Effective pruning for top-k feature search on the basis of SHAP values

Lisa Chabrier, Anton Crombach, Sergio Peignier, Christophe Rigotti

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 August 2024, accepted 24 October 2024, date of publication 1 November 2024, date of current version 13 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3489958

RESEARCH ARTICLE

Effective Pruning for Top-k Feature Search on the Basis of SHAP Values

LISA CHABRIER[1,2], (Student Member, IEEE), ANTON CROMBACH[1,2], SERGIO PEIGNIER[3], AND CHRISTOPHE RIGOTTI[1,2]

[1]Inria, Centre de Lyon, 69603 Villeurbanne, France
[2]INSA Lyon, CNRS, Universite Claude Bernard Lyon 1, LIRIS, UMR5205, 69621 Villeurbanne, France
[3]INSA Lyon, INRAE, BF2I, UMR0203, 69621, Villeurbanne, France

Corresponding author: Christophe Rigotti (christophe.rigotti@insa-lyon.fr)

This work was supported in part by the Programmes et Equipements Prioritaires de Recherche (PEPR) Santé Numérique under Project 22-PESN-0002, in part by the Fondation ARC under Grant ARCPJA22020060002212, in part by the Institut National du Cancer under Grant PLBIO22-071, in part by the National Institutes of Health (NIH) under Award R01DC020478, in part by the Bonus Qualité Recherche (BQR) INSA Lyon Neuro-Info 2023, and in part by the Agence Nationale de la Recherche (ANR) under Project C2R-IA ANR-22-CE56-0005.

ABSTRACT With the ever-increasing influence of machine learning models, it has become necessary to explain their predictions. The SHAP framework provides a solution to this problem by assigning a score to each feature of a model such that it reflects the feature contribution to the prediction. Although SHAP is widely used, it is hampered by its computational cost when preserving model-agnosticism. This paper proposes a model-agnostic algorithm, TopShap, to efficiently approximate the SHAP values of the top-k most important features. TopShap uses confidence interval bounds of the approximate SHAP values to determine on the fly which features can no longer be part of the top-k and then removes them from the computation, thus saving computational resources. This cost reduction makes TopShap better suited than competing model-agnostic methods for top-k SHAP value computation. The evaluation of TopShap shows that it performs efficient pruning of the feature search space, in turn leading to a substantial reduction in the execution time when compared to the existing most efficient agnostic approach, Kernel SHAP. The experiments presented in this work cover a wide range of numbers of features and instances, using the following public datasets: Concrete, Wine quality, Appliances energy, PBMC gene expression, Mercedes, CT locations, and a synthetic regression. Various models were used to demonstrate model-agnosticism: *Regression Forest*, *Multi-Layer Perceptron*, *RBF-kernel Support Vector Regression*, and *Stacked Generalization*.

INDEX TERMS Explainable artificial intelligence, local feature attribution, model-agnostic method, Shapley value, top-k selection.

I. INTRODUCTION

Machine learning models have become more and more sophisticated over the last decade, and the need to explain their predictions has been increasing similarly [1]. Indeed, understanding how machine learning models make their predictions can help to decipher and counter bias in training data, and enables domain-experts to judge model output, which in turn builds trust in model decisions. One way to explain the predictions of a model is to provide importance values for the features used in the model input. This can be done at global level, encompassing all predictions, or at the local level, where feature importances are assessed for a single prediction made for a given input instance.

Among local feature importance approaches, so-called SHAP values [2] have been shown to give meaningful insight into machine learning predictions (e.g., [3], [4]). SHAP values are based on a game theory approach developed by Shapley [5], addressing the distribution of gain between players in cooperative multiplayer games. In the context of

The associate editor coordinating the review of this manuscript and approving it for publication was Ugur Guvenc.

explaining predictions made by machine learning models, the goal is to quantify the influence of each feature on the model output. SHAP values are the resulting framework, using the desirable properties of Shapley values [5] and the requirement of additive feature attributions as found in a larger group of explainability methods [6], [7].

One of the main difficulties of using SHAP values is their computational cost, because they require to take into account the contribution of each subset of features, as they need to obtain the expectation of the model output when only such a subset is known. An important research direction is the reduction of these costs, while being independent of the type of models, i.e., being model-agnostic, in order to be of benefit to any current (or emerging) model types. In existing works (e.g., [8], [9]), these gains are made by using sampling strategies to compute approximations of feature importance measures. It should be noted that, for some types of models, other improvements can be obtained by taking advantage of the model structure itself, though this implies losing the property of being model-agnostic [3].

A. PROBLEM STATEMENT

While knowing the importance of all features can deliver meaningful insight, an explanation of model behavior that highlights which features contribute most to a prediction could be sufficient for the user in order to develop an understanding of that prediction. In this case, especially if only a few top-k features are of interest, the problem we face is that the SHAP values of all features need to be computed to know which ones are the most important.

B. RESEARCH OBJECTIVE

Our goal is to reduce the time required in this context, by avoiding computations related to features that are not in the top-k. Current applications based on SHAP values that need to select the most influential features (e.g., [10], [11], [12]), use the following workflow. Firstly, they run an algorithm to approximate SHAP values (e.g., [2], [8], [9], [13]) of each feature, and secondly, they select the features having the highest SHAP values in a post-processing step. The novelty of the method proposed in this paper is that it starts by quickly computing a rough approximation of the SHAP values of all features, and then iteratively improves these approximations while discarding on the fly the features whose SHAP values converge towards too low values. The contribution of this work is an important reduction in execution time compared to the above-mentioned post-processing strategy. When looking for the top-k features and their corresponding SHAP values, this gain is obtained by effective pruning of the search space which avoids the need to compute the precise importance of most features not in the top-k. As the computational cost of SHAP values is exponential in the number of features, such a reduction of the search space is particularly interesting for datasets with a large number of features.

To perform such a pruning, the algorithm TopShap is proposed, based on an iterative sampling strategy that interleaves bound improvement of SHAP value estimates and pruning of the set of features that are candidates to lie in the top-k. Experiments on various datasets and representative types of models show a rapid reduction in the number of candidates, thus avoiding the correspondingly large number of computationally expensive sampling operations. When compared to current existing alternative approaches, this leads to a large reduction in the execution cost. In addition, a use case from biology shows that such a top-k selection provides meaningful insight in the context of gene expression data analysis.

The rest of this paper is organized as follows. The next section recalls the necessary preliminaries. Section III presents related works. The algorithm to compute the top-k features is described in Section IV. Section V reports the experiments, while a concluding summary is given in Section VI. Complementary experiment results are reported in the Supplementary Materials.

II. PRELIMINARIES

In the following, the term *SHAP value* designates the importance of a feature in a prediction as defined by both Strumbelj et al. [7] and the unified framework of SHapley Additive exPlanations [2].

Let f be an already trained regression model that, given an instance x described by a set of features F, can be used to compute $f(x) \in \mathbb{R}$, the prediction associated with x. For each feature $i \in F$, the SHAP value $\phi_i(f, x)$ assesses the contribution of i in the prediction $f(x)$.

Let $f_S(x)$, where $S \subseteq F$, denote the output of f when only the values of the features in S are known for x. For instance, $f_\emptyset(x)$ is the default prediction of f when no feature is known.

The main property of the SHAP values is the local accuracy, which guarantees that their addition to $f_\emptyset(x)$ sums to the prediction for input x:

$$f(x) = f_\emptyset(x) + \sum_{i \in F} \phi_i(f, x). \tag{1}$$

Note that in this additive explanation framework the contribution of a feature can be positive or negative, and that features with high importance are those with large absolute $\phi_i(f, x)$ values. The common interpretation of Eq. 1 is that the sum of the SHAP values over all features corresponds to the difference between $f(x)$, the output of the model for the chosen instance x, and the default output of the model when none of the feature values are known (i.e., $f_\emptyset(x)$). In general the function ϕ_i is not bounded by a known maximum or minimum, and the importance of a feature cannot be assessed by computing its SHAP value alone. To determine if a feature plays a major role in a prediction, it is necessary to compare its SHAP value to those of the other features, to find out if this value lies among the main positive or negative contributions in the sum in Eq. 1.

Following the formal framework of [5], it has been shown that, under reasonable fairness axioms (e.g., symmetry, consistency), there exists only one function ϕ_i and that it can be defined as [2], [7]:

$$\phi_i(f, x) = \sum_{S \subseteq F \setminus \{i\}} w(S)\big(f_{S\cup\{i\}}(x) - f_S(x)\big), \tag{2}$$

where w is a weight function depending on the size of S and F:

$$w(S) = \frac{|S|!\,(|F| - |S| - 1)!}{|F|!}. \tag{3}$$

This follows the cooperative game theory results of [5], which proposes a solution to the problem of allocating benefits to players under fairness axioms. The analogy is that the set of features F is replaced by a set of players, and the function $f_S(x)$ is replaced by a function $\nu(S)$ for $S \subseteq F$ that gives the value of the coalition of players S in the game. In this original context, the allocation of a player $i \in F$ was defined as:

$$\mathcal{A}_i(\nu) = \sum_{S \subseteq F \setminus \{i\}} w(S)\big(\nu(S \cup \{i\}) - \nu(S)\big). \tag{4}$$

Having recalled the definitions necessary to introduce the TopShap algorithm, it is noted that, for the sake of clarity, the above preliminaries have been restricted to regression tasks. However, SHAP values can be easily reformulated for classification tasks. For instance, in the case of a binary classification with classes $\mathcal{C}_0$ and $\mathcal{C}_1$, if $f(x)$ outputs the probability to belong to class $\mathcal{C}_1$, with $1 - f(x)$ being the probability that x lies in $\mathcal{C}_0$, then the framework can be applied directly.

