Artificial Intelligence 298 (2021) 103502

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Explaining individual predictions when features are dependent: More accurate approximations to Shapley values [☆]

Kjersti Aas [*], Martin Jullum, Anders Løland

Norwegian Computing Center, P.O. Box 114, Blindern, N-0314 Oslo, Norway

A R T I C L E   I N F O

Article history:
Received 3 October 2019
Received in revised form 5 January 2021
Accepted 29 March 2021
Available online 31 March 2021

Keywords:
Feature attribution
Shapley values
Kernel SHAP
Dependence

A B S T R A C T

Explaining complex or seemingly simple machine learning models is an important practical problem. We want to explain individual predictions from such models by learning simple, interpretable explanations. Shapley value is a game theoretic concept that can be used for this purpose. The Shapley value framework has a series of desirable theoretical properties, and can in principle handle any predictive model. Kernel SHAP is a computationally efficient approximation to Shapley values in higher dimensions. Like several other existing methods, this approach assumes that the features are independent. Since Shapley values currently suffer from inclusion of unrealistic data instances when features are correlated, the explanations may be very misleading. This is the case even if a simple linear model is used for predictions. In this paper, we extend the Kernel SHAP method to handle dependent features. We provide several examples of linear and non-linear models with various degrees of feature dependence, where our method gives more accurate approximations to the true Shapley values.

1. Introduction

Interpretability is crucial when a complex machine learning model is to be applied in areas such as medicine [1], fraud detection [2] or credit scoring [3]. In many applications, complex hard-to-interpret machine learning models like deep neural networks, random forests and gradient boosting machines are currently outperforming the traditional, and to some extent interpretable, linear/logistic regression models. However, often there is a clear trade-off between model complexity and model interpretability, meaning that it is often hard to understand why these sophisticated models perform so well. This lack of explanation constitutes a practical issue – can I trust the model? [4], and a legal issue – those who develop the model can be required by law to explain what the model does to those who are exposed to automated decisions (the General Data Protection Regulation [5]). In response, a new line of research has emerged that focuses on helping users to interpret the predictions from advanced machine learning methods.

Existing work on explaining complex models can be divided into two main categories; global and local explanations. The former tries to describe the model as a whole, in terms of which variables/features that influenced the general model the most. Two common methods for such an overall explanation are permutation based feature importance [6] or partial dependence plots [7]. Local explanations, on the other hand, try to identify how the different input variables/features influenced a specific prediction/output from the model, and are often referred to as individual prediction explanation methods. Such

[☆] This paper is part of the Special Issue on Explainable AI.
[*] Corresponding author.
E-mail addresses: kjersti.aas@nr.no (K. Aas), martin.jullum@nr.no (M. Jullum), anders.loland@nr.no (A. Løland).

https://doi.org/10.1016/j.artint.2021.103502

explanations are particularly useful for complex models which behave rather different for different feature combinations, meaning that the global explanation is not representative for the local behavior.

Local explanation methods may further be divided into two categories: model-specific and model-agnostic (general) explanation methods. In this paper the focus is on the latter. The methods in this category usually try to explain individual predictions by determining simple, interpretable explanations of the model specifically for a given prediction. Three examples are Explanation Vectors [8], LIME (Local Interpretable Model-agnostic Explanations) [4] and Shapley values [9–11]. The latter approach, which builds on concepts from cooperative game theory [12], has a series of desirable theoretical properties [11].

The Shapley value is a method originally invented for assigning payouts to players depending on their contribution towards the total payout. In the explanation setting, the features are the players and the prediction is the total payout. In this framework, the difference between the prediction and the average prediction is perfectly distributed among the features. This property distinguishes Shapley values from other methods like for example LIME, which does not guarantee perfectly distributed effects. It should be noted that LIME and Shapley values actually explain two different things. For instance, if the prediction to be explained is the probability of person A crashing his car, the sum of the Shapley values for all features is equal to the difference between this prediction and the mean probability of a person crashing his car, where the mean is taken over all persons having a driver license. The sum of the LIME values is also equal to the difference between this prediction and a mean probability, but here the mean is taken over all persons “similar to” person A. That is, Shapley values explain the difference between the prediction and the global average prediction, while LIME explains the difference between the prediction and a local average prediction. Appropriate model explanations should be consistent with how humans understand that model. In their study, [11] found a much stronger agreement between human explanations and Shapley values than with LIME.

Shapley values have also been used for measuring global feature importance. For instance, it has been used to partition the R^2 quantity among the d features in a linear regression model (“Shapley regression values”), both assuming independent features [13], and more recently also for dependent features [14–16]. A general Shapley framework for global additive importance measures is suggested by [17].

The main disadvantage of the Shapley value is that the computational complexity grows exponentially and becomes intractable for more than, say, ten features. This has led to approximations like the Shapley Sampling Values [9,10] and Kernel SHAP [11]. The latter requires less computational power to obtain a similar approximation accuracy. Hence, in this paper, the focus is on the Kernel SHAP method. While having many desirable properties, this method assumes feature independence. In observational studies and machine learning problems, it is very rare that the features are statistically independent, meaning that the Shapley value methods suffer from inclusion of predictions based on unrealistic data instances when features are correlated. This is the case even if a simple linear model is used.

The main contribution of this paper is to extend the Kernel SHAP method to handle dependent features. The methodology has been implemented in the `R`-package `shapr` available on CRAN [18]. Our paper is a revised version of an unpublished paper [19], which, to the best of our knowledge, was the first to address and account for dependence within Shapley value based individual prediction explanation.

Later there has been several papers discussing the difference between our approach and the original Kernel SHAP method, termed respectively the observational and the interventional approach in the succeeding literature. Lundberg and Lee [11] advocate the observational approach, but uses the interventional approach for computational reasons. Janzing et al. [20] argue for a causal interpretation of Shapley values, where they replace conventional “conditioning by observation” with “conditioning by intervention”, as in Pearl’s do-calculus [21]. Frye et al. [22] suggest so-called asymmetric Shapley values as a way to incorporate causal knowledge in the real world by restricting the possible permutations of the features when computing the Shapley values to those consistent with a partial causal ordering. In line with our approach, they then apply conventional conditioning by observation to make sure that the explanations respect the multivariate distribution of the data, which they denote the “data manifold”. Heskes et al. [23] generalize the work on causal Shapley values further, partly based on our approach. They define our approach as “symmetric conditional Shapley values”. Chen et al. [24] argue that neither is preferable in general, but that the choice between the observational (“true to the data”) or interventional (“true to the model”) approach is application dependent.

All the above-mentioned approaches are model-agnostic in the sense that they can be used to explain any machine learning method. In addition to these methods, there is a method called TreeSHAP [25] which is specially designed for tree ensemble methods like XGBoost [26]. According to the authors, TreeSHAP accounts for some of the feature dependence, but not all. As will be apparent from our simulation experiments, this method can be inaccurate when the features are dependent.

The rest of this paper is organized as follows. Section 2 reviews the Shapley values and the Kernel SHAP method. Section 3 contains the main contribution of the paper, that is the proposed approaches for accounting for the dependence, while Section 4 gives the results from several experiments validating these methods. Finally, Section 5 contains some concluding remarks.

2. The exact Shapley value and the Kernel SHAP approximation

In this section we first give the definition of the Shapley value from game theory in Section 2.1, then explain its use in the context of explaining individual predictions in Section 2.2. In Section 2.3 we finally describe the Kernel SHAP method.

2.1. The exact Shapley value and cooperative game theory

Consider a cooperative game with M players aiming at maximizing a payoff, and let $\mathcal{S} \subseteq \mathcal{M} = \{1, \ldots, M\}$ be a subset consisting of $|\mathcal{S}|$ players. Assume that we have a contribution function $v(\mathcal{S})$ that maps subsets of players to the real numbers, called the worth or contribution of coalition $\mathcal{S}$. It describes the total expected sum of payoffs the members of $\mathcal{S}$ can obtain by cooperation. The Shapley value [12] is one way to distribute the total gains to the players, assuming that they all collaborate. It is a "fair" distribution in the sense that it is the only distribution with certain desirable properties listed below. According to the Shapley value, the amount that player j gets is

$$\phi_j(v) = \phi_j = \sum_{\mathcal{S} \subseteq \mathcal{M} \setminus \{j\}} \frac{|\mathcal{S}|!(M - |\mathcal{S}| - 1)!}{M!} (v(\mathcal{S} \cup \{j\}) - v(\mathcal{S})), \quad j = 1, \ldots, M, \tag{1}$$

that is, a weighted mean over contribution function differences for all subsets $\mathcal{S}$ of players not containing player j. Note that the empty set $\mathcal{S} = \emptyset$ is also part of this sum. The formula can be interpreted as follows: Imagine the coalition being formed for one player at a time, with each player demanding their contribution $v(\mathcal{S} \cup \{j\}) - v(\mathcal{S})$ as a fair compensation. Then, for each player, compute the average of this contribution over all permutations of all possible coalitions, yielding a weighted mean over the unique coalitions.

To illustrate the application of (1), let us consider a game with three players such that $\mathcal{M} = \{1, 2, 3\}$. Then, there are eight possible subsets; $\emptyset$, $\{1\}$, $\{2\}$, $\{3\}$, $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, and $\{1, 2, 3\}$. Using (1), the Shapley values for the three players are given by

$$\begin{aligned}
\phi_1 &= \frac{1}{3}\big(v(\{1,2,3\}) - v(\{2,3\})\big) + \frac{1}{6}\big(v(\{1,2\}) - v(\{2\})\big) + \frac{1}{6}\big(v(\{1,3\}) - v(\{3\})\big) + \frac{1}{3}\big(v(\{1\}) - v(\emptyset)\big),\\
\phi_2 &= \frac{1}{3}\big(v(\{1,2,3\}) - v(\{1,3\})\big) + \frac{1}{6}\big(v(\{1,2\}) - v(\{1\})\big) + \frac{1}{6}\big(v(\{2,3\}) - v(\{3\})\big) + \frac{1}{3}\big(v(\{2\}) - v(\emptyset)\big),\\
\phi_3 &= \frac{1}{3}\big(v(\{1,2,3\}) - v(\{1,2\})\big) + \frac{1}{6}\big(v(\{1,3\}) - v(\{1\})\big) + \frac{1}{6}\big(v(\{2,3\}) - v(\{2\})\big) + \frac{1}{3}\big(v(\{3\}) - v(\emptyset)\big).
\end{aligned}$$

Let us also define the non-distributed gain $\phi_0 = v(\emptyset)$, that is, the fixed payoff which is not associated to the actions of any of the players, although this is often zero for coalition games.

By summarizing the right hand sides above, we easily see that they add up to the total worth of the game: $\phi_0 + \phi_1 + \phi_2 + \phi_3 = v(\{1, 2, 3\})$.

The Shapley value has the following desirable properties:

Efficiency: The total gain is distributed:

$$\sum_{j=0}^{M} \phi_j = v(\mathcal{M})$$

Symmetry: If i and j are two players who contribute equally to all possible coalitions, i.e.

$$v(\mathcal{S} \cup \{i\}) = v(\mathcal{S} \cup \{j\})$$

for every subset $\mathcal{S}$ which contains neither i nor j, then their Shapley values are identical:

$$\phi_i = \phi_j.$$

Dummy player: If $v(\mathcal{S} \cup \{j\}) = v(\mathcal{S})$ for a player j and all coalitions $\mathcal{S} \subseteq \mathcal{M} \setminus \{j\}$, then $\phi_j = 0$.

Linearity: If two coalition games described by gain functions v and w are combined, then the distributed gains correspond to the gains derived from v and the gains derived from w:

$$\phi_i(v + w) = \phi_i(v) + \phi_i(w),$$

for every i. Also, for any real number a we have that

$$\phi_i(a\,v) = a\phi_i(v).$$

The Shapley values is the only set of values satisfying the above properties, see [12] and [27] for proofs.

2.2. Shapley values for prediction explanation

Consider a classical machine learning scenario where a training set $\{y^i, \boldsymbol{x}^i\}_{i=1,\dots,n_{\text{train}}}$ of size n_{train} has been used to train a predictive model $f(\boldsymbol{x})$ attempting to resemble a response value y as closely as possible. Assume now that we want to explain the prediction from the model $f(\boldsymbol{x}^*)$, for a specific feature vector $\boldsymbol{x} = \boldsymbol{x}^*$. Štrumbel and Kononenko [9,10] and Lundberg and Lee [11] suggest to do this using Shapley values. By moving from game theory to decomposing an individual prediction into feature contributions, the single prediction takes the place of the payout, and the features take the place of the players. We have that the prediction $f(\boldsymbol{x}^*)$ is decomposed as follows

$$f(\boldsymbol{x}^*) = \phi_0 + \sum_{j=1}^{M} \phi_j^*,$$

where $\phi_0 = \mathrm{E}[f(\boldsymbol{x})]$ and ϕ_j^* is the ϕ_j for the prediction $\boldsymbol{x} = \boldsymbol{x}^*$. That is, the Shapley values explain the difference between the prediction $y^* = f(\boldsymbol{x}^*)$ and the global average prediction. A model of this form is an additive feature attribution method, and it is the only additive feature attribution method that adhers to the properties listed in Section 2.1 [11]. In Appendix A we discuss why these properties are useful in the prediction explanation setting.

To be able to compute the Shapley values in the prediction explanation setting, we need to define the contribution function $v(\mathcal{S})$ for a certain subset $\mathcal{S}$. This function should resemble the value of $f(\boldsymbol{x}^*)$ when we only know the value of the subset $\mathcal{S}$ of these features. To quantify this, we follow [11] and use the expected output of the predictive model, conditional on the feature values $\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*$ of this subset:

$$v(\mathcal{S}) = \mathrm{E}[f(\boldsymbol{x})|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*]. \tag{2}$$

Other measures, such as the conditional median, may also be appropriate. However, the conditional expectation summarizes the whole probability distribution and it is the most common estimator in prediction applications. When viewed as a prediction for $f(\boldsymbol{x}^*)$, it is also the minimizer of the commonly used squared error loss function.

