EXPLAINING INDIVIDUAL PREDICTIONS WHEN FEATURES ARE DEPENDENT: MORE ACCURATE APPROXIMATIONS TO SHAPLEY VALUES

KJERSTI AAS[1], MARTIN JULLUM[2], AND ANDERS LØLAND[3]

ABSTRACT. Explaining complex or seemingly simple machine learning models is a practical and ethical question, as well as a legal issue. Can I trust the model? Is it biased? Can I explain it to others? We want to explain individual predictions from a complex machine learning model by learning simple, interpretable explanations. Of existing work on interpreting complex models, Shapley values is the only method with a solid theoretical foundation. Kernel SHAP is a computationally efficient approximation to Shapley values in higher dimensions. Like most other existing methods, this approach assumes independent features, which may give very wrong explanations. This is the case even if a simple linear model is used for predictions. We extend the Kernel SHAP method to handle dependent features. We provide several examples of linear and non-linear models with linear and non-linear feature dependence, where our method gives more accurate approximations to the true Shapley values. We also propose a method for aggregating individual Shapley values, such that the prediction can be explained by groups of dependent variables.

1. INTRODUCTION

Interpretability is crucial when a complex machine learning model is to be applied in areas such as medicine (Obermeyer & Emanuel, 2016), fraud detection (Sudjianto et al., 2010) or credit scoring (Kvamme et al., 2018). 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? (Ribeiro et al., 2016), an ethical issue—is the model biased? (Angwin et al., 2016; Dressel & Farid, 2018) 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 (European Union, 2016). 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 interpreting complex models may be divided into two main 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 learning simple, interpretable explanations of the model specifically for a given prediction. Three examples are Explanation Vectors (Baehrens et al., 2010), LIME (Local Interpretable Model-agnostic Explanations (Ribeiro et al., 2016) and Shapley values (Štrumbel & Kononenko, 2010, 2014; Lundberg & Lee, 2017a). The latter, which is based on concepts from cooperative game theory (Shapley, 1953), is the only explanation method with a solid theoretical foundation. 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

[1,2,3]NORWEGIAN COMPUTING CENTER, P.O. BOX 114, BLINDERN, N-0314 OSLO, NORWAY.
E-mail addresses: kjersti.aas@nr.no, martin.jullum@nr.no, anders.loland@nr.no.

not guarantee perfectly distributed effects. Hence, the Shapley values might be the only explanation method which is compliant with the new GDPR rules (Molnar, 2019). It should also be noted that LIME and the Shapley values actually explain two different things. 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.

The main disadvantage of the Shapley value is that the computational time grows exponentially and becomes intractable for more than say ten features. This has led to approximations like the Shapley Sampling Values (Štrumbelj & Kononenko, 2010, 2014) and Kernel SHAP (Lundberg & Lee, 2017a). 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. If there is high degree of dependence/correlation among some or all the features, the resulting explanations might be very wrong. This is the case even if a simple linear model is used. Recently, Lundberg & Lee (2017b) have proposed a method based on Shapley Values denoted Tree SHAP, which is said to assume "less" feature independence in the sense that it accounts for some of the dependence, but not all. As we shall see, this method is not very accurate for dependent features. Moreover, it is not universally applicable, but specially designed for tree ensemble methods like XGBoost (Chen & Guestrin, 2016).

Shapley values have also been used in a slightly different setting, measuring global feature importance. For instance this framework has been used to partition the R^2 quantity among the d features in a linear regression model ("Shapley regression values"). This has been done for independent (Owen, 2014), and more recently also for dependent features (Song et al., 2016; Owen & Prieur, 2017). In this paper, we extend the Kernel SHAP method to handle dependent features, using some of the ideas from the latter. According to Molnar (2019) there has been no previous research on what dependency between the features means for the Shapley values in the explanation setting, nor a suggestion on how to handle such dependence. Hence, to the best of our knowledge, this paper is the first to incorporate dependence into the Kernel SHAP method. Moreover, we propose a method for clustering Shapley values corresponding to dependent features, improving the presentation of feature contribution for individual predictions in the presence of feature dependence.

The rest of this paper is organized as follows. Section 2 reviews the Shapley values and the Kernel SHAP method. In Section 3 we describe our proposed approaches for capturing dependence, while Section 4 gives the results from several experiments validating these methods. An approach for clustering of Shapley values corresponding to feature dependence is described in Section 5 and, finally, Section 6 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, explain its use in the context of explaining individual predictions in Section 2.2, and then we describe the Kernel SHAP method in Section 2.3.

2.1. **The exact Shapley value and cooperative game theory.** Consider a cooperative game with M players aiming at maximizing a payoff, and let $S \subseteq \mathcal{M} = \{1, \ldots, M\}$ be a subset consisting of $|S|$ players. Assume that we have a "contribution" function $v(S)$ that maps subsets of players to the real numbers, called the worth or contribution of coalition S. It describes the total expected sum of payoffs the members of S can obtain by cooperation. The Shapley value (Shapley, 1953) 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

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

that is, a weighted mean over 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 possible different permutations in which the coalition can be formed, yielding a weighted mean.

To illustrate the application of (1), let us consider a game with three players such that $\mathcal{M} = \{1, 2, 3\}$. Then, there are 8 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

$$\phi_1 = \tfrac{1}{3}\left(v(\{1,2,3\}) - v(\{2,3\})\right) + \tfrac{1}{6}\left(v(\{1,2\}) - v(\{2\})\right) + \tfrac{1}{6}\left(v(\{1,3\}) - v(\{3\})\right) + \tfrac{1}{3}\left(v(\{1\}) - v(\emptyset)\right),$$

$$\phi_2 = \tfrac{1}{3}\left(v(\{1,2,3\}) - v(\{1,3\})\right) + \tfrac{1}{6}\left(v(\{1,2\}) - v(\{1\})\right) + \tfrac{1}{6}\left(v(\{2,3\}) - v(\{3\})\right) + \tfrac{1}{3}\left(v(\{2\}) - v(\emptyset)\right),$$

$$\phi_3 = \tfrac{1}{3}\left(v(\{1,2,3\}) - v(\{1,2\})\right) + \tfrac{1}{6}\left(v(\{1,3\}) - v(\{1\})\right) + \tfrac{1}{6}\left(v(\{2,3\}) - v(\{2\})\right) + \tfrac{1}{3}\left(v(\{3\}) - v(\emptyset)\right).$$

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 $\mathcal{S} \subseteq \mathcal{M}$, then $\phi_j = 0$. There are also some other properties, but they have no practical relevance in the context of feature importance. The Shapley values is the only set of values satisfying the above properties, see Shapley (1953) and Young (1985) for proofs. Hence, it is the only explanation method with a solid theoretical background.

2.2. **Shapley values for prediction explanation.** Consider a classical machine learning scenario where a training set $\{y^i, \boldsymbol{x}^i\}_{i=1,\ldots,n_{\text{train}}}$ 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 & Kononenko (2010, 2014) and Lundberg & Lee (2017a) 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 game being played, and the features takes the place of the players. Typically we want to explain multiple predictions, corresponding to different feature vectors $\boldsymbol{x}^*$. In that case, the method is simply repeated for every prediction, resulting in several individual prediction explanations, one for every prediction.

Lundberg & Lee (2017a) define the contribution function $v(\mathcal{S})$ for a certain subset $\mathcal{S}$, as the expected output of the predictive model, conditional on the feature values $\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*$ of this subset:

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

In full generality, one may very well choose a different contribution function. However, the conditional expectation of the prediction function is by all means the most natural one, mimicking your best guess of the prediction value knowing only $\boldsymbol{x}_{\mathcal{S}}$. We will therefore rely on this contribution function throughout the present paper. In Lundberg & Lee (2017a), this type of Shapley values are denoted SHAP (SHapley Additive exPlanation) values.

Applying the above method in practice requires the expectation in (2) to be known for all $\boldsymbol{x}_{\mathcal{S}}$. The main methodological contribution of the present paper is carefully describing, developing, and comparing methodology to appropriately estimate these expectations. This will be handled in Sections 2.3.2 and 3.

Remark 1. 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 mean prediction. If ϕ_0 is large (in absolute value) compared to $\phi_j, j \neq 0$, then the features are not important for that specific prediction. Moreover, in the prediction explanation setting, the interpretation of the properties of the Shapley value discussed in Section 2.1 are as follows: The efficiency property states that the Shapley values for the different features have to sum up to the difference between the prediction and the global average prediction. Further, according to the symmetry property, the Shapley values for two features should be the same if, when combined with any other subsets of features, they contribute equally to the predictions. Finally, the dummy player property says that a feature which doesn't change the predicted value has a Shapley value of 0, independent of which features it is combined with.

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 $x_j, j = 1, \ldots, M$ are independent, one may show that under (2), the Shapley values take the simple form:

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

When there is dependence between the features, and/or the model is non-linear, no such simple explicit formulae exist, and one is left with (1). 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 the next section, we shall see how a clever approximation method may be used to (partly) overcome this issue.