III. RELATED WORK

Recent studies have reported the successful application of SHAP values in various domains to identify the most important features used by a model. In the prediction of diseases, SHAP values help detect unexpected influences of meteorological conditions on cardiovascular diseases [14]. They also improve user transparency in the interpretation of pneumonia predictions from X-ray images [11]. In the cybersecurity domain, computing SHAP values to compare the classification criteria of two models improves botnet traffic detection [15]. To enhance the design of urban networks, SHAP values are used to decipher the motivations for choosing transportation modes [10]. In a more industrial context, they are also used to contribute to the fine-grained analysis of the important parameters driving steel quality in steelworks [16] and impacting the efficiency of power plants [17], thereby leading to improvements in the design process.

SHAP values remain an active research topic. The approach in [18] proposes taking advantage of SHAP values to find positive and negative feature impacts on a prediction, and then to use this contrast to improve model performance through fine-grained parameter tuning. The Powershap method [12] investigates the use of SHAP values as a feature selection measure. In this approach, features are selected if their SHAP values are statistically significant compared to random features. The measure of local feature importance based on SHAP values is also extended in [19] to handle predictions made by a series of models. This is performed by propagating SHAP values through the models. An example is loan attribution, where a first model predicts a probability of fraud in the declaration and then a second model predicts the loan risk. The handling of dependencies among features is investigated in [20]. In this work, the authors propose to include a causal graph between features in the input and to generalize the SHAP value framework to consider this causal relationship when assessing feature importance. Another aspect that is not related to the model is the presence of noise in the data, which can lead to SHAP values that over- or underestimate feature importance [21]. The authors solve this issue by proposing a framework called WeightedSHAP, which extends SHAP values to identify the most influential features in this context. The SHAP framework can also be adapted to unsupervised learning. This is proposed in [22] where SHAP values are used to assess the feature importance of the distance between an instance and its cluster centroid.

In the aforementioned approaches, SHAP values are computed for all features, even when only the features with the largest SHAP values are of interest in an application. This means that all SHAP values are computed and then, in a post-processing step, most of them are discarded. The TopShap algorithm presented in this paper aims to reduce the computation time by stopping early the computation of any SHAP values that are not in the top-k.

Reducing this cost is important, because for each feature i computing $\phi_i(f, x)$ or $\mathcal{A}_i(\nu)$, according to (2) and (4), suggests a summation over the subsets of F. The cost can be reduced for particular classes of cooperative games (e.g., [23]), however, in general, computing the Shapley values as defined by (4) requires an exponential number of operations [24]. There exist two other equivalent closed-form expressions of the Shapley values. The first relies on the fact that these values can be shown to be the optimal solutions of a Weighted Least Squares (WLS) problem [25]; however, the exact solutions still require the evaluation of an exponential number of terms. The second one is based on rewriting (4) as an average over permutations of F instead of a weighted sum over subsets of F. This gave rise to the method proposed by [26] to estimate the Shapley values for real world problems (when $|F|$ is large) using a uniform random sampling of the permutations.

In the context of local feature attribution of a prediction, the SHAP values suffer from the same exponential computational cost as Shapley values. In fact, it is even worse, because $\phi_i(f, x)$ in (2) requires also determination of $f_S(x)$, that is the expected output of the model when only a subset S of the features of x are known. Reducing the cost of computing SHAP values has received ample attention in two main

research directions. The first one (e.g., [7], Kernel SHAP [2], [9]) aims to be broadly applicable and therefore takes a model-agnostic perspective, where SHAP computations are kept independent from the model type. The second one, on the contrary, proposes non-agnostic approaches (e.g., Tree SHAP [3], Linear SHAP [2]), that are dedicated to one kind of model and speedup the computation by taking advantage of the model structure itself.

Here, we pursue the general model-agnostic option and propose a top-k approach that does not rely on model-specific considerations. In the community, the search for methods to approximate SHAP values while preserving the model-agnostic property has led to two main fruitful families of methods, similar to those used for Shapley values: sampling the permutations of F or solving an equivalent WLS problem.

In the first family, among the approaches based on sampling permutations (e.g., [9]) the one in [13] is of particular interest because it incorporates the approximation of $f_S(x)$ in the sampling process. Indeed, computing $f_S(x)$ is a problem in itself, and most model types cannot output it directly. The algorithm presented in [13] solved this problem by using a joint sampling of permutations and values in the domain of x to estimate $f_S(x)$.

In the second family, the WLS approach led to the method of reference called Kernel SHAP [2], shown to be more efficient than the sampling of [13] mentioned above. As detailed in [27], the approximation made by Kernel SHAP is also based on sampling, but it is a sampling over the subsets of F, not over the permutations of F. Sampling was used by this algorithm to reduce the size of the set of equations involved in solving the WLS problem to obtain SHAP values.

Beyond the reduction of computational cost, other studies tried to put a strict limit on global resource consumption. Such an algorithm was for instance proposed by [13], aiming to obtain the best approximations for all $\phi_i(f, x)$ using no more than a given total number of observations drawn to build all samples, i.e., satisfying a sampling budget constraint. For cooperative games, such a constraint has also been used by [28] to limit the resources allocated to the computation of the $\mathcal{A}_i(\nu)$ of the different players. In addition, the authors proposed explicitly to output only the top-k players, as follows. Given a maximum number of drawing operations allowed by the total budget, the algorithm refines the approximations of the $\mathcal{A}_i(\nu)$ as much as possible. Then, in a post-processing step, the top-k are selected. Computing the top-k $\mathcal{A}_i(\nu)$ has been used in [29] to identify influential nodes in social networks, but the top-k selection was again simply made as a post-processing step.

Our contribution is to show that, when computing the top-k features and their corresponding $\phi_i(f, x)$, an effective pruning can be interleaved within the permutation sampling process. The experiments reported in Section V show an important reduction in the search space during sampling of the permutations, in comparison to a selection of the top-k based on post-processing. In addition, this pruning is compared to Kernel SHAP, the algorithm of reference for model-agnostic SHAP value computation, that has been reported to have better performance than classical sampling of permutations. Our experiments show that this is no longer the case when pruning steps are interleaved within the permutation sampling, and that the gain is strong when k is small in comparison with the total number of features.

IV. PROPOSED METHOD

In this section, we present TopShap, a model-agnostic algorithm to compute the top-k features according to their SHAP values. Since the contribution of a feature to a prediction, as captured by its SHAP value, can be positive or negative, features need to be considered by their absolute SHAP values. In addition, defining the top-k requires ties to be handled, especially as the fairness axioms of the original Shapley values impose ties for equivalent contributors [5]. Thus, we define a top-k feature as having no more than $k-1$ other features with a strictly greater absolute SHAP value.

Definition 1 (Top-k Features): A feature i is a top-k feature of a set of features F, for a model f and an instance x if and only if

$$\left|\{j \in F \mid |\phi_j(f,x)| > |\phi_i(f,x)|\}\right| < k. \tag{5}$$

To reduce computational cost, searching for the top-k is a viable strategy if it can be done without a naive determination of all SHAP values. This means that, during computation, a bounding and pruning mechanism is needed to avoid estimating SHAP values for features that can no longer be in the top-k.

Deriving bounds for player allocations and SHAP values using sampling has been studied by [7] and [26]. Noting that the allocation $\mathcal{A}_i(\nu)$ in (4) can be equivalently obtained as an average over the permutations of the elements of F (detailed later), the authors of [26] showed that $\mathcal{A}_i(\nu)$ can be estimated as the mean of a sequence of $i.i.d.$ random variables, providing an unbiased and consistent estimator. Applying the central limit theorem, they derived a confidence interval for $\mathcal{A}_i(\nu)$. This approach was extended by [7] to estimate $\phi_i(f, x)$ for a model f, while at the same time sampling over the domain of x to estimate the terms $f_S(x)$ and $f_{S\cup\{i\}}$ in (2).

The aforementioned bounding scheme is adopted in TopShap to reduce incrementally the number of candidate features on the basis of improved confidence interval bounds through additional sampling (i.e., increasing the length of the sequence of random variables used). Starting from an initial set of candidate features CF that is equal to the whole feature set, the core algorithm repeats the following three steps: (1) perform sampling for features in CF; (2) compute new bounds of their SHAP values; (3) prune useless candidates from CF.

At first glance, the stopping criterion could be the presence of at most k different $\phi_i(f, x)$ values among the features in CF, where CF has possibly a size greater than k because of

ties. Unfortunately, in the presence of ties, such a condition is unlikely to be reached in a reasonable number of iterations. Indeed, the computed $\phi_i(f, x)$ values are approximations, and two features with the same true SHAP value can still have different intermediate estimates $\phi_i(f, x)$ until their numerical precision reaches machine precision.

To avoid this problem, the stopping criterion in TopShap is based on the stability of the confidence interval bounds. This means that the algorithm terminates when all remaining candidates have only minor bound variations in the recent past iterations.

The corresponding algorithm, TopShap, is presented in Algorithm 1 and detailed hereafter.

