Synthetic Benchmarks for Scientific Research in Explainable Machine Learning

Yang Liu*
Abacus.AI
San Francisco, CA 94103
yang@abacus.ai

Sujay Khandagale*
Abacus.AI
San Francisco, CA 94103
sujay@abacus.ai

Colin White
Abacus.AI
San Francisco, CA 94103
colin@abacus.ai

Willie Neiswanger
Stanford University
Stanford, CA 94305
neiswanger@cs.stanford.edu

Abstract

As machine learning models grow more complex and their applications become more high-stakes, tools for explaining model predictions have become increasingly important. This has spurred a flurry of research in model explainability and has given rise to feature attribution methods such as LIME and SHAP. Despite their widespread use, evaluating and comparing different feature attribution methods remains challenging: evaluations ideally require human studies, and empirical evaluation metrics are often data-intensive or computationally prohibitive on real-world datasets. In this work, we address this issue by releasing XAI-BENCH: a suite of synthetic datasets along with a library for benchmarking feature attribution algorithms. Unlike real-world datasets, synthetic datasets allow the efficient computation of conditional expected values that are needed to evaluate ground-truth Shapley values and other metrics. The synthetic datasets we release offer a wide variety of parameters that can be configured to simulate real-world data. We demonstrate the power of our library by benchmarking popular explainability techniques across several evaluation metrics and identifying surprising failure modes even for the most widely used explainers. The versatility and efficiency of our library will help researchers bring their explainability methods from development to deployment. Our code is available at https://github.com/abacusai/xai-bench.

1 Introduction

The last decade has seen a huge increase in applications of machine learning in a wide variety of high-stakes domains, such as credit scoring, fraud detection, criminal recidivism, and loan repayment [28, 8, 29, 7]. With the widespread deployment of machine learning models in applications that impact human lives, research on model explainability is becoming increasingly more important. The applications of model explainability include debugging, legal obligations to give explanations, recognizing and mitigating bias, data labeling, and faster adoption of machine learning technologies [26, 44, 6, 15]. Many different methods for explainability are actively being explored including logic rules [18, 40, 36], hidden semantics [43], feature attribution [32, 26, 31, 11, 39], and explanation by example [24, 10]. The most common type of explainers are post-hoc, local feature attribution methods [44, 26, 1, 32, 31, 11], which output a set of weights corresponding to the importance of each feature for a given datapoint and model prediction. Although various feature attribution methods are

*Equal contribution

Preprint. Under review.

being deployed in different use cases today, currently there are no widely adopted methods to easily *evaluate and/or compare* different feature attribution algorithms. Indeed, evaluating the effectiveness of explanations is an intrinsically human-centric task that ideally requires human studies. However, it is often desirable to develop new explainability techniques using empirical evaluation metrics before the human trial stage. Although empirical evaluation metrics have been proposed, many of these metrics are either computationally prohibitive or require strong assumptions, to compute on real-world datasets. For example, a popular method for feature attribution is to approximate Shapley values [26, 13, 25, 39], but computing the distance to ground-truth Shapley values requires estimating exponentially many conditional feature distributions, which is not possible to compute unless the dataset contains sufficiently many datapoints across exponentially many combinations of features.

In this work, we overcome these challenges by releasing a suite of synthetic datasets, which make it possible to efficiently benchmark feature attribution methods. The use of synthetic datasets, for which the ground-truth distribution of data is known, makes it possible to exactly compute the conditional distribution over any set of features, thus enabling computations of many feature attribution evaluation metrics such as distance to ground-truth Shapley values [26], remove-and-retrain (ROAR) [20], faithfulness [3], and monotonicity [27]. Our synthetic datasets offer a wide variety of parameters which can be configured to simulate real-world data and have the potential to identify subtle failure modes of explainability techniques. We give examples of how real datasets can be converted to similar synthetic datasets, thereby allowing explainability methods to be benchmarked on realistic synthetic datasets.

We showcase the power of our library by benchmarking popular explainers such as SHAP [26], LIME [32], MAPLE [31], SHAPR [1], and L2X[11], with respect to a broad set of evaluation metrics, across a variety of axes of comparison, such as feature correlation, model type, and data distribution type. Our library is designed to substantially accelerate the time it takes for researchers and practitioners to move their explainability algorithms from development to deployment. All of our code, API docs, and walkthroughs are available at `https://github.com/abacusai/xai-bench`. We welcome contributions and hope to grow the repository to handle a wide variety of use-cases. We expect the scope and breadth of our framework to increase over time.

Our contributions. We summarize our main contributions below.

- We release a set of synthetic datasets with known ground-truth distributions, along with a library that makes it possible to efficiently evaluate feature attribution techniques with respect to ten different metrics. Our synthetic datasets offer a wide variety of parameters that can be configured to simulate real-world applications.
- We demonstrate the power of our library by benchmarking popular explainers such as SHAP [26], LIME [32], MAPLE [31], SHAPR [1], and L2X[11], and identifying their failure modes.

2 Related Work

Model explainability in machine learning has seen a wide range of approaches, and multiple taxonomies have been proposed to classify the different types of approaches. Zhang et al. [44] describe three dimensions of explainability techniques: passive/active, type of explanation, and local/global explainations. The types of explanations they identified are logic rules [18, 40, 36], hidden semantics [43], feature attribution [32, 26, 31, 11, 39, 1], and explanation by example [24, 10]. Other surveys on explainable AI include Arrieta et al. [5], Adadi and Berrada [2], and Došilović et al. [16].

Techniques for feature attribution include approximating Shapley values [26, 13, 25, 39], approximating the model locally with a more explainable model [32], and approximating the mutual information of each feature with the label [11]. In Appendix E, we give descriptions and implementation details of the five feature attribution methods we implemented.

2.1 Benchmarking Explainability Techniques

One recent work [21] gave an experimental survey of explainability methods, testing SHAP [26], LIME [32], Anchors [33], Saliency Maps [37], and Grad-CAM++ [9], and their proposed ExMatchina on image, text, audio, and sensory datasets. They use human labeling via Mechanical Turk as an evaluation metric. Another work [6] gave an experimental survey of several algorithms including local/global, white-box/black-box, and supervised/unsupervised techniques. The only feature attribution algorithms they tested were SHAP and LIME. Another recent work gives a benchmark on

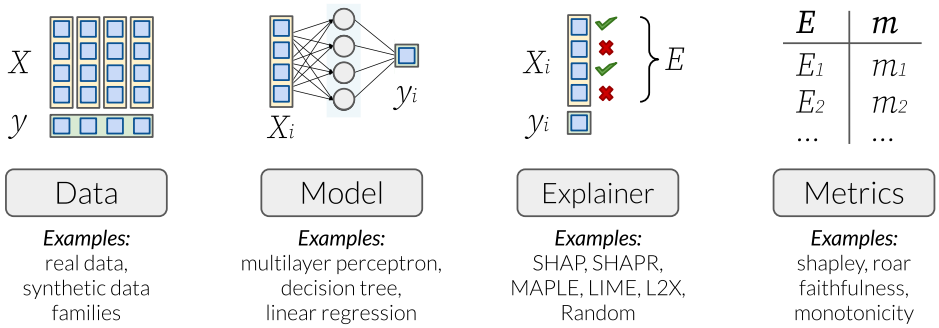


Figure 1: Overview of the main components in XAI-BENCH.

explainability for time-series classification [17]. Another recent work [15] gives a set of benchmark natural language processing (NLP) datasets aimed at comparing explainability methods. This work releases multiple datsets with human-annotated explanations, as well as a few newly proposed metrics specifically chosen to capture the explainability of predictions in NLP applications. To the best of our knowledge, no prior work has released a library with five different evaluation metrics or released a set of synthetic datasets for explainability with more than one tunable parameter.