2.3. **Kernel SHAP.** The Kernel SHAP method of Lundberg & Lee (2017a) 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 Lundberg & Lee (2017a), 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.* Lundberg & Lee (2017a) show that the Shapley values correspond to the solution of a certain weighted least squares (WLS) problem.[1] In its simplest form, the WLS problem can be stated as the problem of minimizing

$$(3) \qquad \sum_{\mathcal{S} \subseteq \mathcal{M}} (v(\mathcal{S}) - (\phi_0 + \sum_{j \in S} \phi_j))^2 k(M, \mathcal{S}),$$

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

---

[1]The proof of this result is found in a supplement to the paper, which may be downloaded from https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions. Note that there are several errors in the proof leading to formula (5). The formula, however, is correct.

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 $2^{M+1} \times 2^{M+1}$ diagonal matrix containing $k(M, |\mathcal{S}|)$, where $\mathcal{S}$ in both cases resembles the feature combinations of the corresponding row in $\boldsymbol{Z}$. Then (3) may be rewritten to

$$(4) \qquad (\boldsymbol{v} - \boldsymbol{Z}\phi)^T \boldsymbol{W} (\boldsymbol{v} - \boldsymbol{Z}\phi),$$

for which the solution is

$$(5) \qquad \phi = \left(\boldsymbol{Z}^T \boldsymbol{W} \boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^T \boldsymbol{W} \boldsymbol{v}.$$

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 (5) is still computationally expensive. However, the trick is to use the weighted least squares formulation to approximate (5). 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, 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 (5):[2]

$$(6) \qquad \phi = \left[\left(\boldsymbol{Z}_{\mathcal{D}}^T \boldsymbol{W}_{\mathcal{D}} \boldsymbol{Z}_{\mathcal{D}}\right)^{-1} \boldsymbol{Z}_{\mathcal{D}}^T \boldsymbol{W}_{\mathcal{D}}\right] \boldsymbol{v}_{\mathcal{D}} = \boldsymbol{R}_{\mathcal{D}} \boldsymbol{v}_{\mathcal{D}}.$$

A practical consequence of using (6) 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 has to be carried out once. All that is needed to explain different predictions from the same model (provided $\boldsymbol{v}_{\mathcal{D}}$ is pre-computed) 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 $\boldsymbol{Z}$. 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

$$(7) \qquad E[f(\boldsymbol{x}) | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*] = E[f(\boldsymbol{x}_{\bar{\mathcal{S}}}, \boldsymbol{x}_{\mathcal{S}}) | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*] = \int f(\boldsymbol{x}_{\bar{\mathcal{S}}}, \boldsymbol{x}_{\mathcal{S}}^*) \, p(\boldsymbol{x}_{\bar{\mathcal{S}}} | \boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*) d\boldsymbol{x}_{\bar{\mathcal{S}}},$$

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 $p(\boldsymbol{x}_{\bar{\mathcal{S}}} | \boldsymbol{x}_{\mathcal{S}})$ by $p(\boldsymbol{x}_{\bar{\mathcal{S}}})$. 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 (7) can be approximated by

$$(8) \qquad v_{\mathrm{KerSHAP}}(\mathcal{S}) = \frac{1}{K} \sum_{k=1}^{K} f(\boldsymbol{x}_{\bar{\mathcal{S}}}^k, \boldsymbol{x}_{\mathcal{S}}^*),$$

---

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

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

3. Incorporating dependence into the Kernel SHAP method

If the features in a given model are highly dependent, the Kernel SHAP method may give a completely wrong answer. 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 $p(\boldsymbol{x}_{\bar{S}}|\boldsymbol{x}_S) = p(\boldsymbol{x}_{\bar{S}})$ is used, is when approximating the integral in (7). Apart from this rough assumption, the Kernel SHAP framework stands out as a clever and fruitful way to compute the Shapley values. It would therefore be desirable to incorporate dependence into the Kernel SHAP method by relaxing the independence assumption. This can be done by estimating/approximating $p(\boldsymbol{x}_{\bar{S}}|\boldsymbol{x}_S = \boldsymbol{x}_S^*)$ directly and generate samples from this distribution, instead of generating them independently from $\boldsymbol{x}_S$ as in Section 2.3.2. We propose three approaches for estimating $p(\boldsymbol{x}_{\bar{S}}|\boldsymbol{x}_S = \boldsymbol{x}_S^*)$; assuming a Gaussian distribution for $p(\boldsymbol{x})$, assuming a Gaussian copula distribution for $p(\boldsymbol{x})$, and approximating $p(\boldsymbol{x}_{\bar{S}}|\boldsymbol{x}_S = \boldsymbol{x}_S^*)$ by an empirical (conditional) distribution. These approaches may also be combined.

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 $p(\boldsymbol{x}_{\bar{S}}|\boldsymbol{x}_S = \boldsymbol{x}_S^*)$ is also multivariate Gaussian. In particular, writing $p(\boldsymbol{x}) = p(\boldsymbol{x}_S, \boldsymbol{x}_{\bar{S}}) = \mathrm{N}_M(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu} = (\boldsymbol{\mu}_S, \boldsymbol{\mu}_{\bar{S}})^\top$ and

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

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

$$\boldsymbol{\mu}_{\bar{S}|S} = \boldsymbol{\mu}_{\bar{S}} + \boldsymbol{\Sigma}_{\bar{S}S}\,\boldsymbol{\Sigma}_{SS}^{-1}(\boldsymbol{x}_S^* - \boldsymbol{\mu}_S) \tag{9}$$

and

$$\boldsymbol{\Sigma}_{\bar{S}|S} = \boldsymbol{\Sigma}_{\bar{S}\bar{S}} - \boldsymbol{\Sigma}_{\bar{S}S}\,\boldsymbol{\Sigma}_{SS}^{-1}\boldsymbol{\Sigma}_{S\bar{S}}. \tag{10}$$

Hence, instead of sampling from the marginal distribution of $\boldsymbol{x}_{\bar{S}}$, we can sample from the Gaussian distribution with expectation vector and covariance matrix given by (9) and (10), 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 $\boldsymbol{x}_{\bar{S}}^k, k = 1, \ldots, K$ from the conditional distribution, the integral in (7) is finally approximated by (8).

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 (Sklar, 1959) states that every multivariate distribution F with marginals $F_1, F_2, \ldots, F_d$ can be written as

$$F(x_1, \ldots, x_d) = C(F_1(x_1), F_2(x_2), \ldots, F_d(x_d)), \tag{11}$$

for some appropriate d-dimensional copula C. In fact, the copula from (11) 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 (9) and (10).

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

- 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[3], 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 (7) is finally approximated by (8).

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 will work very well. For such situations, we propose a non-parametric approach. The classical method for non-parametric density estimation is the kernel estimator (Rosenblatt, 1956), which in the decades following its introduction has been refined and developed in many directions, see for example Holmes et al. (2010); Bertin et al. (2016); Izbicki & B. Lee (2017). 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. Motivated by the idea that samples $(\boldsymbol{x}_{\bar{\mathcal{S}}}, \boldsymbol{x}_{\mathcal{S}})$ with $\boldsymbol{x}_{\mathcal{S}}$ close to $\boldsymbol{x}_{\mathcal{S}}^*$ is informative about the conditional distribution $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}}^*)$, and the use of the training data to represent the empirical distribution of the features, we have developed a heuristic empirical conditional approach to sample approximately from $p(\boldsymbol{x}_{\bar{\mathcal{S}}}|\boldsymbol{x}_{\mathcal{S}}^*)$:

(1) Compute the distance between the instance $\boldsymbol{x}^*$ to be explained and all training instances $\boldsymbol{x}^i, i = 1, \ldots, n_{\text{train}}$. The distance between $\boldsymbol{x}^*$ and instance i is computed as

$$(12) \qquad D_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^i) = \sqrt{\frac{(\boldsymbol{x}_{\mathcal{S}}^* - \boldsymbol{x}_{\mathcal{S}}^i)^T \Sigma_{\mathcal{S}}^{-1}(\boldsymbol{x}_{\mathcal{S}}^* - \boldsymbol{x}_{\mathcal{S}}^i)}{|\mathcal{S}|}},$$

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 $\mathcal{S}$. (12) may be viewed as a scaled version of the Mahalanobis distance (Mahalanobis, 1936).

