Understanding Implications of Dataset Choice for Feature Effect Estimation: A Simulation-Based Investigation through Error Decomposition

Timo Heiß[1]

Department of Statistics, LMU Munich, Ludwigstr. 33, 80539 Munich, Germany
`t.heiss@campus.lmu.de`

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: Explainable AI · Feature effects · Partial dependence plot · Accumulated local effects

1 Introduction

Most Machine Learning (ML) models can be considered black boxes — opaque systems that intrinsically do not allow insight into their internal reasoning, making it often impossible to explain their decisions. However, this can be problematic in many domains and applications, such as the healthcare, legal, or finance sectors, where decisions must be transparent and accountable [1].

Interpretability is crucial to enhance trust [29, 30], address potential biases [13], fairness and ethical concerns [18], and ensure compliance with regulations such as the EU's General Data Protection Regulation (GDPR) [5] and AI Act [6]. To address these challenges, the field of Explainable AI / Interpretable ML has emerged [1]. Although there are many different methods[1], we will focus on feature effect methods like *Partial Dependence Plots (PDP)* [9] and *Accumulated Local Effects (ALE)* [2].

Due to the severity of many applications, it is crucial to utilize these explainability methods correctly. In general, there are many pitfalls to be aware of [27], including whether to compute explanations in-sample, i.e. on training data, or out-of-sample, which we refer to as validation data in the following. In loss-based methods such as *Permutation Feature Importance (PFI)* [4, 7], this choice is crucial and has already been studied (e.g., in [27]). Nevertheless, other explainability methods, such as the *Mean Decrease in Impurity (MDI)* (or *Gini Importance*) of Random Forests, or *SHAP* values [20, 21], have also been found to exhibit biases when computed on training data.

However, to the best of our knowledge, there exist no similar studies for feature effect methods like PDP and ALE. Existing works, including the original papers of PDP and ALE, rely on training data without further justification (e.g., [2, 9, 25]). In contrast, practitioners often advocate for using unseen

[1] For an overview of Explainable AI methods, see e.g. [1, 17, 25].

test/validation data[2], or base their choice on practical constraints like dataset size[3]. While the training set is usually larger and might thus lead to less variance in the feature effect estimates, a too large dataset can increase computation times substantially, particularly for the PDP [9]. On the other hand, although feature effects are not based on generalization error like PFI, it is not clear how much they are affected by overfitting or distribution shifts between training and validation data.

In this paper, we aim to answer this largely unaddressed, fundamental methodological question of whether to use training or validation data to compute feature effects. We perform an empirical simulation study, comparing feature effect error and uncertainty for PDP and ALE across training data, validation data, and cross-validation scenarios, considering various data scenarios and model types. Our main contributions in this paper are as follows.

1. We shed light on the question of whether to compute feature effects on training data, validation data, or in a cross-validated manner, grounded through our comprehensive simulation study, considering feature effect error, bias, and variance across different models and data scenarios.
2. We define methods to quantify feature effect error and uncertainty in a theoretical framework, building upon previous work in this area.
3. We provide an overview of commonly used test functions for simulation studies in Interpretable ML, including applications and purposes.

These contributions have several important implications: Our empirically grounded recommendations enable practitioners and researchers to make informed decisions about which dataset to select for feature effect computation, helping to understand potential implications of their choices. Additionally, our framework for evaluating feature effects and our systematic collection of test functions provide a foundation for future research in Interpretable ML, with the latter specifically facilitating test function choice for simulation studies.

The remainder of this paper is structured as follows. In Section 2, we introduce the considered feature effect methods PDP and ALE, as well as related works on feature effect error and uncertainty, and give an overview of common test functions for simulation studies. In Section **??**, we extend existing works and give our definitions of feature effect errors and uncertainty. We then describe the methodology and set-up of our simulation studies in Section 4, present the results in Section 5, and discuss their implications and limitations in Section 6. In Section 7, we briefly conclude our work.

[2] see, e.g. https://github.com/SauceCat/PDPbox/issues/68 (10/27/2024)
[3] see, e.g. https://forums.fast.ai/t/partial-dependence-plot/98465 (10/27/2024)

2 Background & Related Work

2.1 Feature Effects

The *Partial Dependence Plot (PDP)* by Friedman [9] describes the marginal effect of one or two features on the prediction of a model $\hat{f}$. For a feature set X_S (with $S \subseteq \{1, \dots, p\}$, $|S| = 1$ or $|S| = 2$), the PDP is defined as

$$PDP_{\hat{f},S} = \mathbb{E}_{X_C}[\hat{f}(x_S, X_C)] = \int f(x_S, x_C) d\mathbb{P}(x_C), \tag{1}$$

where X_C is the complement feature subset. $PDP_{\hat{f},S}$ is a function of x_S and can be estimated by Monte Carlo integration:

$$\widehat{PDP}_{\hat{f},S}(x_S) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)}). \tag{2}$$

Here, $x_C^{(i)}$ are the actual complement feature values from the dataset of n instances. To plot this function, a grid of G grid points $\{(x_S^{(g)}, \widehat{PD}_{\hat{f},S}(x_S^{(g)}))\}_{g=1}^{G}$ can be used [26]. Molnar et al. [27] recommend using quantile-based over equidistant grids.

The PDP assumes that the features in S are independent of the features in C. If this is violated, the perturbations may produce unrealistic data points outside the underlying joint distribution of the data. This extrapolation issue can cause misleading interpretations [25, 27].