2.2 Explainability evaluation metrics

While the "correctness" of feature attribution methods may be subjective, comparisons between methods are often based on human studies [22, 34, 35]. However, human studies are not always possible, and several empirical (non-human) evaluation metrics have been proposed. Faithfulness [3] measures the correlation between the weights of the feature attribution algorithm, and the effect of the features on the performance of the model. Monotonicity [27] checks whether iteratively adding features from least weighted feature to most weighted feature, causes the prediction to monotonically improve. By retraining a model with subsets of features ablated, ROAR [20] uses a new model with partially ablated input features to evaluate a feature attribution technique while avoiding problems with distribution shift. Note that all of the above metrics evaluate feature importance by computing the effect of removing the feature from a single set of features S. In contrast, Shapley values [26, 13, 25, 39] evaluate all possible sets S that a feature can be removed from to compute an average effect.

3 Evaluation Metrics

3.1 Preliminaries

Now we give definitions and background information used throughout the next three sections. Given a distribution $\mathcal{D}$, each datapoint is of the form $(\boldsymbol{x}, y) \sim \mathcal{D}$, where $\boldsymbol{x}$ denotes the set of features, and y denotes the label. We assume that $\boldsymbol{x} \in [0,1]^D$ and $y \in [0,1]$, yet all of the concepts we discuss can be generalized to arbitrary categorical and real-valued feature distributions and labels. Assume we have a training set $\mathcal{D}_{\text{train}}$ and a test set $\mathcal{D}_{\text{test}}$, both drawn from $\mathcal{D}$. We train a model $f : [0,1]^D \to [0,1]$ on the training set. Common choices for f include a neural network or a decision tree.

A *feature attribution method* is a function g which can be used to estimate the importance of each feature in making a prediction. That is, given a model f and a datapoint $\boldsymbol{x}$, then $g(\boldsymbol{x}, f) = \boldsymbol{w} \in [-1,1]^D$, where each output weight w_i corresponds to the relative importance of feature i when making the prediction $f(\boldsymbol{x})$. Common choices for g include SHAP [26] or LIME [32].

3.2 Metrics

In this section, we define several different evaluation metrics for explainability methods. Each evaluation metric has pros and cons, and all of them should ideally be used across different datasets to see the clearest picture of the comparative performance of different feature attribution techniques. A *feature attribution evaluation metric* is a function which evaluates the weights of a feature attribution method on a datapoint $\boldsymbol{x}$. For example, given a datapoint $\boldsymbol{x}$ and a set of feature weights $\boldsymbol{w} = g(\boldsymbol{x}, f)$, then a value near zero indicates that g did not provide an accurate feature attribution estimate for $\boldsymbol{x}$, while a value near one indicates that g did provide an accurate feature attribution estimate.

Many evaluation metrics involve evaluating the change in performance of the model when a subset of features of a datapoint are removed. In order to measure the true marginal improvement for a set of features S, we evaluate the model when replacing the features S with their expected values conditioned on the remaining features. Formally, given a datapoint $\boldsymbol{x} \sim \mathcal{D}$ and a set of indices $S \subseteq \{1, \cdots, D\}$, we define $\mathcal{D}(\boldsymbol{x}_S)$ as the conditional probability distribution $\boldsymbol{x}' \sim \mathcal{D}$ such that $x'_i = x_i$ for all $i \in S$. In other words, given $\boldsymbol{x}$ and S, we have

$$p(\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_S)) = p(\boldsymbol{x}' \sim \mathcal{D} \mid x'_i = x_i \text{ for all } i \in S). \tag{1}$$

By this definition, $\mathcal{D}(\boldsymbol{x}_\emptyset) = \mathcal{D}$, and if we define $F = \{1, \cdots, D\}$, then $\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_F)$ is equal to $\boldsymbol{x}$ with probability 1. Given a datapoint $\boldsymbol{x}$, a model f, and a weight vector $\boldsymbol{w}$, the first evaluation metric, **faithfulness** (faith$-$) [3], is defined as follows:

$$\text{faith}- = \text{Pearson}\left(\left|\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{F \setminus i})}[f(\boldsymbol{x}')] - f(\boldsymbol{x})\right|_{1 \le i \le D}, [w_i]_{1 \le i \le D}\right). \tag{2}$$

In other words, faith$-$ computes the Pearson correlation coefficient [42] between the weight vector and the approximate marginal contribution of each feature. We also study a new variant of faithfulness: instead of computing the marginal improvement between $\mathcal{D}(\boldsymbol{x}_{F \setminus i})$ and $\mathcal{D}(\boldsymbol{x}_F)$, we compute the marginal improvement between $\mathcal{D}(\boldsymbol{x}_\emptyset)$ and $\mathcal{D}(\boldsymbol{x}_{\{i\}})$:

$$\text{faith}+ = \text{Pearson}\left(\left|\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{\{i\}})}[f(\boldsymbol{x}')] - \mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_\emptyset)}[f(\boldsymbol{x}')]\right|_{1 \le i \le D}, [w_i]_{1 \le i \le D}\right). \tag{3}$$

The next metric computes the marginal improvement of each feature ordered by the weight vector $\boldsymbol{w}$ *without replacement*, and then computes the fraction of indices i such that the marginal improvement for feature i is greater than the marginal improvement for feature $i+1$. Formally, define $S^-(\boldsymbol{w}, i)$ as the set of i least important weights, define $S^+(\boldsymbol{w}, i)$ as the set of i most important weights, and let $S^-(\boldsymbol{w}, 0) = \emptyset$. Given a datapoint $\boldsymbol{x}$, a model f, and a weight vector $\boldsymbol{w}$, we define **monotonicity** (mono$-$) [27] as follows:

$$\delta_i^- = \mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{S^-(\boldsymbol{w}, i+1)})}[f(\boldsymbol{x}')] - \mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{S^-(\boldsymbol{w}, i)})}[f(\boldsymbol{x}')], \tag{4}$$

$$\text{mono}- = \frac{1}{D-1} \sum_{i=0}^{D-2} \mathbb{I}_{|\delta_i^-| \le |\delta_{i+1}^-|}. \tag{5}$$

Similar to faithfulness, we define a new variant of monotonicity by computing in the opposite order:

$$\delta_i^+ = \mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{S^+(\boldsymbol{w}, i+1)})}[f(\boldsymbol{x}')] - \mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{S^+(\boldsymbol{w}, i)})}[f(\boldsymbol{x}')], \tag{6}$$

$$\text{mono}- = \frac{1}{D-1} \sum_{i=0}^{D-2} \mathbb{I}_{|\delta_i^+| \le |\delta_{i+1}^+|}. \tag{7}$$

The types of metrics discussed so far, faith and mono, each evaluate weight vectors by comparing an estimate of the marginal improvement of a set of features to their corresponding weights. Estimating the marginal improvement requires computing f on different combinations of features, and it is possible that these combinations of features have very low density in $\mathcal{D}$, and are therefore unlikely to occur in $\mathcal{D}_{\text{train}}$. This is especially true for structured data or data where there are large low-density regions in $\mathcal{D}$ and may make the evaluations on f unreliable. To help mitigate this issue, another paradigm of explainability evaluation metrics was proposed: remove-and-retrain (ROAR) [20]. In this paradigm, in order to evaluate the marginal improvement of sets of features, the model is retrained using a new dataset with the features removed. For example, rather than computing $\left|\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{F \setminus i})}[f(\boldsymbol{x}')] - f(\boldsymbol{x})\right|$, we would compute $\left|f^*(\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{F \setminus i})}[\boldsymbol{x}']) - f(\boldsymbol{x})\right|$, where f^* denotes a model that has been trained on a modification of $\mathcal{D}_{\text{train}}$ where each datapoint has its i features with highest weight removed. The original work advocated for reporting a curve of retrained model performance against number of features ablated [20]. In order to report a scalar metric, we propose four new metrics by combining the remove-and-retrain paradigm with faithfulness and monotonicity: **roar-faith+/-** and **roar-mono+/-**. That is, the definitions are similar to faith+/- and mono+/-, but f is replaced with f^* as defined above, accordingly. The formal definitions can be found in Appendix E.1. To compute a ROAR-based

metric on all datapoints in the test set, the explainer must evaluate all datapoints in the training set to construct $D + 1$ ablated datasets, and then the model must be retrained for each of the datasets.