We show in Appendix B that if the predictive model is a linear regression model $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{M} \beta_j x_j$, where all features $x_j, j = 1, \dots, M$ are independent, then, under (2), the Shapley values take the simple form:

$$\phi_0 = \beta_0 + \sum_{j=1}^{M} \beta_j E[x_j], \quad \text{and} \quad \phi_j = \beta_j (x_j^* - E[x_j]), \quad j = 1, \dots, M. \tag{3}$$

Note that for ease of notation, we have here and in the rest of the paper dropped the superscript * for the ϕ_j values. Every prediction $f(\boldsymbol{x}^*)$ to be explained will result in different sets of ϕ_j values.

To the best of our knowledge, no explicit formula like (3) exists for the general case of dependent features with non-linear models. With M features, the number of possible subsets involved in (1) is 2^M. Hence, the number of possible subsets increases exponentially when M increases, meaning that the exact solution to this problem becomes computationally intractable when we have more than a few features. In Section 2.3, we shall see how a clever approximation method may be used to (partly) overcome this issue.

In addition to a computationally tractable approximation for computing the Shapley values, applying the above method in practice requires an estimate of the expectation in (2) for all $\boldsymbol{x}_{\mathcal{S}}$. The main methodological contribution of this paper is describing, developing, and comparing methodology to appropriately estimate these expectations. In Section 2.3.2 we describe the state-of-art method for determining the expectations before we describe our proposed approaches in Section 3.

2.3. Kernel SHAP

The Kernel SHAP method of [11] aims at estimating Shapley values under (2) in practical situations. The method may be divided in two separate parts:

(i) A clever computationally tractable approximation for computing the Shapley values of (1)
(ii) A simple method for estimating $v(\mathcal{S})$

In [11], the method is presented in a somewhat limited form without full detail. In order to facilitate the understanding of the method and our consecutive improvements to the method, we shall therefore in this section carefully re-state the Kernel SHAP method. We will first present part i), assuming $v(\mathcal{S})$ is known, and then present the method for estimating $v(\mathcal{S})$ used by Kernel SHAP.

2.3.1. Approximated weighted least squares

There are several alternative equivalent formulas for the Shapley values. Charnes et al. [28] and later Lundberg and Lee [11] define the Shapley values as the optimal solution of a certain weighted least squares (WLS) problem.

In its simplest form, the WLS problem can be stated as the problem of minimizing

$$\sum_{\mathcal{S}\subseteq\mathcal{M}} (v(\mathcal{S}) - (\phi_0 + \sum_{j\in\mathcal{S}} \phi_j))^2 k(M,\mathcal{S}), \tag{4}$$

with respect to $\phi_0, \ldots, \phi_M$, where $k(M,\mathcal{S}) = (M-1)/(\binom{M}{|\mathcal{S}|}|\mathcal{S}|(M-|\mathcal{S}|))$, are denoted the Shapley kernel weights. Let us write $\boldsymbol{Z}$ for the $2^M \times (M+1)$ binary matrix representing all possible combinations of inclusion/exclusion of the M features, where the first column is 1 for every row, while entry $j+1$ of row l is 1 if feature j is included in combination l, and 0 otherwise. Let also $\boldsymbol{v}$ be a vector containing $v(\mathcal{S})$, and $\boldsymbol{W}$ be the $2^M \times 2^M$ diagonal matrix containing $k(M,|\mathcal{S}|)$, where $\mathcal{S}$ in both cases resembles the feature combinations of the corresponding row in $\boldsymbol{Z}$. Letting $\boldsymbol{\phi}$ be a vector containing $\phi_0, \ldots, \phi_M$, (4) may be rewritten to

$$(\boldsymbol{v} - \boldsymbol{Z}\boldsymbol{\phi})^T \boldsymbol{W}(\boldsymbol{v} - \boldsymbol{Z}\boldsymbol{\phi}), \tag{5}$$

for which the solution is

$$\boldsymbol{\phi} = \left(\boldsymbol{Z}^T\boldsymbol{W}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{W}\boldsymbol{v}. \tag{6}$$

In practice, the infinite Shapley kernel weights $k(M,M) = k(M,0) = \infty$ may be set to a large constant c, for example $c = 10^6$, or imposing the constraints that $\phi_0 = v(\emptyset)$ and $\sum_{j=0}^{M}\phi_j = v(\mathcal{M})$ into the problem.

When the model contains more than a few features M, computing the right hand side of (6) is still computationally expensive. The Kernel SHAP trick is to use the weighted least squares formulation to approximate (6). The Shapley kernel weights have very different sizes, meaning that the majority of the subsets $\mathcal{S}$, that is, the rows in $\boldsymbol{Z}$, contributes very little to the Shapley value. Hence, assuming that we have a proper approximation for the elements in $\boldsymbol{v}$, a consistent approximation may be obtained by sampling (with replacement) a subset $\mathcal{D}$ of $\mathcal{M}$ from a probability distribution following the Shapley weighting kernel, and using only those rows $\boldsymbol{Z}_{\mathcal{D}}$ of $\boldsymbol{Z}$ and elements $\boldsymbol{v}_{\mathcal{D}}$ of $\boldsymbol{v}$ in the computation. As the Shapley kernel weights are used in the sampling, the sampled subsets are weighted equally in the new least squares problem. Note that $\mathcal{S} = \emptyset$ and $\mathcal{S} = \mathcal{M}$ are excluded from the sampling procedure. As above, their corresponding $\boldsymbol{Z}$-rows are appended to $\boldsymbol{Z}_{\mathcal{D}}$, their $v(\mathcal{S})$-elements are appended to $\boldsymbol{v}_{\mathcal{D}}$, and the diagonal matrix $\boldsymbol{W}_{\mathcal{D}}$ is extended with two diagonal elements equal to c. This procedure gives the following approximation to (6)[1]:

$$\boldsymbol{\phi} = \left[\left(\boldsymbol{Z}_{\mathcal{D}}^T\boldsymbol{W}_{\mathcal{D}}\boldsymbol{Z}_{\mathcal{D}}\right)^{-1}\boldsymbol{Z}_{\mathcal{D}}^T\boldsymbol{W}_{\mathcal{D}}\right]\boldsymbol{v}_{\mathcal{D}} = \boldsymbol{R}_{\mathcal{D}}\boldsymbol{v}_{\mathcal{D}}. \tag{7}$$

A practical consequence of using (7) as opposed to (1) is that when explaining several predictions, which typically is the case, the matrix operations producing the $(M+1)\times|\mathcal{D}|$ matrix $\boldsymbol{R}_{\mathcal{D}}$ only have to be carried out once. Provided that $\boldsymbol{v}_{\mathcal{D}}$ is pre-computed, all that is needed to explain different predictions from the same model is to perform the matrix multiplication of $\boldsymbol{R}_{\mathcal{D}}$ and $\boldsymbol{v}_{\mathcal{D}}$ for the different $\boldsymbol{v}_{\mathcal{D}}$.

2.3.2. Estimating the contribution function under feature independence

When computing the vector $\boldsymbol{v}$, we need the $v(\mathcal{S})$ values for all possible feature subsets represented by the matrix $\boldsymbol{Z}$ (or $\boldsymbol{Z}_{\mathcal{D}}$ if we use the approximation in (7)). As stated in Section 2.1, the contribution value $v(\mathcal{S})$ for a certain subset $\mathcal{S}$ is defined as

$$v(\mathcal{S}) = \mathrm{E}[f(\boldsymbol{x})|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*].$$

Let $\bar{\mathcal{S}}$ denote the complement of $\mathcal{S}$, such that $\boldsymbol{x}_{\bar{\mathcal{S}}}$ is the part of $\boldsymbol{x}$ not in $\boldsymbol{x}_{\mathcal{S}}$. Then, the expected value may be computed as follows

$$\begin{aligned} E[f(\boldsymbol{x})|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*] &= E[f(\boldsymbol{x}_{\bar{\mathcal{S}}}, \boldsymbol{x}_{\mathcal{S}})|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*] \\ &= \int f(\boldsymbol{x}_{\bar{\mathcal{S}}}, \boldsymbol{x}_{\mathcal{S}}^*)\, p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*) d\boldsymbol{x}_{\bar{\mathcal{S}}}, \end{aligned} \tag{8}$$

where $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*)$ is the conditional distribution of $\boldsymbol{x}_{\bar{\mathcal{S}}}$ given $\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*$. Hence, to be able to compute the exact $v(\mathcal{S})$ values we need the conditional distribution $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*)$, which seldom is known. In this step of the procedure, the Kernel SHAP method assumes feature independence, replacing $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*)$ by $p(\boldsymbol{x}_{\bar{\mathcal{S}}})$ in (8). Hence, by assuming feature independence, Kernel SHAP "intervenes" on the features by breaking the dependence between the features in $\mathcal{S}$ and $\bar{\mathcal{S}}$ and therefore produces interventional Shapley values.

[1] The bulk of information regarding the approximation of the WLS problem was obtained through personal communication with Scott Lundberg.

Using the training set, that is, the data used to train the model $f(\cdot)$, as the empirical distribution of $\boldsymbol{x}$, the integral in (8) can be approximated by Monte Carlo integration:

$$v_{\text{KerSHAP}}(\mathcal{S}) = \frac{1}{K}\sum_{k=1}^{K} f(\boldsymbol{x}^k_{\bar{\mathcal{S}}}, \boldsymbol{x}^*_{\mathcal{S}}), \tag{9}$$

Here, $\boldsymbol{x}^k_{\bar{\mathcal{S}}}, k = 1, \ldots, K$ are samples from the training data. Due to the independence assumption, they are sampled independently of $\boldsymbol{x}_{\mathcal{S}}$.

3. Incorporating dependence into the Kernel SHAP method

If the features in a given model are highly dependent, the Kernel SHAP method may not give a correct answer, due to predictions using non-representative data instances. As stated in Section 1, it is very rare that features in real datasets are statistically independent. The only place in the Kernel SHAP method where the independence assumption $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}}) = p(\boldsymbol{x}_{\bar{\mathcal{S}}})$ is used, is when approximating the integral in (8). Apart from this rough assumption, the Kernel SHAP framework stands out as a clever and fruitful way to approximate the Shapley values. It would therefore be desirable to incorporate dependence into the Kernel SHAP method by avoiding the independence assumption. This can be done by estimating/approximating $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}})$ directly and generate samples from this distribution, instead of generating them independently from $\boldsymbol{x}_{\mathcal{S}}$ as described in Section 2.3.2. We propose four approaches for estimating $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}})$; (i) assuming a Gaussian distribution for $p(\boldsymbol{x})$, (ii) assuming a Gaussian copula distribution for $p(\boldsymbol{x})$, (iii) approximating $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}})$ by an empirical (conditional) distribution and (iv) a combination of the empirical approach and either the Gaussian or the Gaussian copula approach.

3.1. Multivariate Gaussian distribution

If we assume that the feature vector $\boldsymbol{x}$ stems from a multivariate Gaussian distribution with some mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, the conditional distribution $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}})$ is also multivariate Gaussian. In particular, writing $p(\boldsymbol{x}) = p(\boldsymbol{x}_{\mathcal{S}}, \boldsymbol{x}_{\bar{\mathcal{S}}}) = \mathrm{N}_M(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu} = (\boldsymbol{\mu}_{\mathcal{S}}, \boldsymbol{\mu}_{\bar{\mathcal{S}}})^\top$ and

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathcal{S}\mathcal{S}} & \boldsymbol{\Sigma}_{\mathcal{S}\bar{\mathcal{S}}} \\ \boldsymbol{\Sigma}_{\bar{\mathcal{S}}\mathcal{S}} & \boldsymbol{\Sigma}_{\bar{\mathcal{S}}\bar{\mathcal{S}}} \end{bmatrix},$$

gives $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}}) = \mathrm{N}_{|\bar{\mathcal{S}}|}(\boldsymbol{\mu}_{\bar{\mathcal{S}}|\mathcal{S}}, \boldsymbol{\Sigma}_{\bar{\mathcal{S}}|\mathcal{S}})$, with

$$\boldsymbol{\mu}_{\bar{\mathcal{S}}|\mathcal{S}} = \boldsymbol{\mu}_{\bar{\mathcal{S}}} + \boldsymbol{\Sigma}_{\bar{\mathcal{S}}\mathcal{S}}\, \boldsymbol{\Sigma}^{-1}_{\mathcal{S}\mathcal{S}}(\boldsymbol{x}^*_{\mathcal{S}} - \boldsymbol{\mu}_{\mathcal{S}}) \tag{10}$$

and

$$\boldsymbol{\Sigma}_{\bar{\mathcal{S}}|\mathcal{S}} = \boldsymbol{\Sigma}_{\bar{\mathcal{S}}\bar{\mathcal{S}}} - \boldsymbol{\Sigma}_{\bar{\mathcal{S}}\mathcal{S}}\, \boldsymbol{\Sigma}^{-1}_{\mathcal{S}\mathcal{S}} \boldsymbol{\Sigma}_{\mathcal{S}\bar{\mathcal{S}}}. \tag{11}$$

Hence, instead of sampling from the marginal distribution of $\boldsymbol{x}_{\bar{\mathcal{S}}}$, we can sample from the Gaussian distribution with expectation vector and covariance matrix given by (10) and (11), where the full expectation vector $\boldsymbol{\mu}$ and the full covariance matrix $\boldsymbol{\Sigma}$ are estimated by the sample mean and covariance matrix of the training data, respectively. Using the samples $\boldsymbol{x}^k_{\bar{\mathcal{S}}}, k = 1, \ldots, K$ from the conditional distribution, the integral in (8) is finally approximated by (9).

3.2. Gaussian copula

When the features are far from multivariate Gaussian distributed, one may instead represent the marginals by their empirical distributions, and model the dependence structure by a Gaussian copula. The definition of a d-dimensional copula is a multivariate distribution, C, with uniformly distributed marginals U(0,1) on [0,1]. Sklar's theorem [29] states that every multivariate distribution F with marginals $F_1, F_2, \ldots, F_d$ can be written as

$$F(x_1, \ldots, x_d) = C(F_1(x_1), F_2(x_2), \ldots., F_d(x_d)), \tag{12}$$

for some appropriate d-dimensional copula C. In fact, the copula from (12) has the expression

$$C(u_1, \ldots, u_d) = F(F_1^{-1}(u_1), F_2^{-1}(u_2), \ldots, F_d^{-1}(u_d)),$$

where the F_j^{-1}s are the inverse distribution functions of the marginals. While other copulas may be used, the Gaussian copula has the benefit that we may use the analytical expressions for the conditionals in (10) and (11).