A. MAIN LOOP

The following symbols are used as global constants. N is the total number of features. F is the set of all feature identifiers, each feature being represented by an integer identifier in $\{1, \ldots, N\}$. D is the set of instances used to train the model. The symbols f and x denote the model and instance for which the SHAP values are computed. The explicit parameters are: k the requested number of top features; *warmUp* the size of the sample used to approximate each SHAP value in the initialization stage; and the confidence level γ used to determine the confidence intervals.

Algorithm 1 starts by initializing V, *up* and *low*. V is an array where element $V[i]$ contains the list of values to be used to estimate $\phi_i(f, x)$. The variables *up* and *low* have the same structure as V, with $up[i]$ (resp. $low[i]$) containing the list of upper bounds (resp. lower bounds) of the confidence interval for $\phi_i(f, x)$. These bounds are computed and stored at each iteration and will be used to assess stability.

The initial set of candidate features CF is the whole set of features F (line 3), and a stage of *warmUp* iterations is performed to compute initial estimates. A first pruning of the candidates is then applied (line 7). Next, the algorithm iteratively computes new estimations for all (remaining) candidates and attempts to prune them further, until the confidence interval bounds of all remaining candidates are stable. Algorithm 1 returns the estimates of $\phi_i(f, x)$ for the features i in CF, together with their corresponding confidence intervals.

B. ESTIMATION AND BOUNDING

We adopt the estimation framework of [13], defined as follows. The population P of the sampling process is the set of all pairs $(\mathcal{O}, w)$, where $\mathcal{O}$ is a permutation of the set of features F, and w is an instance. Let $Pre^i(\mathcal{O})$ be the set of features preceding feature i in permutation $\mathcal{O}$. For a pair $(\mathcal{O}, w)$, and a feature i, we consider the characteristic $\delta_i(\mathcal{O}, w)$ that represents the marginal contribution of feature i to the value of $f(x)$ when i completes the features in $Pre^i(\mathcal{O})$, and the values of the other features are set to those of the random instance w.

This characteristic $\delta_i(\mathcal{O}, w)$ is computed as follows. Let z and z' be two instances with feature values $z_1, \ldots, z_N$ and $z'_1, \ldots, z'_N$ set by merging x and w in the following way: z_j is set to x_j if $j \in Pre^i(\mathcal{O})$, and to w_j otherwise; z'_j is set to x_j if $j = i$, and to z_j otherwise. Then, $\delta_i(\mathcal{O}, w)$ is given by $\delta_i(\mathcal{O}, w) = f(z') - f(z)$.

Let M be a sample of size m of pairs $(\mathcal{O}, w)$ of the population P. Then, $\hat{\phi}_i(f, x)$, the estimate of $\phi_i(f, x)$, is defined as the mean of $\delta_i(\mathcal{O}, w)$ over M. This estimator was shown to be unbiased and consistent [13]. Furthermore, applying the central limit theorem, it was derived that $\hat{\phi}_i(f, x) - \phi_i(f, x)$ is approximately normally distributed with mean 0 and variance $\frac{\sigma^2}{m}$, where σ^2 is the variance of $\delta_i(\mathcal{O}, w)$ [13]. This allows a straightforward computation of bounds of a confidence interval for a confidence level γ. For a set $V[i]$ of the values of $\delta_i(\mathcal{O}, w)$ over M, we note BOUND$(V[i], \gamma)$ the function that returns these bounds.

In TopShap, the estimation is performed iteratively calling ESTIMATE (Algorithm 2), where, for each feature i remaining in CF, a new value $\delta_i(\mathcal{O}, w)$ is computed and appended to current $V[i]$ (line 5). The upper and lower bounds of the confidence interval are computed by BOUND from $V[i]$ using the unbiased sample variance as estimator for the population variance. TopShap computes these bounds for all the features in CF. So, to ensure an overall confidence level γ, we apply the extension of the Bonferroni correction to confidence intervals as proposed by [30]. This correction is obtained by using for each interval the confidence level $\gamma' = 1 - \frac{1-\gamma}{|CF|}$. This is done when calling BOUND (line 7).

We note that this sampling process assumes feature independence (as most other frameworks, e.g., [2]); in case of dependencies, various extensions can handle them (see [8]).

C. PRUNING AND STABILITY

Pruning is performed in PRUNE (Algorithm 3) when there are more than k candidate features. The top-k features are defined using absolute SHAP values, whereas the confidence interval bounds in *up* and *low* are those corresponding to the SHAP values themselves, which can be positive or negative. Thus, first, bounds are transformed into new bounds *absUp* and *absLow* corresponding to absolute SHAP values. To this end, the most recent bounds are retrieved by $last(up[i])$ and $last(low[i])$. Note that for a feature i, $up[i]$ and $low[i]$ contain the lists of all upper and lower bounds computed for the SHAP value of feature i. The last elements of these lists are the most recent bounds. Next, for a given feature, *absUp* is simply the maximum of the absolute values of the two bounds (line 8). For *absMin*, if the signs of the two bounds are the same, then *absMin* is the minimum of the absolute values of the bounds (line 11). If they have different signs, then it is zero. The pruning itself is performed as follows. First, the k^{th} highest element among the new lower bounds *absLow*, termed θ, is selected. Then, only the features that have an absolute upper bound *absUp* greater than or equal to θ are kept as candidates (line 18), since the others can no longer be in the set of top-k features. Note that *absUp* is stored as an array to allow constant time direct access. Algorithm 3

finishes by returning the new set of candidate features, and its correctness is shown below.

Theorem 1: The selection made by Algorithm 3 is correct.

Proof: The input of Algorithm 3 contains the parameter k, the set of candidate features CF, and two arrays up and low. The indices of these two arrays are the feature numbers, with $last(up[i])$ and $last(low[i])$ being the current upper and lower bounds of the SHAP value of feature i. The algorithm uses these bounds to prune the features that can no longer be in the top-k highest absolute SHAP values. The output of the algorithm is the new set of candidate features $newCF$. The correctness of Algorithm 3 is shown by proving that a feature is discarded and not output in $newCF$ if and only if it cannot be in the top-k according to the current bounding intervals given by up and low.

Case $|CF| \leq k$: This implies that no more than k candidate features remain. In this case, all candidates are in the top-k. The algorithm handles this in line 3 by placing all these features in the resulting set $newCF$. Therefore, the algorithm is correct in this case.

Case $|CF| > k$: There are more than k candidate features, and thus some may be pruned depending on the bounds of their SHAP values. Because the top-k are selected based on absolute SHAP values (Definition 1), the algorithm computes for each feature the upper (resp. lower) bounds of these absolute values. This is performed from lines 5 to 15 as detailed in the algorithm description (Section IV-C). This part of the algorithm builds two data structures: $absUp$ and $absLow$. The former is an array, where $absUp[i]$ is the upper bound of the absolute SHAP value of feature i. The latter, $absLow$, is a list containing the lower bound of the absolute SHAP value of each feature in CF.

Now consider the algorithm's lines 16 to 18. First, we show that a feature belonging to the top-k is not suppressed and is output in $newCF$. Let i be a feature in CF that is in the top-k. Because i is in the top-k, there cannot be k features with strictly greater absolute SHAP values than i. Thus, the upper bound for feature i cannot be strictly less than the lower bounds of k features. Let θ be the k^{th} largest lower bound, as computed in lines 16-17. Then, the upper bound for feature i, $absUp[i]$, is not strictly less than θ. Therefore, $absUp[i] \geq \theta$ and i is selected in line 18 to be in the return set of features $newCF$.

The complementary property that needs to be shown to ensure correctness is that if a feature cannot be in the top-k due to its current bounds, then it is not placed in $newCF$. Let i be a feature in CF that can no longer be in the top-k because there are at least k other features having a lower bound strictly greater than the upper bound of i. Then, the k^{th} largest lower bound is strictly greater than $absUp[i]$ and thus $absUp[i] < \theta$. Therefore, i is not selected in line 18 and is not returned in $newCF$.

Hence, a feature is not output by Algorithm 3 if and only if it cannot be in the top-k according to the current bounds. □

In Algorithm 1, pruning is interleaved with the computation of new estimations until reaching stable bounds. This stability is checked by ALL-CF-STABLE which returns *true* if and only if, for all remaining candidate features, the variation in size of the confidence interval was strictly less than 0.1% between any two consecutive estimations over the previous 100 iterations.

Algorithm 1 TopShap

1: **Input** k, γ, $warmUp$
2: Initialize V, up and low as arrays of size N, each containing N empty lists.
3: $CF \leftarrow F$
4: **for** initialEstim $\leftarrow 1$ to $warmUp$ **do**
5: $V, up, low \leftarrow$ ESTIMATE(CF, V, up, low, γ)
6: **end for**
7: $CF \leftarrow$ PRUNE(CF, up, low, k)
8: **while** *not*(ALL-CF-STABLE(CF, up, low)) **do**
9: $V, up, low \leftarrow$ ESTIMATE(CF, V, up, low, γ)
10: $CF \leftarrow$ PRUNE(CF, up, low, k)
11: **end while**
12: **Output** Set CF, estimates $mean(V[i])$ for the features i in CF, and corresponding confidence intervals $[last(low[i]), last(up[i])]$.