A caveat for all of the aforementioned metrics is that they evaluate each feature weight by computing the effect of removing the feature from a single set of features S. While this evaluation is sufficient for linear models, it may lead to unreliable measurements for nonlinear models such as neural networks. To address this, we use *Shapley values* [26, 13, 25, 39], which take into account the marginal improvement of a feature i across *all possible* exponentially many sets with and without i. We consider two Shapley-based metrics: **shapley-mse** and **shapley-corr**, which involve computing the ground-truth Shapley values [26] for each feature, and then computing either the mean squared error (MSE) or Pearson correlation, respectively, between the weight vector and the set of ground-truth Shapley values. We give the formal definitions of the Shapley-based metrics in Appendix E. See Table 1 for a summary of the properties of each metric.

Table 1: Summary of evaluation metrics. **Linearity** indicates whether model linearity is an implicit assumption. **Retrain** indicates whether computing a metric requires retraining the original model. To compute the evaluation metric on the entire test set, the model must be retrained $\Theta(D)$ times.

Metric	Type	Model evaluations	Retrain	Linearity
faith+/-	correlation	$\Theta(D)$		✓
mono+/-	ranking	$\Theta(D)$		✓
roar-faith+/-	correlation	$\Theta(D)$	✓	✓
roar-mono+/-	ranking	$\Theta(D)$	✓	✓
shapley-mse	accuracy	$\Theta(2^D)$		
shapley-corr	correlation	$\Theta(2^D)$		

Researchers may use any or all of the above metrics for evaluating and comparing different feature attribution explaination techniques. The metric to rely on the most depends on the specific use-case, dataset, feature attribution technique, or computational constraints. For example, researchers evaluating Shapley-based methods such as Kernel-SHAP [26], SHAPR [1], or other SHAP variants [39, 4, 19], may wish to use the shapley-mse and shapley-corr metrics. However, the runtime of these metrics is exponential in the number of features. For tasks involving highly structured data such as image data, ROAR-based metrics may perform better because the faithfulness and monotonicity may compute the function in low-density areas. For tasks involving high-dimensional data and a large cost to retrain the model, faithfulness and monotonicity will be much less computationally intensive than ROAR-based or Shapley-based metrics.

4 Synthetic Datasets

In this section, we describe the synthetic datasets used in our library. We start by discussing the benefits of synthetic datasets when evaluating feature attribution methods. Next, we describe our multivariate Gaussian and mixture of Gaussian datasets.

4.1 The case for synthetic data

As shown in Section 3.2, key to the metrics is computing the conditional expectation $\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_S)}[f(\boldsymbol{x}')]$ for a subset S, datapoint $\boldsymbol{x}$, and trained model f. On real-world datasets, the conditional distribution $\mathcal{D}(\boldsymbol{x}_S)$ (defined in Equation 1) can only be approximated, and the approximation may be very poor when the conditional distribution defines a low-dimensional region of the feature space. Since all evaluation metrics require computing $\Theta(D)$ or $\Theta(2^D)$ such expectations, for each datapoint $\boldsymbol{x}$, is is likely that some evaluations will make use of a poor approximation. However, for the synthetic datasets that we define, the conditional distributions are known, allowing exact computation of the evaluation metrics.

Additionally, as we show in Section 5, synthetic datasets allow one to explicitly control all attributes of the dataset, which allows for targeted experiments, for example, investigating explainer performance as a function of feature correlation. For explainers such as SHAP [26] which assume feature independence, this type of experiment may be very beneficial. Finally, synthetic datasets can be used to simulate real datasets, which enables fair benchmarking of explainers with quantitative metrics.

4.2 Mutivariate Gaussian and mixture of Gaussians features

Now we describe the synthetic datasets in our library. In general, the datasets are expressed as $y = h(\boldsymbol{x})$, with y as label and $\boldsymbol{x}$ as feature vector. The generation is split into two parts, generating features $\boldsymbol{x}$, and defining a function to generate labels y from $\boldsymbol{x}$.

We first describe feature generation, beginning with multivariate normal and mixture of Gaussians synthetic features. The multivariate normal distribution of a $\boldsymbol{D}$-dimensional random vector $\boldsymbol{X} = (X_1, ..., X_D)^T$ can be written as $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is the $\boldsymbol{D}$-dimensional mean vector, and $\boldsymbol{\Sigma}$ is the $\boldsymbol{D} \times \boldsymbol{D}$ covariance matrix. Without loss of generality, we can partition the $\boldsymbol{D}$-dimensional vector $\boldsymbol{x}$ as $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2)^T$. To compute the distribution of $\boldsymbol{X}_1$ conditional on $\boldsymbol{X_2} = \boldsymbol{x_2^*}$ where $\boldsymbol{x_2^*}$ is a $\boldsymbol{K}$-dimensional vector with $0 < \boldsymbol{K} < \boldsymbol{D}$, we can then partition $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ accordingly:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}.$$

Then the conditional distribution is a new multivariate normal $(\boldsymbol{X}_1 | \boldsymbol{X}_2 = \boldsymbol{x}_2^*) \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ where

$$\boldsymbol{\mu}^* = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2^* - \boldsymbol{\mu}_2), \quad \boldsymbol{\Sigma}^* = \boldsymbol{\Sigma}_{11} + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}. \tag{8}$$

For any $\boldsymbol{x}_2^* \in \mathbb{R}^K$, one can compute $\boldsymbol{\mu}^*$ and $\boldsymbol{\Sigma}^*$ and then generate samples from the conditional distribution. $\boldsymbol{\mu}$ can take any value, and $\boldsymbol{\Sigma}$ must be symmetric and positive definite. Similarly, we also include mixture of multivariate Gaussian features with derivation in Appendix D.

4.3 Labels

After creating a distribution of features by either a multivariate Gaussian or mixture of Gaussians, we then create the distribution of labels. The distributions we implement are `linear`, `piecewise constant`, and `nonlinear additive`.

Data labels are computed in two steps: *(1)* raw labels are computed from features, i.e. $y_{\text{raw}} = \sum_{n=1}^{D} \Psi_n(x_n)$ where Ψ_n is a function that operates on feature n, and *(2)* final labels are normalized to have zero mean and unit variance. The normalization ensures that a baseline ML model, which always predicts the mean of the dataset, has an MSE of 1. This allows results derived from different types of datasets to be comparable at scale.

For `linear` datasets, $\Psi_n(x_n)$ are scalar weights, and we can rewrite the raw labels as $y_{raw} = \boldsymbol{w}^T\boldsymbol{x}$. For `piecewise constant` datasets, $\Psi_n(x_n)$ are piecewise constant functions made up of different threshold values (similar to Aas et al. [1]). For `nonlinear additive` datasets, $\Psi_n(x_n)$ are nonlinear functions including *absolute*, *cosine*, and *exponent* function adapted from Chen et al. [11]. Detailed specifications can be found in Appendix F.

5 Experiments

In this section, we describe our experiments in benchmarking several popular feature attribution methods across synthetic datasets.

5.1 Feature attribution methods

We compare six different feature attribution methods: SHAP [26], SHAPR [1], brute-force Kernel SHAP (BF-SHAP) [26], LIME [32], MAPLE [31], and L2X [11]. We also compare the methods to RANDOM explainer, which outputs random weights drawn from a standard normal distribution. See Appendix E for descriptions and implementation details for all methods. We report mean and standard deviation from three trials for all experiments.

5.2 Parameterized synthetic data experiments

Now we run experiments using multivariate Gaussian datasets described in Section 4. Without loss of generality, we can assume that the feature set is normalized (in other words, $\boldsymbol{\mu}$ is set to 0, and the diagonal of $\boldsymbol{\Sigma}$ is set to 1). In all sections except Section 5.3, we set the non-diagonal terms of $\boldsymbol{\Sigma}$ to $\boldsymbol{\rho}$, which allows for convenient parameterization of a global level of feature dependence and has been used in prior work [1].

