get the final result. To do so, we adapt the proof of Theorem 1 from Scornet et al. [2015], which shows the convergence of Breiman's forests for additive models.

The proof only differs for the approximation error. Indeed, we need to show that the variation of the regression function vanishes in a cell of the empirical PRF. Scornet et al. [2015] show that this is always true in the original forest for additive models. Here, the result is valid for all regression functions, using the fact that the random forest is slightly modified: splits cannot be too close from the edges of cells (at least a fraction of γ observations in children nodes), and $mtry$ is set to 1 at each node with a small probability δ. Under these small modifications, Lemma 2 from Meinshausen [2006] gives that the diameter of each cell of the original forest vanishes, i.e,

$$\lim_{n\to\infty}\text{diam}(A_n(\mathbf{X},\Theta))=0,$$

where $A_n(\mathbf{X},\Theta)$ is the cell of the forest where the new query point $\mathbf{X}$ falls, and the diameter of a cell A is the length of the longest line fitting in A, formally

$$\text{diam}(A)=\sup_{\mathbf{x},\mathbf{x}'\in A}||\mathbf{x}-\mathbf{x}'||_2.$$

By definition of the PRF algorithm, the projected cell where $\mathbf{X}^{(U)}$ falls is included in $A_n(\mathbf{X},\Theta)$, and therefore the diameter of the projected cell also vanishes as n increases. Additionally, the regression function m is continuous by Assumption (A1), and consequently the approximation error converges to 0. Finally, the PRF estimate is $\mathbb{L}^2$-consistent, and we deduce the final result,

$$\hat{v}_{M,n}(U)\xrightarrow{p}v^\star(U).$$

□

Proof of Lemma 3. The loss function $\ell_{M,n}$ contains three sources of randomness: the data $\mathscr{D}_n$, the forest randomization Θ, and the importance sampling of the subsets U. The discrete distribution used to sample the subsets U is built using the occurrence frequency in the forest $\hat{p}_{M,n}(U)$, which depends on $\mathscr{D}_n$ and Θ. This subtle relation between the data, the forest, and the importance sampling prevent a straightforward proof for this lemma. We reshape the loss function and use the law of total variance to handle separately the multiple sources of randomness. We assume that Assumptions (A1), (A2), and (A3) are satisfied.

First, we have

$$\begin{aligned}\ell_{M,n}(\beta) &= \frac{1}{K_n}\sum_{U\in\mathcal{U}_{n,K}}\frac{w(U)}{\hat{p}_{M,n}(U)}(\hat{v}_{M,n}(U)-\beta^T I(U))^2\\ &= \sum_{U\subset\{1,\dots,p\}}\frac{w(U)N_n(U)}{K_n\hat{p}_{M,n}(U)}\mathbb{1}_{\hat{p}_{M,n}(U)>0}(\hat{v}_{M,n}(U)-\beta^T I(U))^2,\end{aligned}$$

where $N_n(U)$ is the number of times where U is drawn in $\mathcal{U}_{n,K}$ (with the convention $0/0=0$). Since the sum is finite, it is enough to study the convergence of the terms one by one. Let us consider a given variable subset U. First, we define

$$\Delta_{n,K_n} = \frac{N_n(U)\mathbb{1}_{\hat{p}_{M,n}(U)>0}}{K_n\hat{p}_{M,n}(U)}.$$

Next, we derive the limit of $\mathbb{V}[\Delta_{n,K_n}]$ using the law of total variance. We have

$$\mathbb{V}[\Delta_{n,K_n}] = \mathbb{E}[\mathbb{V}[\Delta_{n,K_n}|\mathscr{D}_n,\Theta]] + \mathbb{V}[\mathbb{E}[\Delta_{n,K_n}|\mathscr{D}_n,\Theta]].$$