Assuming a Gaussian copula, we may thus use the following procedure for generating samples from $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}})$:

- Convert each marginal X_j of the feature distribution $\boldsymbol{X}$ to a Gaussian feature V_j by $V_j = \Phi^{-1}(\hat{F}_j(X_j))$, where $\hat{F}_j$ is the empirical distribution function of marginal j.
- Assume that $\boldsymbol{V}$ is distributed according to a multivariate Gaussian,[2] and sample from the conditional distribution $p(\boldsymbol{v}_{\bar{\mathcal{S}}}|\boldsymbol{v}_{\mathcal{S}} = \boldsymbol{v}^*_{\mathcal{S}})$ using the method described in Section 3.1.
- Convert the margins V_j in the conditional distribution to the original distribution using $\hat{X}_j = \hat{F}_j^{-1}(\Phi(V_j))$.

With a series of samples generated as described above, the integral in (8) is finally approximated by (9).

3.3. Empirical conditional distribution

If both the dependence structure and the marginal distributions of $\boldsymbol{x}$ are very far from the Gaussian, neither of the two aforementioned methods are expected to work very well. For such situations, we propose a non-parametric approach. The classical method for non-parametric density estimation is the kernel estimator [30], which in the decades following its introduction has been refined and developed in many directions, see for example [31–33]. However, the kernel estimator suffers greatly from the curse of dimensionality, which quickly inhibits its use in multivariate problems. Moreover, very few methods exist for the non-parametric estimation of conditional densities, especially when either $\boldsymbol{x}_{\mathcal{S}}$ or $\boldsymbol{x}_{\bar{\mathcal{S}}}$ are not one-dimensional. Finally, most kernel estimation approaches gives a non-parametric density estimate, only, while we need to be able to generate samples from the estimated distribution.

Hence, we have developed an empirical conditional approach to sample approximately from $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}^*_{\mathcal{S}})$. The method, which is motivated by the idea that samples $(\boldsymbol{x}_{\bar{\mathcal{S}}}, \boldsymbol{x}_{\mathcal{S}})$ with $\boldsymbol{x}_{\mathcal{S}}$ close to $\boldsymbol{x}^*_{\mathcal{S}}$ are informative about the conditional distribution $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}^*_{\mathcal{S}})$, consists of the following steps:

1. Compute the distance between the instance $\boldsymbol{x}^*$ to be explained and all training instances $\boldsymbol{x}^i, i = 1, \dots, n_{\text{train}}$. The distance between $\boldsymbol{x}^*$ and instance i is computed as
$$D_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^i) = \sqrt{\frac{(\boldsymbol{x}^*_{\mathcal{S}} - \boldsymbol{x}^i_{\mathcal{S}})^T \Sigma_{\mathcal{S}}^{-1} (\boldsymbol{x}^*_{\mathcal{S}} - \boldsymbol{x}^i_{\mathcal{S}})}{|\mathcal{S}|}}, \tag{13}$$
where $\Sigma_{\mathcal{S}}$ is the sample covariance matrix for the n_{train} instances of $\boldsymbol{x}_{\mathcal{S}}$. That is, when we compute the distance we only use the elements in the subset $\mathcal{S}$. Equation (13) may be viewed as a scaled version of the Mahalanobis distance [34].
2. Compute weights for all training instances $\boldsymbol{x}^i, i = 1, \dots, n_{\text{train}}$ from the distances similarly to a Gaussian distribution kernel:
$$w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^i) = \exp\left(-\frac{D_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^i)^2}{2\sigma^2}\right),$$
where σ may be viewed as a smoothing parameter or bandwidth that needs to be specified.
3. Sort the weights $w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^i)$ in increasing order, and let $\boldsymbol{x}^{[k]}$ be the training instance corresponding to the kth largest weight.
4. Approximate the integral in (8) with a weighted version of (9):
$$v_{\text{condKerSHAP}}(\mathcal{S}) = \frac{\sum_{k=1}^{K} w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^{[k]}) f(\boldsymbol{x}^{[k]}_{\bar{\mathcal{S}}}, \boldsymbol{x}^*_{\mathcal{S}})}{\sum_{k=1}^{K} w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^{[k]})}. \tag{14}$$

Note that we could have used (9), with the $\boldsymbol{x}^k_{\bar{\mathcal{S}}}$ sampled (with replacement) from the training data with weights $w_{\mathcal{S}}(\boldsymbol{x}^i), i = 1, \dots, n_{\text{train}}$. Our approach is, however, more sampling effective as it uses each training observation only once, and uses their weights in the integral computation, rather than as input for the sampling only.

The number of samples K to be used in the approximate prediction in step 4 can for instance be chosen such that most of the total sum of weights is accounted for by the K largest weights:
$$K = \min_{L \in \mathbb{N}} \left\{ \frac{\sum_{k=1}^{L} w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^{[k]})}{\sum_{i=1}^{n_{\text{train}}} w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^i)} > \eta \right\}, \tag{15}$$
where η is set to for instance 0.9. If K in (15) exceeds a certain limit, for instance 5,000, it might be set to that limit.

Essentially all kernel based estimation procedures (such as kernel density estimation) require selection of one or more bandwidth parameters. Our method is no exception. The choice of the bandwidth parameter σ may be viewed as a bias-variance trade-off. A small σ puts most of the weight to a few of the closest training observations and thereby gives low

[2] The quality of this assumption will depend on how close the Gaussian copula is to the true copula.

bias, but high variance. A large σ spreads the weight to a higher number of (more distant) training observations and thereby gives high bias, but low variance. Typically, when the features are highly dependent, a small σ is needed such that the bias does not dominate. When the features are essentially independent, there is no bias for any σ and a larger σ is preferable. As $\sigma \to \infty$ our method approximates the original Kernel SHAP method in Section 2.3.2.

By viewing the estimation of $E[f(\boldsymbol{x})|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}}]$ as a regression problem with response $f(\boldsymbol{x}^i_{\bar{\mathcal{S}}}, \boldsymbol{x}^*_{\mathcal{S}})$ and covariates $\boldsymbol{x}^i_{\mathcal{S}}, i = 1, \ldots, n_{\text{train}}$, it turns out that our empirical conditional distribution approach (with $K = n_{\text{train}}$) is equivalent to the Nadaraya-Watson estimator [35]. Hurvich et al. [36] have developed a small-sample-size corrected version of Akaike information criterion (AICc) to select bandwidth parameters in such nonparametric regression problems. The connection to the Nadaraya-Watson estimator allows us to apply the AICc directly to select σ.

The strategy used in AICc to find a suitable smoothing parameter is to choose the σ which is the minimizer of

$$\text{AICc} = \log(\hat{\tau}^2) + \Phi(H), \tag{16}$$

where

$$\hat{\tau}^2 = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \left(f(\boldsymbol{x}^i_{\bar{\mathcal{S}}}, \boldsymbol{x}^*_{\mathcal{S}}) - \frac{\sum_{j=1}^{n_{\text{train}}} w_{\mathcal{S}}(\boldsymbol{x}^j, \boldsymbol{x}^i) f(\boldsymbol{x}^j_{\bar{\mathcal{S}}}, \boldsymbol{x}^*_{\mathcal{S}})}{\sum_{j=1}^{n_{\text{train}}} w_{\mathcal{S}}(\boldsymbol{x}^j, \boldsymbol{x}^i)} \right)^2,$$

and

$$\Phi(H) = \frac{1 + \text{tr}(H)/n_{\text{train}}}{1 - (\text{tr}(H) + 2)/2}.$$

Here, H is the $n_{\text{train}} \times n_{\text{train}}$ matrix with indexes

$$H_{i,j} = \frac{w_{\mathcal{S}}(\boldsymbol{x}^j, \boldsymbol{x}^i)}{\sum_{l=1}^{n_{\text{train}}} w_{\mathcal{S}}(\boldsymbol{x}^l, \boldsymbol{x}^i)}$$

commonly called the smoother or hat matrix, and $\text{tr}(H)$ is the trace of H.

Thus, to select σ we compute the AICc in (16) for various σ values and select the σ corresponding to the smallest AICc value. This should be done for all subsets $\mathcal{S}$ and every new observation $\boldsymbol{x}^*$ to be explained, meaning that it is a computationally intensive approach. To reduce the computational burden, we have experimented with different approximations, and ended up with one where σ is assumed to have the same value for all subsets $\mathcal{S}$ of the same size $|\mathcal{S}|$. In this approach, which is denoted the approximate AICc method in Section 4, the sum of the AICc values for all subsets of the same size is minimized instead of the AICc values for each subset.

Even the approximate AICc method is quite time consuming. Hence, in Section 4 we have also experimented with a fixed $\sigma = 0.1$ for all subsets $\mathcal{S}$.

3.4. A combined approach

When performing the experiments to be described in Section 4, it turned out that the empirical conditional distribution method works very well if the dimension of $\boldsymbol{x}_{\mathcal{S}} \le D^*$, where D^* is a small number, while it is outperformed by the multivariate Gaussian method and the Gaussian copula method if we condition on more features. This is in accordance with previous literature, see for instance [37] and references therein. Very few papers attempt to estimate $f(\boldsymbol{z}|\boldsymbol{x})$ when $\boldsymbol{x}$ has more than $D = 3$ dimensions, even if $\boldsymbol{z}$ is one-dimensional. In higher dimensions, the previously proposed methods typically rely on a prior dimension reduction step, which can result in significant loss of information. Hence, it might be wise to combine the empirical approach with either the multivariate Gaussian or the Gaussian copula approach, simulating the conditional distributions for which $\dim(\boldsymbol{x}_{\mathcal{S}}) \le D^*$ using the empirical method, and all other conditional distributions using the parametric method. In this combined approach we at least partly avoid the curse of dimensionality, since the empirical approach is used only when conditioning on a few features and the conditional distributions can be analytically computed for the Gaussian approach. To determine D^*, one may use for instance cross validation. In our experiments we have, however, determined D^* based on experience.

4. Experiments

A problem with evaluating prediction explanation methods is that there generally is no ground truth. Hence, to verify that our approaches are more accurate than the original Kernel SHAP method described in Section 2.3.2 when we have dependent features, we have to turn to simulated data for which we may compute the true Shapley values.

As computing the exact Shapley values for a single prediction requires solving $O(2^M)$ integrals of the type in (8), which are of dimension 1 to $M - 1$, we cannot perform accurate experiments in high dimensional settings. We will however perform experiments in a low dimensional ($M = 3$) and moderate dimensional ($M = 10$) setting, using various multivariate distributions for the features $\boldsymbol{x}$, sampling models for $y|\boldsymbol{x}$, and forms of the predictive model $f(\cdot)$. The experiments with

$M = 3$ and $M = 10$ are treated in Sections 4.2 and 4.3, respectively. Due to the low/moderate dimension of the features in these experiments, it is computationally tractable to use the exact version of Kernel SHAP in (6). Hence, we do not have to turn to the approximation in (7).

As the AICc dependent approximation methods are directly dependent on the sampled training set, we run the experiments in 10 batches. In each batch, we sample a new training set of size $n_{\text{train}} = 2,000$, and use the model fitted to those training data to explain predictions in a test set of size 100. This means that the quality of the conditional expectation approximations is measured based on a total of $n_{\text{test}} = 10 \cdot 100 = 1,000$ test observations. Sampling new training data for each batch also reduces the influence of the exact form of the fitted predictive model, compared to using a single training set across all simulations.

The Shapley value approximations we compare with the original Kernel SHAP method (original) are:

- The Kernel SHAP with the Gaussian conditional distribution (Gaussian)
- The Kernel SHAP with the Gaussian copula and empirical margins (copula)
- The Kernel SHAP with the empirical conditional distribution determining σ with exact AICc (empirical-AICc-exact)
- The Kernel SHAP with the empirical conditional distribution determining σ with approximate AICc (empirical-AICc-approx)
- The Kernel SHAP with the empirical conditional distribution setting $\sigma = 0.1$ for all conditional distributions (empirical-0.1)
- The Kernel SHAP with the combined approach using the empirical approach for subsets with dimension ≤ 3 and the Gaussian approach otherwise (empirical-0.1+Gaussian and empirical-AICc-approx+Gaussian)
- The Kernel SHAP with the combined approach using the empirical approach for subsets with dimension ≤ 3 and the copula approach otherwise (empirical-0.1+copula and empirical-AICc-approx+copula)

Due to the computational complexity, the empirical-AICc-exact method is only used in the 3 dimensional experiments. Furthermore, the combined approaches are only used in the 10 dimensional experiments. For the experiments where XGBoost is used to fit the predictive model, we will also include the so-called TreeSHAP method [25] in the comparison.

4.1. Evaluation measures

To quantify the accuracy of the different methods, we rely on the mean absolute error (MAE) of the Shapley value approximations, averaged over all features and all test samples, that is

$$\text{MAE}(\text{method } q) = \frac{1}{M n_{\text{test}}} \sum_{j=1}^{M} \sum_{i=1}^{n_{\text{test}}} |\phi_{j,\text{true}}^{(i)} - \phi_{j,q}^{(i)}|,$$

where $\phi_{j,q}^{(i)}$ denotes the Shapley value of feature j, for prediction i, and computed with approximation method q, while $\phi_{j,\text{true}}^{(i)}$ is the corresponding true value. In order to determine the superiority of the various proposed methods over the original Kernel SHAP method, we rely on the so-called skill score [38] associated with the aforementioned MAE. The skill score for method q takes the form

$$\begin{aligned}\text{Skill}(\text{MAE, method } q) &= \frac{\text{MAE}(\text{method } q) - \text{MAE}(\text{original})}{\text{MAE}(\text{optimal}) - \text{MAE}(\text{original})} \\ &= 1 - \frac{\text{MAE}(\text{method } q)}{\text{MAE}(\text{original})},\end{aligned}$$

where MAE(optimal) is the MAE of an optimal method, being equal to zero. The skill score measures the superiority of a method compared to the reference method (here the original Kernel SHAP). It is standardized in such a way that it takes the value 1 for a perfect approximation and 0 for the reference method. When the approximation method is worse than the reference method, the skill score becomes negative.