We run experiments that compare six feature attribution methods on the ten different evaluation metrics defined in Section 3.1 across several different datasets and ML models. In this section, we conduct experiments by varying one or two of these dimensions at a time while holding the other dimensions fixed (for example, we compare different datasets while keeping the ML model fixed) and in Appendix E.1, we give the exhaustive set of experiments.

Performance across metrics As shown in Table 2, the relative performance of explainers varies dramatically across metrics for a fixed decision tree model trained on a `piecewise constant` dataset with $\rho = 0$. It is not surprising that SHAPR, which is an improvement of SHAP, performs well in Shapley metrics. In fact, SHAP, BF-SHAP, SHAPR, and LIME offer accurate approximation of ground truth Shapley values (>0.9 shapley-corr). In addition, LIME achieves top performance in three out of four ROAR-based metrics. Unexpectedly, none of the explainers outperformed random on mono-, suggesting that this metric is not helpful for this dataset and model. Another surprising observation is that while MAPLE performs well for faith+/-, and roar-mono+/-, it fails for roar-faith+/- by producing large negative scores, suggesting that it systematically ranks feature importance in an order opposite to the marginal improvement-based rankings in roar-faith+/-.

Table 2: Explainer performance across metrics. All performance numbers are from explaining a multilayer perceptron trained on the Gaussian piecewise constant dataset with $\rho = 0$.

	RANDOM	SHAP	BF-SHAP	SHAPR	LIME	MAPLE	L2X
faith+(↑)	$-0.028_{\pm 0.022}$	$\mathbf{0.922}_{\pm 0.020}$	$0.887_{\pm 0.031}$	$0.918_{\pm 0.039}$	$0.859_{\pm 0.035}$	$0.626_{\pm 0.050}$	$-0.004_{\pm 0.100}$
faith-(↑)	$-0.022_{\pm 0.023}$	$0.970_{\pm 0.006}$	$0.937_{\pm 0.017}$	$\mathbf{0.977}_{\pm 0.004}$	$0.918_{\pm 0.010}$	$0.647_{\pm 0.045}$	$0.002_{\pm 0.080}$
mono+(↑)	$0.538_{\pm 0.012}$	$\mathbf{0.720}_{\pm 0.018}$	$0.676_{\pm 0.027}$	$0.719_{\pm 0.019}$	$0.667_{\pm 0.032}$	$0.712_{\pm 0.008}$	$0.562_{\pm 0.024}$
mono-(↑)	$\mathbf{0.467}_{\pm 0.006}$	$0.433_{\pm 0.019}$	$0.449_{\pm 0.027}$	$0.435_{\pm 0.012}$	$0.428_{\pm 0.014}$	$0.440_{\pm 0.017}$	$0.430_{\pm 0.040}$
roar-faith+(↑)	$0.003_{\pm 0.028}$	$0.461_{\pm 0.095}$	$0.496_{\pm 0.016}$	$0.468_{\pm 0.082}$	$\mathbf{0.585}_{\pm 0.046}$	$-0.429_{\pm 0.018}$	$0.045_{\pm 0.060}$
roar-faith-(↑)	$0.008_{\pm 0.049}$	$0.581_{\pm 0.024}$	$0.535_{\pm 0.067}$	$0.559_{\pm 0.026}$	$\mathbf{0.621}_{\pm 0.019}$	$-0.339_{\pm 0.013}$	$0.052_{\pm 0.038}$
roar-mono+(↑)	$0.474_{\pm 0.016}$	$0.747_{\pm 0.028}$	$\mathbf{0.771}_{\pm 0.015}$	$0.730_{\pm 0.022}$	$0.707_{\pm 0.024}$	$0.425_{\pm 0.009}$	$0.500_{\pm 0.027}$
roar-mono-(↑)	$0.492_{\pm 0.019}$	$0.721_{\pm 0.032}$	$0.683_{\pm 0.038}$	$0.713_{\pm 0.044}$	$\mathbf{0.745}_{\pm 0.020}$	$0.471_{\pm 0.016}$	$0.451_{\pm 0.041}$
shapley-corr(↑)	$0.001_{\pm 0.014}$	$0.992_{\pm 0.005}$	$0.956_{\pm 0.007}$	$\mathbf{0.998}_{\pm 0.001}$	$0.955_{\pm 0.009}$	$0.735_{\pm 0.038}$	$0.073_{\pm 0.084}$
shapley-mse(↓)	$1.134_{\pm 0.040}$	$0.003_{\pm 0.001}$	$0.008_{\pm 0.001}$	$\mathbf{0.000}_{\pm 0.000}$	$0.026_{\pm 0.001}$	$0.071_{\pm 0.007}$	$0.188_{\pm 0.022}$

Performance across dataset types and feature correlations Next, we explore how the type of dataset and feature correlation affects performance of explainers on a decision tree model with the faithfulness metric. As shown in Figure 2, a general trend is that explainers become less faithful as feature correlation increases. Explainers such as SHAP assume feature independence and tend to perform well when features are indeed independent ($\rho = 0$). This is especially apparent with the `linear` dataset, where all performance of most methods cluster around 0.9 at $\rho = 0$. However, LIME's performance drops as much as $\sim 90\%$ when features are almost perfectly correlated ($\rho = 0.99$). On the other hand, for both the `nonlinear additive` and `piecewise constant` datasets, MAPLE's performance stayed relative stable across values of ρ.

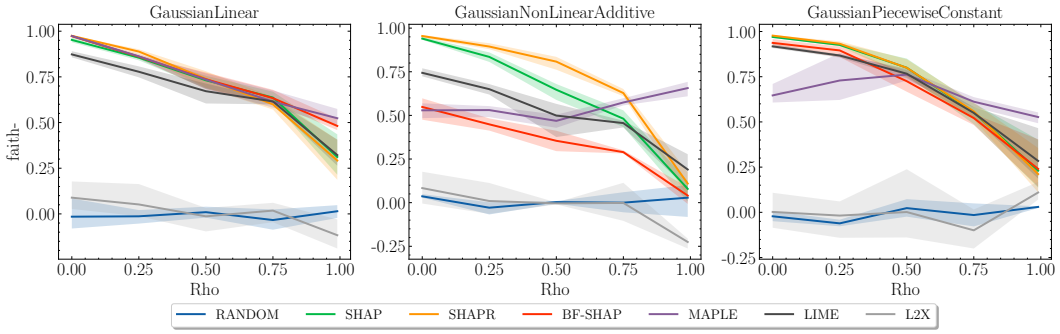


Figure 2: Results for faith- on a multilayer perceptron trained on three types of Gaussian datasets.

Performance across ML models Next, we train three ML models: linear regression, decision tree, and multilayer perceptron, with a `piecewise constant` dataset and compare faith-. Figure 3 shows that as in Figure 2, explainer performance drops as features become more correlated. Most explainers perform well for linear regression up to $\rho = 0.75$. The performance of SHAP, SHAPR, and LIME

remain relatively consistent across ML models. In contrast, BF-SHAP performs significantly worse on the tree model. The nearly consistent negative faith- score of MAPLE on the tree model provides additional evidence to Table 2 that in some cases, the most important feature weights MAPLE predicts tend be the least important.

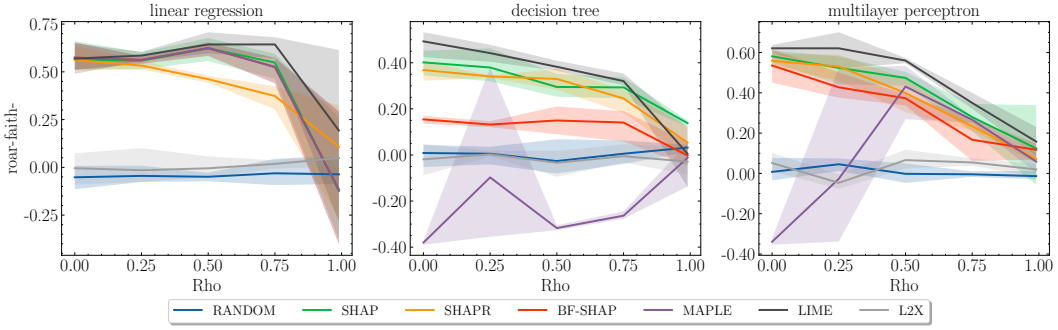


Figure 3: Results for faith- for three types of ML models– linear regression, decision tree, and multilayer perceptron– trained on a Gaussian piecewise constant dataset.