The *Accumulated Local Effects (ALE)* plot is an alternative to the PDP that solves the extrapolation issue [2]. Using the notation above, for $|S| = 1$, the ALE plot is defined as

$$ALE_{\hat{f},S}(x_S) = \int_{x_{\min,s}}^{x_S} \mathbb{E}_{X_C|X_S}\left[\hat{f}^S(X_S, X_C)|X_S = z_S\right] dz_S - \text{constant} \tag{3}$$

$$= \int_{x_{\min,s}}^{x_S} \int_{x_C} \hat{f}^S(z_S, x_C)\mathbb{P}(x_C|z_S) dx_C dz_S - \text{constant}, \tag{4}$$

where $\hat{f}^S(x_S, x_C) = \frac{\partial \hat{f}(x_S, x_C)}{\partial x_S}$. The constant is chosen so that $\widehat{ALE}_{\hat{f},S}(X_S)$ is centered with a mean of 0 w.r.t. the marginal distribution of X_S. The uncentered ALE can be estimated by

$$\widetilde{\widehat{ALE}}_{\hat{f},S}(x) = \sum_{k=1}^{k_S(x)} \frac{1}{n_S(k)} \sum_{i: x_S^{(i)} \in N_S(k)} \left[\hat{f}(z_{k,S}, x_C^{(i)}) - \hat{f}(z_{k-1,S}, x_C^{(i)})\right]. \tag{5}$$

Here, $\{N_S(k) = (z_{k-1,S}, z_{k,S}]\}_{k=1}^{K}$ partitions the samples $\{x_S^{(i)}\}_{i=1}^{n}$ into K intervals or neighborhoods $N_S(k)$. $n_S(k)$ denotes the number of observations in the kth interval $N_S(k)$, $k_S(x)$ represents the index of the interval to which a particular value x of feature x_S belongs. The uncentered ALE is centered by

$$\widehat{ALE}_{\hat{f},S}(x) = \widehat{\widetilde{ALE}}_{\hat{f},S}(x) - \frac{1}{n}\sum_{i=1}^{n}\widehat{\widetilde{ALE}}_{\hat{f},S}(x_S^{(i)}) \tag{6}$$

to have a mean effect of 0. For the grid that defines the intervals, the quantiles of the empirical distribution of $\{x_S^{(i)}\}_{i=1}^n$ can be used [2].

Other feature effect methods include the *M-Plot (Marginal Plot)*, which, however, suffers from the omitted variable bias [2, 9], or *functional ANOVA (fANOVA)*, which decomposes feature effects into main and interaction effects [16].

Furthermore, methods exist that extend previous ones, such as *Robust and Heterogeneity-aware ALE (RHALE)* [11], or *Accumulated Total Derivative Effect (ATDEV)* plots, which can be decomposed into ALE and *Accumulated Cross Effects (ACE)* [19].

In addition to these global feature effect methods, there are regional effect plots such as *REPID* [15], as well as local methods, including *ICE (Individual Conditional Expectation)* curves [12] or SHAP dependence plots (e.g., [25]) based on SHAP values [22].

For the remainder of this paper, we will focus on PDP and ALE as the most popular global feature effect methods and refer to them when speaking of feature effects.

2.2 Feature Effect Error Decomposition

To quantify the error of a computed feature effect, a "ground truth" needs to be defined first. We follow the approach of Molnar et al. [26] and define ground truth versions of PDP and ALE directly on the data generating process (DGP) by applying PDP and ALE to the underlying ground truth function f (instead of model $\hat{f}$).

For PDP, we can directly use the definition of Molnar et al. [26]:

Definition 1 (Definition 1 from [26]). *The PDP ground truth is the PDP applied to function $f : \mathcal{X} \rightarrow \mathcal{Y}$ of the data generating process.*

$$PDP_{f,S}(x_S) = \mathbb{E}_{X_C}[f(x_S, X_C)] \tag{7}$$

As stated by Molnar et al. [26], their results also apply to conditional variants of the PDP such as ALE. We now make this definition explicit:

Definition 2. *The ALE ground truth is the ALE applied to function $f : \mathcal{X} \rightarrow \mathcal{Y}$ of the data generating process.*

$$ALE_{f,S}(x_S) = \int_{x_{min,s}}^{x_S} \mathbb{E}_{X_C|X_S}\left[f^S(X_S, X_C)|X_S = z_S\right] dz_S - constant \tag{8}$$

where $f^S(x_S, x_C) = \frac{\partial f(x_S, x_C)}{\partial x_S}$ *and constant chosen such that the effect has a mean of* 0 *w.r.t. the marginal distribution of* X_S.

Note that different ground truth effects may also be defined, and our choices come with certain implications and limitations, such as omitting the *aggregation bias*[4] [24].

With more complex ground truth functions f, it may become increasingly difficult to derive the ground truth feature effects analytically, especially for ALE. In these cases, we therefore propose to also estimate those effects by Monte Carlo integration, yielding $\widehat{PDP}_{f,S}(x_S)$ and $\widehat{ALE}_{f,S}(x_S)$ (obtained by plugging in f instead of $\hat{f}$ into the estimators in equations (2) and (5) / (6)).