(2) Compute weights for all training instances $\boldsymbol{x}^i, i = 1, \ldots, n_{\text{train}}$ 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 (7) with a weighted version of (8):

$$v_{\text{condKerSHAP}}(\mathcal{S}) = \frac{\sum_{k=1}^K w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^{[k]}) f(\boldsymbol{x}_{\bar{\mathcal{S}}}^{[k]}, \boldsymbol{x}_{\mathcal{S}}^*)}{\sum_{k=1}^K w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^{[k]})}.$$

Note that we could have used (8), with the $\boldsymbol{x}_{\bar{\mathcal{S}}}^k$ sampled (with replacement) from the training data with weights $w_{\mathcal{S}}(\boldsymbol{x}^i), i = 1, \ldots, n_{\text{train}}$. 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:

$$(13) \qquad K = \min_{L \in \mathbb{N}} \left\{ \frac{\sum_{k=1}^L w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^{[k]})}{\sum_{i=1}^{n_{\text{train}}} w_{\mathcal{S}}(\boldsymbol{x}^*, \boldsymbol{x}^i)} > \eta \right\},$$

---

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

where η is set to for instance 0.9. If K in (13) 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) requires 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 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 and a larger σ is preferable. As $\sigma \to \infty$ our method approximates the original Kernel SHAP method in Section 2.3.2.

We have chosen to use the small-sample-size corrected version of Akaike information criterion (AICc) proposed by Hurvich et al. (1998) to estimate the bandwidth parameter. This criterion is constructed for nonparametric regression, and not for empirical conditional distributions. It turns out, however, that viewing the estimation of $E[f(\boldsymbol{x})|\boldsymbol{x}_{\mathcal{S}} = \boldsymbol{x}_{\mathcal{S}}^*]$ as a regression problem with response $f(\boldsymbol{x}_{\bar{\mathcal{S}}}^i, \boldsymbol{x}_{\mathcal{S}}^*)$ and covariates $\boldsymbol{x}_{\mathcal{S}}^i, i = 1, \ldots, n_{\text{train}}$, our empirical conditional distribution approach (with $K = n_{\text{train}}$) is equivalent to the Nadaraya-Watson estimator (Bierens, 1994).

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

$$(14) \qquad \text{AICc} = \log(\hat{\tau}^2) + \Phi(H),$$

where

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

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.

Thus, to select σ we compute the AICc in (14) 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 $\boldsymbol{x}_{\mathcal{S}} \leq D^*$, 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 Izbicki & Lee (2017) 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 $\dim(\boldsymbol{x}_{\mathcal{S}}) \leq D^*$ using the

empirical method, and all other conditional distributions using the parametric method. To determine D^*, one may use for instance cross validation. In our experiments we have, however, determined D^* using an ad-hoc procedure.

4. Experiments

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 compare the various methods to the true Shapley value in several controlled experiments. As computing the exact Shapley values for a single prediction requires solving $O(2^M)$ integrals of the type in (7), 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 medium high ($M = 10$) dimensional setting, using various multivariate distributions for the features $\boldsymbol{x}$, sampling models for $y|\boldsymbol{x}$, and forms of the predictive model $f()$. The experiments with $M = 3$ and $M = 10$ are treated in Sections 4.2 and 4.3, respectively.

In all experiments we train the model on $n_{\text{train}} = 2,000$ independent samples and measure the quality of the Shapley value approximations of a total of $n_{\text{test}} = 1,000$ predictions with features sampled from the same distribution. As the AICc dependent approximation methods are directly dependent on the sampled training set, we run the experiments in 10 batches, where each batch contains a new training set of size $n_{\text{train}} = 2,000$, and 100 of the $n_{\text{test}} = 1,000$ test observations. This also reduces the influence of the exact form of the fitted predictive model.

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 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 (Lundberg & Lee, 2017b) 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

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

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 (Gneiting & Raftery, 2007) associated with the aforementioned MAE.

The skill score for method q takes the form

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

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 $K = 1000$ 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 experients 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 (7) 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 $p(\boldsymbol{x}) = \text{N}_3(\boldsymbol{0}, \boldsymbol{\Sigma}(\rho))$, where the covariance matrix $\boldsymbol{\Sigma}(\rho)$ takes the form

$$(15) \qquad \boldsymbol{\Sigma}(\rho) = \begin{bmatrix} 1 & \rho & \rho \\ \rho & 1 & \rho \\ \rho & \rho & 1 \end{bmatrix}.$$

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 Browne & McNicholas (2015), 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 $\boldsymbol{X} \sim \text{GH}(\lambda, \omega, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\beta})$, if it can be represented by

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

where $W \sim \text{GIG}(\lambda, \omega, \omega)$, $\boldsymbol{U} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma})$ and W is independent of $\boldsymbol{U}$. GIG is the Generalised Inverse Gaussian distribution introduced by Good (1953). Appendix A contains more details on this distribution. We use the following parameter values in our experiments:

$$\lambda = 1$$
$$\omega = 0.5$$
$$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(0) = \text{diag}(\boldsymbol{1})$$
$$\boldsymbol{\beta} = (1/4)\kappa * \boldsymbol{1}$$
$$\boldsymbol{\mu} = \boldsymbol{0} - \text{E}[W](1/4)\kappa,$$

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. $\text{E}[W]$ 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 distribution 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 $\boldsymbol{\mu}_1(\lambda) = -\boldsymbol{\mu}_2(\lambda) = \gamma \cdot (1, -0.5, 1)^\top$. The covariance matrix $\boldsymbol{\Sigma}$ is assumed to be on the form in (15). 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 modelled 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 $\mathrm{fun}_1, \mathrm{fun}_2,$ and fun_3 are displayed in Figure 1. We fit the model $g(\boldsymbol{x})$ with the
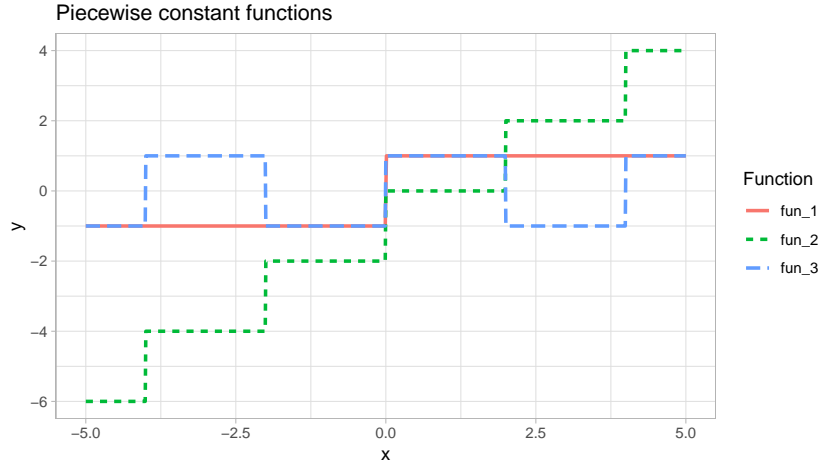
FIGURE 1. Piecewise constant functions used in experiment D, E and F.

XGBoost framework (Chen & Guestrin, 2016) using default values of all hyperparameters, the histogram tree learning method (`tree_method="hist"`) and 50 boosting iterations.

4.2.6. *Results.* The results from the 3D-experiments are visualized in Figures 2—7. In experiment A we have used a linear sampling model and Gaussian features. As seen from Figure 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 methods 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 Figure 3, when κ increases, the MAE of the copula and Gaussian methods is 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. Figure 4 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 Figures 5-7, 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 shows 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.
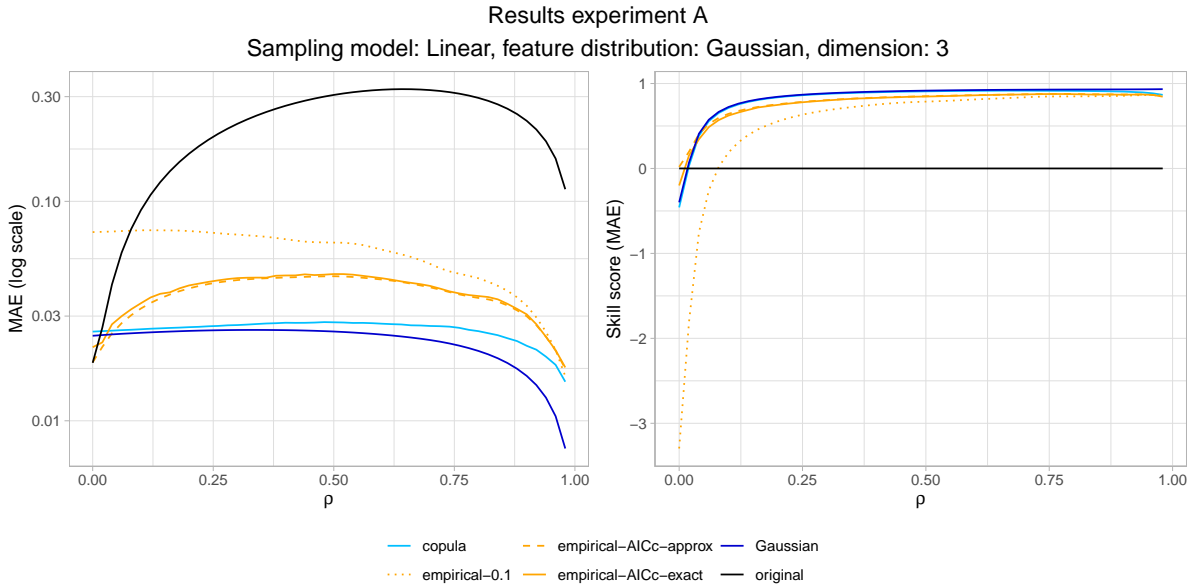

FIGURE 2. Dimension 3: MAE and skill score for linear model with Gaussian features.