5.3 Simulating the wine dataset

In this section, we demonstrate the power and flexibility of synthetic datasets by simulating the popular wine dataset [12, 38] with synthetic features so that it can be used to efficiently benchmark feature attribution methods.

The white wine dataset has 11 continuous features ($\boldsymbol{x}_{\text{real}}$) and one integer quality rating (y_{real}) between 0 and 10. In this section, it is formulated as a regression task, but it can also be formulated as a multi-class classification task. The features are first normalized to have zero mean and unit variance, then an empirical covariance matrix is computed (Appendix Figure 5), which is then used as the input covariance matrix to generate synthetic multivariate Gaussian features ($\boldsymbol{x}_{\text{sim}}$). Simulated wine quality (y_{sim}) is labeled by a k-nearest neighbor model based on real datapoints ($\boldsymbol{x}_{\text{real}}, y_{\text{real}}$).

We evaluate how close the simulated dataset is to the real one in two steps. First, we compute the Jensen-Shannon Divergence (JSD) [41] of the real and synthetic wine datasets. JSD measures the similarity between two distributions, it is bounded between 0 and 1, and lower JSD suggests higher similarity between two distributions. The JSD of marginal distributions between the real empirical features and the synthetic Gaussian features has a mean of 0.20, and the JSD of real and synthetic targets is 0.23, suggesting a good fit. Second, we train three types of ML models on both simulated and real wine datasets and compare the MSE of explanations on a common held-out real test set. As shown in Appendix Table 5, consistent low MSE across ML models and explainers suggest that the simulated dataset is a good proxy for the original wine dataset for evaluating explainers.

Next, we compute evaluation metrics for five different explainers on the synthetic wine dataset. Note that computing these metrics accurately is not possible on the real wine dataset, as the conditional distribution is unknown. As shown in Table 3, SHAP performs well on the Shapley metrics, consistent with Table 2. Both LIME and MAPLE outperform SHAP on faith+. MAPLE achieves top performance on mono-, however, none of the explainers significantly outperform RANDOM.

Table 3: Explainer performance on the simulated wine dataset across metrics. All performance numbers are from explainers explaining a decision tree model.

	RANDOM	SHAP	LIME	MAPLE	L2X
faith- (↑)	$0.012_{\pm 0.011}$	$\mathbf{0.461}_{\pm 0.034}$	$0.237_{\pm 0.031}$	$-0.007_{\pm 0.036}$	$-0.010_{\pm 0.032}$
faith+ (↑)	$0.025_{\pm 0.038}$	$0.488_{\pm 0.023}$	$\mathbf{0.595}_{\pm 0.022}$	$0.556_{\pm 0.021}$	$0.055_{\pm 0.035}$
mono- (↑)	$0.490_{\pm 0.004}$	$0.502_{\pm 0.010}$	$0.500_{\pm 0.013}$	$\mathbf{0.506}_{\pm 0.011}$	$0.492_{\pm 0.001}$
mono+ (↑)	$0.523_{\pm 0.010}$	$\mathbf{0.556}_{\pm 0.012}$	$0.539_{\pm 0.005}$	$0.513_{\pm 0.008}$	$0.522_{\pm 0.008}$
shapley-corr (↑)	$0.011_{\pm 0.027}$	$\mathbf{0.815}_{\pm 0.024}$	$0.692_{\pm 0.019}$	$0.669_{\pm 0.007}$	$0.035_{\pm 0.055}$
shapley-mse (↓)	$1.032_{\pm 0.022}$	$\mathbf{0.014}_{\pm 0.003}$	$0.032_{\pm 0.005}$	$0.041_{\pm 0.001}$	$0.055_{\pm 0.001}$

5.4 Recommended usage

Throughout Section 5, we gave a sample of the types of experiments that can be done using XAI-BENCH (recall that our comprehensive experiments are in Appendix E.1). For researchers looking to develop new explainability techniques, we recommend benchmarking new algorithms across all metrics using our synthetic multivariate Gaussian and mixture of Gaussian datasets with different values of ρ. These datasets give a good initial picture of the efficacy of new techniques. For researchers with a dataset and application in mind, we recommend converting the dataset into a synthetic dataset using the technique described in Section 5.3. Note that converting to a synthetic dataset also gives the ability to evaluate explainability techniques on perturbations of the original covariance matrix, to simulate robustness to distribution shift. Finally, researchers can decide on the evaluation metric that is most suitable to the application at hand, based on the model, intended use-case, size and number of features of the dataset, and level of feature correlation. For example, ROAR-based metrics may be the most appropriate for structured data that is prone to unreliable function evaluations in low-density regions of the feature space, and faithfulness and monotonicity are the most lightweight options for applications in which high-dimensional data and high cost of model training make ROAR-based and Shapley-based metrics infeasible.

6 Societal Impact

Machine learning models are more prevalent now than ever before. With the widespread deployment of models in applications that impact human lives, explainability is becoming increasingly more important for the purposes of debugging, legal obligations, and mitigating bias [26, 44, 6, 15]. Given the importance of high-quality explanations, it is essential that the explainability methods are reliable across all types of datasets. Our work seeks to speed up the development of explainability methods, with a focus on catching edge cases and failure modes, to ensure that new explainability methods are robust before they are used in the real world. Of particular importance are improving the reliability of explainability methods intended to recognize biased predictions, for example, ensuring that the features used to predict criminal recidivism are not based on race or gender [23]. Frameworks for evaluating and comparing explainability methods are an important part of creating inclusive and unbiased technology. As pointed out in prior work [14], while methods for explainability or debiasing are important, they must be part of a larger, socially contextualized project to examine the ethical considerations of the machine learning application.

7 Conclusions and Limitations

In this work, we released a set of synthetic datasets along with a library for benchmarking feature attribution algorithms. The use of synthetic datasets with known ground-truth distributions makes it possible to exactly compute the conditional distribution over any set of features, enabling computations of several explainability evaluation metrics, including ground-truth Shapley values, ROAR, faithfulness, and monotonicity. Our synthetic datasets offer a variety of parameters which can be configured to simulate real-world data and have the potential to identify failure modes of explainability techniques. We showcase the power of our library by benchmarking several popular explainers with respect to ten evaluation metrics across a variety of settings.

Furthermore, despite the fact that the synthetic datasets aim to cover a broad range of feature distributions, correlations, scales, and target generation functions, there is almost certainly a gap between synthetic and real-world datasets. However, as discussed before, it is often the case that we do not know the ground truth generative model of real datasets, thus making it impossible to compute many objective metrics. Hence, there is a trade-off between data realism and ground truth availability.

Note that our library is **not** meant to be a replacement for human interpretability studies. Since the goals of explainability methods are inherently human-centric, the only foolproof method of evaluating explanation methods are to use human trials. Rather, our library is meant to substantially speed up the process of development, refinement, and identifying failure modes, before reaching human trials.

Overall, we recommend developing new explainability methods in this library, and then conducting human trials on real data. Our library is designed to substantially accelerate the time it takes to move new explainability algorithms from development to deployment. With the release of API documentation, walkthroughs, and a contribution guide, we hope that the scope of our library can increase over time.

References

[1] Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, page 103502, 2021.

[2] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[3] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*, 2018.

[4] Marco Ancona, Cengiz Oztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *International Conference on Machine Learning*, pages 272–281. PMLR, 2019.

[5] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[6] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*, 2019.

[7] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *NIPS Tutorial*, 2017.

[8] Miranda Bogen and Aaron Rieke. Help wanted: An examination of hiring algorithms, equity, and bias, 2018.

[9] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847. IEEE, 2018.

[10] Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. *arXiv preprint arXiv:1806.10574*, 2018.

[11] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 883–892. PMLR, 2018.

[12] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4):547–553, 2009.