Summarizing, we have now defined four quantities per feature effect: $PDP_{f,S}$ $\widehat{PDP}_{f,S}$, $PDP_{\hat{f},S}$, and $\widehat{PDP}_{\hat{f},S}$ (analogue for ALE). We can now define different errors between each of these quantities. In this paper, we focus on the MSE, as it can be decomposed into bias and variance (see e.g. [10]). Taking, for example, $PDP_{f,S}$ as ground truth, we can define the MSE of $PDP_{\hat{f},S}$ at a point x_S as follows [26]:

$$\text{MSE}(x_S; PDP_{f,S}, PDP_{\hat{f},S}) = \mathbb{E}_F[(PDP_{f,S}(x_S) - \widehat{PDP}_{\hat{f},S}(x_S))^2] \tag{9}$$

$$= \underbrace{(PDP_{f,S}(x_S) - \mathbb{E}_F[PDP_{\hat{f},S}(x_S)])^2}_{Bias^2} + \underbrace{\text{Var}_F[PDP_{\hat{f},S}(x_S)]}_{Variance} \tag{10}$$

Here, F denotes the distribution of trained models. The bias is linked to the bias of the model, the variance comes from the variance in the model fits (randomness in training data, randomness in model training procedure).

Since we usually cannot determine $PDP_{\hat{f},S}$, we need to estimate it by Monte Carlo integration, yielding $\widehat{PDP}_{\hat{f},S}(x_S)$. This, however, introduces an additional variance term (we use the random variable X_{mc} for the Monte Carlo samples (e.g., training or validation data)):

$$\begin{aligned}\text{MSE}(x_S; PDP_{f,S}, \widehat{PDP}_{\hat{f},S}) &= \underbrace{(PDP_{f,S}(x_S) - \mathbb{E}_F[PDP_{\hat{f},S}(x_S)])^2}_{Bias^2} \\ &+ \underbrace{\text{Var}_F[PDP_{\hat{f},S}(x_S)]}_{Variance} + \underbrace{\mathbb{E}_F \text{Var}_{X_{mc}}[\widehat{PDP}_{\hat{f},S}(x_S)]}_{MC-Variance}\end{aligned} \tag{11}$$

Proof. For better readability, we will omit the subscript S as well as the point x_S and use $X = X_{mc}$ in this proof:

[4] for details, see [15]

$$\begin{tikzcd}
\widehat{PDP}_f & PDP_f \arrow[l, "\text{MC}"'] \arrow[r, "\text{Bias}"] & \mathbb{E}_F PDP_{\hat{f}} \arrow[r, "\text{Variance}"] & PDP_{\hat{f}} \arrow[r, "\text{MC}"] & \widehat{PDP}_{\hat{f}} \\
 & f \arrow[u] \arrow[d] \arrow[r, "\text{Model Bias}"] & \mathbb{E}_F[\hat{f}] \arrow[u] \arrow[d] \arrow[r, "\substack{\text{Model}\\ \text{Variance}}"] & \hat{f} \arrow[u] \arrow[d] & \\
\widehat{ALE}_f & ALE_f \arrow[l, "\text{MC}"'] \arrow[r, "\text{Bias}"] & \mathbb{E}_F ALE_{\hat{f}} \arrow[r, "\text{Variance}"] & ALE_{\hat{f}} \arrow[r, "\text{MC}"] & \widehat{ALE}_{\hat{f}}
\end{tikzcd}$$

Fig. 1. Error chain in feature effect estimation (modified, original version can be found in [26])

$$\begin{aligned}
\text{MSE}(PDP_f, \widehat{PDP}_{\hat{f}}) &= \mathbb{E}_F \mathbb{E}_X[(PDP_f - \widehat{PDP}_{\hat{f}})^2] \\
&= \mathbb{E}_F \mathbb{E}_X[PDP_f^2 - 2PDP_f \widehat{PDP}_{\hat{f}} + \widehat{PDP}_{\hat{f}}^2] \\
&= PDP_f^2 - 2PDP_f \mathbb{E}_F[PDP_{\hat{f}}] + \mathbb{E}_F \mathbb{E}_X[\widehat{PDP}_{\hat{f}}^2] \\
&= PDP_f^2 - 2PDP_f \mathbb{E}_F[PDP_{\hat{f}}] + \mathbb{E}_F \text{Var}_X[\widehat{PDP}_{\hat{f}}] + \mathbb{E}_F[\mathbb{E}_X[\widehat{PDP}_{\hat{f}}]^2] \\
&= PDP_f^2 - 2PDP_f \mathbb{E}_F[PDP_{\hat{f}}] + \mathbb{E}_F \text{Var}_X[\widehat{PDP}_{\hat{f}}] + \text{Var}_F(\mathbb{E}_X[\widehat{PDP}_{\hat{f}}]) \\
&\quad + \mathbb{E}_F(\mathbb{E}_X[\widehat{PDP}_{\hat{f}}])^2 \\
&= PDP_f^2 - 2PDP_f \mathbb{E}_F[PDP_{\hat{f}}] + Var_F[PDP_{\hat{f}}] + \mathbb{E}_F[PDP_{\hat{f}}]^2 + \mathbb{E}_F \text{Var}_X[\widehat{PDP}_{\hat{f}}] \\
&= (PDP_f - \mathbb{E}_F[PDP_{\hat{f}}])^2 + \text{Var}_F[PDP_{\hat{f}}] + \mathbb{E}_F[\text{Var}_X(\widehat{PDP}_{\hat{f}})]
\end{aligned}$$

At multiple points, we use the fact that $\mathbb{E}_X[\widehat{PDP}_{\hat{f}}] = PDP_{\hat{f}}$ (cf. [26]).

We see that the variance due to MC integration also depends on the model distribution F. Similarly, one could use the estimate $\widehat{PDP}_{f,S}$ as groundtruth, introducing an additional variance term $Var_{X_{mc2}}$when estimated on different MC sample (proof in Appendix A). An overview of all error terms in this chain can be found in **Fig. 1**.