Algorithm 2 ESTIMATE

1: **Input** CF, V, up, low, γ
2: $\mathcal{O} \leftarrow$ random permutation of F
3: $w \leftarrow$ random instance drawn from D
4: **for** i in CF **do**
5: append $\delta_i(\mathcal{O}, w)$ to $V[i]$
6: $\gamma' \leftarrow 1 - \frac{1-\gamma}{|CF|}$
7: $u, \ell \leftarrow$ BOUND($V[i], \gamma'$)
8: append u to $up[i]$
9: append ℓ to $low[i]$
10: **end for**
11: **Output** V, up, low

D. TIME COMPLEXITY

An iteration (Algorithm 1, line 8) starts by a call to ALL-CF-STABLE performed in $O(|CF|)$. Then, in ESTIMATE, a permutation of F is drawn in $O(|F|)$, followed by the computation of $\delta_i(\mathcal{O}, w)$ needing to merge x and w which can be done once before the *for* loop (line 4) in $O(|F|)$. It also requires the model f to be evaluated on two instances, with a cost $modelEvalCost$ that depends on the type of model. BOUND can be evaluated in $O(1)$ using an incremental update of the mean and of the variance [31]. Thus, the cost of ESTIMATE is in $O(|F| + |CF| \times modelEvalCost)$.

In PRUNE, setting/inserting elements in $absUp$, $absLow$ and $newCF$ (loop line 7 and line 18) is in $O(|CF|)$. Finding the k^{th} highest element (lines 16-17) of $absLow$ (of size $|CF|$) can be done in $O(|CF| \log k)$ using a min-heap.

Thus, the overall cost of an iteration in TopShap is in $O(|F| + |CF|(modelEvalCost + \log k))$. This is to be

Algorithm 3 PRUNE

Input CF, up, low, k
if $|CF| \leq k$ **then**
  $newCF \leftarrow CF$
else
  Initialize $absUp$ as an array of size N.
  $absLow \leftarrow$ empty list
  for i in CF **do**
    $absUp[i] \leftarrow max(|last(up[i])|, |last(low[i])|)$
    if $sign(last(up[i])) = sign(last(low[i]))$ **then**
      append $min(|last(up[i])|, |last(low[i])|)$ to $absLow$
    else
      append 0 to $absLow$
    end if
  end for
  sort $absLow$ in decreasing order
  $\theta \leftarrow absLow[k]$
  $newCF \leftarrow \{i \in CF \mid absUp[i] \geq \theta\}$
end if
Output $newCF$

compared to the complexity of a top-k search based on sampling but without pruning, where the top-k selection would be performed in a post-processing step. In the latter case, the time complexity of an iteration would be $O(|F| \times modelEvalCost)$, requiring more model evaluations.

V. EXPERIMENTS AND DISCUSSION

In this section, experiments are presented, reporting the performance of TopShap when computing the top-k features according to their SHAP values. The goal of the experiments is to assess pruning when compared to classical sampling and to Kernel SHAP. A Python implementation of TopShap, as well as code to reproduce the experiments, are publicly available as a Git repository at https://gitlab.inria.fr/topshap/topshap_and_experiments.

A. DATASETS

The pruning capabilities of TopShap were assessed on six real-world datasets and a synthetic one. These datasets exhibited different numbers of instances and features, among which only the numerical ones were selected. The final corresponding sizes are reported in Table 1. Most of these datasets, namely *Concrete* [32], *Wine Quality* [33], *Appliances Energy* [34] and *CT location* [35], came from the UCI Machine Learning Repository [36]. The *Mercedes* dataset was provided by Mercedes-Benz Greener Manufacturing and was accessible via Kaggle [37]. We used the *make_regression* function from scikit-learn [38] to generate the *Synthetic* dataset. This dataset contained uncorrelated random features following a normal distribution with zero mean and unitary standard deviation. The regression target variable was a random linear combination of all input features. Finally, we relied on the 10x Genomics Peripheral Blood Mononuclear Cells (*PBMC*) dataset [39] to explore the potential of the algorithm in a bioinformatics-oriented use case.

TABLE 1. Number of numerical features and instances of the datasets.

Dataset	# Features	# Instances
Concrete	8	1031
Wine Quality	11	1600
Appliances Energy	25	19735
PBMC	127	2638
Mercedes	375	4209
CT location	384	53500
Synthetic	400	5000

B. MODEL TYPES

Since TopShap is a model-agnostic approach, it was tested with different machine-learning approaches for regression. The experiments presented in this paper include four models: *Regression Forest* (*RF*), *Multi-Layer Perceptron* (*MLP*), *RBF-kernel Support Vector Regression* (*SVR*) and *Stacked Generalization* (*STK*). For all models, scikit-learn (v1.2.2) [38] was used.

TopShap, classical sampling without pruning, and Kernel SHAP were applied to the same trained models. This evaluation did not require the best possible models to be learned, thus the following simple, standard hyperparameter tuning was performed. For the first three models (*RF*, *MLP*, *SVR*), the hyperparameters were optimized by a grid search over the usual hyperparameter ranges for these models (details can be found in the Git repository). *STK* consisted of a stacking of the three other optimized models, combined using a gradient boosting regressor with hyperparameters set to their default values.

C. EXPERIMENTAL SETUP

For each dataset, models were trained on a randomly chosen 70% of the instances, and the remaining 30% of the dataset were used as a test set. Then, TopShap was executed on 100 randomly chosen test instances, to search their top-k features and estimate their corresponding SHAP values. The experiments presented in the main text were run with TopShap parameters set to $k = 5$, confidence $\gamma = 0.95$ and $warmUp = 100$. For datasets with enough features for a meaningful selection, experiments were run also with $k = 15$. To demonstrate robustness of our results, additional experiments where $k = 10$ and $\gamma = 0.99$ are provided in Supplementary Materials.

D. COMPARISON WITH CLASSICAL SAMPLING

The pruning proposed in TopShap is interleaved with the permutation sampling process. However, a classical sampling of permutations could be used to compute all SHAP values. Then, when stability is reached, the top-k could be selected in a post-processing step. The interest of using TopShap,

in comparison to such a post-processing strategy, is studied in this section.

In order to illustrate the global behavior of TopShap, Fig. 1a shows the evolution of the SHAP values for a *Regression Forest* model and a single instance picked from the *Wine Quality* dataset. For each of the 11 features, the absolute SHAP values, surrounded by their confidence intervals, are given along the iterations. In this example, the width of the confidence intervals shrank quickly, which enabled 6 features to be pruned from the candidates, and only 5 features were left before reaching the first 1000 iterations. The stability criterion was reached at iteration 4234 (Fig. 1a, vertical line), where TopShap stopped.

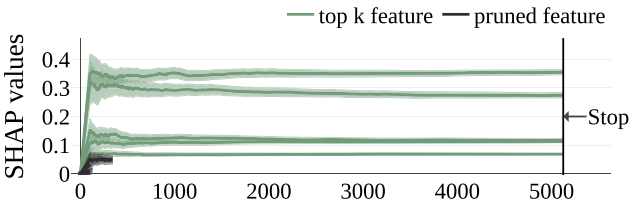
Fig. 1b and 1c illustrate the pruning performance of TopShap, for the same regression model, on 100 randomly selected test instances from the *Wine Quality* dataset. The median and average number of candidate features along the iterations are depicted in Fig. 1b. The green area demarcates the distribution of the number of candidates between the maximal and minimal values, while the left (resp. right) vertical line denotes the iteration when the first (resp. last) instance reached the stability criterion. This implies that TopShap stopped between these two lines for the 100 instances.

The behavior for one instance, as shown in Fig. 1a, is also observed over the 100 instances. Both the average and median number of candidate features decreased (from 11 to 6 features) quickly during the first 2,000 iterations (Fig. 1b). The pruning was effective, and the mean and median number of candidates tended to plateau in the subsequent iterations. The number of iterations necessary to reach stability, i.e., the iteration at which TopShap stopped, was different for each instance. The histogram of the number of iterations needed for each of the 100 instances is reported in Fig. 1c. In this histogram, each bin is split according to the number of candidates remaining when TopShap stopped. If more than 5 candidates were still present, they corresponded to ties among the top-k. These ties could be features with the same SHAP values, or with overlapping confidence intervals due to the approximation framework. The precise count in this example is 59 instances that reached stability with only 5 candidates remaining, 36 instances with 6 or 7 candidates, and 5 instances with 8 candidates.

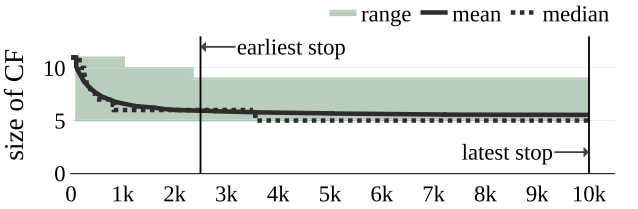
We observed a correspondingly robust behavior of TopShap for all combinations of models and datasets. A subset of combinations is presented for $k = 5$ in Fig. 2, and for datasets with more than 25 features, for $k = 15$ in Fig. 3 (see Supplementary Materials Fig. S1, S2, and S4 for the other combinations). In all cases, TopShap exhibited a pattern of feature pruning similar to what was reported above (Fig. 1). Also, the distributions of the number of iterations needed to reach stability were consistent.