[13] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pages 598–617. IEEE, 2016.

[14] Emily Denton, Ben Hutchinson, Margaret Mitchell, and Timnit Gebru. Detecting bias with generative counterfactual face attribute augmentation. *arXiv preprint arXiv:1906.06439*, 2019.

[15] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. Eraser: A benchmark to evaluate rationalized nlp models. *arXiv preprint arXiv:1911.03429*, 2019.

[16] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.

[17] Kevin Fauvel, Véronique Masson, and Elisa Fromont. A performance-explainability framework to benchmark machine learning methods: Application to multivariate time series classifiers. *arXiv preprint arXiv:2005.14501*, 2020.

[18] LiMin Fu. Rule learning by searching on adapted nets. In *AAAI*, volume 91, pages 590–595, 1991.

[19] Tom Heskes, Evi Sijben, Ioan Gabriel Bucur, and Tom Claassen. Causal shapley values: Exploiting causal knowledge to explain individual predictions of complex models. *arXiv preprint arXiv:2011.01625*, 2020.

[20] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *arXiv preprint arXiv:1806.10758*, 2018.

[21] Jeya Vikranth Jeyakumar, Joseph Noor, Yu-Hsi Cheng, Luis Garcia, and Mani Srivastava. How can i explain this to you? an empirical study of deep neural network explanation methods. *Advances in Neural Information Processing Systems*, 2020.

[22] Isaac Lage, Andrew Slavin Ross, Been Kim, Samuel J Gershman, and Finale Doshi-Velez. Human-in-the-loop interpretability prior. *arXiv preprint arXiv:1805.11571*, 2018.

[23] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. *ProPublica (5 2016)*, 9, 2016.

[24] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[25] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[26] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30:4765–4774, 2017.

[27] Ronny Luss, Pin-Yu Chen, Amit Dhurandhar, Prasanna Sattigeri, Yunfeng Zhang, Karthikeyan Shanmugam, and Chun-Chen Tu. Generating contrastive explanations with monotonic attribute functions. *arXiv preprint arXiv:1905.12698*, 2019.

[28] Amitabha Mukerjee, Rita Biswas, Kalyanmoy Deb, and Amrit P Mathur. Multi–objective evolutionary algorithms for the risk–return trade–off in bank loan management. *International Transactions in operational research*, 2002.

[29] Eric WT Ngai, Yong Hu, Yiu Hing Wong, Yijun Chen, and Xin Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision support systems*, 50(3):559–569, 2011.

[30] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *arXiv preprint arXiv:2003.12206*, 2020.

[31] Gregory Plumb, Denali Molitor, and Ameet Talwalkar. Model agnostic supervised local explanations. *arXiv preprint arXiv:1807.02910*, 2018.

[32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " 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*, 2016.

[33] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[34] Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[35] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[36] Rudy Setiono and Huan Liu. Understanding neural networks via rule extraction. In *IJCAI*, volume 1, pages 480–485. Citeseer, 1995.

[37] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[38] Mateusz Staniak and Przemyslaw Biecek. Explanations of model predictions with live and breakdown packages. *arXiv preprint arXiv:1804.01955*, 2018.

[39] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

[40] Geoffrey G Towell and Jude W Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine learning*, 13(1):71–101, 1993.

[41] Andrew KC Wong and Manlai You. Entropy and distance of random graphs with application to structural pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1985.

[42] Sewall Wright. Correlation and causation. *Journal of Agricultural Research*, 20:557–580, 1921.

[43] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018.

[44] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *arXiv preprint arXiv:2012.14261*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Our abstract and introduction accurately reflect our paper.
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We discuss the ethics guidelines in Section 6.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We did not include theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A] We did not include theoretical results.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All code, data, and instructions needed to reproduce our experimental results are available at `https://github.com/abacusai/xai-bench`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All training details are specified in our repository and discussed in the appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We included error bars for the experiments (Section 5).
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We give runtime and hardware details in Section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite all creators of explainability methods and metrics implemented, both in our repository and in the appendix.
   (b) Did you mention the license of the assets? [Yes] We mentioned the licenses for all assets in our repository.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include new assets in our repository.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We did not gather any new data. All of our datasets are synthetic.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] All of the datasets we use are synthetic.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not have human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We did not have human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We did not have human subjects.

A Dataset Documentation and Intended Use

Our code is available at `https://github.com/abacusai/xai-bench`.

A.1 Author responsibility

We bear all responsibility in case of violation of rights, etc. The license of our repository is the **Apache License 2.0**. For more information, see `https://github.com/abacusai/xai-bench/blob/main/LICENSE`.

A.2 Maintenance plan and contributing policy.

We plan to actively maintain the repository, and we welcome contributions from the explainability community and machine learning community at large. For more information, see `https://github.com/abacusai/xai-bench`. As our benchmarks are synthetic, we will host the code to generate the datasets on GitHub.

A.3 Code of conduct

Our Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at `https://www.contributor-covenant.org/version/2/0/code_of_conduct.html`. The policy is copied below.

> "We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, caste, color, religion, or sexual identity and orientation."

B Reproducibility Checklist

To ensure reproducibility, we use the Machine Learning Reproducibility Checklist v2.0, Apr. 7, 2020 [30]. An earlier verision of this checklist (v1.2) was used for NeurIPS 2019 [30].

- For all **models** and **algorithms** presented,
  - **A clear description of the mathematical setting, algorithm, and/or model.** We clearly describe all of the settings and algorithms in Section 3.1 and Appendix Section E.
  - **A clear explanation of any assumptions.** Some of the explainability techniques implemented in our repository make assumptions about the dataset (e.g., that all features are independent). We give this information in Appendix E.
  - **An analysis of the complexity (time, space, sample size) of any algorithm.** We the complexity analysis in Section 3.1 and Appendix Section E.
- For any **theoretical claim**,
  - **A clear statement of the claim.** We do not make theoretical claims.
  - **A complete proof of the claim.** We do not make theoretical claims.
- For all **datasets** used, check if you include:
  - **The relevant statistics, such as number of examples.** We used a real dataset in Section 5.3. We give the statistics for this dataset in the same section.
  - **The details of train / validation / test splits** We give this information in our repository.
  - **An explanation of any data that were excluded, and all pre-processing step.** We did not exclude any data or perform any preprocessing.
  - **A link to a downloadable version of the dataset or simulation environment.** Our repository contains all of the instructions to download and run experiments on the datasets in our work. See `https://github.com/abacusai/xai-bench`.

- **For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control.** We release new synthetic datasets, so there was no collection process. The code to generate the synthetic datasets is hosted on GitHub.

- For all shared **code** related to this work, check if you include:
  - **Specification of dependencies.** We give installation instructions in the README of our repository.
  - **Training code.** The training code is available in our repository.
  - **Evaluation code.** The training code is available in our repository.
  - **(Pre-)trained model(s).** We do not release any pre-trained models. The code to run all experiments in our work can be found in the GitHub repository.
  - **README file includes table of results accompanied by precise command to run to produce those results.** We include a README with detailed instructions to reproduce our experiments.

- For all reported **experimental results**, check if you include:
  - **The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results.** We use default configuration for explainers except SHAPR, tuning details in Appendix E.2
  - **The exact number of training and evaluation runs.** We report 3 runs for each experiment.
  - **A clear definition of the specific measure or statistics used to report results.** We define our metrics in Section 3.2 and Appendix E.1.
  - **A description of results with central tendency (e.g. mean) & variation (e.g. error bars).** We report mean and standard deviation for all experiments.
  - **The average runtime for each result, or estimated energy cost.** We report the runtimes in Section H.
  - **A description of the computing infrastructure used.** We use CPUs for all experiments. We give details of our experiments in Appendix Section H.

C Multivariate Gaussian distribution

The probability density function of a non-degenerative multi-variate normal distribution is

$$f_x(x_1, ..., x_D) = \frac{\exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}))}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}}, \tag{9}$$

with parameters $\boldsymbol{\mu} \in \mathbb{R}^D$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$.