To compare the methods on equal terms, all methods are restricted to use $K = 1,000$ samples from the training set for each feature combination and test observation.

4.2. Dimension 3

For the three dimensional setting, we will use two different sampling models for $y|\boldsymbol{x}$ (linear and piecewise constant), and combine these with three different multivariate distributions for the features $\boldsymbol{x}$ (Gaussian, Generalized Hyperbolic distribution and Gaussian mixture), such that we get a total of six experimental setups A-F. In Sections 4.2.1–4.2.3 we describe the multivariate feature distributions, while the sampling models are discussed in Sections 4.2.4 and 4.2.5. Finally, Section 4.2.6 contains the results.

The noise term ε_i is common for all experiments and assumed to follow the distribution $\varepsilon_i \overset{\text{d.}}{=} \varepsilon \sim \text{N}(0, 0.1^2)$. Due to the low dimension, the exact Shapley value in (8) can be computed using numerical integration.

4.2.1. *Gaussian distributed features*

The first feature distribution $p(\boldsymbol{x})$ we shall consider is a multivariate Gaussian distribution $p(\boldsymbol{x}) = \mathrm{N}_3(\mathbf{0}, \boldsymbol{\Sigma}(\rho))$, where the covariance matrix $\boldsymbol{\Sigma}(\rho)$ takes the form

$$\boldsymbol{\Sigma}(\rho) = \begin{bmatrix} 1 & \rho & \rho \\ \rho & 1 & \rho \\ \rho & \rho & 1 \end{bmatrix}. \tag{17}$$

In the various experiments, the correlation coefficient ρ varies between 0 and 0.98, representing an increasing positive correlation among the features.

4.2.2. *Skewed and heavy-tailed distributed features*

The second feature distribution $p(\boldsymbol{x})$ to be considered is the Generalized Hyperbolic(GH)-distribution. Following [39], a random vector $\boldsymbol{X}$ is said to follow a GH-distribution with index parameter λ, concentration parameter ω, location vector $\boldsymbol{\mu}$, dispersion matrix $\boldsymbol{\Sigma}$, and skewness vector $\boldsymbol{\beta}$, denoted by $\boldsymbol{X} \sim \mathrm{GH}(\lambda, \omega, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\beta})$, if it can be represented by

$$\boldsymbol{X} = \boldsymbol{\mu} + W\,\boldsymbol{\beta} + \sqrt{W}\,\boldsymbol{U},$$

where $W \sim \mathrm{GIG}(\lambda, \omega, \omega)$, $\boldsymbol{U} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$ and W is independent of $\boldsymbol{U}$. GIG is the Generalized Inverse Gaussian distribution introduced by [40]. Appendix C contains more details on this distribution. We use the following parameter values in our experiments:

$$\begin{aligned}
&\lambda = 1\\
&\omega = 0.5\\
&\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(0) = \mathrm{diag}(\mathbf{1})\\
&\boldsymbol{\beta} = (1/4)\kappa * \mathbf{1}\\
&\boldsymbol{\mu} = \mathbf{0} - \mathrm{E}[W](1/4)\kappa,
\end{aligned}$$

where the skewness coefficient κ varies from 1 to 10 in different experiments, resulting in an increasingly more skewed, heavy-tailed and positively correlated distribution. $\mathrm{E}[W]$ denotes the mean of the GIG distribution, equal to approximately 4.56 for the above parameter values. The special form of $\boldsymbol{\mu}$ is chosen such that the GH-distribution has mean zero.

4.2.3. *Multi-modal distributed features*

The last feature distribution is a mixture of two Gaussian distributions with different means. That is,

$$p(\boldsymbol{x}) = \sum_{k=1}^{2} \pi_k \mathrm{N}(\boldsymbol{\mu}_k(\gamma), \boldsymbol{\Sigma}(0.2)),$$

with mixture probabilities $\pi_1 = \pi_2 = 0.5$ and mean functions $\boldsymbol{\mu}_1(\gamma) = -\boldsymbol{\mu}_2(\gamma) = \gamma \cdot (1, -0.5, 1)^\top$. The covariance matrix $\boldsymbol{\Sigma}$ is assumed to be on the form in (17). The γ parameter represents the distance between the two Gaussian distributions, and will range from 0.5 to 10 in the different experiments.

4.2.4. *Linear model*

The first sampling model $y|\boldsymbol{x}$ is a simple linear model:

$$y = g(\boldsymbol{x}) = x_1 + x_2 + x_3 + \varepsilon.$$

Data from such a distribution will be modeled by fitting the parameters $\beta_j, j = 0, \ldots, 3$ in the linear model

$$f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon,$$

using ordinary linear regression.

4.2.5. *Piecewise constant model*

The second sampling model $y|\boldsymbol{x}$ is a piecewise constant model constructed by summing three different piecewise constant functions:

$$y = g(\boldsymbol{x}) = \mathrm{fun}_1(x_1) + \mathrm{fun}_2(x_2) + \mathrm{fun}_3(x_3) + \varepsilon.$$

The functions fun_1, fun_2, and fun_3 are displayed in Fig. 1. We fit the model $g(\boldsymbol{x})$ with the XGBoost framework [26] using default values of all hyperparameters, the histogram tree learning method and 50 boosting iterations.

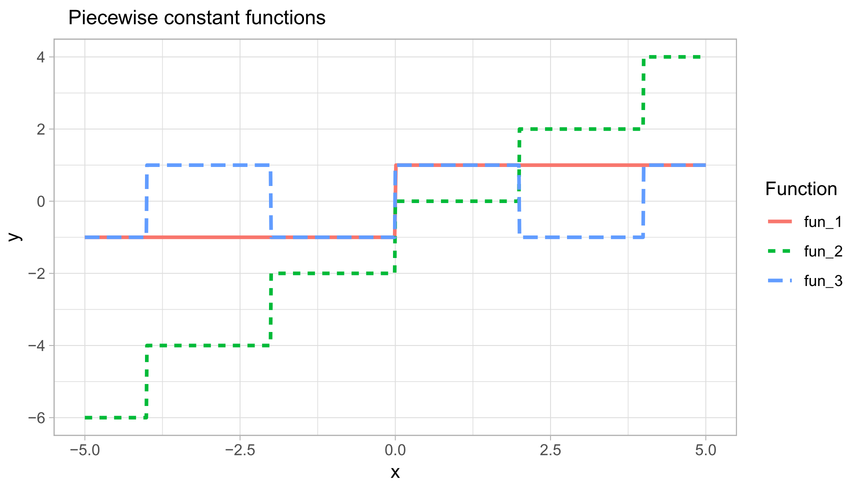

Fig. 1. Piecewise constant functions used in experiment D, E and F. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

4.2.6. Results

The results from the 3D-experiments are visualized in Figs. 2 and 3. In experiment A we have used a linear sampling model and Gaussian features. As seen from the upper row of Fig. 2, the original Kernel SHAP method works well when the features are independent, but it is outperformed by all other methods when ρ is greater than 0.05. The Gaussian model generally shows the best performance. It should also be noted that the AICc method for determining σ works better than using the fixed value of 0.1.

In experiment B, we still have the linear sampling model, but now we have skewed and heavy-tailed features. Like for the previous experiment, the original Kernel SHAP method is outperformed by all other approaches. As shown in the middle row of Fig. 2, when κ increases, the MAE values of the copula and Gaussian methods are reduced. This might be due to the increased variance of the features that comes as a by-product of increasing the κ parameter. For this experiment, the empirical approach with a fixed $\sigma = 0.1$ performs uniformly better than those based on AICc.

In experiment C, we have the same sampling model, but now with bimodal Gaussian mixture distributed features. Again, all our methods perform uniformly better than the original Kernel SHAP method. The lower row of Fig. 2 further shows that the empirical methods outperform the Gaussian and copula methods, especially when γ (that is, the distance between the modes of the feature distribution) is large.

In experiments D-F we use the same feature distributions as in A-C, but for these experiments we use a piecewise constant model instead of the linear. The results are largely the same as in the linear model case. The original Kernel SHAP method is outperformed by all other approaches. Further the Gaussian model is best when we have Gaussian features, while the empirical approaches are best when the feature distribution is bi-modal. In the case with skewed and heavy-tailed features, the Gaussian and copula methods again seem to be preferable for larger values of κ.

For the experiments with the piecewise constant model, we have used the TreeSHAP method in addition to the other ones. As shown in Fig. 3, the performance of this method is just slightly better than that of the original kernel SHAP method for experiments D and E, and worse in experiment F. This is surprising, since the TreeSHAP method is supposed to handle dependence better than the original kernel SHAP method.

Overall, our 3 dimensional experiments clearly show that it is important to account for the dependence between the features when computing the Shapley values. Which of the suggested methods that is best, depends on the underlying feature distribution. Further, since the results for our empirical method using the approximate AICc version are fairly similar (and in some cases even better) to those using the slower exact version, we recommend the former.

4.3. Dimension 10

In the 10 dimensional case we have restricted ourselves to three types of experiments. In the first two, we use Gaussian distributed features on the same form as described for the 3 dimensional case in Section 4.2.1. That is, $p(\boldsymbol{x}) = \mathrm{N}_{10}(\mathbf{0}, \boldsymbol{\Sigma}(\rho))$, where $\boldsymbol{\Sigma}(\rho)$ is the 10 dimensional extension of (17) such that all features have variance 1 and the pairwise correlation between any two features is ρ.

The first experiment uses a sampling model $y|\boldsymbol{x}$ directly extending the 3 dimensional linear model in Section 4.2.4, that is,

$$y = g(\boldsymbol{x}) = \sum_{j=1}^{9} x_j + \varepsilon,$$

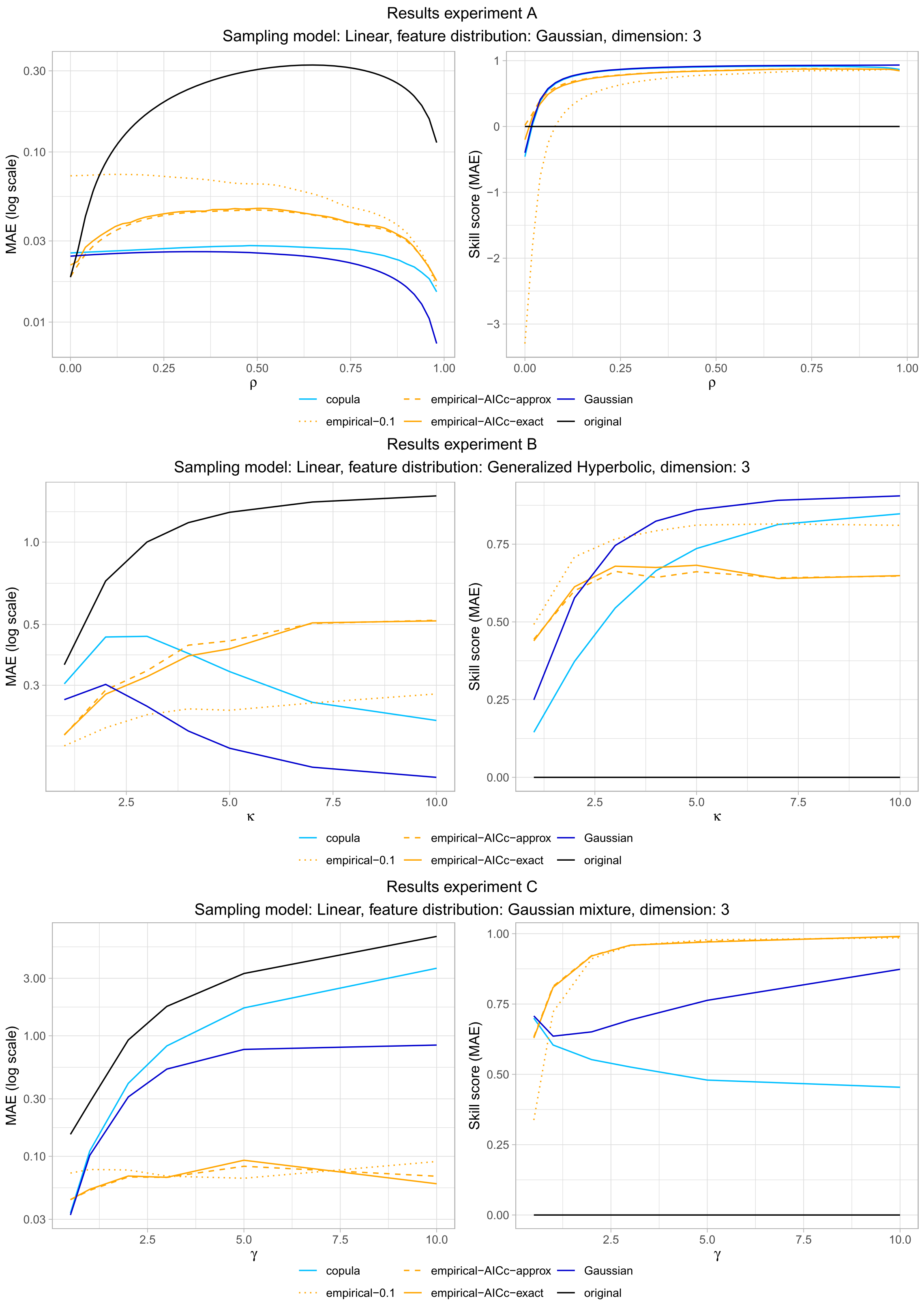

Fig. 2. Dimension 3: MAE and skill score for the **linear model**. Upper row: Gaussian features. Middle row: GH-distributed features. Lower row: Gaussian mixture distributed features.