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, $p(\boldsymbol{x}) = \mathrm{N}_{10}(\boldsymbol{0}, \boldsymbol{\Sigma}(\rho))$, where $\boldsymbol{\Sigma}(\rho)$ is the 10 dimensional extension of (15) such that all features have variance 1 and the pairwise correlation between any two features is ρ.
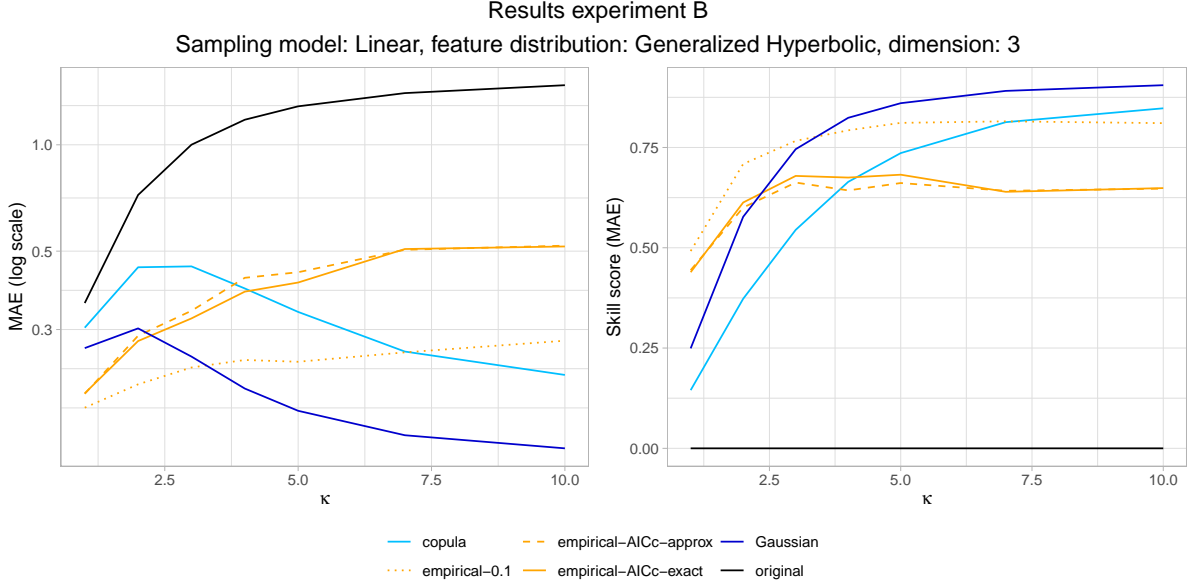
FIGURE 3. Dimension 3: MAE and skill score for linear model with GH-distributed features.
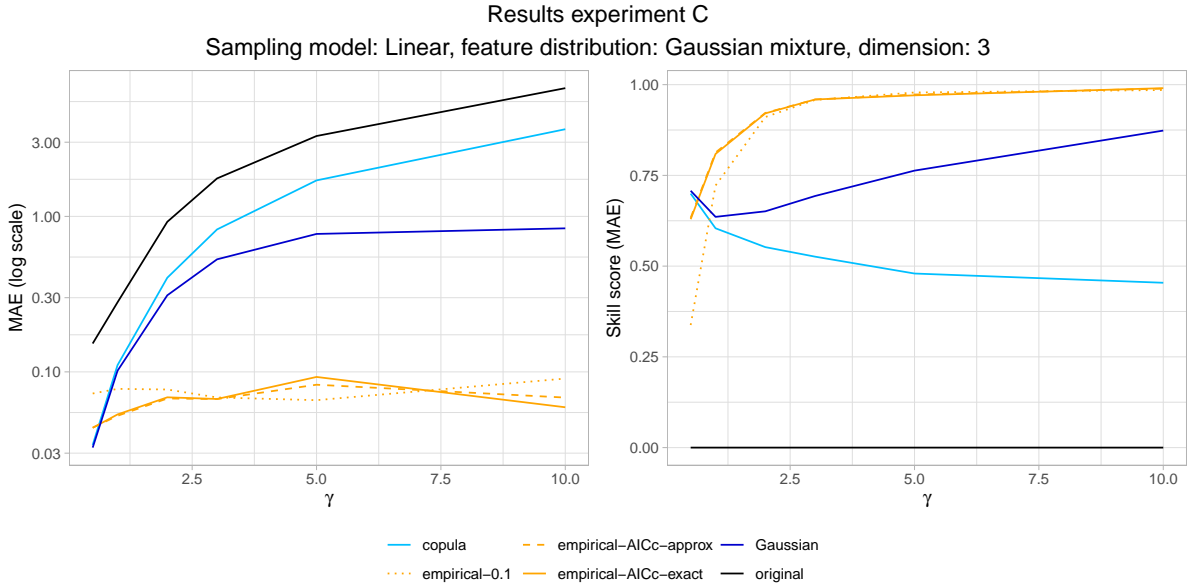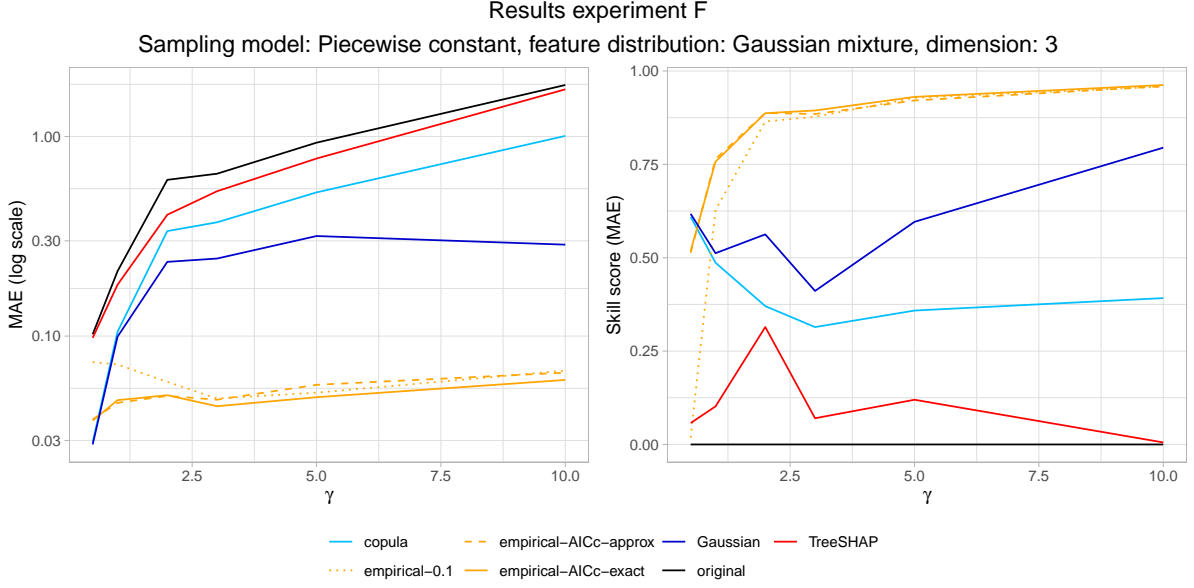

FIGURE 4. Dimension 3: MAE and skill score for linear model with Gaussian mixture distributed features.