$\mathbb{E}_F$ and Var_F could be estimated by averaging over multiple models of the same inducer fitted to M different training data sets sampled independently from the DGP. We propose the following estimators:

$$\widehat{\text{MSE}}(x_S; PDP_{f,S}, \widehat{PDP}_{\hat{f},S}) = \frac{1}{M} \sum_{m=1}^{M} \left(PDP_{f,S}(x_S) - \widehat{PDP}_{\hat{f}^{(m)},S}(x_S)\right)^2 \quad (12)$$

$$\widehat{\text{Bias}}(x_S; PDP_{f,S}, \widehat{PDP}_{\hat{f},S}) = (PDP_{f,S}(x_S) - \frac{1}{M} \sum_{m=1}^{M} \widehat{PDP}_{\hat{f}^{(m)},S}(x_S)) \quad (13)$$

$$\widehat{\text{Variance}}(x_S; \widehat{PDP}_{\hat{f},S}) = \frac{1}{M-1} \sum_{m=1}^{M} \left(\widehat{PDP}_{\hat{f}^{(m)},S}(x_S) - \frac{1}{M} \sum_{m=1}^{M} \widehat{PDP}_{\hat{f}^{(m)},S}(x_S) \right)^2 \tag{14}$$

Note that these are similar to the approach in [26], but we do not specify which data points to use for MC integration. The variance captures both the variance in the model fits and the variance due to MC integration. To estimate the MC variance, we propose the following estimators:

$$\widehat{\text{Variance}}_{MC}(x_S; PDP_f, \widehat{PDP}_f) = \frac{1}{K} \sum_{k=1}^{K} (PDP_f(x_S) - \widehat{PDP}_f^{(k)}(x_S))^2 \tag{15}$$

and

$$\widehat{\text{Variance}}_{MC}(x_S; \widehat{PDP}_{\hat{f}}) = \frac{1}{M(K-1)} \sum_{m=1}^{M} \sum_{k=1}^{K} \left(\widehat{PDP}_{\hat{f}^{(m)},S}^{(k)}(x_S) - \frac{1}{K} \sum_{k=1}^{K} \widehat{PDP}_{\hat{f}^{(m)},S}^{(k)}(x_S) \right)^2 \tag{16}$$

For more convenient analysis of the errors, one could also aggregate them over the marginal distribution of X_S (e.g., estimated by averaging over the grid points if chosen appropriately) to obtain a single error measure per feature effect.

While our definitions are based on the PDP, they can be directly applied to the ALE as well.

3 Test Functions for Simulation Studies

Test functions play a crucial role in research, e.g. for evaluating and comparing different methodological approaches, or when conducting simulation studies. This section synthesizes commonly used test functions across various domains, providing structured guidance for researchers — particularly those in Interpretable ML — in selecting appropriate test functions for their simulation studies. By examining test functions from different fields and their purposes of application, we aim to facilitate experimental design decisions. While this overview is not exhaustive, it highlights key test functions and their characteristics to help researchers make informed choices for their specific research needs.

Test Functions in Optimization. The field of optimization, where test functions are essential to enable the assessment and comparison of optimization algorithms, has established a rich foundation of test functions. A fundamental approach involves using simple mathematical expressions like the sphere function [28]. These basic functions are often combined with more complex ones like Branin or Rosenbrock to create comprehensive test suites that incorporate important properties such as nonlinearity, non-separability, and scalability [33]. A notable framework in this domain is the Comparing Continuous Optimizer (COCO) platform with its Black Box Optimization Benchmark (BBOB), offering a structured approach to testing continuous optimization algorithms through artificial test functions [14]. These classical test function suites are well-established in optimization and may also serve as a basis for Interpretable ML researchers. However, it is important to note that the ability of these artifical test functions to represent complex real-world behavior is limited [34].

Physics-Inspired Test Functions. Physics-derived functions offer a compelling source of real-world test cases, with the Feynman Symbolic Regression Database (FSReD) being a prominent example. FSReD comprises 100 physics equations from the seminal *Feynman's Lectures on Physics (34–36)*, supplemented by 20 more challenging equations from other seminal physics texts [32]. These equations span diverse physics domains including classical mechanics, electromagnetism, and quantum mechanics. They involve between one and nine independent variables and incorporate various elementary functions such as arithmetic operations, trigonometric functions, and exponentials. From that, tabular datasets were generated through random sampling from defined value ranges.

Building upon this foundation, Matsubara et al. [23] addressed several limitations of the original FSReD. Their enhanced framework introduces a three-tiered categorization of problems (easy, medium, hard) based on their complexity, incorporates dummy variables to simulate irrelevant features, and implements more realistic sampling ranges and strategies. Detailed specifications for all formulas, including their sampling parameters, are available in their work.

While initially developed for symbolic regression tasks, many Feynman equations may serve as suitable test functions for simulation studies in Interpretable ML. Their basis in physical principles provides real-world relevance, though researchers should carefully select equations that align with their specific analytical objectives and complexity requirements.

Test Functions for Interpretable Machine Learning. The Interpretable ML field itself has developed several specialized test functions designed to evaluate specific aspects of interpretability methods.

Goldstein et al. [12] used several simple test functions to demonstrate the behavior of Individual Conditional Expectation (ICE) curves. These include a simple additive function to demonstrate the absence of interactions, simple interactions to reveal heterogeneity that might be obscured by averaging procedures such as PDPs, and a specially designed function with an empty quadrant for assessing extrapolation behavior.

Similarly, Liu et al. [19] focus on simple functions before progressing to more complex ones. They begin with basic two-variable scenarios — using additive functions, interaction functions, and combinations thereof — and examine these under both independent and correlated feature conditions to compare various feature effect methods. The advantage of these simple test functions is that solutions (e.g., feature effects) can also be computed analytically, and that they allow for deeper and more fine-grained analysis of individual aspects.