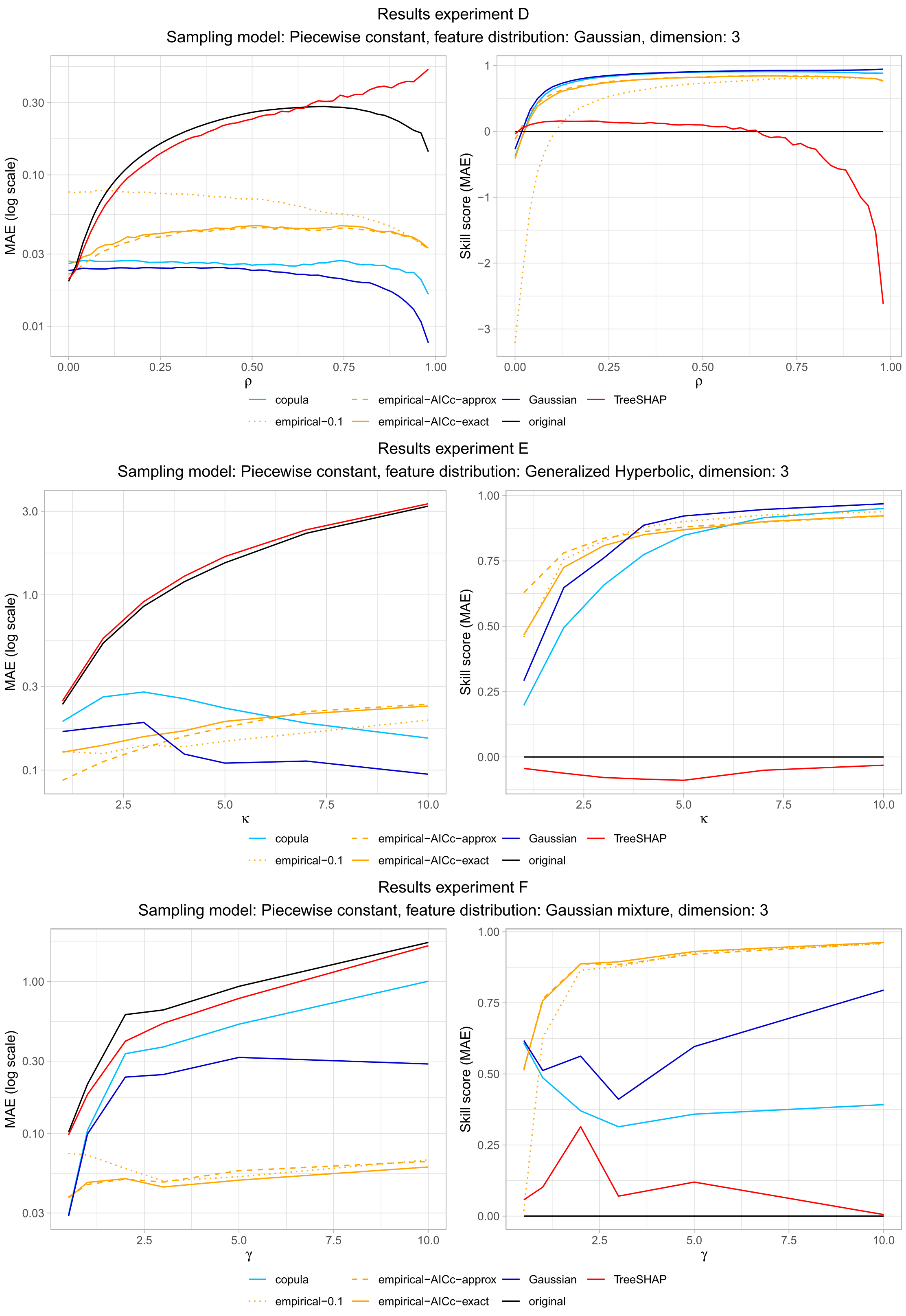

Fig. 3. Dimension 3: MAE and skill score for the **piecewise constant model**. Upper row: Gaussian features. Middle row: GH-distributed features. Lower row: Gaussian mixture distributed features.

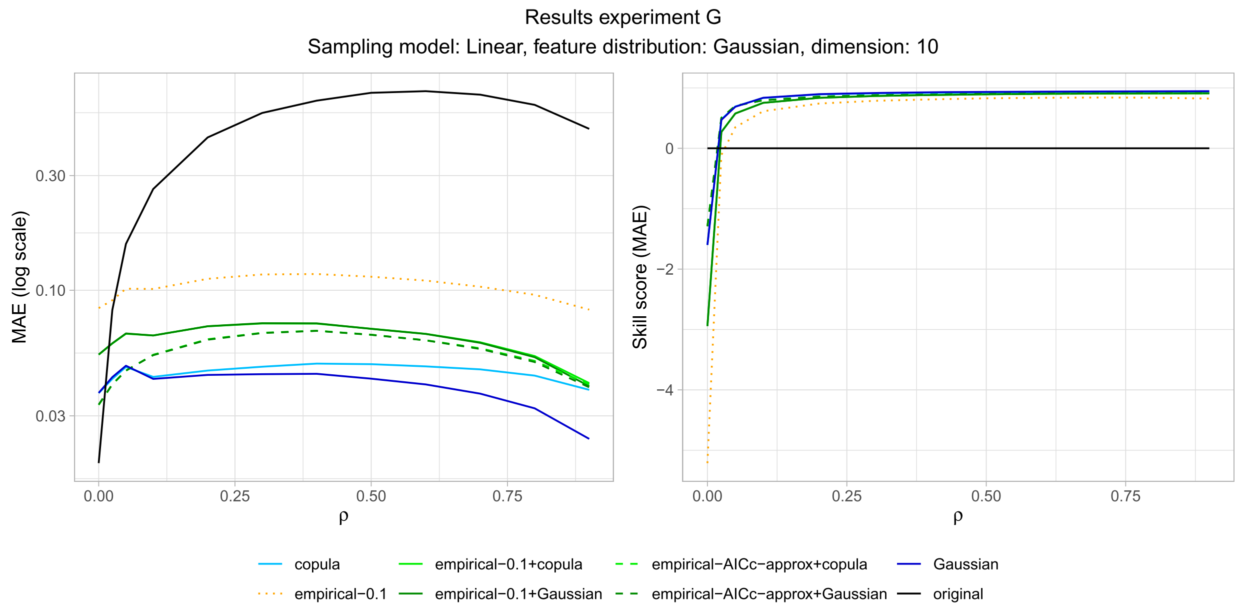

Fig. 4. Dimension 10: MAE and skill score for linear model with Gaussian distributed features.

where we use the same error term $\varepsilon_i \overset{d.}{=} \varepsilon \sim N(0, 0.1^2)$ as in the 3 dimensional experiments. Analogous to the 3 dimensional case, y is modeled by ordinary linear regression

$$f(\boldsymbol{x}) = \sum_{j=1}^{10} \beta_j x_j + \varepsilon.$$

The second experiment extends the 3 dimensional experiment described in Section 4.2.5, taking the form

$$y = g(\boldsymbol{x}) = \sum_{j\in\{1,2,3\}} \text{fun}_1(x_j) + \sum_{j\in\{4,5,6\}} \text{fun}_2(x_j) + \sum_{j\in\{7,8,9\}} \text{fun}_3(x_j) + \varepsilon,$$

again excluding the effect of the 10th feature. As for the 3 dimensional case, the model $g(\boldsymbol{x})$ is fitted using the XGBoost framework, using the same settings. Note that for both the above settings, feature 10 has no *direct* influence on y.

In the last experiment, we simulate data from a 10 dimensional GH-distribution with the following parameter values

$$\lambda = 1$$
$$\omega = 0.5$$
$$\boldsymbol{\mu} = (3, 3, 3, 3, 3, 3, 3, 3, 3, 3)$$
$$\boldsymbol{\Sigma} = \text{diag}(1, 2, 3, 1, 2, 3, 1, 2, 3, 3)$$
$$\boldsymbol{\beta} = (1, 1, 1, 1, 1, 0.5, 0.5, 0.5, 0.5, 0.5).$$

As sampling model, we use the piecewise constant model described above.

We didn't use the empirical-AICc-exact approach for any of the 10 dimensional experiments. As previously stated, this approach is computationally intensive. Moreover, the 3 dimensional experiments showed that the performance of the empirical-AICc-exact and the empirical-AICc-approx approaches were very similar. For the 3 dimensional experiment with the Generalized Hyperbolic features and piecewise constant model, the MAE and skill scores for the three empirical approaches were almost equal, meaning that it is not necessary to use the significantly more computational heavy AICc approach. Hence, the only empirical approach used in the last experiment was the empirical-0.1 method. In the combined approaches, we use the empirical approach for subsets with dimension ≤ 3, with the bandwidth parameter σ either determined by the approximate AICc method (in the linear case, only) or fixed to 0.1.

The results from the simulation experiments are shown in Figs. 4 and 5 and Table 1. While numerical integration was used to compute the exact Shapley values in the 3 dimensional experiments, we have to turn to Monte Carlo integration for the 10 dimensional case. That is, we use (9) even when computing the true Shapley value, but using samples from the true conditional distribution instead of samples from the estimated conditional distribution.

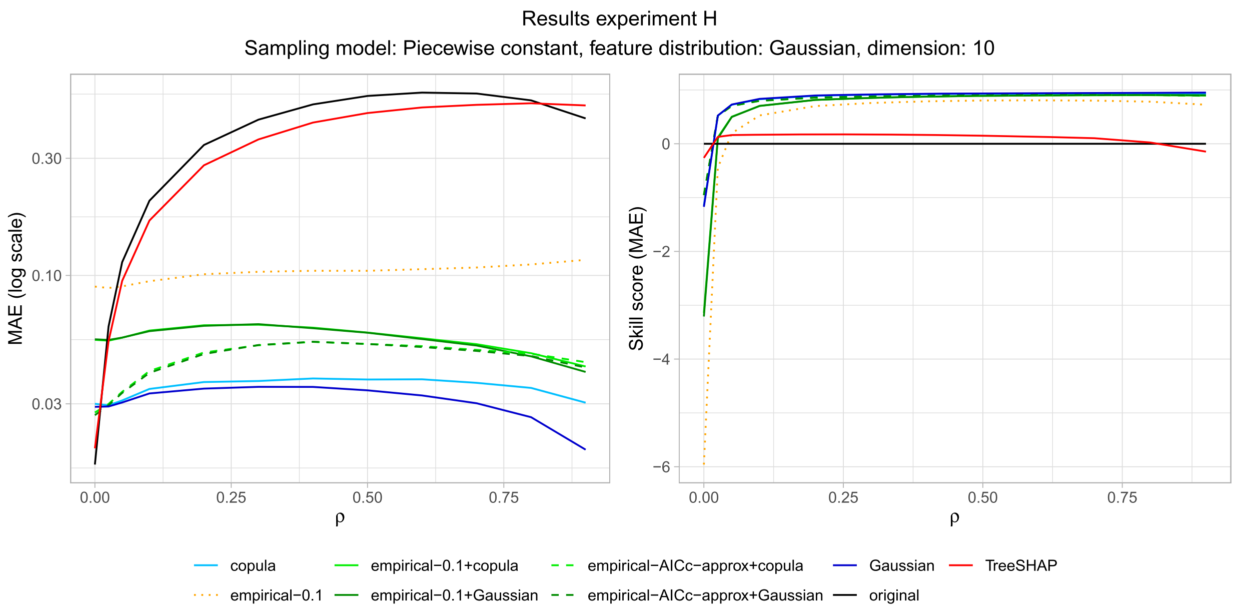

Fig. 5. Dimension 10: MAE and skill score for piecewise constant model with Gaussian distributed features.

Table 1
Dimension 10: MAE and skill score for piecewise constant model with GH-distributed features.

Approach	MAE	Skill score
original	1.182	0.000
Gaussian	0.377	0.633
copula	0.526	0.504
empirical-0.1	0.307	0.737
empirical-0.1+Gaussian	0.199	0.821
empirical-0.1+Copula	0.236	0.791
TreeSHAP	1.181	0.014

From the figures, we see that the results with Gaussian distributed features in dimension 10 are mostly the same as for the 3 dimensional counterparts in Fig. 2, with the Gaussian method generally showing the best performance. The combined empirical and Gaussian/copula approaches also work well. For the piecewise constant model, the TreeSHAP method behaves as in the 3 dimensional case: slightly better than the original kernel SHAP method for small and medium sized dependence, but worse when there is high dependence between the features.

For the skewed and heavy-tailed data, Table 1 shows that the empirical-0.1+Gaussian method gives the best performance, having slightly better MAE values and skill scores than the empirical-0.1+copula method. Finally, like for the other experiments, all our suggested approaches outperform the original Kernel SHAP and the TreeSHAP method.

4.4. Real data example

In our last example, we use a real data set. The data set consists of 28 features extracted from 6 transaction time series. It has previously been used for predicting mortgage default, relating probability of default to transaction information [3]. The transaction information consists of the daily balances on consumers' credit (`kk`), checking (`br`), and savings (`sk`) accounts, in addition to the daily number of transactions on the checking account (`tr`), the amount transferred into the checking account (`inn`), and the sum of the checking, savings, and credit card accounts (`sum`). For each of these time series, which were of length 365 days, the mean (`mean`), maximum (`max`), minimum (`min`), standard deviation (`std`), and the coefficient of variation (`cv`) were computed, resulting in 28 features. These are scaled to have mean 0 and standard deviation 1 before they are used in our computations.

Fig. 6 shows histograms for nine of the features (the remaining features have similar distributions). The feature distributions are skewed and heavy-tailed. The pairwise rank correlations measured by Kendall's τ [41] are shown in Fig. 7. We use Kendall's τ instead of the linear correlation coefficient because our variables are far from linearly dependent, see Fig. 9 for some examples. Most rank correlations in Fig. 7 are close to 0, but there are groups of features with high mutual correlations. Hence, we expect our approach to give more accurate approximations to the true Shapley values than the original Kernel SHAP method since the latter assumes independent features.