Increasing k from 5 to 15 led to an increase in the number of iterations needed to reach the stopping criterion. This can be observed when comparing Fig. 3 to the bottom panels of Fig. 2. This behavior was anticipated, since adding

(a) Pruning of features and reduction of confidence intervals for the absolute SHAP values of one instance.



(b) Reduction in the number of candidates along the iterations.



(c) Histogram of iterations for which stability was reached and TopShap stopped.

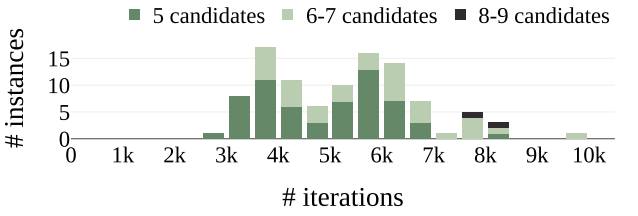


FIGURE 1. **TopShap convergence, pruning, and stability.**

more features is likely to require more iterations to reach stability. In agreement with this explanation, an intermediate increase in the number of iterations was observed for the intermediate setting of $k = 10$ (see Supplementary Materials Fig. S3).

Performing the same experiments with a higher confidence level, $\gamma = 0.99$, resulted in the presence of more ties in the top-k. This occurs because a higher confidence level implies wider confidence intervals and hence more opportunities for features to overlap. Having more ties has a similar effect to choosing a larger k, and accordingly a slight increase was observed in the number of iterations needed to reach stability (Supplementary Materials Fig. S5 and S6).

To summarize, across a variety of datasets and for a range of parameter settings, our results demonstrated the effectiveness of TopShap in pruning candidate features to avoid 'pointless' computational effort. The gain is obvious if our pruning strategy is compared to a post-processing strategy to discover the top-k features, that is to say to a permutation sampling without pruning. As was reported in Section IV-D, the computational cost of an iteration of the sampling process is driven by $|CF| \times modelEvalCost$, where CF is the set of features that are still candidates to be in the top-k. Obviously, the post-processing strategy would require at least as many iterations to reach stability as the top-k search with pruning. To be precise, without pruning, the size of CF would remain

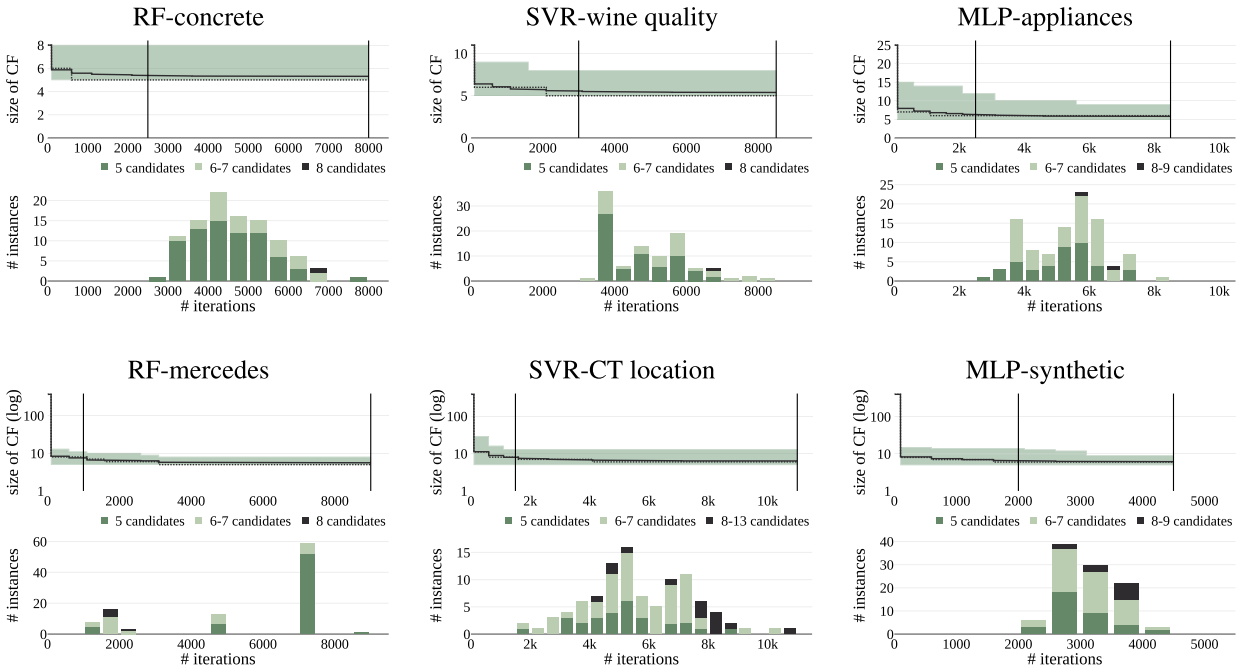


FIGURE 2. Behavior of TopShap when selecting the top $k = 5$ features of 100 instances.

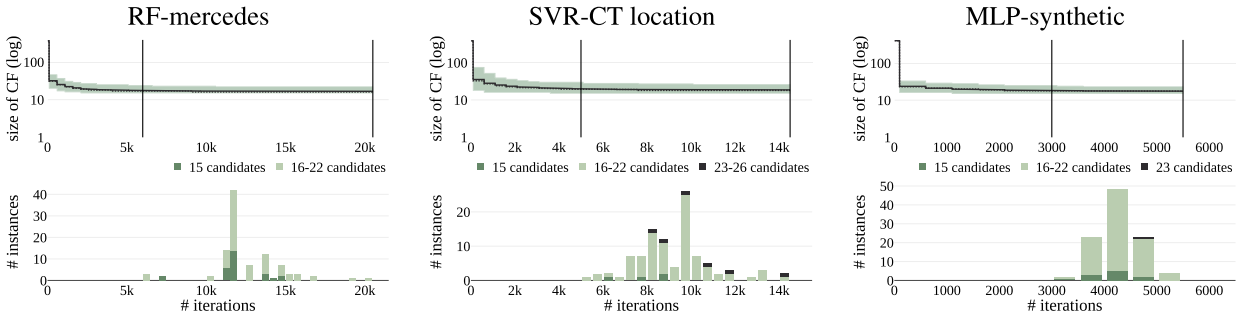


FIGURE 3. Behavior of TopShap when selecting the top $k = 15$ features of 100 instances.

equal to the initial number of features $|F|$ for all iterations. As shown here, TopShap started pruning almost immediately after the warm-up, with $|CF|$ quickly decreasing to be close to k. This means that, roughly speaking, TopShap avoided a cost of $(|F|-k) \times modelEvalCost$ per sampling iteration after warm-up, which is an effective cost reduction of the overall process, especially if $k \ll |F|$.

E. COMPARISON WITH KERNEL SHAP

The previous section showed the gain of incorporating TopShap pruning within the general approximation process as done by the permutation sampling family of approaches. As presented in Section III, the other predominant research direction to approximate SHAP values is solving an equivalent WLS problem. In the case of model-agnostic methods, this led to a reference algorithm named Kernel SHAP [2]. In this section, the execution times and the output of TopShap are compared to top-k searches based on Kernel SHAP.

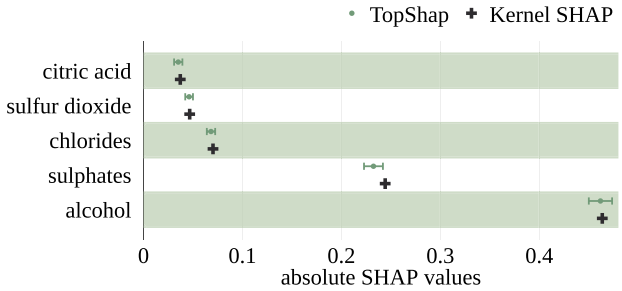


FIGURE 4. TopShap and Kernel SHAP approximations, with confidence intervals for TopShap.

As detailed in [27], the approximation made by Kernel SHAP is also based on sampling, but it is a sampling over the powerset of the features, not a sampling of the permutation of the features. This sampling is performed only once, to build the equations needed to formulate

TABLE 2. Comparison of Kernel SHAP and TopShap.

dataset	model	Kernel SHAP time (s)	TopShap time (s)	time ratio	Comparative scores recall-like	Comparative scores matching	TopShap overlaps
concrete	RF	25.63	17.07	1.5	100.0	100.0	0.62
concrete	MLP	18.78	23.73	0.79	100.0	97.8	0.59
concrete	SVR	13.49	12.00	1.12	100.0	99.2	0.54
concrete	STK	51.62	92.37	0.56	100.0	99.8	0.61
wine quality	RF	152.3	17.79	8.56	100.0	97.5	0.81
wine quality	MLP	98.85	40.03	2.47	100.0	100.0	0.92
wine quality	SVR	117.25	14.17	8.28	100.0	100.0	0.62
wine quality	STK	348.62	147.43	2.36	100.0	98.3	1.27
appliance energy	RF	3112.69	35.82	86.9	99.8	90.4	1.52
appliance energy	MLP	479.94	38.17	12.57	99.8	94.0	0.92
appliance energy	SVR	> 5400	552.15	> 9.78	-	-	8.22
appliance energy	STK	> 5400	419.44	> 12.87	-	-	2.12
PBMC	RF	201.92	23.23	8.69	99.6	84.8	0.68
PBMC	MLP	645.27	117.77	5.48	100.0	91.5	1.53
PBMC	SVR	242.62	24.47	9.91	100.0	100.0	0.74
PBMC	STK	968.62	367.46	2.64	99.2	87.4	1.49
mercedes	RF	904.45	39.24	23.05	99.4	99.3	1.09
mercedes	MLP	727.6	45.90	15.85	100.0	93.6	1.48
mercedes	SVR	> 5400	167.20	> 32.30	-	-	1.56
mercedes	STK	> 5400	670.74	> 8.05	-	-	1.26
CT location	RF	-	86.18	-	-	-	1.75
CT location	MLP	-	353.37	-	-	-	2.90
CT location	SVR	-	2638.82	-	-	-	1.61
CT location	STK	-	5314.55	-	-	-	1.88
synthetic	RF	1892.09	62.57	30.24	99.6	80.8	1.31
synthetic	MLP	927.03	56.15	16.51	100.0	98.8	1.39
synthetic	SVR	684.85	25.30	27.07	100.0	100.0	1.41
synthetic	STK	3276.0	163.07	20.09	100.0	99.0	1.45

the WLS problem, which is then solved by Kernel SHAP to approximate the SHAP values of all the features. Such an approach implies that reducing the search space is not straightforward for Kernel SHAP when we are only concerned with the top-k features. Indeed, for our comparison, the only reasonable option was to select these features in a post-processing step after running Kernel SHAP.