A more complex test function suite was proposed by Tsang et al. [31], specifically designed to evaluate the detection of variable interactions. Their functions incorporate various types of interactions with different orders, strengths, non-linearities, and overlaps. While this makes them particularly valuable for interaction detection, they are also useful for evaluating other interpretability methods in scenarios with complex interactions.

The Friedman functions [3, 8] serve as classical benchmarks applicable across various interpretability tasks. These three functions combine linear and non-linear effects with interactions, incorporating dummy variables and random noise terms to reflect realistic complexity. For detailed specifications, see [3].

When choosing test functions for simulation studies in Interpretable ML, researchers should consider several criteria, including the specific aspects of interpretability being evaluated, the desired complexity level and number of variables, the presence of specific challenges such as correlation between features or interactions, the need for analytical solutions for validation, as well as the relevance to real-world applications in the domain of interest.

4 Methodology & Experimental Set-Up

Datasets. Building upon Section 3, we employ three distinct datasets of varying complexity for our simulation study:

- **SimpleNormalCorrelated** consists of four standard-normally distributed features, where the first two features exhibit strong correlation ($\rho = 0.9$) while the others are independent dummy variables. The target variable is given by this simple formula:

$$f_1(\boldsymbol{x}) = x_1 + \frac{x_2^2}{2} + x_1 x_2 \tag{17}$$

  This test function is inspired by [19] and aims to focus on the impact of correlation and interactions.
- **Friedman1** implements the classical Friedman1 benchmark function [3, 8] with seven uniformly distributed features between 0 and 1, all mutually independent:

$$f_2(\boldsymbol{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - \frac{1}{2})^2 + 10x_4 + 5x_5 \tag{18}$$

  This dataset includes a mix of linear and different non-linear effects with interactions.
- **Feynman I.29.16** is based on the Feynman equation I.29.16 describing wave interference, comprising six independent features. Using the refined sampling strategies from Matsubara et al. [23], two log-uniformly distributed variables on $[0.1, 10]$, two angles uniformly distributed over $[0, 2\pi]$, and two uniformly distributed dummy variables on [0,1]:

$$f_3(\boldsymbol{x}) = \sqrt{x_1^2 + x_2^2 + 2x_1 x_2 \cos(\theta_1 - \theta_2)} \tag{19}$$

  This dataset is designed to reflect a physics-based relationship for more real-world relevance.

To generate the datasets, a standard normally distributed noise term ϵ is added to each function, scaled by a signal-to-noise ratio of five[5]. For each test function, we consider two dataset sizes: $(n_{train} = 1000, n_{val} = 250)$ and $(n_{train} = 8000, n_{val} = 2000)$.

Model Inducers. We consider the following model inducers for our simulation study:

- **LinReg**: a simple linear regression model as baseline.
- **GAM_OT**: a Generalized Additive Model (GAM) with spline terms and tensor splines for first-order interactions, number of splines and penalization optimally tuned.
- **GAM_OT**: as above, but with hyperparameters chosen to overfit.
- **SVM_OT**: a Support Vector Machine (SVM) with optimally tuned hyperparameters.
- **SVM_OF**: a Support Vector Machine (SVM) with hyperparameters chosen to overfit.
- **XGBoost_OT**: an XGBoost model with optimally tuned hyperparameters.
- **XGBoost_OF**: an XGBoost model with hyperparameters chosen to overfit.

Hyperparameters are pre-selected, i.e. carefully hand-picked for overfitting scenarios and tuned on a different sample for optimal tuning. Details on the hyperparameters and the performance of the models can be found in Appendix B.

5 Results

6 Conclusions

6.1 Summary & Discussion

6.2 Limitations & Future Work

Acknowledgments. A bold run-in heading in small font size at the end of the paper is used for general acknowledgments, for example: This study was funded by X (grant number Y).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

[5] To determine the factor by which the noise is multiplied, the standard deviation of the signal is computed over 100'000 randomly drawn samples of y and divided by the signal-to-noise ratio of five.

References

1. Adadi, A., Berrada, M.: Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)
2. Apley, D.W., Zhu, J.: Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. Journal of the Royal Statistical Society Series B: Statistical Methodology **82**(4), 1059–1086 (Sep 2020)
3. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123–140 (Aug 1996)
4. Breiman, L.: Random Forests. Machine Learning **45**(1), 5–32 (Oct 2001)
5. European Parliament and Council: 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 **L 119**(2016/679), 1–88 (5 2016)
6. European Parliament and Council: Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act). Official Journal of the European Union (2024/1689) (7 2024), PE/24/2024/REV/1
7. Fisher, A., Rudin, C., Dominici, F.: All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. Journal of Machine Learning Research : JMLR **20**(177), 1–81 (2019)
8. Friedman, J.H.: Multivariate Adaptive Regression Splines. The Annals of Statistics **19**(1), 1–67 (Mar 1991), publisher: Institute of Mathematical Statistics
9. Friedman, J.H.: Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics **29**(5), 1189–1232 (2001)
10. Geman, S., Bienenstock, E., Doursat, R.: Neural Networks and the Bias/Variance Dilemma. Neural Computation **4**(1), 1–58 (1992)
11. Gkolemis, V., Dalamagas, T., Ntoutsi, E., Diou, C.: RHALE: Robust and Heterogeneity-aware Accumulated Local Effects (Sep 2023), arXiv:2309.11193v1 [cs.LG]
12. Goldstein, A., Kapelner, A., Bleich, J., Pitkin, E.: Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. Journal of Computational and Graphical Statistics **24**(1), 44–65 (Jan 2015)
13. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A Survey of Methods for Explaining Black Box Models. ACM Computing Surveys **51**(5), 1–42 (Sep 2019)
14. Hansen, N., Auger, A., Mersmann, O., Tusar, T., Brockhoff, D.: COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. Optimization Methods and Software **36** (Mar 2016)
15. Herbinger, J., Bischl, B., Casalicchio, G.: REPID: Regional Effect Plots with implicit Interaction Detection. In: Proceedings of The 25th International Conference on Artificial Intelligence and Statistics. pp. 10209–10233. PMLR (May 2022)
16. Hooker, G.: Discovering additive structure in black box functions. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 575–580. KDD '04, Association for Computing Machinery, New York, NY, USA (Aug 2004)