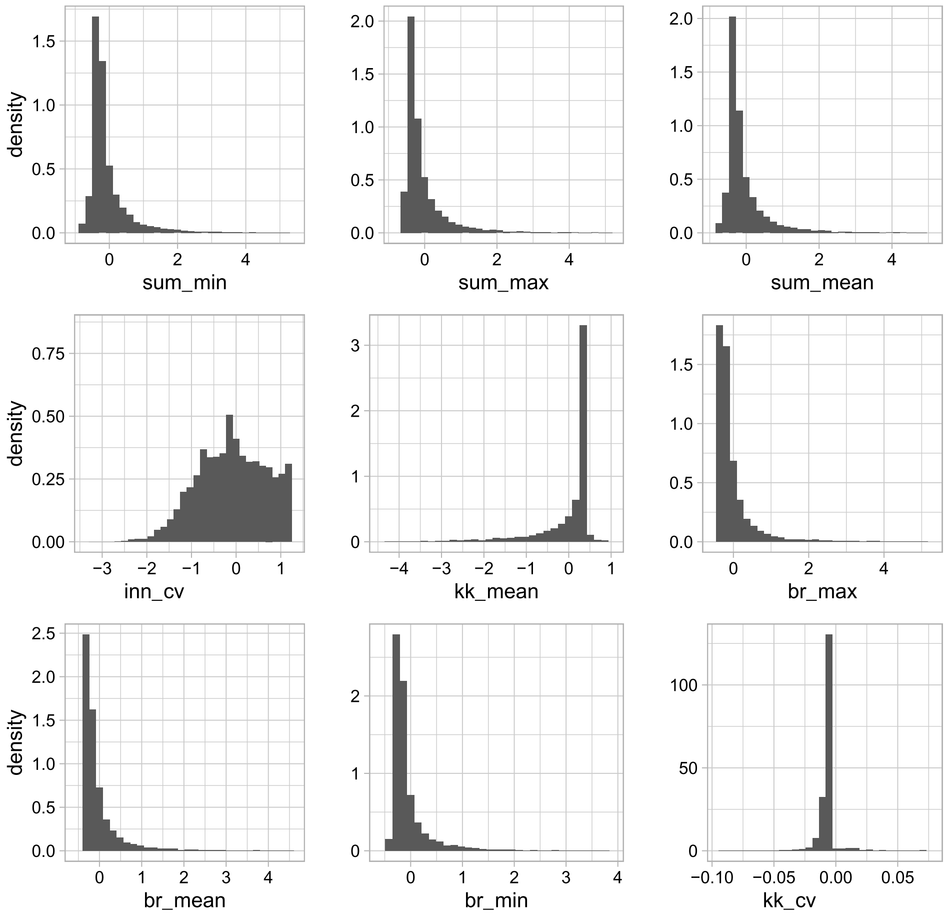

Fig. 6. Histograms for nine of the variables in the real data set.

The data set was divided into a training set and a test set, containing 12,696 and 1,921 observations respectively. We fitted an XGBoost model with 50 trees to the training data, using default parameter settings. The resulting AUC for the test data was 0.88.

The last example in Section 4.3 showed that for the case with skewed and heavy-tailed features and a piecewise constant model, the combined approaches were superior to the other, with the empirical-0.1+Gaussian approach as the best performing method. We assume that this is also the case for the real data set, and compare the performance of this method with the original Kernel SHAP approach.

Fig. 8 shows the Shapley-values for two of the individuals in the data set. The Shapley values are quite different. For individual A, there are e.g. large differences for the variables `br_mean`, `br_min` and `br_max` and for the variables `sum_min`, `sum_mean` and `sum_max`. From the Kendall's τ matrix in Fig. 7 we see that all these variables are positively correlated with many other variables. Hence, it is not surprising that the Shapley values for the two methods are different. For individual B, we also observe large differences for the variables `sum_min`, `sum_mean` and `sum_max`. In addition, it is worth noticing that the Shapley values for variable `kk_cv` have opposite signs. This variable is strongly negatively correlated to the variable `kk_std`. In addition it is positively correlated to `kk_min`, `kk_mean` and `kk_max`.

As previously stated, a problem with evaluating Shapley values for real data is that there is no ground truth. Hence, we have to justify the results in other ways. As shown in (1), the Shapley value is a weighted sum of differences $v(\mathcal{S} \cup \{j\}) - v(\mathcal{S})$ for several subsets $\mathcal{S}$. Both in our method and in the original Kernel SHAP method $v(\mathcal{S})$ is estimated by (9). The two methods differ, however, in that the original Kernel SHAP method assumes feature independence when generating samples from the conditional distribution $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}})$. Hence, if we are able to show that the samples from the conditional distributions generated using our method are more representative than the samples generated using the original Kernel SHAP method, it is likely that the Shapley values obtained using our method are more accurate than those obtained using the original Kernel SHAP method.

Since there are very many conditional distributions involved in the Shapley formula when we have 28 variables, it is impossible to show all here. However, we have included some examples that illustrate that our method gives more correct approximations to the true conditional distributions than the original Kernel SHAP approach. First, Fig. 9 shows plots of

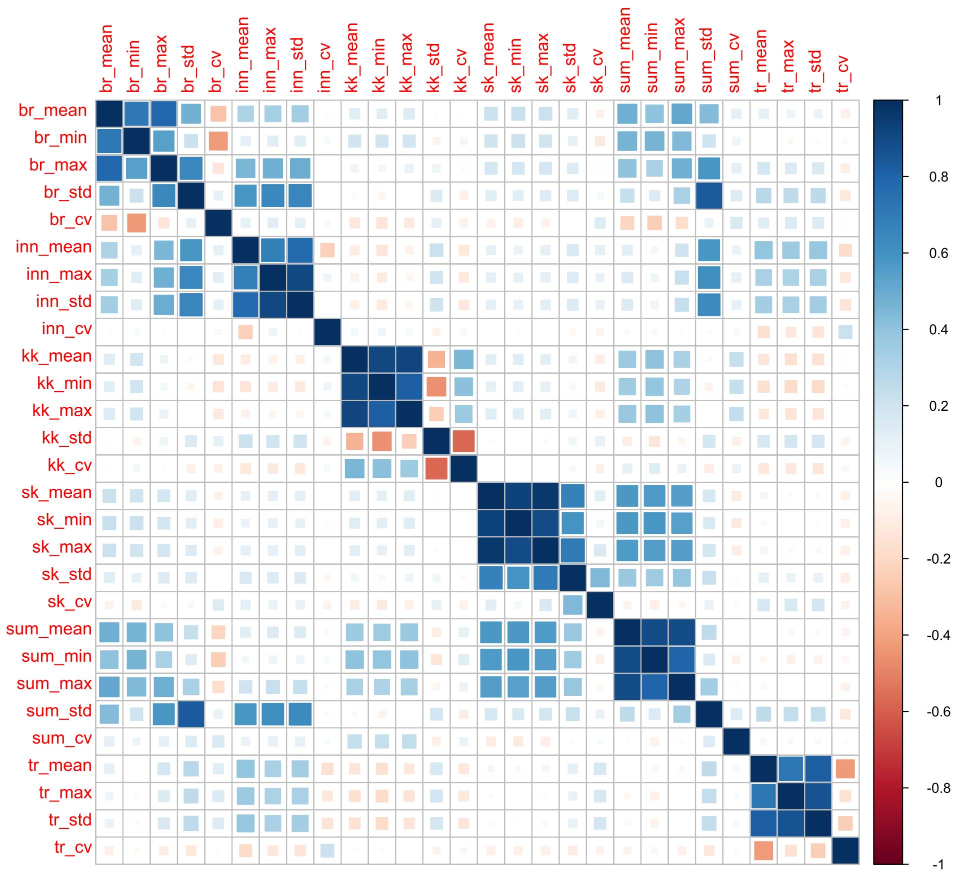

Fig. 7. Kendall's τ correlation matrix for the real data set.

`br_min` against `br_max` and `br_std` against `br_max`. The black dots are the training data. The turquoise dots are the samples from the conditional distribution of the variable at the x-axis given that `br_max` is equal to zero generated using our method, while the red dots are the corresponding samples generated using the original Kernel SHAP approach. The original Kernel SHAP approach generates samples that are unrealistic, in the sense that they are far outside the range of what is observed in the training data. It is well known that evaluation of predictive machine learning models far from the domain at which they have been trained, can lead to spurious predictions. Thus, it is important that the explanation methods are evaluating the predictive model at appropriate feature combinations. The samples generated by our method are inside the range of what is observed in the training data.

Further, in Fig. 10 we study three different conditional distributions involved in the Shapley formula:

- The conditional distribution of `sum_min`, `sum_mean` and `sum_max` given all the other variables.
- The conditional distribution of all variables except `inn_cv` given `inn_cv`.
- The conditional distribution of all variables except `kk_mean` and `br_max` given `kk_mean` and `br_max`.

For all the three distributions, we generate 1,000 samples for four of the individuals in the test data set using our method and the original Kernel SHAP approach. That is, we condition on four different sets of values. For each combination of individual, conditional distribution and method, we compute the mean Mahalanobis distance between each sample and its ten nearest training samples, resulting in 1,000 different mean distances. Each panel of Fig. 10 shows the probability density of such mean distances after scaling the mean distances with the average for the original method. Hence, the distances are sized relative to the mean distance of the original method. If the generated samples are realistic, we would expect the majority of the mean distances to be small.

Starting with the leftmost column, the mode of the density corresponding to our method is smaller that corresponding to the original Kernel SHAP approach. This indicates that the samples generated by our approach are more realistic than those generated by the original Kernel SHAP approach. From the Kendall's τ matrix in Fig. 7 we see that the variables `sum_min`, `sum_mean` and `sum_max` are positively correlated with many of the other variables. Hence, it makes sense that our method provides a better estimate of this conditional distribution.

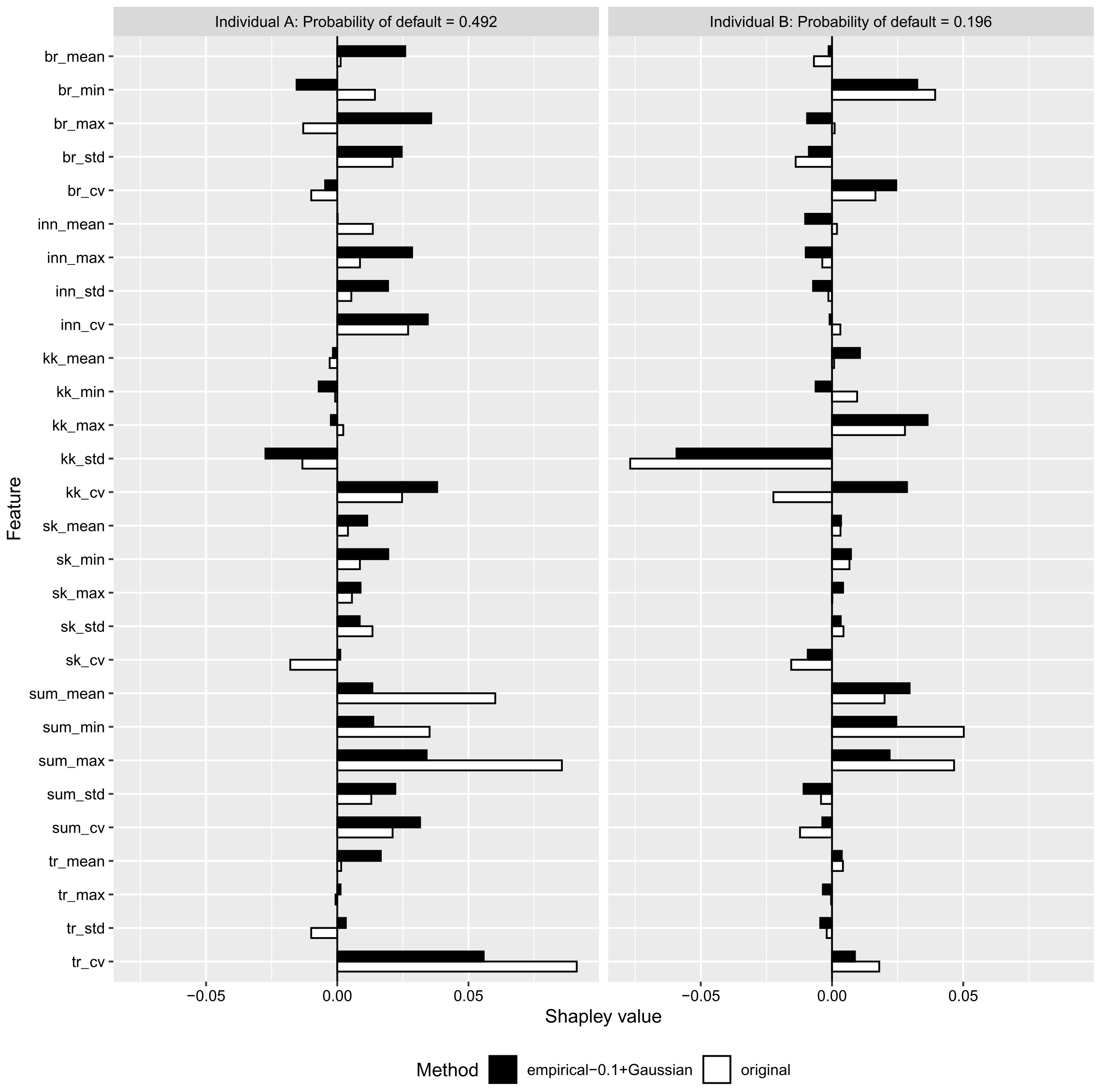

Fig. 8. Shapley values for two persons in the real data set computed using our method and the original Kernel SHAP method.

If we proceed to the second column, the densities for the two approaches are very similar. This is not surprising, since, as shown in Fig. 7, the variable `inn_cv` is almost uncorrelated with all other variables.

Finally, in the rightmost column, the mode of the density corresponding to our method is again smaller than that corresponding to the original Kernel SHAP approach, but the differences are not as large as those in the first column. We believe that this can be explained by the fact that one of the variables we condition on, `kk_mean`, is strongly correlated to only a few of the other variables.

To summarize, we have illustrated that the Shapley values computed using our method and the original method are different. We have tried to justify that this is due to the fact that our method gives more correct approximations to the true conditional distributions than the original Kernel SHAP approach.