On one hand, since K_n is a constant and $\hat{p}_{M,n}(U)$ only depends on $\mathscr{D}_n$ and Θ, we have

$$\mathbb{V}[\Delta_{n,K_n}|\mathscr{D}_n,\Theta] = \mathbb{V}\big[\frac{N_n(U)\mathbb{1}_{\hat{p}_{M,n}(U)>0}}{K_n\hat{p}_{M,n}(U)}|\mathscr{D}_n,\Theta\big] = \Big(\frac{\mathbb{1}_{\hat{p}_{M,n}(U)>0}}{K_n\hat{p}_{M,n}(U)}\Big)^2\mathbb{V}\big[N_n(U)|\mathscr{D}_n,\Theta\big].$$

By definition, $N_n(U)=\sum_{k=1}^{K_n}\mathbb{1}_{U_k=U}$, where $U_1,\dots,U_{K_n}$ are the variable subsets drawn at each iteration of the importance sampling. Since $U_1,\dots,U_{K_n}$ are independent conditional on $\mathscr{D}_n$ and Θ, and U is drawn with probability $\hat{p}_{M,n}(U)$,

$$\mathbb{V}\big[N_n(U)|\mathscr{D}_n,\Theta\big] = K_n\mathbb{V}[\mathbb{1}_{U_1=U}|\mathscr{D}_n,\Theta] = K_n\hat{p}_{M,n}(U)[1-\hat{p}_{M,n}(U)],$$

and finally

$$\mathbb{E}[\mathbb{V}[\Delta_{n,K_n}|\mathscr{D}_n,\Theta]] = \frac{1}{K_n}\mathbb{E}\big[\frac{1-\hat{p}_{M,n}(U)}{\hat{p}_{M,n}(U)}\mathbb{1}_{\hat{p}_{M,n}(U)>0}\big].$$

Therefore,

$$\mathbb{E}[\mathbb{V}[\Delta_{n,K_n}|\mathscr{D}_n,\Theta]] \leq \frac{1}{K_n}\mathbb{E}\big[\frac{\mathbb{1}_{\hat{p}_{M,n}(U)>0}}{\hat{p}_{M,n}(U)}\big].$$

The number of paths in the forest is upper bounded by $n\times M_n$, and therefore if $\hat{p}_{M,n}(U)$ is not null, it is lower bounded by $1/(n.M_n)$. Thus

$$\mathbb{E}[\mathbb{V}[\Delta_{n,K_n}|\mathscr{D}_n,\Theta]] \leq \frac{n.M_n}{K_n},$$

which converges to 0 by Assumption (A3).

On the other hand,

$$\mathbb{E}[\Delta_{n,K_n}|\mathscr{D}_n,\Theta] = \frac{\mathbb{1}_{\hat{p}_{M,n}(U)>0}}{K_n\hat{p}_{M,n}(U)}\mathbb{E}[N_n(U)|\mathscr{D}_n,\Theta] = \mathbb{1}_{\hat{p}_{M,n}(U)>0},$$

and then

$$\begin{aligned}\mathbb{V}[\mathbb{E}[\Delta_{n,K_n}|\mathscr{D}_n,\Theta]] =& \mathbb{P}(\hat{p}_{M,n}(U)>0)[1-\mathbb{P}(\hat{p}_{M,n}(U)>0)]\\ =& \mathbb{P}(\hat{p}_{M,n}(U)>0)\mathbb{P}(\hat{p}_{M,n}(U)=0).\end{aligned}$$

Lemma 1 gives that $\mathbb{P}\big(\hat{p}_{M,n}(U)=0\big)\longrightarrow 0$, which implies the convergence of $\mathbb{V}[\mathbb{E}[\Delta_{n,K_n}|\mathscr{D}_n,\Theta]]$ towards 0.