17. Kamath, U., Liu, J.: Introduction to Interpretability and Explainability. In: Kamath, U., Liu, J. (eds.) Explainable Artificial Intelligence: An Introduction to Interpretable Machine Learning, pp. 1–26. Springer International Publishing, Cham (2021)
18. Lipton, Z.C.: The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue **16**(3), 31–57 (Jun 2018)
19. Liu, X., Chen, J., Vaughan, J., Nair, V., Sudjianto, A.: Model Interpretation: A Unified Derivative-based Framework for Nonparametric Regression and Supervised Machine Learning (Sep 2018), arXiv:1808.07216v2 [cs, stat]
20. Loecher, M.: Debiasing MDI Feature Importance and SHAP Values in Tree Ensembles. In: Holzinger, A., Kieseberg, P., Tjoa, A.M., Weippl, E. (eds.) Machine Learning and Knowledge Extraction. pp. 114–129. Springer International Publishing, Cham (2022)
21. Loecher, M.: Debiasing SHAP scores in random forests. AStA Advances in Statistical Analysis **108**(2), 427–440 (Jun 2024)
22. Lundberg, S.M., Lee, S.I.: A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
23. Matsubara, Y., Chiba, N., Igarashi, R., Ushiku, Y.: Rethinking Symbolic Regression Datasets and Benchmarks for Scientific Discovery (Mar 2024), arXiv:2206.10540v5 [cs.LG]
24. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A Survey on Bias and Fairness in Machine Learning. ACM Comput. Surv. **54**(6) (2021)
25. Molnar, C.: Interpretable machine learning: a guide for making black box models explainable. Christoph Molnar, Munich, Germany, second edition edn. (2022)
26. Molnar, C., Freiesleben, T., König, G., Herbinger, J., Reisinger, T., Casalicchio, G., Wright, M.N., Bischl, B.: Relating the partial dependence plot and permutation feature importance to the data generating process. In: Longo, L. (ed.) Explainable Artificial Intelligence. pp. 456–479. Springer Nature Switzerland, Cham (2023)
27. Molnar, C., König, G., Herbinger, J., Freiesleben, T., Dandl, S., Scholbeck, C.A., Casalicchio, G., Grosse-Wentrup, M., Bischl, B.: General Pitfalls of Model-Agnostic Interpretation Methods for Machine Learning Models. In: Holzinger, A., Goebel, R., Fong, R., Moon, T., Müller, K.R., Samek, W. (eds.) xxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers, pp. 39–68. Springer International Publishing, Cham (2022)
28. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing Unconstrained Optimization Software. ACM Transactions on Mathematical Software **7**(1), 17–41 (Mar 1981)
29. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144. KDD '16, Association for Computing Machinery, New York, NY, USA (Aug 2016)
30. Teach, R.L., Shortliffe, E.H.: An analysis of physician attitudes regarding computer-based clinical consultation systems. Computers and Biomedical Research, an International Journal **14**(6), 542–558 (Dec 1981)
31. Tsang, M., Cheng, D., Liu, Y.: Detecting Statistical Interactions from Neural Network Weights (May 2017), arXiv:1705.04977v4 [stat.ML]

32. Udrescu, S.M., Tegmark, M.: AI Feynman: A physics-inspired method for symbolic regression. Science Advances **6**(16), eaay2631 (Apr 2020), publisher: American Association for the Advancement of Science
33. Whitley, L.D., Mathias, K.E., Rana, S.B., Dzubera, J.: Building Better Test Functions. In: ICGA. pp. 239–247. Citeseer (1995)
34. Zaefferer, M., Fischbach, A., Naujoks, B., Bartz-Beielstein, T.: Simulation-based test functions for optimization algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 905–912. GECCO '17, Association for Computing Machinery, New York, NY, USA (Jul 2017)

A Proof: Monte Carlo Variance for Groundtruth

Proof. For better readability, we will omit the subscript S as well as the point x_S and use $X = X_{mc}$ in this proof:

$$\begin{aligned}
\mathbb{E}_F \mathbb{E}_X[(PDP_f - \widehat{PDP}_f)^2] &= \mathbb{E}_F \mathbb{E}_X[PDP_f^2 - 2PDP_f\widehat{PDP}_f + \widehat{PDP}_f^2] \\
&= PDP_f^2 - 2PDP_f \mathbb{E}_F \mathbb{E}_X[\widehat{PDP}_f] + \mathbb{E}_F \mathbb{E}_X[\widehat{PDP}_f^2] \\
&= PDP_f^2 - 2PDP_f^2 + \mathbb{E}_F \mathrm{Var}_X[\widehat{PDP}_f] + \mathbb{E}_F[\mathbb{E}_X[\widehat{PDP}_f]^2] \\
&= PDP_f^2 - 2PDP_f^2 + \mathbb{E}_F \mathrm{Var}_X[\widehat{PDP}_f] + PDP_f^2 \\
&= \mathbb{E}_F \mathrm{Var}_X[\widehat{PDP}_f] \\
&= \mathrm{Var}_X[\widehat{PDP}_f]
\end{aligned}$$

Again we use $\mathbb{E}_X[\widehat{PDP}_f] = PDP_f$ (cf. [26]) as well as the fact that all quantities based on f do not depend on the model distribution F.