5. Summary and discussion

Shapley values is a model-agnostic method for explaining individual predictions with a solid theoretical foundation. The main disadvantage with this method is that the computational complexity grows exponentially with the number of features. This has led to approximations, of which the Kernel SHAP method is the most known. A key ingredient of the Kernel SHAP method is the conditional distribution of a subset $\bar{\mathcal{S}}$ of the features, conditional on the features in $\mathcal{S}$ that are not in this subset. In the original version of the Kernel SHAP method it is implicitly assumed that the features in the two subsets are independent, meaning that the conditional distribution may be replaced by the marginal distribution of

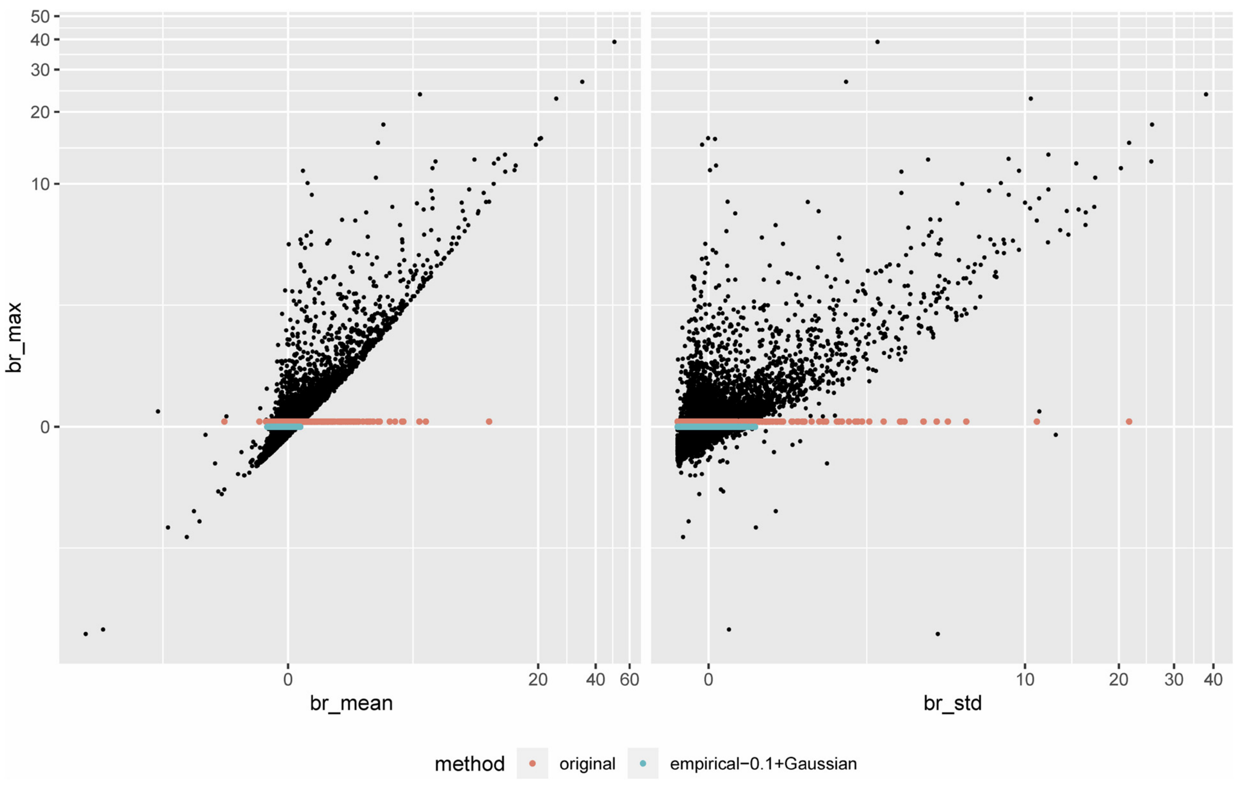

Fig. 9. Plots of `br_min` against `br_max` (left) and `br_std` against `br_max` (right). The black dots are the training data. The turquoise dots are the samples from the conditional distribution of the variable at the x-axis given that `br_max` is equal to zero generated using our method, while the red dots are the corresponding samples generated using the original Kernel SHAP approach.

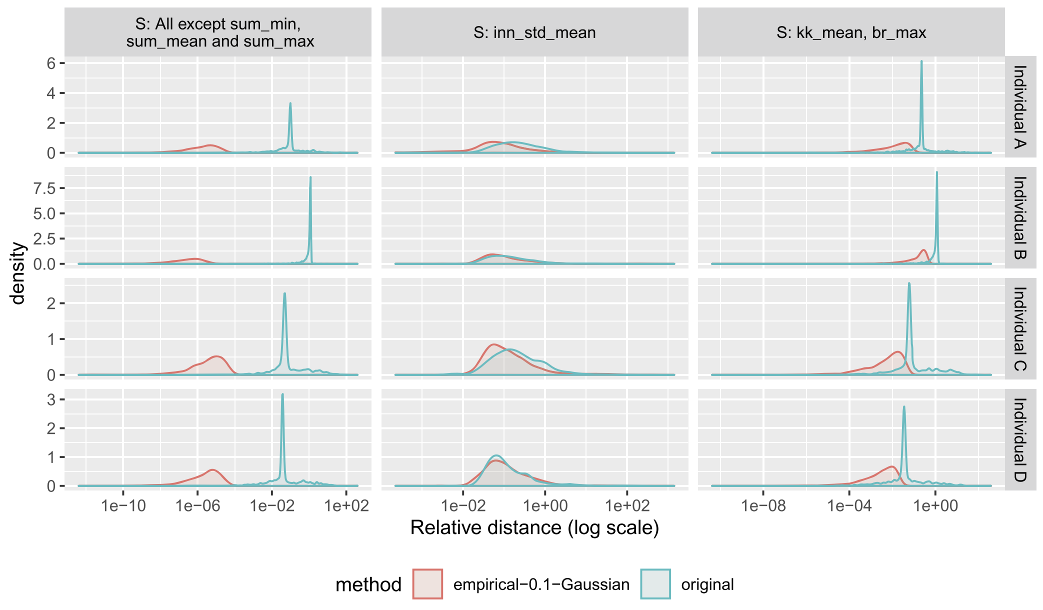

Fig. 10. Probability densities of mean Mahalanobis distances for three different conditional distributions and three different individuals. See the text for a further description.

the features in $\bar{\mathcal{S}}$. If there is a high degree of dependence among some or all the features, the independence assumption may lead to unrealistic combinations of feature values. Since evaluation of predictive machine learning models far from the domain at which they have been trained can lead to spurious predictions, the resulting Shapley values may not be correct. This paper introduces a modified version of the Kernel SHAP method, which handles dependent features. We have proposed four different approaches for estimating the conditional distribution; assuming a Gaussian multivariate distribution for all features, assuming a Gaussian copula with empirical margins, using an empirical approach and a combination of the empirical approach and either the Gaussian or the Gaussian copula approach.

We have performed a comprehensive simulation study with linear and non-linear models, Gaussian and non-Gaussian distributions, and dimensions 3 and 10, where our methods give more accurate approximations to the true Shapley values than the original Kernel SHAP approach. For the non-linear models, these methods clearly outperformed the TreeSHAP method, which, to the best of our knowledge, is the only Shapley approach which tries to handle dependence between features in the prediction explanation setting. When performing our experiments, it turned out that the non-parametric approach was superior when conditioning on a small number of the features, while it was outperformed by the Gaussian and copula methods if we condition on more variables. Hence, we regard the combined approach to be the most promising of our proposed methods. This approach was applied to a real case with 28 variables, where the predictions to be explained were produced by a XGBoost classifier designed to predict mortgage default. In this case, the true Shapley values are not known. However, we provide results, which indicate that our combined approach provides more sensible approximations than the original Kernel SHAP methods.

While having many desirable properties, our method has one obvious drawback: the computational time. The most time-consuming part is the AICc computation of the bandwidth parameter in the non-parametric approach. Using a fixed bandwidth parameter instead, the computational time of the combined approach is the same as that of the original Kernel SHAP method. Recently, there has been some attempts at using the underlying graph structure of the input data to reduce the computational complexity [42,43]. If several conditional independence requirements are satisfied, the full graph may first be divided into separate communities and then the Kernel SHAP method may be applied to each community individually. The methods proposed by [42,43] were tested on predictions obtained from text and image classification. In such settings the data often has a graph structure, enabling factorization into separate communities. The main challenge with using this method for tabular data is the potentially huge number of tests for conditional dependence which has to be performed. However, it is definitely worth a closer look.

In this paper, we assume that the aim is to explain the actual predictions from the model. For some models, the prediction is a probability. If the Shapley framework is used to decompose probabilities, summing over a subset of the ϕ_i-values may in theory produce values that are not in the range [0,1]. Hence, in such cases it might be more natural to assume that the importance of features is additive in the log odds space rather than in the space of probabilities. There are however problems even with this solution, since it is not straightforward for a human to interpret a log odds contribution to the probability. Hence, what to decompose seems to be more a practical than a mathematical question.

Tabular data often contain ordered or even non-ordered categorical data. The proposed non-parametric approach may still be used if the categorical variables are converted into numerical ones. The simplest solution is to use one-hot-encoding. However, large data sets often handle categorical features with hundreds of categories, meaning that such a method needs to handle a large number of binary attributes. An alternative approach is to use ideas from the clustering literature [44,45] defining distance functions that handle both categorical and mixed-type features. There are also generalizations of the Mahalanobis distance treating data with a mixture of nominal, ordinal and continuous variables that might be used instead of the approach described above, see for instance [46]. When it comes to the parametric approaches, categorical data represents a greater challenge. The most promising alternative might be to use entity embedding [47] to convert the categorical features into numerical ones, and then treat these features similarly to the other numerical features.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors are grateful to Scott Lundberg for valuable advice concerning the Kernel SHAP method. We also want to thank Ingrid Hobæk Haff for suggesting the copula approach, and Nikolai Sellereite for help implementing the methods in the R-package `shapr`. This work was supported by the Norwegian Research Council grant 237718 (Big Insight).

Appendix A. Shapley properties in the prediction explanation setting

When using Shapley values for prediction explanation, $\phi_0 = v(\emptyset) = \mathrm{E}[f(\boldsymbol{x})]$ actually plays an important role, quantifying how much of a prediction which is not due to *any* of the features, but merely to the global average prediction. If ϕ_0 is large (in absolute value) compared to ϕ_j, $j \neq 0$, the features are said to be 'unimportant' for that specific prediction. Moreover, in the prediction explanation setting, the interpretations of the properties of the Shapley value discussed in Section 2.1 are as follows:

Efficiency: The sum of the Shapley values for the different features is equal to the difference between the prediction and the global average prediction: $\sum_{j=1}^{M} \phi_j = f(\boldsymbol{x}^*) - \mathrm{E}[f(\boldsymbol{x})]$. This property ensures that the part of the prediction value which is not explained by the global mean prediction is fully devoted and explained by the features, and that the $\phi_j, j = 1, \ldots, M$ can be compared across different predictions $f(\boldsymbol{x}^*)$.

Symmetry: The Shapley values for two features are equal if, when combined with any other subsets of features, they contribute equally to the prediction. Failure to fulfill such a criterion, that is, that the same contribution from two different features does not give the same explanation, would be inconsistent and give untrustworthy explanations.

Dummy player: A feature that does not change the prediction, no matter which other features it is combined with, has a Shapley value of 0. Assigning a nonzero explanation value to a feature that has no influence on the prediction would be very odd, so this is a natural requirement.

Linearity: When a prediction function consists of a sum of prediction functions, the Shapley value for a feature is identical to the sum of that feature's Shapley values from each of the individual prediction functions. This also holds for linear combinations of prediction functions. This property ensures that models on this form, such as Random Forests or other structurally simple ensemble models, can be explained and interpreted individually.

Failure to fulfill any of these basic and advantageous properties gives an odd, undesirable or inconsistent explanation framework. There is no other additive explanation method than Shapley values which satisfies all these criteria [11].

Appendix B. Shapley values when the model is linear

In this section we first give a proof for the explicit formula for the Shapley values when the predictive model is a linear regression model, and all features are independent. Then, we show how to obtain the contribution function $v(\mathcal{S})$ when the model still is linear, but the features might be dependent.

B.1. Linear model and independent features

When $v(\mathcal{S}) = \mathrm{E}[f(\boldsymbol{x})|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}}]$, the predictive model is a linear regression model, and all features are independent, then the Shapley values take the simple form

$$\phi_j = \beta_j \,(x_j^* - E[x_j]), \quad j = 1, \ldots, M.$$

Proof. First, we derive the expression for $v(\mathcal{S})$ in this case:

$$\begin{aligned}
v(\mathcal{S}) &= E[f(\boldsymbol{x})|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}}] \\
&= \int f(\boldsymbol{x}_{\bar{\mathcal{S}}}, \boldsymbol{x}^*_{\mathcal{S}})\, p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}})\, d\boldsymbol{x}_{\bar{\mathcal{S}}}, && \text{(B.1)}\\
&= \int \left(\sum_{i \in \bar{\mathcal{S}}} \beta_i \, x_i + \sum_{i \in \mathcal{S}} \beta_i \, x_i^* \right) p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}^*_{\mathcal{S}})\, d\boldsymbol{x}_{\bar{\mathcal{S}}} && \text{(B.2)}\\
&= \sum_{i \in \bar{\mathcal{S}}} \beta_i \int x_i \; p(x_i)\, dx_i + \sum_{i \in \mathcal{S}} \beta_i \, x_i^* \int p(\boldsymbol{x}_{\bar{\mathcal{S}}})\, d\boldsymbol{x}_{\bar{\mathcal{S}}} && \text{(B.3)}\\
&= \sum_{i \in \bar{\mathcal{S}}} \beta_i \, E[x_i] + \sum_{i \in \mathcal{S}} \beta_i \, x_i^*.
\end{aligned}$$

The transition from step (B.1) to (B.2) follows from the assumption of a linear model, while the transition from (B.2) to (B.3) follows from the independent features assumption.