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

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

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

$$f(\boldsymbol{x}) = \sum_{j=1}^{10} \beta_j x_j + \varepsilon.$$
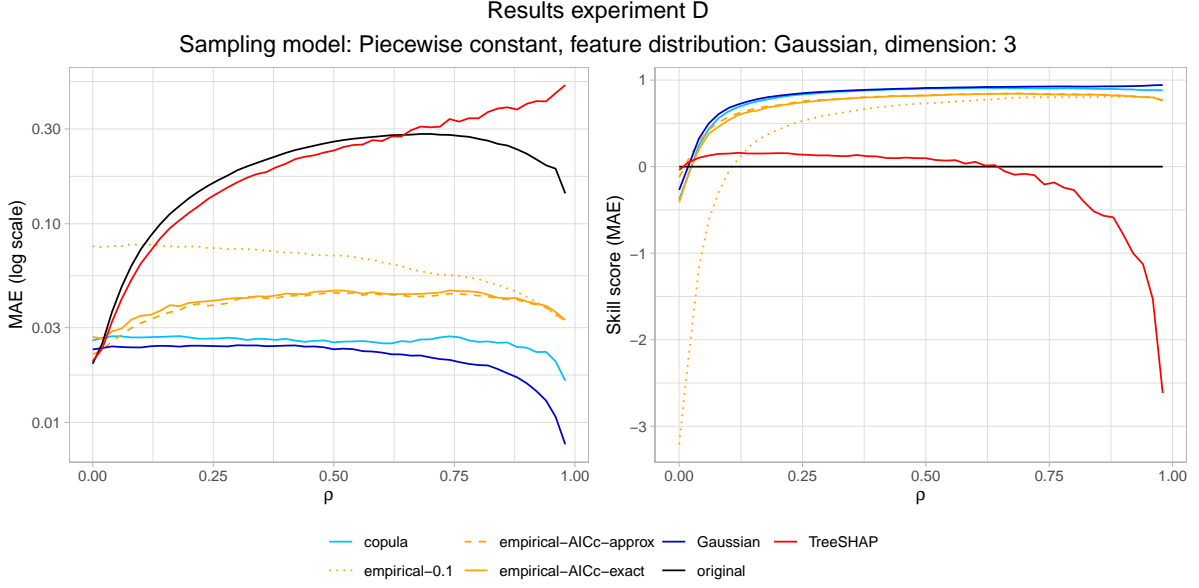
FIGURE 5. Dimension 3: MAE and skill score for piecewise constant model with Gaussian features.
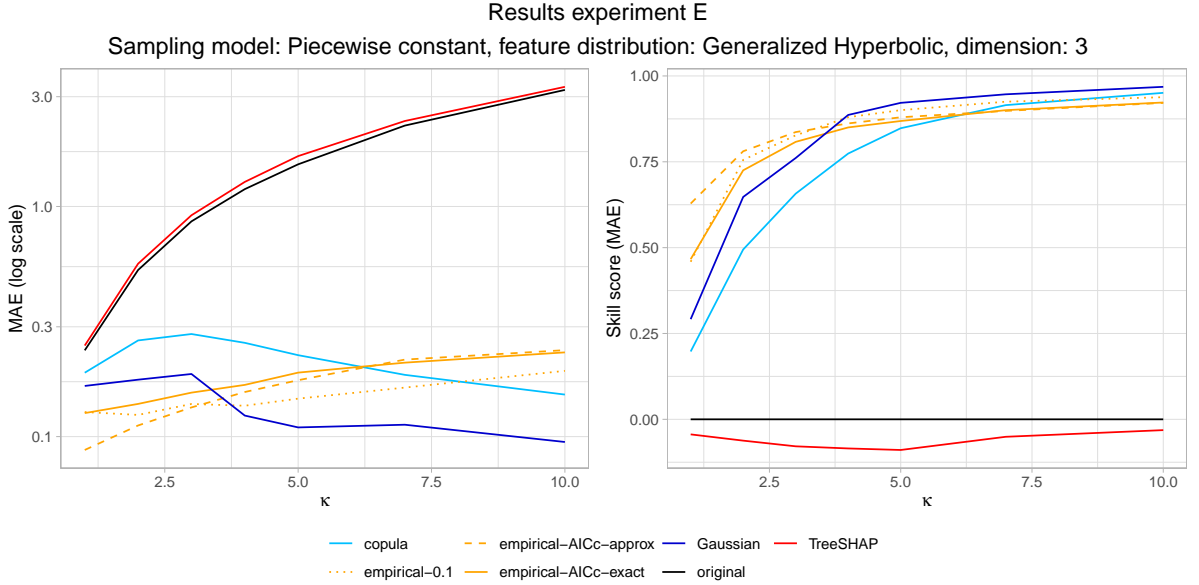
FIGURE 6. Dimension 3: MAE and skill score for piecewise constant model with GH-distributed features.

Note that even though we have 10 features, one of them has no influence on y.

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

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

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.

FIGURE 7. Dimension 3: MAE and skill score for piecewise constant model with Gaussian mixture distributed features.

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

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

The parameter values are chosen as to resemble the feature distribution in our real data example to be described in Section 4.4. 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 approches 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 Figures 8 and 9 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. 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 Figures 2 and 3, with the Gaussian method generally showing the best performance. The combined empirical and Gaussian/copula approaches also works 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.
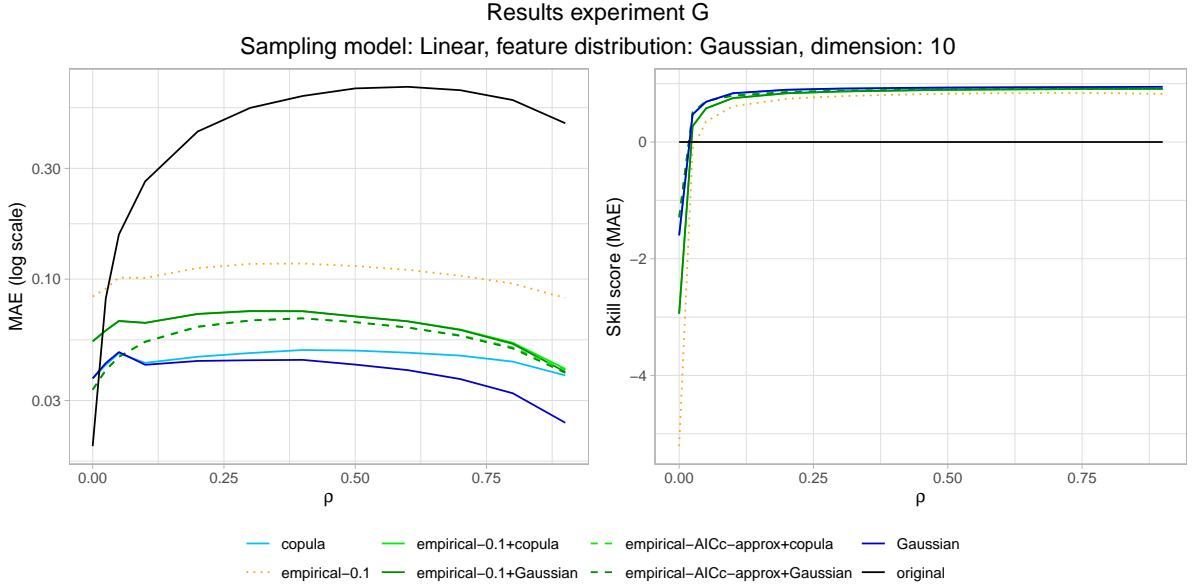

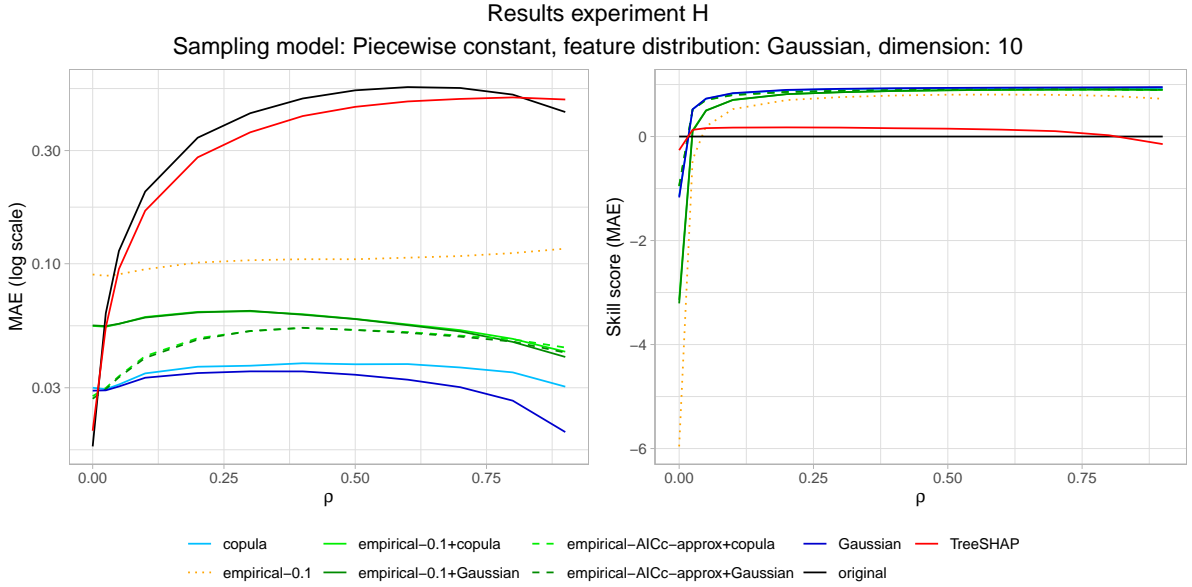FIGURE 8. Dimension 10: MAE and skill score for linear model with Gaussian distributed features.

FIGURE 9. Dimension 10: MAE and skill score for piecewise constant model with Gaussian distributed features.