D Mixture of Gaussian features

Now we describe the mixture of Gaussian features. Suppose now that $\boldsymbol{X} = (X_1, ..., X_D)^T$ is a $\boldsymbol{D}$-dimensional random vector distributed as a mixture of k Gaussians. We write this as $\boldsymbol{X} \sim \sum_{j=1}^{k} \pi_j \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, where each $\boldsymbol{\mu}_j$ is a $\boldsymbol{D}$-dimensional mean vector for the j^{th} mixture component, and $\boldsymbol{\Sigma}_j$ is the $\boldsymbol{D} \times \boldsymbol{D}$ covariance matrix for the j^{th} mixture component.

Suppose, as before we use the partition defined by $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \end{bmatrix}$ and partition the parameters of each mixture component accordingly as

$$\boldsymbol{\mu}_j = \begin{bmatrix} \boldsymbol{\mu}_{j,1} \\ \boldsymbol{\mu}_{j,2} \end{bmatrix}, \boldsymbol{\Sigma}_j = \begin{bmatrix} \boldsymbol{\Sigma}_{j,11} & \boldsymbol{\Sigma}_{j,12} \\ \boldsymbol{\Sigma}_{j,21} & \boldsymbol{\Sigma}_{j,22} \end{bmatrix}$$

for $j = 1, \ldots, k$. Then, given $X_2 = \boldsymbol{x}_2^*$, the conditional distribution is also a mixture of Gaussians, written $(\boldsymbol{X}_1 | \boldsymbol{X}_2 = \boldsymbol{x}_2^*) \sim \sum_{j=1}^k \pi_j^* \mathcal{N}(\boldsymbol{\mu}_j^*, \boldsymbol{\Sigma}_j^*)$, where the parameters of each mixture component can be written

$$\boldsymbol{\mu_j^*} = \boldsymbol{\mu}_{j,1} + \boldsymbol{\Sigma}_{j,12}\boldsymbol{\Sigma}_{j,22}^{-1}(\boldsymbol{x}_2^* - \boldsymbol{\mu}_{j,2}) \tag{10}$$

$$\boldsymbol{\Sigma_j^*} = \boldsymbol{\Sigma}_{j,11} + \boldsymbol{\Sigma}_{j,12}\boldsymbol{\Sigma}_{j,22}^{-1}\boldsymbol{\Sigma}_{j,21} \tag{11}$$

$$\pi_j^* = \frac{\pi_j f_{j,2}(\boldsymbol{x}_2^*)}{\sum_{\ell=1}^k \pi_\ell f_{\ell,2}(\boldsymbol{x}_2^*)} \tag{12}$$

and where $f_{j,2}$ denotes the probability density function of the multivariate normal distribution $\mathcal{N}(\mu_{j,2}, \Sigma_{j,22})$.

E Descriptions of Explainability Metrics and Explainers

E.1 Metrics

In this section, we give the formal definitions for the rest of the evaluation metrics from Section 3. We start by giving the definition of the ROAR-based metrics.

Recall that the major difference between ROAR-based metrics and other metrics is that in order to evaluate the marginal improvement of sets of features, ROAR-based metrics retrain the model using a new dataset with the features removed. For example, rather than computing $\left|\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}\left(\boldsymbol{x}_{F \setminus i}\right)}[f(\boldsymbol{x}')] - f(\boldsymbol{x})\right|$, we would compute $\left|f^*(\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}\left(\boldsymbol{x}_{F \setminus i}\right)}[\boldsymbol{x}']) - f(\boldsymbol{x})\right|$, where f^* denotes a model that has been trained on a modification of $\mathcal{D}_{\text{train}}$ where each datapoint has its i features with highest weight removed. Given a datapoint $\boldsymbol{x}$ and a set of features $S \subseteq F$, we start by defining $\bar{\boldsymbol{x}}_S$, the expected value of a datapoint conditioned on the features S from $\boldsymbol{x}$:

$$\bar{\boldsymbol{x}}_S = \begin{cases} x_i \text{ for indices } i \in S \\ \mathbb{E}\left[x_i' \mid \boldsymbol{x'} \sim \mathcal{D} \text{ s.t. } x_j' = x_j \text{ for } j \in S\right] \text{ for indices } i \notin S \end{cases} \tag{13}$$

Let $\mathcal{D}_{\text{train}}^{i-}$ denote a new training set by replacing each $\boldsymbol{x} \sim \mathcal{D}_{\text{train}}$ with $\bar{\boldsymbol{x}}_{F \setminus o^-(\boldsymbol{w}(\boldsymbol{x}), i)}$, where $\boldsymbol{w}(\boldsymbol{x})$ denotes the weight vector for $\boldsymbol{x}$ and $o^-(\boldsymbol{w}(\boldsymbol{x}), i)$ denotes the ith most important feature for $\boldsymbol{x}$ according to $\boldsymbol{w}$. That is, $\mathcal{D}_{\text{train}}^{i-}$ is the training set modified by removing the ith most important features for each datapoint. Let f^{i-} denote the model f retrained on $\mathcal{D}_{\text{train}}^{i-}$ instead of $\mathcal{D}_{\text{train}}$. Then we define **roar-faith-** as follows:

$$\text{roar-faith-} = \texttt{Pearson}\left(\left|f^{i-}(\bar{\boldsymbol{x}}_{F \setminus i}) - f(\boldsymbol{x})\right|_{1 \le i \le D}, [w_i]_{1 \le i \le D}\right), \tag{14}$$

Next, let $\mathcal{D}_{\text{train}}^{i+}$ denote a new training set by replacing each $\boldsymbol{x} \sim \mathcal{D}_{\text{train}}$ with $\bar{\boldsymbol{x}}_{F \setminus o^+(\boldsymbol{w}(\boldsymbol{x}), i)}$, where $\boldsymbol{w}(\boldsymbol{x})$ denotes the weight vector for $\boldsymbol{x}$ and $o^+(\boldsymbol{w}(\boldsymbol{x}), i)$ denotes the ith most important feature for $\boldsymbol{x}$ according to $\boldsymbol{w}$. That is, $\mathcal{D}_{\text{train}}^{i+}$ is the training set modified by removing the ith most important features for each datapoint. Let f^{i+} denote the model f retrained on $\mathcal{D}_{\text{train}}^{i+}$ instead of $\mathcal{D}_{\text{train}}$. Similar to roar-faith-, we define **roar-faith+** as follows:

$$\text{roar-faith+} = \texttt{Pearson}\left(\left|f^{i+}(\bar{\boldsymbol{x}}_{\{i\}}) - f^{i+}(\bar{\boldsymbol{x}}_{\{\emptyset\}})\right|_{1 \le i \le D}, [w_i]_{1 \le i \le D}\right), \tag{15}$$

Recall from Section 3 that $S^-(\boldsymbol{w}, i)$ denotes the set of i least important weights, $S^+(\boldsymbol{w}, i)$ denotes the set of i most important weights, and $S^-(\boldsymbol{w}, 0) = \emptyset$. Let $\mathcal{D}_{\text{train}}^{S(k)-}$ denote a new training set by replacing each $\boldsymbol{x} \sim \mathcal{D}_{\text{train}}$ with $\bar{\boldsymbol{x}}_{F \setminus S^-(\boldsymbol{w}(\boldsymbol{x}), k)}$, where $\boldsymbol{w}(\boldsymbol{x})$ denotes the weight vector for $\boldsymbol{x}$. That is,

$\mathcal{D}_{\text{train}}^{k-}$ is the training set modified by removing the k least important features for each datapoint. Let $f^{S(k)-}$ denote the model f retrained on $\mathcal{D}_{\text{train}}^{S(k)-}$ instead of $\mathcal{D}_{\text{train}}$. We define **roar-mono-** as follows:

$$\bar{\delta}_i^- = f^{S(k)-}(\bar{\boldsymbol{x}}_{S_{i+1}^-(\boldsymbol{w})}) - f^{S_i^-}(\bar{\boldsymbol{x}}_{S_i^-(\boldsymbol{w})}), \tag{16}$$

$$\text{roar-mono-} = \frac{1}{D-1}\sum_{i=0}^{D-2} \mathbb{I}_{|\delta_i^-| \leq |\bar{\delta}_{i+1}^-|} \tag{17}$$