Kernel SHAP was called with its default parameters, the number of sampled coalitions in this default setting being equal to twice the number of features plus 2048, and the top 5 features were retained. The TopShap parameters were set to $k = 5$, confidence $\gamma = 0.95$ and $warmUp = 100$. Both methods performed approximations of $f_S(x)$, that is the output of a model f when only a subset S of the features of x is known. To allow for a fair comparison of global execution time and of the SHAP values themselves, these approximations were made in the same way for both methods using the training dataset as a reference to set values for the unknown part of x.

All times are reported for single-threaded executions on a desktop computer running Ubuntu Linux (2.1 GHz Intel Xeon, 192 GB RAM). The implementation of TopShap is provided in the git repository, while the implementation of Kernel SHAP is KernelExplainer from the SHAP library [40] (version 0.44.0). Both methods are implemented in Python, using numpy vectorization. The models were learned using scikit-learn, and the same models were given to Kernel SHAP and TopShap as input. In both the Kernel SHAP and TopShap implementations, the calls to the prediction functions of the models were not made one instance at a time. Each call was made for a batch of instances to amortize the possible overhead due to data structure preparation in prediction functions of the scikit-learn models.

1) COMPARING EXECUTION TIME

For both Kernel SHAP and TopShap, execution time was measured for the same set of 100 randomly chosen instances of the test dataset. Average execution times per instance are reported in Table 2. The time-ratio column corresponds to the average time per instance for Kernel SHAP divided by the average time per instance for TopShap. Note that executions that took more than one hour and a half (5400 seconds) were stopped. Note also that for *CT location*, one of the largest datasets, Kernel SHAP approximations could not be completed due to memory problems, and corresponding comparisons could not be reported (Table 2).

For nearly all models and datasets, TopShap was much faster than Kernel SHAP. Overall, run time was reduced

almost 10-fold (median ≈ 9.24), ranging from 2-fold to more than 85-fold. For the smallest dataset, *Concrete Compressive Strength*, TopShap reduced execution times only in half of the combinations. This dataset contained only 8 features whilst we searched for the top-5. As expected, TopShap was unable to benefit from a substantial amount of pruning. However, for all other datasets, a clear reduction in execution time was observed.

2) COMPARING SHAP VALUES

A comparison is now made of the top-k features and their approximated SHAP values output by TopShap and Kernel SHAP. The corresponding scores are given in Table 2 and, as for execution time, were obtained as an averages over a set of 100 randomly chosen instances of the test dataset.

TopShap provides confidence intervals, and the selection of the top-k is based on these intervals. Thus, it can report more than k features, because of ties due to overlapping intervals. Kernel SHAP does not provide confidence intervals, but, in principle, it could report ties if the same SHAP values were obtained for two features. However, this never occured in the experiments discussed here.

We compared the output of the two methods on three scores. First, we considered Kernel SHAP as a reference, and a recall-like score was computed as the percentage of the top-k features obtained with Kernel SHAP that also appeared in those reported by TopShap. Table 2 shows a nearly perfect recall-like score, with only 6 combinations out of 20 for which the average score is not 100% but slightly below. Note that a similar computation of a precision-like score would not be meaningful when Kernel SHAP is taken as a reference. This is because the top-k features of TopShap can include ties due to overlapping confidence intervals, whereas such ties are not taken into account by Kernel SHAP and would not be present in its output.

Next, to assess the extent to which the outputs of the two algorithms agreed for the SHAP values themselves, a matching score was computed. This score was the percentage of the confidence intervals of the top-k features of TopShap that included the SHAP value given by Kernel SHAP for the corresponding feature. An example of a matching score of 80% (i.e., 4/5) is illustrated in Fig. 4 for the same instance as in Fig. 1a from the *Wine Quality* dataset. Fig. 4 presents the SHAP values approximation of the top-5 features computed by TopShap and Kernel SHAP. All TopShap intervals encompassed the approximation made by Kernel SHAP, except for the feature *sulphates*, the value of which is nevertheless close to the interval's upper bound. The matching scores reported in Table 2 reflected a strong agreement of both algorithms on the SHAP values obtained.

3) CONFIDENCE INTERVAL OVERLAPS

The above observed high matching scores are of interest, if confidence intervals tend to isolate the SHAP values, i.e., if these intervals have limited or no overlap. This was assessed by counting for each interval the number of overlaps, that is, non-empty intersections with other intervals. The mean overlap counts are reported in Table 2 (column *TopShap overlaps*). Most combinations of datasets and models showed a mean overlap count much less than 2.00, indicating a good separation of the SHAP values by their intervals. A few combinations led to a greater number of overlaps, corresponding to cases where several ties were obtained in the output of TopShap. An extreme average of 8.22 is reported in Table 2 for the *Appliances Energy* dataset and *SVR* model. This was due to a large number of ties, leading to an output containing on average between 10 and 15 features, even if only the top 5 features were requested (Supplementary Materials Fig. S1). Even though ties may seem undesirable at first sight, it should be emphasized that ties simply mean that several features have similar importance in the prediction made by the model.

To summarize, TopShap and Kernel SHAP agreed well on their approximations of SHAP values. Moreover, the confidence intervals computed by TopShap provided a quantification of the remaining uncertainty. These intervals tended to be rather narrow, implying that, in general, few ties (overlap of intervals between features) were found.

F. USE CASE: THE PBMC DATASET

A major challenge in biology is to understand how genes interact. We know that understanding gene regulation is key: the question to answer is which transcription factors (TFs, genes that can influence the expression of other genes, including themselves) are responsible for regulation of a given gene on the genome? Even though we know for a few well-studied cases how TFs regulate a particular gene, generally speaking within a cell it is unknown which TF regulates which gene. This, however, can be predicted from data, such as single-cell gene expression data. From such data, regulatory links can be extracted using machine learning and feature importance techniques, in order to reveal which TFs are likely to be associated with a given target gene. Indeed, if we take for example one of the best currently available workflows, PySCENIC [41], its first computing step consists in learning a tree-based predictor of the expression of the target gene, from which an ordered list of features (i.e., TFs) is extracted through impurity-based feature importances [38]. In this manner, PySCENIC identifies the putative interactions between TFs and genes, from which a single regulatory network can be built for a population of cells.

In a similar way, we applied TopShap to the *PBMC* gene expression dataset, which is commonly used in the single-cell biology domain to demonstrate new methods. After quality control, it consists of 2638 high-quality cells (instances), mainly comprising cell types of the human immune system. One of these cell types is the B cell type. In this use case, a non-trivial biological question was asked regarding B cells: ''Which TFs are potentially regulating gene MS4A1, a well-known marker gene for B cells?''.