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 (Kvamme et al., 2018). 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

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

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

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 standard deviation scaled by the mean (`std_mean`) were computed, resulting in 28 features. These are scaled to have mean 0 and standard deviation 1 before they are used in our computations. Figure 10 shows histograms for nine of the features (the remaining features have similar distributions), while the pairwise rank correlations (measured by Kendall's τ, see Section 5.1 for more information) for this data set are shown in Figure 11. The feature distributions are skewed and heavy-tailed. Moreover, most correlations are weak, but there are quite a few strong dependencies.

The data set was divided into a training set and a test set, containing 12,696 and 1,921 observations respectively. The default rate in the training set was 10%, while it was 5% in the test set. 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. Figure 12 shows plots of Shapley values estimated with the original Kernel SHAP approach against those estimated with the empirical-0.1+Gaussian method for four of the features. For the features `inn_std_mean` and `sk_std_mean` there is a high degree of correspondence, while for `tr_std` and `br_max`, the Shapley values for the two methods are very different. These results may at least partly be explained by Figure 11. The features `inn_std_mean` and `sk_std_mean` are almost independent from all other features, while `tr_std`, and especially `br_max`, have a high degree of dependence with many other features. Hence, the original Kernel SHAP method is likely to perform well for `inn_std_mean` and `sk_std_mean` and poor for `tr_std` and `br_max`.

5. CLUSTERING OF SHAPLEY VALUES CORRESPONDING TO FEATURE DEPENDENCE

Using any of our suggested Shapley value approximations, the dependencies between features are handled in a proper way. One way to then visually present the explanation of a particular prediction could be to rank the absolute Shapley values and present them and their corresponding features in descending order, for example for the ten most important features. However, if some or many of the features are dependent, a grouping or clustering of dependent features may ease the interpretation and use of the Shapley values.

Assume that we have many features that are highly correlated, but none of them is among the ten most important features. If presented as a group, however, their combined Shapley value might turn out to be on the top ten list. Even when the individual Shapley values are well approximated, clustering can therefore aid in presenting the model explanation more correctly.

This clustering of features can in principle be different for each individual, but that would make model explanations across individuals opaque. We therefore suggest to cluster the features
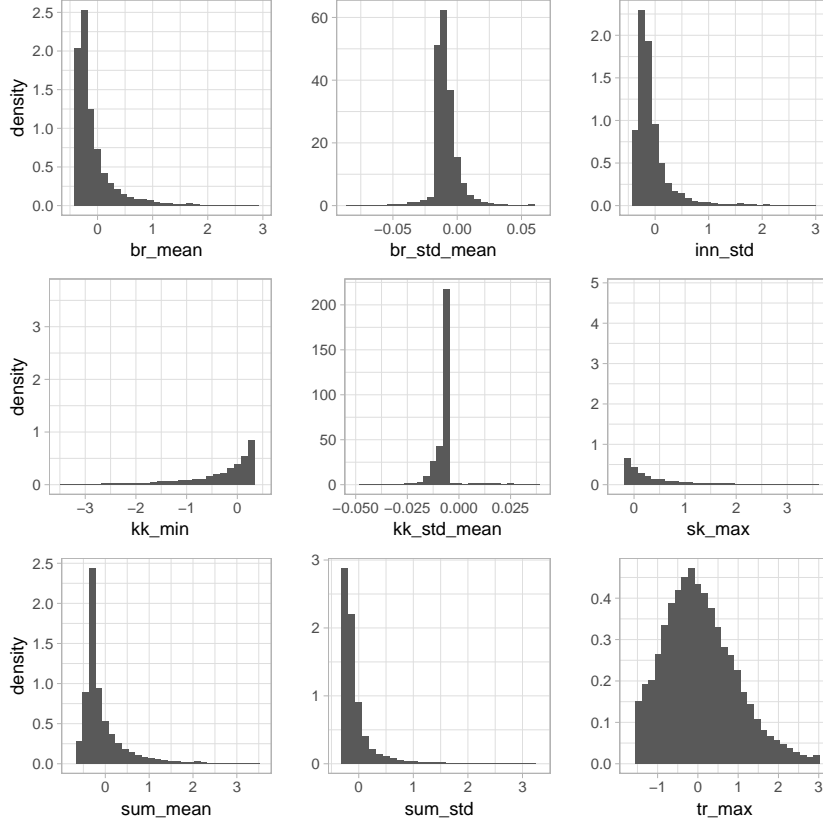
FIGURE 10. Histograms for nine of the variables in the real data set.

once on 1) the training data, or 2) test data, given that the test data set is large enough for reliable clustering to be done. We are then left with the choice of dependence measure and clustering method. Our preferred choices are not canonical in any sense and other choices may suit your particular application better.

5.1. **Dependence measure.** The Pearson correlation coefficient measures the linear correlation. As our Shapley values are not restricted to linear dependence, we suggest to apply a more robust measure of dependence with fewer implicit assumptions, namely the rank correlation, more specifically as measured by Kendall's τ (Kendall, 1938). Kendall's τ for two random features x_j and x_k is defined as

$$T_{j,k} = \frac{1}{n(n-1)} \sum_{i \neq l} \operatorname{sign}(x_j^i - x_j^l) \operatorname{sign}(x_k^i - x_k^l),$$

where $i, l = 1, \ldots n$ are the samples we use to compute this measure.

5.2. **Clustering method and cluster size.** First, we define a dissimilarity matrix, that is distances between pairs of features. Since we would like to group features together that are strongly correlated, regardless of whether this correlation is positive or negative, we define a dissimilarity matrix $\boldsymbol{D}$ having indicies

(16) $D_{j,k} = 1 - |T_{j,k}|, \quad j, k = 1, \ldots, M,$

where $|(\cdot)|$ denotes the absolute value. Hence, two features that are perfectly correlated will have a dissimilarity value of 0, while a pair of independent features will have a dissimilarity value of 1. We then do hierarchical, agglomerative clustering with the agglomeration method commonly denoted as "complete" (Müllner, 2013) and $\boldsymbol{D}$ in (16) as dissimilarity matrix. This means that when merging of two clusters J and K is considered, the cluster dissimilarity is
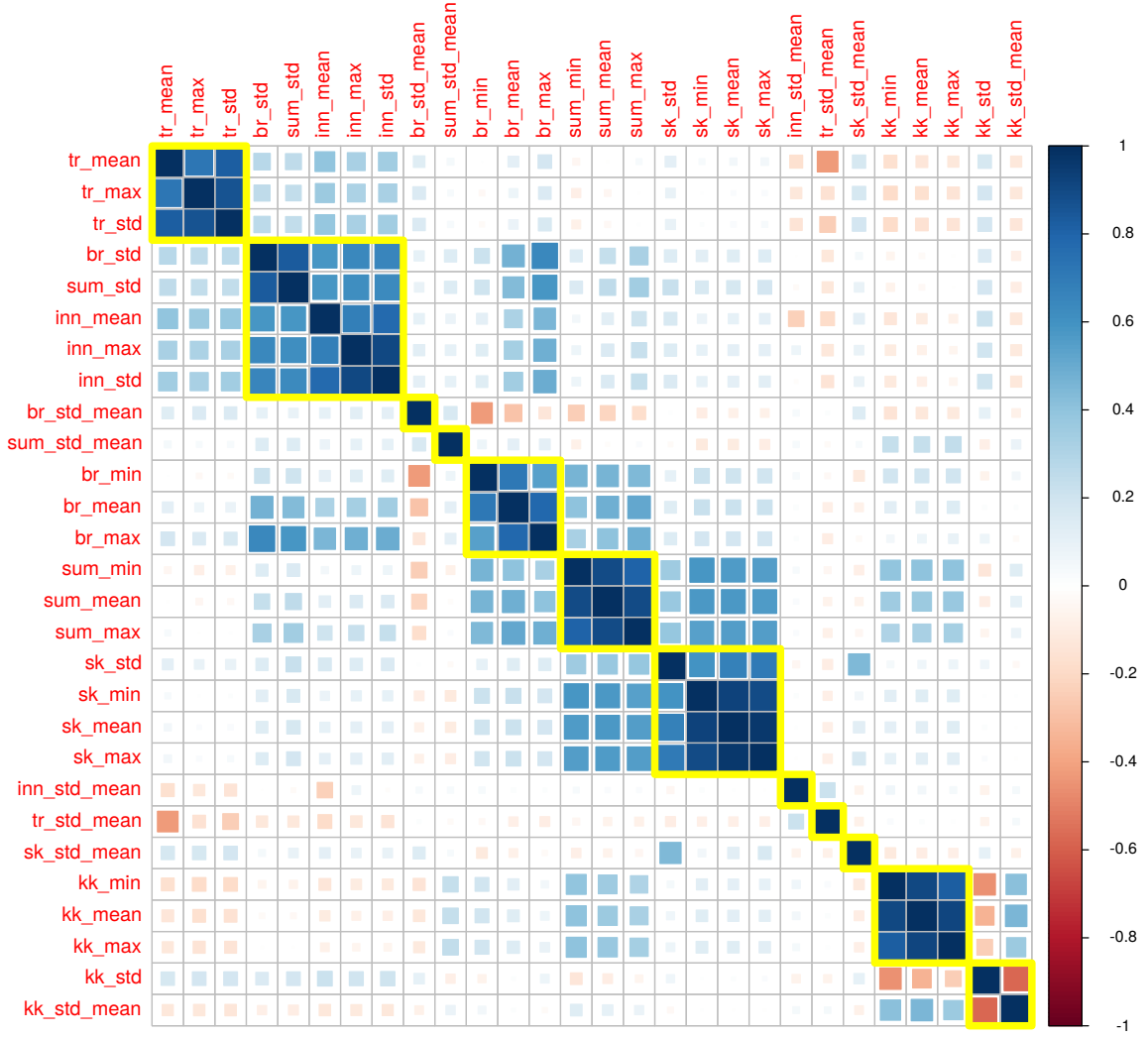
FIGURE 11. Kendall's τ values for the real data set. The chosen clusters are marked with a yellow rectangle.