Having computed $v(\mathcal{S})$, the expression for $v(\mathcal{S} \cup \{j\})$ can be simply found as

$$v(\mathcal{S} \cup \{j\}) = v(\mathcal{S}) + \beta_j \, x_j^* - \beta_j \, E[x_j],$$

meaning that

$$v(\mathcal{S} \cup \{j\}) - v(\mathcal{S}) = \beta_j \left(x_j^* - E[x_j] \right).$$

From the above, we see that in this case, the difference $v(\mathcal{S} \cup \{j\}) - v(\mathcal{S})$ is independent of $\mathcal{S}$. Hence, the Shapley formula may be written as

$$\phi_j = \beta_j \left(x_j^* - E[x_j] \right) \sum_{\mathcal{S} \subseteq \mathcal{M} \setminus \{j\}} \frac{|\mathcal{S}|!(M - |\mathcal{S}| - 1)!}{M!}.$$

Since the sum of the Shapley weights is 1, we have

$$\phi_j = \beta_j \left(x_j^* - E[x_j] \right),$$

for $j = 1, \ldots, M$. □

B.2. Linear model and dependent features

If the model is linear, but the features are *not* independent, $v(\mathcal{S})$ instead may be derived as follows

$$\begin{aligned} v(\mathcal{S}) &= E[f(\boldsymbol{x}) | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*] \\ &= \int f(\boldsymbol{x}_{\bar{\mathcal{S}}}, \boldsymbol{x}_{\mathcal{S}}^*)\, p(\boldsymbol{x}_{\bar{\mathcal{S}}} | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*)\, d\boldsymbol{x}_{\bar{\mathcal{S}}}, \qquad \text{(B.4)} \\ &= \int \left(\sum_{i \in \bar{\mathcal{S}}} \beta_i x_i + \sum_{i \in \mathcal{S}} \beta_i x_i^* \right) p(\boldsymbol{x}_{\bar{\mathcal{S}}} | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*)\, d\boldsymbol{x}_{\bar{\mathcal{S}}} \qquad \text{(B.5)} \\ &= \sum_{i \in \bar{\mathcal{S}}} \beta_i \int x_i \; p(x_i | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*)\, dx_i + \sum_{i \in \mathcal{S}} \beta_i x_i^* \int p(\boldsymbol{x}_{\bar{\mathcal{S}}})\, d\boldsymbol{x}_{\bar{\mathcal{S}}} \qquad \text{(B.6)} \\ &= \sum_{i \in \bar{\mathcal{S}}} \beta_i \, E[x_i | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*] + \sum_{i \in \mathcal{S}} \beta_i x_i^* \\ &= f(\boldsymbol{x}_{\bar{\mathcal{S}}} = E[\boldsymbol{x}_{\bar{\mathcal{S}}} | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*], \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*) \qquad \text{(B.7)} \end{aligned}$$

In this case, one may therefore avoid time-consuming simulations if one is able to analytically obtain a proper estimate of $E[\boldsymbol{x}_{\bar{\mathcal{S}}} | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*]$. This is not straightforward in general.

Appendix C. Generalized hyperbolic distribution

The density of a d-dimensional Generalized Hyperbolic random vector $\boldsymbol{X}$ is

$$p(\boldsymbol{x}) = \left[\frac{\omega + \delta(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\omega + \boldsymbol{\beta}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\beta}} \right]^{\frac{\lambda - d/2}{2}} \frac{K_{\lambda - d/2}\left(\sqrt{(\omega + \delta(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}))(\omega + \boldsymbol{\beta}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\beta})} \right)}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2} K_\lambda(\omega) \exp\left\{ -(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\beta} \right\}},$$

where $\delta(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})$ is the squared Mahalanobis distance between $\boldsymbol{x}$ and $\boldsymbol{\mu}$, and K_λ is the modified Bessel function of the third kind with index λ. The mean vector and covariance matrix of $\boldsymbol{X}$ are

$$\mathrm{E}(\boldsymbol{X}) = \boldsymbol{\mu} + \mathrm{E}(W) \boldsymbol{\beta} \qquad \text{and} \qquad \mathrm{Var}(\boldsymbol{X}) = \mathrm{E}(W) \boldsymbol{\Sigma} + \mathrm{Var}(W) \boldsymbol{\beta} \boldsymbol{\beta}^T.$$

It can be shown [48] that if $\boldsymbol{X}$ is partitioned as $(\boldsymbol{X}_1^T, \boldsymbol{X}_2^T)^T$, where $\boldsymbol{X}_1$ is d_1-dimensional and $\boldsymbol{X}_2$ is d_2-dimensional, the conditional distribution of $\boldsymbol{X}_2$ given that $\boldsymbol{X}_1 = \boldsymbol{x}_1$ is Generalized Hyperbolic distributed, that is, $\boldsymbol{X}_2 | \boldsymbol{X}_1 = \boldsymbol{x}_1 \sim \mathrm{GH}^*(\lambda_{2|1}, \chi_{2|1}, \psi_{2|1}, \boldsymbol{\mu}_{2|1}, \boldsymbol{\Sigma}_{2|1}, \boldsymbol{\beta}_{2|1})$, where

$$\begin{aligned} \lambda_{2|1} &= \lambda - d_1/2, & \boldsymbol{\mu}_{2|1} &= \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{12}^T \boldsymbol{\Sigma}_{11}^{-1} (\boldsymbol{x}_1 - \boldsymbol{\mu}_1), \\ \chi_{2|1} &= \omega + (\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_{11}^{-1} (\boldsymbol{x}_1 - \boldsymbol{\mu}_1), & \boldsymbol{\Sigma}_{2|1} &= \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{12}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{12}, \\ \psi_{2|1} &= \omega + \boldsymbol{\beta}_1^T \Sigma_{11}^T \boldsymbol{\beta}_1, & \boldsymbol{\beta}_{2|1} &= \boldsymbol{\beta}_2 - \boldsymbol{\Sigma}_{12}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta}_1. \end{aligned}$$

Here, $\mathrm{GH}^*(\cdot)$ is a slightly different parameterization of the GH distribution is used for the conditional distribution. For technical reasons, we use the parameterization proposed by [49]:

$$p(\boldsymbol{x}) = \left[\frac{\chi + \delta(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\psi + \boldsymbol{\beta}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\beta}} \right]^{\frac{\lambda - d/2}{2}} \frac{(\psi/\chi)^{\lambda/2} K_{\lambda - d/2}\left(\sqrt{(\chi + \delta(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}))(\psi + \boldsymbol{\beta}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\beta})} \right)}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2} K_\lambda(\sqrt{\chi\psi}) \exp\left\{ -(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\beta} \right\}}.$$

References

[1] Z. Obermeyer, E.J. Emanuel, Predicting the future – big data, machine learning, and clinical medicine, N. Engl. J. Med. 375 (2016) 1216.
[2] A. Sudjianto, S. Nair, M. Yuan, A. Zhang, D. Kern, F. Cela-Díaz, Statistical methods for fighting financial crimes, Technometrics 52 (2010) 5–19.
[3] H. Kvamme, N. Sellereite, K. Aas, S. Sjursen, Predicting mortgage default using convolutional neural networks, Expert Syst. Appl. 102 (2018) 207–217.
[4] M.T. Ribeiro, S. Singh, C. Guestrin, Why should I trust you? Explaining predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, ACM, 2016, pp. 1135–1144.
[5] European Union, Regulation (EU) 2016/679 of the European Parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (general data protection regulation), Off. J. Eur. Union 59 (2016).
[6] A. Fisher, C. Rudin, F. Dominici, All models are wrong, but many are useful: learning a variable's importance by studying an entire class of prediction models simultaneously, J. Mach. Learn. Res. 20 (2019) 1–81.
[7] C. Molnar, Interpretable machine learning - a guide for making black box models explainable, https://christophm.github.io/interpretable-ml-book/, 2020.
[8] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K.-R. Müller, How to explain individual classification decisions, J. Mach. Learn. Res. 11 (2010) 1803–1831.
[9] E. Štrumbel, I. Kononenko, An efficient explanation of individual classifications using game theory, J. Mach. Learn. Res. 11 (2010) 1–18.
[10] E. Štrumbel, I. Kononenko, Explaining prediction models and individual predictions with feature contributions, Knowl. Inf. Syst. 41 (2014) 647–665.
[11] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, Curram Associates Inc., 2017, pp. 4768–4777.
[12] L.S. Shapley, A value for N-person games, in: Contributions to the Theory of Games, vol. 2, 1953, pp. 307–317.
[13] A.B. Owen, Sobol indices and Shapley value, SIAM/ASA J. Uncertain. Quantificat. 2 (2014) 245–251.
[14] E. Song, B.L. Nelson, J. Staum, Shapley effects for global sensitivity analysis: theory and computation, SIAM/ASA J. Uncertain. Quantificat. 4 (2016) 1060–1083.
[15] A.B. Owen, C. Prieur, On Shapley value for measuring importance of dependent inputs, SIAM/ASA J. Uncertain. Quantificat. 5 (2017) 986–1002.
[16] P. Giudici, E. Raffinetti, Shapley-Lorenz explainable artificial intelligence, Expert Syst. Appl. (2020) 114104.
[17] I. Covert, S.M. Lundberg, S.-I. Lee, Understanding global feature contributions with additive importance measures, Adv. Neural Inf. Process. Syst. 33 (2020).
[18] N. Sellereite, M. Jullum, shapr: an r-package for explaining machine learning models with dependence-aware Shapley values, J. Open Sour. Softw. 5 (2020) 2027.
[19] K. Aas, M. Jullum, A.L. land, Explaining individual predictions when features are dependent: more accurate approximations to Shapley values, arXiv:1903.10464v1, 2019.
[20] D. Janzing, L. Minorics, P. Blöbaum, Feature relevance quantification in explainable AI: a causal problem, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2907–2916.
[21] J. Pearl, Causality, Cambridge University Press, 2009.
[22] C. Frye, C. Rowat, I. Feige, Asymmetric Shapley values: incorporating causal knowledge into model-agnostic explainability, Adv. Neural Inf. Process. Syst. 33 (2020).
[23] T. Heskes, E. Sijben, I.G. Bucur, T. Claassen, Causal Shapley values: exploiting causal knowledge to explain individual predictions of complex models, Adv. Neural Inf. Process. Syst. 33 (2020).
[24] H. Chen, J.D. Janizek, S. Lundberg, S.-I. Lee, True to the model or true to the data?, in: 2020 ICML Workshop on Human Interpretability in Machine Learning, WHI 2020, 2020.
[25] S.M. Lundberg, S.-I. Lee, Consistent feature attribution for tree ensembles, in: Proceedings of the 34th International Conference on Machine Learning, JMLR: W&CP, 2017, pp. 15–21.
[26] T. Chen, C. Guestrin, XGBoost: a scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, ACM, 2016, pp. 785–794.
[27] H.P. Young, Monotonic solutions of cooperative games, Int. J. Game Theory 14 (1985) 65–72.
[28] A. Charnes, B. Golany, M. Keane, J. Rousseau, Extremal principle solutions of games in characteristic function form: core, in: Chebychev and Shapley Value Generalizations, Springer, Netherlands, Dordrecht, 1988, pp. 123–133.
[29] A. Sklar, Fonctions de répartition à n dimensions et leurs marges, Publ. Inst. Stat. Univ. Paris 8 (1959) 229–231.
[30] M. Rosenblatt, Remarks on some nonparametric estimates of a density function, Ann. Math. Stat. 27 (1956) 832–837.
[31] M.P. Holmes, A.G. Gray, C.L. Isbell, Fast kernel conditional density estimation: a dual-tree Monte Carlo approach, Comput. Stat. Data Anal. 54 (2010) 1707–1718.
[32] K. Bertin, C. Lacour, V. Rivoirard, Adaptive pointwise estimation of conditional density function, Ann. Inst. Henri Poincaré Probab. Stat. 52 (2016) 939–980.
[33] R. Izbicki, A.B. Lee, Converting high-dimensional regression to high-dimensional conditional density estimation, Electron. J. Stat. 11 (2017) 2800–2831.
[34] P.C. Mahalanobis, On the generalised distance in statistics, in: Proceedings of the National, Institute of Sciences of India, 1936, pp. 49–55.
[35] H.J. Bierens, The Nadaraya-Watson kernel regression function estimator, in: Topics in Advanced Econometrics, Cambridge University Press, 1994, pp. 212–247.
[36] C.M. Hurvich, J.S. Simonoff, C.-L. Tsai, Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion, J. R. Stat. Soc., Ser. B, Stat. Methodol. 60 (1998) 271–293.
[37] R. Izbicki, A.B. Lee, Converting high-dimensional regression to high-dimensional conditional density estimation, Electron. J. Stat. 11 (2017) 2800–2831.
[38] T. Gneiting, A.E. Raftery, Strictly proper scoring rules, prediction, and estimation, J. Am. Stat. Assoc. 102 (2007) 359–378.
[39] R.P. Browne, P.D. McNicholas, A mixture of generalized hyperbolic distributions, Can. J. Stat. 43 (2015) 176–198.
[40] I.J. Good, The population frequencies of species and the estimation of population parameters, Biometrika 40 (1953) 237–264.
[41] M.G. Kendall, A new measure of rank correlation, Biometrika 30 (1938) 81–93.
[42] J. Chen, L. Song, M.J. Wainwright, M.I. Jordan, L-Shapley and C-Shapley: efficient model interpretation for structured data, CoRR, arXiv:1808.02610 [abs], 2018.
[43] X. Li, N.C. Dvornek, Y. Zhou, J. Zhuang, P. Ventola, J.S. Duncan, Efficient interpretation of deep learning models using graph structure and cooperative game theory: application to ASD biomarker discovery, in: International Conference on Information Processing in Medical Imaging, Springer, 2019, pp. 718–730.
[44] Z. Huang, Clustering large data sets with mixed numeric and categorical variables, in: Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, 1997, pp. 21–34.
[45] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical variables, Data Min. Knowl. Discov. 2 (1998) 283–304.
[46] A.R. de Leon, K.C. Carriere, A generalized Mahalanobis distance for mixed data, J. Multivar. Anal. 92 (2005) 174–185.

[47] C. Guo, F. Berkhahn, Entity embeddings of categorical variables, arXiv:1604.06737v1, 2016.
[48] Y. Wei, Y. Tang, P.D. McNicholas, Mixtures of generalized hyperbolic distributions and mixtures of skew-t distributions for model-based clustering with incomplete data, Comput. Stat. Data Anal. 130 (2019) 18–41.
[49] A.J. McNeil, R. Frey, P. Embrechts, Quantitative Risk Management: Concepts, Techniques and Tools, Princeton University Press, 2006.