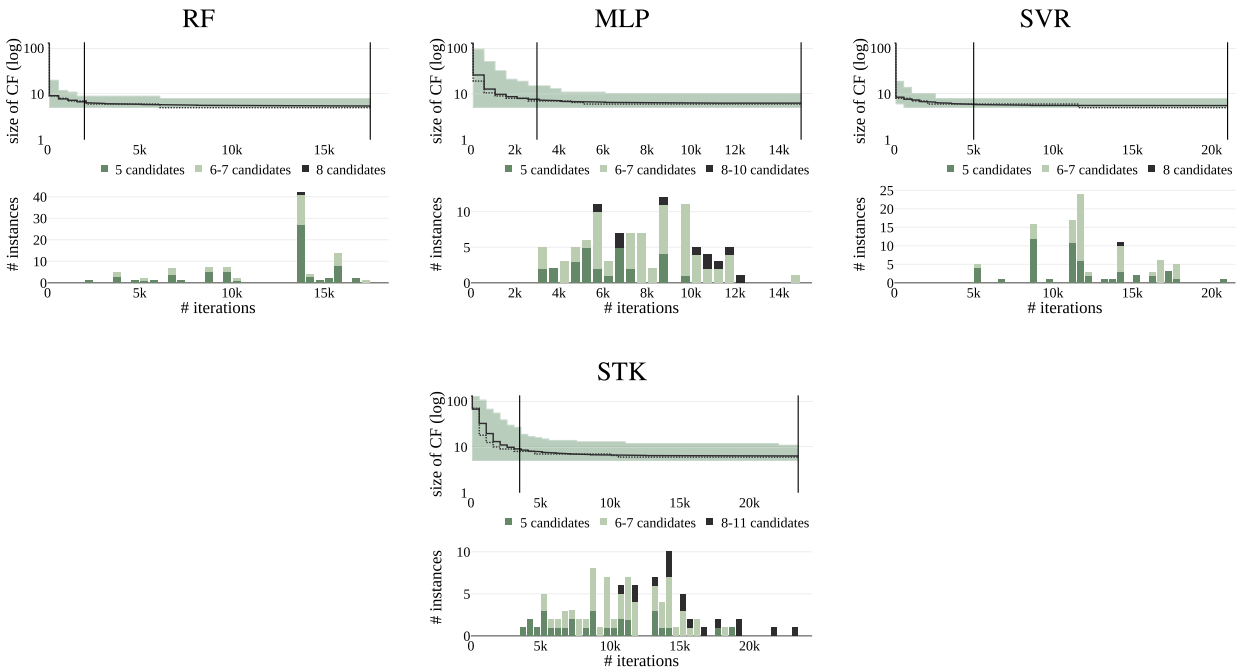


FIGURE 5. Behavior of TopShap for different models and 100 instances of the *PBMC* dataset. Parameter $k = 15$.

Four model types were trained, applying the same experimental setup as for the above reported experiments (see Section V-C): Random Forest (RF), Multi-Layer Perceptron (MLP), Support Vectors Regressor (SVR), and stacked regressors (STK). Except for STK, all approaches were previously used on single-cell RNA-seq data [41], [42], [43]. After splitting the dataset in training and test sets (respectively 70% and 30% of the 2638 cells) and guaranteeing all cell types were present, we trained the four models.

TopShap was then applied to each of the 105 B cells in the test data. Results were summarized in Table 3, where the column *Counts* indicates the number of times a TF is in the top-5 of a cell, per model (RF, MLP, SVR, STK). The five most-occurring top-5 TFs were indicated in black font color, and the counts were completed for any TF not present in the top-5 of all four models (green font color). Six out of eight TFs were found to be either known from literature for their capability to bind in the regulatory genomic regions around MS4A1 or for their involvement in B-cell functioning, as detailed by an entry in the column *Literature evidence* of Table 3.

The transcription factors SPIB and IRF8 were the most frequent features in the top-5 for all models. For the other six genes, the models did not fully agree, suggesting that each regression method provides a different insight into the data. The transcription factors POU2AF1, JUN, and GTF3A were all detected by three methods, and only JUN did not show evidence of links with MS4A1 or B cells in the literature. It may play a role, however, as JUN is a pioneer transcription factor involved in general regulatory processes of transcription. Alternatively, it might be present due to its involvement in the cellular stress response caused by the experimental procedure of single-cell RNA sequencing. The genes CEBPB, NFATC1, and PAX5 were picked up by only one method each, and two of them could be associated with literature evidence of their role in the regulation of MS4A1 and B cells. It could be considered that the variation between the methods is an opportunity to generate new hypotheses for biologists.

TABLE 3. Transcription factors predicted to regulate MS4A1 (marker gene for B cells).

Transcription factor	Counts				Literature evidence
	RF	MLP	SVR	STK	
SPIB	**104**	**71**	**100**	**73**	[44] [45]
IRF8	**93**	**79**	**90**	**82**	[46]
POU2AF1	**57**	21	**63**	**33**	[47] [48]
JUN	5	**49**	**21**	**48**	
GTF3A	**36**	**37**	7	**36**	[49]
CEBPB	16	**27**	8	24	
NFATC1	**41**	6	6	7	[50]
PAX5	12	7	**20**	12	[50]

For all models, from the viewpoint of algorithm performance, it was observed that the width of confidence intervals shrank quickly and that pruning was (again) very effective (see Fig. 5).

VI. CONCLUSION

In this paper, TopShap was proposed, a model-agnostic algorithm for searching the k most important features for a prediction. TopShap operates within the SHAP framework to determine local feature importance, the so-called SHAP

values. It drastically reduces computational costs by iteratively interleaving sampling steps to improve bounds of SHAP values, and pruning steps to stop any computation for features that can no longer be in the top-k. Effectiveness of TopShap was demonstrated by applying it to various datasets, including a use case in the domain of single-cell gene expression analysis. Correctness of its output was verified by comparing our method to the state-of-the-art technique of Kernel SHAP. Moreover, TopShap was shown to be an order of magnitude faster than Kernel SHAP, if the total number of features in a dataset were much larger than the top-k that one would like to use to explain a model prediction.

Despite the advantages of TopShap, it has some limitations. The first is that it computes approximations of SHAP values, like the other model-agnostic approaches. A useful follow-up would be to further improve this approximation scheme to provide tighter bounds for SHAP values. A second limitation is reported in the experiments. When computing the top-k SHAP values, the reduction in execution time was most clearly observed when k was small compared to the total number of features in the dataset. Future work on the sampling strategy, such as adopting an antithetic sampling instead of a uniform one, could permit faster convergence and thus could allow for a reduction of computational cost when k is close to the total number of features. As a third limitation, it should be noted that because TopShap outputs the top-k features and their SHAP values, unnecessary computations may be performed if only the top-k features and their ranks are of interest. The algorithm stops when two conditions are satisfied, namely only k features remain (including possible ties) and their SHAP value approximations are stable. However, this can be too strict, if precise SHAP values are not required. In this case, a useful improvement would be to design an alternative criterion, for instance to stop as soon as the ranking of the top-k features is stable.

REFERENCES

[1] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *J. Artif. Intell. Res.*, vol. 70, pp. 245–317, Jan. 2021.

[2] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 4765–4774.

[3] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable AI for trees," *Nature Mach. Intell.*, vol. 2, no. 1, pp. 56–67, Jan. 2020.

[4] D. Wang, S. Thunéll, U. Lindberg, L. Jiang, J. Trygg, and M. Tysklind, "Towards better process management in wastewater treatment plants: Process analytics based on SHAP values for tree-based machine learning methods," *J. Environ. Manage.*, vol. 301, Jan. 2022, Art. no. 113941.

[5] L. S. Shapley, "A value for n-person games," in *Contributions To the Theory of Games (AM-28)*. Princeton, NJ, USA: Princeton Univ. Press, 1953, pp. 307–318.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1135–1144.

[7] E. Strumbelj and I. Kononenko, "An efficient explanation of individual classifications using game theory," *J. Mach. Learn. Res.*, vol. 11, pp. 1–18, Mar. 2010.

[8] L. H. B. Olsen, I. K. Glad, M. Jullum, and K. Aas, "A comparative study of methods for estimating conditional Shapley values and when to use them," 2023, *arXiv:2305.09536*.

[9] R. Mitchell, J. Cooper, E. Frank, and G. Holmes, "Sampling permutations for Shapley value estimation," *J. Mach. Learn. Res.*, vol. 23, no. 1, pp. 2082–2127, 2022.

[10] F. Hatami, Md. M. Rahman, B. Nikparvar, and J.-C. Thill, "Non-linear associations between the urban built environment and commuting modal split: A random forest approach and SHAP evaluation," *IEEE Access*, vol. 11, pp. 12649–12662, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10035376

[11] R.-K. Sheu, M. S. Pardeshi, K.-C. Pai, L.-C. Chen, C.-L. Wu, and W.-C. Chen, "Interpretable classification of pneumonia infection using eXplainable AI (XAI-ICP)," *IEEE Access*, vol. 11, pp. 28896–28919, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10065420

[12] J. Verhaeghe, J. Van Der Donckt, F. Ongenae, and S. Van Hoecke, "Powershap: A power-full Shapley feature selection method," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2022, pp. 71–87.

[13] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowl. Inf. Syst.*, vol. 41, no. 3, pp. 647–665, Dec. 2014.