given as

$$d_{J,K} = \max_{j \in J, k \in K} D_{j,k}.$$

Further, suppose now that J and K are clusters that are joined into a new cluster, and C is a any other cluster. The distance between the joined cluster $J \cup K$ and cluster C is then defined as

$$\max(d_{J,C}, d_{K,C}).$$

The number of clusters can be chosen either by a user chosen dissimilarity level or some sort of optimal choice of clusters. We used the minimum value of the Kelley-Gardner-Sutcliffe penalty function for a hierarchical cluster tree (Kelley et al., 1996). In the implementation of this method in White & Gramacy (2012), there is an additional tuning parameter α with default value equal to 1 that can be used to scale the optimal number of clusters.
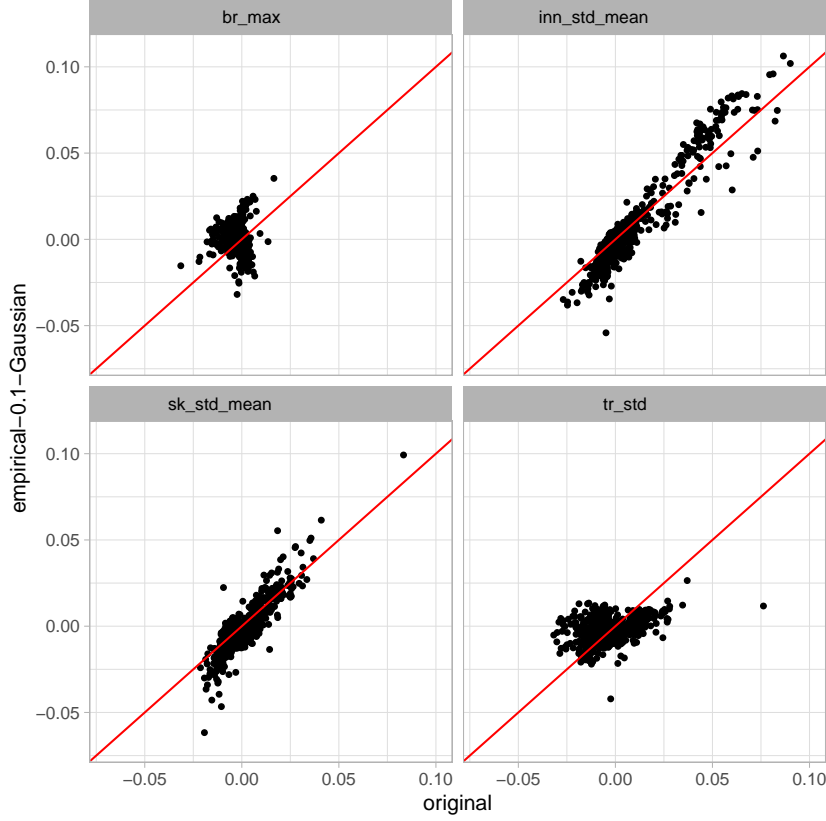
FIGURE 12. Shapley values estimated with the original Kernel SHAP approach against those estimated with the empirical-0.1+Gaussian method for four of the features.

5.3. **Real data example continued.** Here, we continue with the example from Section 4.4. Using the Kelley-Gardner-Sutcliffe penalty function with $\alpha = 0.1$, the features were clustered into twelve groups, denoted g1, g2, . . . , g12 from left to right in Figure 11. Hence, g1 contains the features `tr_mean`, `tr_max` and `tr_std` and g3 contains `br_std_mean` only. Figure 13 shows the combined Shapley values for the twelve groups for three example predictions; one with high, one with low and one with medium predicted probability of default. In all three examples it is apparent that the original Kernel SHAP method has a ranking of groups that is very different from the empiricial-0.1+Gaussian method, which we believe is more correct. Take for instance the high-probability example in the upper left corner of the figure. According to the empirical-0.1+Gaussian method, g12 is the most important group, g5 is the second most important group and so on. The original Kernel SHAP method also ranks *g5* as the second most important group. However, for this method g6 is the most important, while g12 is not among the top 5.

Even the sign of the effect can be wrong with the original Kernel SHAP method. For the medium probability prediction, the contribution from g12 is negative for the original Kernel SHAP method, while it is positive for the empricial-0.1+Gaussian method. Note also that for the empricial-0.1+Gaussian method, some groups are very important for some predictions, while they are much less important for others. For example, g12 is the most important group for the medium and high probability predictions, while it is ranked 7th for the low probability one.

6. Summary and discussion

Shapley values is the only method for explaining individual predictions with a solid theoretical foundation. The main disadvantage with this method is that the computational time
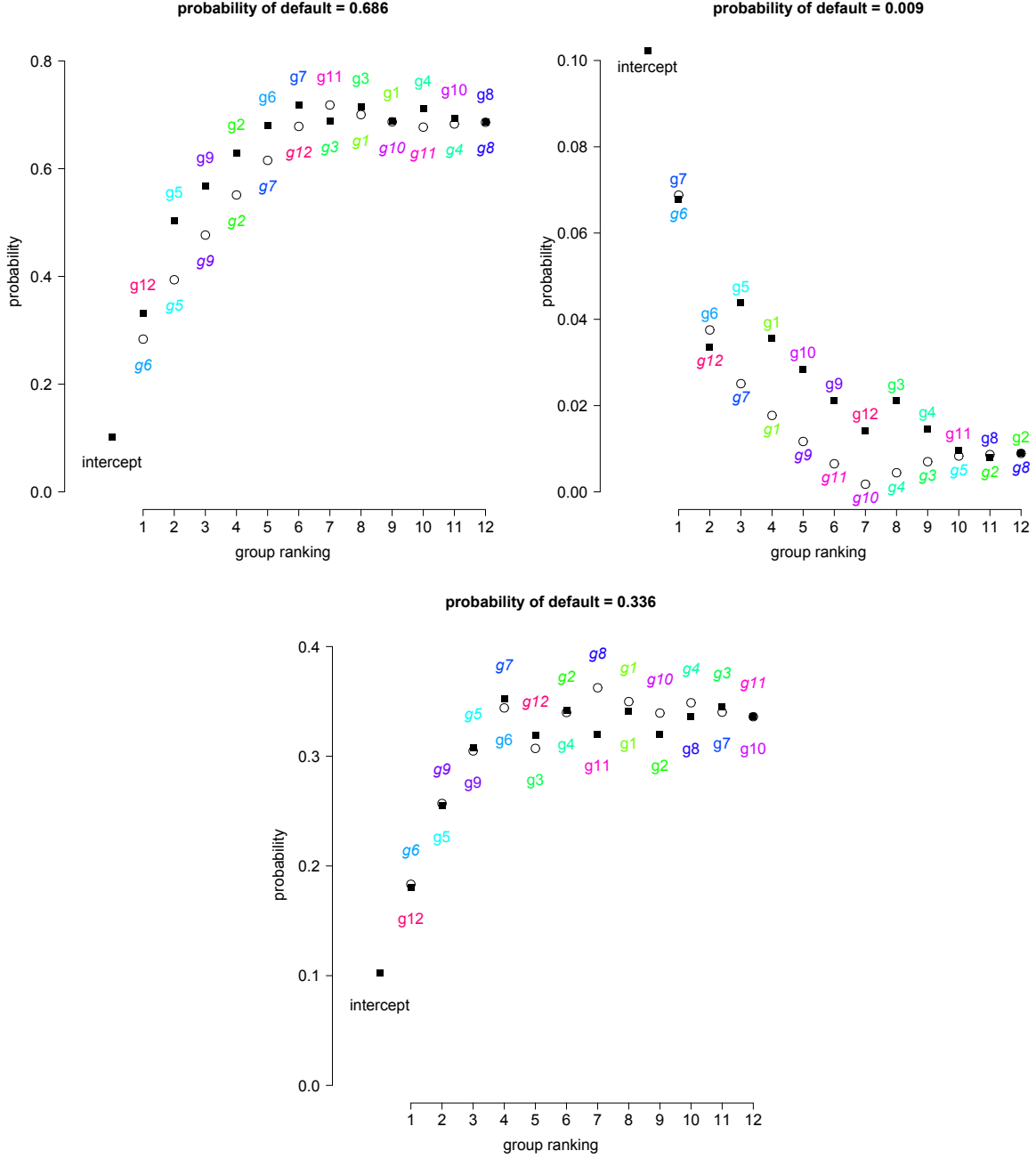
FIGURE 13. Contribution to the probability of default from the twelve groups of features according to the clustering in Figure 11. The filled squares and uppercase group numbers represent the empirical-0.1+Gaussian method, while the open cirles and group numbers in *italics* represent the original Kernel SHAP method.