Overall, the law of total variance gives that

$$\mathbb{V}[\Delta_{n,K_n}] \longrightarrow 0.$$

Since $\mathbb{E}[\Delta_{n,K_n}] = \mathbb{P}\big(\hat{p}_{M,n}(U) > 0\big) \longrightarrow 1$ and $\mathbb{L}^2$-convergence implies convergence in probability, we have

$$\Delta_{n,K_n} \xrightarrow{p} 1.$$

Next, using Lemma 2, we obtain

$$\frac{w(U)N_n(U)}{K_n \hat{p}_{M,n}(U)} \mathbb{1}_{\hat{p}_{M,n}(U)>0} (\hat{v}_{M,n}(U) - \beta^T I(U))^2 \xrightarrow{p} w(U)(v^\star(U) - \beta^T I(U))^2.$$

If Z is a discrete random variable taking values in the set of all subsets of $\{1,\dots,p\}$, excluding the full and empty sets, and distributed with the scaled weights $w(U)$, we finally have

$$\ell_{M,n}(\beta) \xrightarrow{p} \mathbb{E}[(v^\star(Z) - \beta^T I(Z))^2].$$

□

C Formulas of Theoretical Shapley Effects for Experiments

Experiment 1. For a linear model with a Gaussian input vector of dimension p, the theoretical Shapley effects are given by Theorem 2 in [Owen and Prieur, 2017] as

$$Sh^\star(X^{(j)}) = \frac{1}{p} \sum_{U \subset \{1,\dots,p\} \setminus j} \binom{p-1}{|U|}^{-1} \frac{\mathrm{Cov}[X^{(j)}, \mathbf{X}^{(-U)T} \beta^{(-U)} | \mathbf{X}^{(U)}]^2}{\mathbb{V}[X^{(j)} | \mathbf{X}^{(U)}]} \Big(1 - \frac{\sigma_\varepsilon^2}{\mathbb{V}[Y]}\Big),$$

where the conditional covariances and variances can be easily computed using standard formulas for Gaussian vectors, and σ_ε^2 is the noise variance.

In Experiment 1, several copies of a given input $X^{(k)}$ are added to the data. We denote by r the number of redundant variables. We easily deduce the updated value $Sh'^\star(X^{(j)})$ from the original Shapley effects $Sh^\star(X^{(j)})$ for all variables. Then, we have

$$Sh'^\star(X^{(k)}) = \frac{1}{p+r} \sum_{U \subset \{1,\dots,p\} \setminus k} \binom{p+r-1}{|U|}^{-1} \frac{\mathrm{Cov}[X^{(k)}, \mathbf{X}^{(-U)T} \beta^{(-U)} | \mathbf{X}^{(U)}]^2}{\mathbb{V}[X^{(k)} | \mathbf{X}^{(U)}]} \Big(1 - \frac{\sigma_\varepsilon^2}{\mathbb{V}[Y]}\Big).$$

If $j \in \{1,\dots,p\} \setminus k$, we have

$$\begin{aligned} Sh'^\star(X^{(j)}) = \frac{1}{p+r} & \sum_{\substack{U \subset \{1,\dots,p\} \setminus j \\ \text{s.t. } k \notin U}} \binom{p+r-1}{|U|}^{-1} \frac{\mathrm{Cov}[X^{(j)}, \mathbf{X}^{(-U)T} \beta^{(-U)} | \mathbf{X}^{(U)}]^2}{\mathbb{V}[X^{(j)} | \mathbf{X}^{(U)}]} \Big(1 - \frac{\sigma_\varepsilon^2}{\mathbb{V}[Y]}\Big) \\ + \frac{1}{p+r} & \sum_{\substack{U \subset \{1,\dots,p\} \setminus j \\ \text{s.t. } k \in U}} \Big[\sum_{\ell=0}^{r} \binom{r}{\ell} \binom{p+r-1}{|U|+\ell}^{-1} + \sum_{\ell=1}^{r} \binom{r}{\ell} \binom{p+r-1}{|U|+\ell-1}^{-1}\Big] \\ & \qquad \times \frac{\mathrm{Cov}[X^{(j)}, \mathbf{X}^{(-U)T} \beta^{(-U)} | \mathbf{X}^{(U)}]^2}{\mathbb{V}[X^{(j)} | \mathbf{X}^{(U)}]} \Big(1 - \frac{\sigma_\varepsilon^2}{\mathbb{V}[Y]}\Big). \end{aligned}$$

Finally, for $j \in \{p+1,\dots,p+r\}$, clearly

$$Sh'^\star(X^{(j)}) = Sh'^\star(X^{(k)}),$$

and dummy variables have a null Shapley effect.