[14] G. Castronuovo, G. Favia, V. Telesca, and A. Vammacigno, "Analyzing the interactions between environmental parameters and cardiovascular diseases using random forest and SHAP algorithms," *Rev. Cardiovascular Med.*, vol. 24, no. 11, p. 330, 2023. [Online]. Available: https://www.imrpress.com/journal/RCM/24/11/10.31083/j.rcm2411330

[15] N. Kostopoulos, D. Kalogeras, D. Pantazatos, M. Grammatikou, and V. Maglaris, "SHAP interpretations of tree and neural network DNS classifiers for analyzing DGA family characteristics," *IEEE Access*, vol. 11, pp. 61144–61160, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10151849

[16] J. Takalo-Mattila, M. Heiskanen, V. Kyllönen, L. Määttä, and A. Bogdanoff, "Explainable steel quality prediction system based on gradient boosting decision trees," *IEEE Access*, vol. 10, pp. 68099–68110, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9804717/?arnumber=9804717

[17] Z. Song, S. Cao, and H. Yang, "An interpretable framework for modeling global solar radiation using tree-based ensemble machine learning and Shapley additive explanations methods," *Appl. Energy*, vol. 364, Jun. 2024, Art. no. 123238. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261924006214

[18] M. Kuroki and T. Yamasaki, "BSED: Baseline Shapley-based explainable detector," *IEEE Access*, vol. 12, pp. 57959–57973, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10505291/

[19] H. Chen, S. M. Lundberg, and S.-I. Lee, "Explaining a series of models by propagating Shapley values," *Nature Commun.*, vol. 13, no. 1, p. 4512, Aug. 2022. [Online]. Available: https://www.nature.com/articles/s41467-022-31384-3

[20] J. Wang, J. Wiens, and S. Lundberg, "Shapley flow: A graph-based approach to interpreting model predictions," in *Proc. 24th Int. Conf. Artif. Intell. Stat.*, 2021, pp. 721–729. [Online]. Available: https://proceedings.mlr.press/v130/wang21b.html

[21] Y. Kwon and J. Y. Zou, "WeightedSHAP: Analyzing and improving Shapley based feature attributions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 34363–34376. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/de1739eba209c682a90ec3669229ab2d-Abstract-Conference.html

[22] L. Gan, B. Liu, A. Meng, F. Zhang, and G. Qu, "SATTree: A SHAP-augmented threshold tree for clustering explanation," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Dec. 2023, pp. 931–938. [Online]. Available: https://ieeexplore.ieee.org/document/10411540/

[23] J. Castro, D. Gómez, and J. Tejada, "A polynomial rule for the problem of sharing delay costs in PERT networks," *Comput. Oper. Res.*, vol. 35, no. 7, pp. 2376–2387, Jul. 2008.

[24] U. Faigle and W. Kern, "The Shapley value for cooperative games under precedence constraints," *Int. J. Game Theory*, vol. 21, no. 3, pp. 249–266, Sep. 1992.

[25] A. Charnes, B. Golany, M. Keane, and J. Rousseau, "Extremal principle solutions of games in characteristic function form: Core, Chebychev and Shapley value generalizations," in *Econometrics of Planning and Efficiency*. Dordrecht, The Netherlands: Springer, 1988, pp. 123–133.

[26] J. Castro, D. Gómez, and J. Tejada, "Polynomial calculation of the Shapley value based on sampling," *Comput. Oper. Res.*, vol. 36, no. 5, pp. 1726–1730, May 2009.

[27] K. Aas, M. Jullum, and A. Løland, "Explaining individual predictions when features are dependent: More accurate approximations to Shapley values," *Artif. Intell.*, vol. 298, Sep. 2021, Art. no. 103502. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370221000539
[28] P. Kolpaczki, V. Bengs, and E. Hüllermeier, "Identifying top-k players in cooperative games via Shapley bandits," in *Proc. LWDA*, 2021, pp. 133–144.
[29] N. R. Suri and Y. Narahari, "Determining the top-k nodes in social networks using the Shapley value," in *Proc. 7th Int. Conf. Autonomous Agents Multiagent Syst.*, 2008, pp. 1509–1512.
[30] O. J. Dunn, "Multiple comparisons among means," *J. Amer. Stat. Assoc.*, vol. 56, no. 293, p. 52, Mar. 1961.
[31] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, Aug. 1962.
[32] I.-C. Yeh. (1998). *Concrete Compressive Strength*. [Online]. Available: https://archive.ics.uci.edu/dataset/165
[33] A. C. Paulo Cortez. (2009). *Wine Quality*. [Online]. Available: https://archive.ics.uci.edu/dataset/186
[34] L. Candanedo. (2017). *Appliances Energy Prediction*. [Online]. Available: https://archive.ics.uci.edu/dataset/374
[35] H.-P. K. F. Graf. (2011). *Relative Location of CT Slices on Axial Axis*. [Online]. Available: https://archive.ics.uci.edu/dataset/206
[36] *UCI Machine Learning Repository*. Accessed: Jun. 27, 2024. [Online]. Available: https://archive.ics.uci.edu/
[37] *Kaggle: Your Machine Learning and Data Science Community*. Accessed: Jun. 27, 2024. [Online]. Available: https://www.kaggle.com/
[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[39] *Datasets—Single Cell Gene Expression -Official 10× Genomics Support*. Accessed: Jun. 27, 2024. [Online]. Available: https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/pbmc3k
[40] *SHapley Additive ExPlanations*. Accessed: Jun. 27, 2024. [Online]. Available: https://github.com/shap/shap
[41] B. Van de Sande, C. Flerin, K. Davie, M. De Waegeneer, G. Hulselmans, S. Aibar, R. Seurinck, W. Saelens, R. Cannoodt, Q. Rouchon, T. Verbeiren, D. De Maeyer, J. Reumers, Y. Saeys, and S. Aerts, "A scalable SCENIC workflow for single-cell gene regulatory network analysis," *Nature Protocols*, vol. 15, no. 7, pp. 2247–2276, Jul. 2020.
[42] R. Magnusson, J. N. Tegnér, and M. Gustafsson, "Deep neural network prediction of genome-wide transcriptome signatures—Beyond the black-box," *NPJ Syst. Biol. Appl.*, vol. 8, no. 1, p. 9, Feb. 2022.
[43] B. Yang, W. Bao, B. Chen, and D. Song, "Single_cell_GRN: Gene regulatory network identification based on supervised learning method and single-cell RNA-seq data," *BioData Mining*, vol. 15, no. 1, pp. 1–18, Dec. 2022.
[44] G. H. Su, H. S. Ip, B. S. Cobb, M. M. Lu, H. M. Chen, and M. C. Simon, "The Ets protein Spi-B is expressed exclusively in B cells and T cells during development.," *J. Exp. Med.*, vol. 184, no. 1, pp. 203–214, Jul. 1996.
[45] D. Ray, R. Bosselut, J. Ghysdael, M. G. Mattei, A. Tavitian, and F. Moreau-Gachelin, "Characterization of Spi-B, a transcription factor related to the putative oncoprotein Spi-1/PU.1," *Mol. Cellular Biol.*, vol. 12, no. 10, pp. 4297–4304, Oct. 1992.
[46] L. Grzelak, F. Roesch, A. Vaysse, A. Biton, R. Legendre, F. Porrot, P.-H. Commère, C. Planchais, H. Mouquet, M. Vignuzzi, T. Bruel, and O. Schwartz, "IRF8 regulates efficacy of therapeutic anti-CD20 monoclonal antibodies," *Eur. J. Immunol.*, vol. 52, no. 10, pp. 1648–1661, 2022.
[47] G. Pavlasova and M. Mraz, "The regulation and function of CD20: An 'enigma' of B-cell biology and targeted therapy," *Haematologica*, vol. 105, no. 6, pp. 1494–1506, Jun. 2020.
[48] M. Gstaiger, L. Knoepfel, O. Georgiev, W. Schaffner, and C. M. Hovens, "A B-cell coactivator of octamer-binding transcription factors," *Nature*, vol. 373, no. 6512, pp. 360–362, Jan. 1995.
[49] D. Huang and I. Ovcharenko, "Epigenetic and genetic alterations and their influence on gene regulation in chronic lymphocytic leukemia," *BMC Genomics*, vol. 18, no. 1, pp. 1–15, Dec. 2017.
[50] J. C. Romero-Masters, S. M. Huebner, M. Ohashi, J. A. Bristol, B. E. Benner, E. A. Barlow, G. L. Turk, S. E. Nelson, D. C. Baiu, N. Van Sciver, E. A. Ranheim, J. Gumperz, N. M. Sherer, P. J. Farrell, E. C. Johannsen, and S. C. Kenney, "B cells infected with type 2 Epstein-Barr virus (EBV) have increased NFATc1/NFATc2 activity and enhanced lytic gene expression in comparison to type 1 EBV infection," *PLOS Pathogens*, vol. 16, no. 2, Feb. 2020, Art. no. e1008365. [Online]. Available: https://dx.plos.org/10.1371/journal.ppat.1008365

LISA CHABRIER (Student Member, IEEE) is currently pursuing the Ph.D. degree with Inria, in the Beagle Team that is based in Lyon. Her research interests include machine learning explainability and feature ranking, oriented toward applications in systems biology, such as gene regulatory network inference. Naturally, SHAP values were of interest in this context.

ANTON CROMBACH is a permanent Researcher with Inria, the French National Research Institute For the Digital Sciences. He is a Systems Biologist using computer simulations, bioinformatic analyses, and maths to answer questions on the evolution and functioning of gene regulatory networks. Currently, he focuses on topics in single-cell data analysis, such as explainable machine learning and its application in biomedical contexts.

SERGIO PEIGNIER received the Ph.D. degree in computer science from INSA-Lyon, in 2017, in the field of evolutionary subspace clustering. He is an Assistant Professor with INSA-Lyon, University of Lyon. He carried out his research with the BF2i Laboratory (UMR0203). Since then, he has mainly conducted research projects on the development of transfer learning techniques for random forests, inference of gene regulatory networks based on machine learning, and hyper-spectral images processing applied to pest detection.

CHRISTOPHE RIGOTTI received the Ph.D. degree in computer science from INSA-Lyon, in 1996, in the fields of object-oriented programming and deductive databases. He is an Assistant Professor with INSA-Lyon, University of Lyon. He is with the LIRIS Laboratory (UMR5205 CNRS) and the Inria team Beagle. Since then, he has been working on multi-dimensional databases, constraint programming, and data mining. In data mining, his main research interests include spatio-temporal pattern mining, condensed representations for pattern extraction, and cluster analysis.