grows exponentially. 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. It is assumed that the features in the two subsets are independent, meaning that the conditional distribution may be replaced by the marginal distribution of the features in $\bar{\mathcal{S}}$. If there is a high degree of dependence among some or all the features, the resulting explanations might be very wrong. This paper introduces a modified version of the Kernel SHAP method, which handles

dependent features. We have proposed three different approaches for estimating the conditional distribution; assuming a Gaussian multivariate distribution for all features, assuming a Gaussian copula with empirical margins, and using a non-parametric 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 Tree SHAP method, which, as far as we are concerned, 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 have proposed a fourth approach, in which the non-parametric and the parametric methods are combined. All approaches were 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 Tree SHAP and original Kernel SHAP methods.

If features are dependent, it is more challenging to interpret the resulting Shapley values. We have therefore proposed a method for aggregating individual Shapley values. Using hierarchical agglomerative clustering of the training data, the features are divided into a number of clusters. Having obtained the individual Kernel SHAP values using one of the approaches above, we compute the sum of these values for each cluster and visually present these sums instead of the individual values.

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. Very recently, there has been some attempts at using the underlying graph structure of the input data to reduce the computational complexity (Chen et al., 2018; Li et al., 2018). 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 main challenge with this method 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 is likely to 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 a such method needs to handle a large number of binary attributes. An alternative approach is to use ideas from the clustering literature (Huang, 1997, 1998) 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 de Leon & Carriere (2005). When it comes to the parametric approaches, categorical data represents a greater challenge. The most promising alternative might be to use entity embedding (Guo & Berkhahn, 2016) to convert the categorical

features into numerical ones, and then treat these features similarly to the other numerical features.

Appendix A. 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 (Wei et al., 2019) 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

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

Note that GH* means that a slightly different parameterization of the GH distribution is used for the conditional distribution. Here, due to technical reasons, we use the parameterization proposed by McNeil et al. (2006):

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

References

Angwin, J., Larson, J., Mattu, S. & Kirchner, L. (2016). Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. *ProPublica, May* **23**.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K. & Müller, K.-R. (2010). How to explain individual classification descisions. *Journal of Machine Learning Research* **11**, 1803–1831.

Bertin, K., Lacour, C. & Rivoirard, V. (2016). Adaptive pointwise estimation of conditional density function. *Ann. Inst. H. Poincar Probab. Statist.* **52**, 939–980.

Bierens, H. J. (1994). The Nadaraya-Watson kernel regression function estimator. In *Topics in Advanced Econometrics*. Cambridge University Press.

Browne, R. P. & McNicholas, P. D. (2015). A mixture of generalized hyperbolic distributions. *Canadian Journal of Statistics* **43**, 176–198.

Chen, J., Song, L., Wainwright, M. J. & Jordan, M. I. (2018). L-shapley and c-shapley: Efficient model interpretation for structured data. *CoRR* **abs/1808.02610**.

CHEN, T. & GUESTRIN, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM.

DE LEON, A. R. & CARRIERE, K. C. (2005). A generalized mahalanobis distance for mixed data. *Journal of Multivariate Analysis* **92**, 174 – 185.

DRESSEL, J. & FARID, H. (2018). The accuracy, fairness, and limits of predicting recidivism. *Science Advances* **4**, 1–5.

EUROPEAN UNION (2016). 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). *Official Journal of the European Union* **59**.

GNEITING, T. & RAFTERY, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association* **102**, 359–378.

GOOD, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika* **40**, 237–264.

GUO, C. & BERKHAHN, F. (2016). *Entity Embeddings of Categorical Variables.* arXiv:1604.06737v1.

HOLMES, M. P., GRAY, A. G. & ISBELL, C. L. (2010). Fast kernel conditional density estimation: A dual-tree monte carlo approach. *Computational Statistics & Data Analysis* **54**, 1707–1718.

HUANG, Z. (1997). Clustering large data sets with mixed numeric and categorical variables. In *Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference*.

HUANG, Z. (1998). Extensions to the k-Means Algorithm for Clustering Large Data sets with Categorical variables. *Data Mining and Knowledge Discovery* **2**, 283–304.

HURVICH, C. M., SIMONOFF, J. S. & TSAI, C.-L. (1998). Smoothing parameter selection in nonparametric regression using an improved akaike information criterion. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* **60**, 271–293.

IZBICKI, R. & B. LEE, A. (2017). Converting high-dimensional regression to high-dimensional conditional density estimation. *Electron. J. Statist.* **11**, 2800–2831.

IZBICKI, R. & LEE, A. B. (2017). Converting high-dimensional regression to high-dimensional conditional density estimation. *Electronical Journal of Statistics* **11**, 2800–2831.

KELLEY, L. A., GARDNER, S. P. & SUTCLIFFE, M. J. (1996). An automated approach for clustering an ensemble of NMR-derived protein structures into conformationally related subfamilies. *Protein Engineering, Design and Selection* **9**, 1063–1065.

KENDALL, M. G. (1938). A new measure of rank correlation. *Biometrika* **30**, 81–93.

KVAMME, H., SELLEREITE, N., AAS, K. & SJURSEN, S. (2018). Predicting mortgage default using convolutional neural networks. *Expert Systems with Applications* **102**, 207–217.

LI, X., DVORNEK, N. C., ZHOU, Y., ZHUANG, J., VENTOLA, P. & DUNCAN, J. S. (2018). Efficient interpretation of deep learning models using graph structure and cooperative game theory: Application to ASD biomarker discovery. *CoRR* **abs/1812.06181**.

LUNDBERG, S. M. & LEE, S.-I. (2017a). A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*. Curram Associates Inc.

LUNDBERG, S. M. & LEE, S.-I. (2017b). Consistent feature attribution for tree ensembles. In *Proceedings of the 34 th International Conference on Machine Learning*. JMLR: W&CP.

MAHALANOBIS, P. C. (1936). On the generalised distance in statistics. In *Proceedings of the National Institute of Sciences of India*.

MCNEIL, A. J., FREY, R. & EMBRECHTS, P. (2006). *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press.

MOLNAR, C. (2019). *Interpretable Machine Learning - A Guide for Making Black Box Models explainable.* https://leanpub.com.

MÜLLNER, D. (2013). fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. *Journal of Statistical Software* **53**, 1–18.

OBERMEYER, Z. & EMANUEL, E. J. (2016). Predicting the Future – Big Data, Machine Learning, and Clinical Medicine. *New England Journal of Medicine* **375**, 1216.

Owen, A. B. (2014). Sobol indices and Shapley value. *SIAM/ASA Journal on Uncertainty Quantification* **2**, 245–251.

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

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016). Why should I trust you? Explaining predictions of any classifier. In *Proceeedings of te 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM.

Rosenblatt, M. (1956). Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics* **27**, 832–837.

Shapley, L. S. (1953). A Value for N-Person Games. *Contributions to the Theory of Games* **2**, 307–317.

Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publ. Inst. Stat. Univ. Paris* **8**, 229–231.

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

Sudjianto, A., Nair, S., Yuan, M., Zhang, A., Kern, D. & Cela-Díaz, F. (2010). Statistical methods for fighting financial crimes. *Technometrics* **52**, 5–19.

Štrumbel, E. & Kononenko, I. (2010). An Efficient Explanation of Individual Classifications using Game Theory. *Journal of Machine Learning Research* **11**, 1–18.

Štrumbel, E. & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems* **41**, 647–665.

Wei, Y., Tang, Y. & McNicholas, P. D. (2019). Mixtures of generalized hyperbolic distributions and mixtures of skew-t distributions for model-based clustering with incomplete data. *Computational Statistics & Data Analysis* **130**, 18 – 41.

White, D. & Gramacy, R. B. (2012). *maptree: Mapping, pruning, and graphing tree models*. R package version 1.4-7.

Young, H. P. (1985). Monotonic solutions of cooperative games. *International Journal of Game Theory* **14**, 65–72.