B Model Hyperparameters and Performance

In this appendix, we give details on the models used in the simulation study. Specifically, we provide details on the hyperparameter selection and the finally used hyperparameters, as well as the performance of the models.

B.1 Hyperparameters

An overview of all hyperparameters used for the different models can be found in **Table 1**. Note that the linear regression is excluded as we do not have any hyperparameters to tune.

Hyperparameters for the overfitting models (OF) were carefully hand-picked to achieve strong performance on the training data while performing relatively poorly on the validation data.

The optimal hyperparameters (OT) were chosen by tuning the models on a separate data sample. Training and validation data were sampled independently from the correspondings DGPs, with a training size of n_{train} and a validation size of 10000 to get a reliable performance estimate. While this scenario is unrealistic in practice, it allows as to consider hyperparameters as "pre-selected" and avoid costly nested resampling strategies for the simulation study. Each model was tuned for 200 trials using a *Tree-structured Parzen Estimator (TPE)* [**?**] using validation MSE as objective to minimize (equivalent to maximizing R2-score).

B.2 Model Performance Evaluation

To ensure that the models perform as expected, we evaluate their performance both on the training data and a holdout test data with the latter consisting of

Dataset	**n_train**	**Model**	**Hyperparameters**
Simple Normal Correlated	**1000**	**GAM_OF**	n_bases: 50; lam: 0.0005;
		GAM_OT	n_bases: 20; lam: 15.6807;
		SVM_OF	C: 800; gamma: 10;
		SVM_OT	C: 917.9061; gamma: 0.0030;
		XGBoost_OF	n_estimators: 1200; max_depth: 16; learning_rate: 0.35; subsample: 1.0; min_child_weight: 1; colsample_bytree: 1.0; colsample_bylevel: 1.0; lambda: 0; alpha: 0;
		XGBoost_OT	n_estimators: 1640; max_depth: 5; learning_rate: 0.0062; subsample: 0.5601; min_child_weight: 1.6999; colsample_bytree: 0.7632; colsample_bylevel: 0.6944; lambda: 0.0156; alpha: 0.0660;
	8000	**GAM_OF**	n_bases: 64; lam: 1e-05;
		GAM_OT	n_bases: 5; lam: 0.0010;
		SVM_OF	C: 1000; gamma: 10;
		SVM_OT	C: 864.4724; gamma: 0.0085;
		XGBoost_OF	n_estimators: 1500; max_depth: 18; learning_rate: 0.4000; subsample: 1.0; min_child_weight: 1; colsample_bytree: 1.0; colsample_bylevel: 1.0; lambda: 0; alpha: 0;
		XGBoost_OT	n_estimators: 2586; max_depth: 5; learning_rate: 0.0044; subsample: 0.9484; min_child_weight: 1.4257; colsample_bytree: 0.8471; colsample_bylevel: 0.8672; lambda: 5.1002; alpha: 0.0026;
Friedman1	**1000**	**GAM_OF**	n_bases: 50; lam: 0.0001;
		GAM_OT	n_bases: 21; lam: 0.0402;
		SVM_OF	C: 1000; gamma: 15;
		SVM_OT	C: 917.1949; gamma: 0.2102;
		XGBoost_OF	n_estimators: 1000; max_depth: 14; learning_rate: 0.3; subsample: 1.0; min_child_weight: 1; colsample_bytree: 1.0; colsample_bylevel: 1.0; lambda: 0; alpha: 0;
		XGBoost_OT	n_estimators: 2621; max_depth: 8; learning_rate: 0.0335; subsample: 0.6192; min_child_weight: 5.4066; colsample_bytree: 0.7651; colsample_bylevel: 0.5224; lambda: 11.6021; alpha: 4.5342;
	8000	**GAM_OF**	n_bases: 80; lam: 1e-08;
		GAM_OT	n_bases: 22; lam: 0.0657;
		SVM_OF	C: 1000; gamma: 18;
		SVM_OT	C: 901.1903; gamma: 0.2602;
		XGBoost_OF	n_estimators: 1200; max_depth: 14; learning_rate: 0.3; subsample: 1.0; min_child_weight: 1; colsample_bytree: 1.0; colsample_bylevel: 1.0; lambda: 0; alpha: 0;
		XGBoost_OT	n_estimators: 3691; max_depth: 5; learning_rate: 0.0070; subsample: 0.6643; min_child_weight: 1.4075; colsample_bytree: 0.8403; colsample_bylevel: 0.8186; lambda: 0.0399; alpha: 5.0734;
Feynman I.29.16	**1000**	**GAM_OF**	n_bases: 50; lam: 0.0001;
		GAM_OT	n_bases: 31; lam: 0.3260;
		SVM_OF	C: 200; gamma: 8;
		SVM_OT	C: 11.4317; gamma: 0.1394;
		XGBoost_OF	n_estimators: 1000; max_depth: 14; learning_rate: 0.3; subsample: 1.0; min_child_weight: 1; colsample_bytree: 1.0; colsample_bylevel: 1.0; lambda: 0; alpha: 0;
		XGBoost_OT	n_estimators: 4246; max_depth: 9; learning_rate: 0.0327; subsample: 0.8004; min_child_weight: 6.3792; colsample_bytree: 0.8420; colsample_bylevel: 0.8357; lambda: 14.1131; alpha: 6.0556;
	8000	**GAM_OF**	n_bases: 64; lam: 5e-07;
		GAM_OT	n_bases: 32; lam: 0.4500;
		SVM_OF	C: 400; gamma: 10;
		SVM_OT	C: 22.5167; gamma: 0.1114;
		XGBoost_OF	n_estimators: 1000; max_depth: 14; learning_rate: 0.3; subsample: 1.0; min_child_weight: 1; colsample_bytree: 1.0; colsample_bylevel: 1.0; lambda: 0; alpha: 0;
		XGBoost_OT	n_estimators: 3962; max_depth: 7; learning_rate: 0.0372; subsample: 0.9351; min_child_weight: 4.0962; colsample_bytree: 0.8640; colsample_bylevel: 0.7728; lambda: 29.2036; alpha: 5.1631;