Experiment 2. Recall that in the second experiment, we consider two independent blocks of 5 interacting variables. The input vector is Gaussian, centered, and of dimension 10. All variables have unit variance, and all covariances are null, except $\mathrm{Cov}(X^{(1)}, X^{(2)}) = \mathrm{Cov}(X^{(6)}, X^{(7)}) = \rho_1$, and $\mathrm{Cov}(X^{(4)}, X^{(5)}) = \mathrm{Cov}(X^{(9)}, X^{(10)}) = \rho_2$. The output Y is defined as a specific case of

$$\begin{aligned} Y = {} & a\sqrt{\alpha} \times X^{(1)}X^{(2)}\mathbb{1}_{X^{(3)}>0} + b\sqrt{\alpha} \times X^{(4)}X^{(5)}\mathbb{1}_{X^{(3)}<0} \\ & + c\sqrt{\beta} \times X^{(6)}X^{(7)}\mathbb{1}_{X^{(8)}>0} + d\sqrt{\beta}X^{(9)}X^{(10)}\mathbb{1}_{X^{(8)}<0} + \varepsilon. \end{aligned}$$

The Shapley effects of the input variables are given by

$$Sh^\star(X^{(1)}) = Sh^\star(X^{(2)}) = \frac{\alpha}{\alpha V_1 + \beta V_2 + \sigma_\varepsilon^2}\Big(\frac{(a\rho_1)^2}{8} + \frac{5}{24}a^2\Big),$$

$$Sh^\star(X^{(4)}) = Sh^\star(X^{(5)}) = \frac{\alpha}{\alpha V_1 + \beta V_2 + \sigma_\varepsilon^2}\Big(\frac{(b\rho_2)^2}{8} + \frac{5}{24}b^2\Big),$$

$$Sh^\star(X^{(3)}) = \frac{\alpha}{\alpha V_1 + \beta V_2 + \sigma_\varepsilon^2}\Big(\frac{(a\rho_1 - b\rho_2)^2}{4} + \frac{(a\rho_1)^2}{4} + \frac{(b\rho_2)^2}{4} + \frac{a^2}{12} + \frac{b^2}{12}\Big),$$

where

$$V_1 = \Big(\frac{(a\rho_1 - b\rho_2)^2}{4} + \frac{(a\rho_1)^2}{2} + \frac{(b\rho_2)^2}{2} + \frac{a^2}{2} + \frac{b^2}{2}\Big),$$

and

$$V_2 = \Big(\frac{(c\rho_1 - d\rho_2)^2}{4} + \frac{(c\rho_1)^2}{2} + \frac{(d\rho_2)^2}{2} + \frac{c^2}{2} + \frac{d^2}{2}\Big).$$

Symmetrically, we have

$$Sh^\star(X^{(6)}) = Sh^\star(X^{(7)}) = \frac{\beta}{\alpha V_1 + \beta V_2 + \sigma_\varepsilon^2}\Big(\frac{(c\rho_1)^2}{8} + \frac{5}{24}c^2\Big),$$

$$Sh^\star(X^{(9)}) = Sh^\star(X^{(10)}) = \frac{\beta}{\alpha V_1 + \beta V_2 + \sigma_\varepsilon^2}\Big(\frac{(d\rho_2)^2}{8} + \frac{5}{24}d^2\Big),$$

$$Sh^\star(X^{(8)}) = \frac{\beta}{\alpha V_1 + \beta V_2 + \sigma_\varepsilon^2}\Big(\frac{(c\rho_1 - d\rho_2)^2}{4} + \frac{(c\rho_1)^2}{4} + \frac{(d\rho_2)^2}{4} + \frac{c^2}{12} + \frac{d^2}{12}\Big).$$

Clearly, $Sh^\star(X^{(11)}) = Sh^\star(X^{(12)}) = Sh^\star(X^{(13)}) = Sh^\star(X^{(14)}) = Sh^\star(X^{(15)}) = 0$.