Table 1. Hyperparameters for the models used in the simulation study

10000 samples to get a reliable performance estimate. The performances evaluated over the 30 repetitions are aggregated in **Fig. 2** showing the R2-scores. As intended, the overfitting models perform better on the training data than the optimally tuned models, while being outperformed on the holdout test data and also exhibiting higher variance in their generalization performance. Note that the linear regression model serves only as a baseline. The overfitted SVM (SVM_OF) show substantially worse performances on test data compared to the other models. Therefore, we excluded it from further analysis.

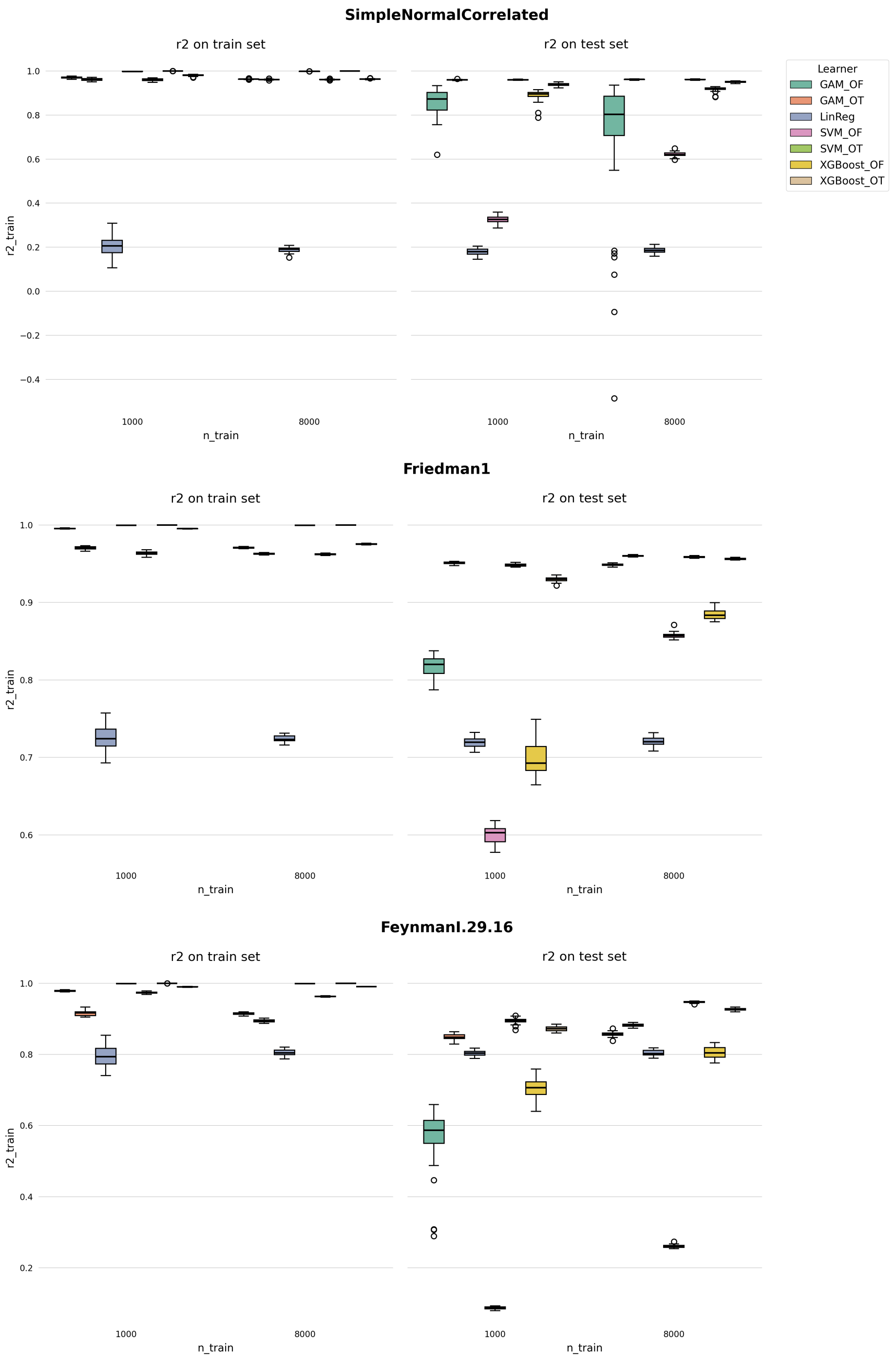

Fig. 2. Performance of the models on training (left) and holdout test data (right), each boxplot aggregates 30 repetitions