Similarly, let $\mathcal{D}_{\text{train}}^{S(k)+}$ denote a new training set by replacing each $\boldsymbol{x} \sim \mathcal{D}_{\text{train}}$ with $\bar{\boldsymbol{x}}_{F\setminus S^+(\boldsymbol{w}(\boldsymbol{x}),k)}$, where $\boldsymbol{w}(\boldsymbol{x})$ denotes the weight vector for $\boldsymbol{x}$. That is, $\mathcal{D}_{\text{train}}^{k+}$ is the training set modified by removing the k most important features for each datapoint. Let $f^{S(k)+}$ denote the model f retrained on $\mathcal{D}_{\text{train}}^{S(k)+}$ instead of $\mathcal{D}_{\text{train}}$. We define **roar-mono+** as follows:

$$\bar{\delta}_i^+ = f^{S(k)+}(\bar{\boldsymbol{x}}_{S_{i+1}^+(\boldsymbol{w})}) - f^{S(k)+}(\bar{\boldsymbol{x}}_{S_i^+(\boldsymbol{w})}), \tag{18}$$

$$\text{roar-mono+} = \frac{1}{D-1}\sum_{i=0}^{D-2} \mathbb{I}_{|\delta_i^+| \leq |\bar{\delta}_{i+1}^+|} \tag{19}$$

Now we give the formal definition for Shapley values. Given a datapoint $\boldsymbol{x}$, the Shapley value v_i is defined as follows.

$$v_i = \sum_{S \subseteq F\setminus\{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!}(\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{S\cup\{i\}})}[f(\boldsymbol{x}')] - \mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_S)}[f(\boldsymbol{x}')]), \tag{20}$$

where $\mathcal{D}(\boldsymbol{x}_S)$ is defined as in Equation 1. Then for a datapoint $\boldsymbol{x}$, shapley-corr is defined as the correlation between the weight vector $\boldsymbol{w}$ and the set of Shapley values for $\boldsymbol{x}$, and shapley-mse is defined as the mean squared error between the weight vector $\boldsymbol{w}$ and the set of Shapley values for $\boldsymbol{x}$. Formally,

$$\text{shapley-corr} = \texttt{Pearson}\left([v_i]_{1\leq i\leq D}, [w_i]_{1\leq i\leq D}\right), \tag{21}$$

$$\text{shapley-mse} = \sum_{i=1}^{D}(v_i - w_i)^2. \tag{22}$$

The main drawback of this metric is its time complexity, which is $\Theta(2^D)$ for a D-dimensional dataset. Computation quickly becomes infeasible as D scales up.

E.2 Local Feature Attribute Explainers

In this section, we give descriptions and implementation details of all of the explainability methods and metrics implemented in our library.

E.2.1 SHAP

Lundberg et al. [26] proposed a few methods such as BF-SHAP to estimate Shapley values defined by Equation 20. Due to the unavailability of the generative model of conditional distribution for real datasets, one can not accurately compute $\mathbb{E}[f_S(\boldsymbol{x}_S)]$. BF-SHAP makes two assumptions: (1) model linearity, which makes $\mathbb{E}[f_S(\boldsymbol{x}_S)] = f_S(\mathbb{E}[\boldsymbol{x}_S])$, (2) feature independence assumption: $\mathbb{E}[\boldsymbol{x}_S]$ with **marginal** expectation instead of **conditional** expectation. In this work, we refer the official implementation of SHAP as SHAP, and re-implemented brute-force kernel SHAP as BF-SHAP.

E.2.2 SHAPR

Aas et al. [1] proposes several techniques to relax both assumptions and improve BF-SHAP such as "Gaussian", "copula", and "empirical". Because the "empirical" method with a fixed σ performs well across tasks in the original paper, we re-implemented the original R package in python with a tuned from $\{0.1, 0.2, 0.4, 0.8\}$ and fixed $\sigma = 0.4$ and refer it as SHAPR.

E.2.3 LIME

Local Interpretable Model-agnostic Explanations (LIME) [32] interprets individual predictions based on locally approximating the model around a given prediction. We use LIME from the official SHAP repository.

E.2.4 MAPLE

MAPLE [31] is another technique that combines local neighborhood selection with local feature selection. We use official implementation from the official SHAP repository.

E.2.5 L2X

L2X [11] used a mutual information-based approach to explainability. The L2X explainer has a hyperparameter k which needs to be defined by the user to decide the top k most important features to pick. For each D-dimensional data point, L2X outputs a D-dimensional binary vector I_k with 1 indicating important features and 0 indicating unimportant features. Because k is often unknown a priori, we modified L2X as follows:

$$\boldsymbol{w} = \frac{2}{k(k+1)} \sum_{k=1}^{D} I_k, \tag{23}$$

where $\frac{2}{k(k+1)}$ is a scaling factor to ensure the elements in $\boldsymbol{w}$ sum up to 1. The original L2X model uses 1 million training samples to achieve good performance, due to the computation limitation of metrics calculation, we limit the training set size of synthetic experiment to 1000, and experiments show that L2X often fails to achieve good performance.

E.2.6 RANDOM

RANDOM explainer is implemented to serve as a baseline model. The explainer generates random weights from standard normal distribution.

F Dataset details

For 5-dimensional datasets, linear $w = [4, 3, 2, 1, 0]$,

piecewise constant:

$$\Psi_1(x_1) = \begin{cases} 1, \ x_1 >= 0 \\ -1, \ x_1 < 0 \end{cases} \tag{24}$$

$$\Psi_2(x_2) = \begin{cases} -2, \ x_2 < -0.5 \\ -1, \ -0.5 \leq x_2 < 0 \\ 1, \ -0 \leq x_2 < 0.5 \\ 2, \ x_2 \geq 0.5 \end{cases} \tag{25}$$

$$\Psi_3(x_3) = floor(2cos(\pi x_3)) \tag{26}$$

$$\Psi_i(x_i) = 0, \ i = 4, 5 \tag{27}$$

where $floor()$ is a rounding function that rounds a real number to the nearest integer with the lowest absolute value.

Nonlinear additive:

$$\Psi_1(x_1) = sin(x_1) \tag{28}$$

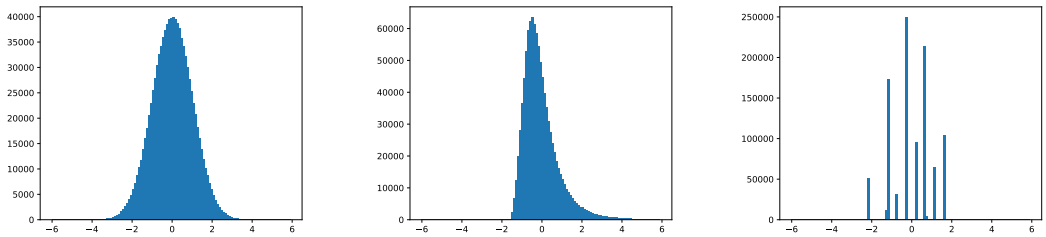
$$\Psi_2(x_2) = |x_2| \tag{29}$$

$$\Psi_3(x_3) = x_3^2 \tag{30}$$

$$\Psi_4(x_4) = e^{x_4} \tag{31}$$

$$\Psi_5(x_5) = 0 \tag{32}$$

where $floor()$ is a rounding function that rounds a real number to the nearest integer with lowest absolute value.



(a) (b) (c)

Figure 4: Label distribution of (a) Gaussian Linear, (b) Gaussian Nonlinear Additive, and (c) Gaussian Piecewise Constant datasets. 1 million datapoints are generated for each dataset, and 120 equal sizedd bins from -6 to 6 are used for discretizing the distribution.

G Higher dimensional experiments

Two factors limit experiments with high dimensional features: *(1)* SHAPR, BF-SHAP, and the Shapley metrics evaluate a model $\Theta(2^D)$ times per datapoint, *(2)* ROAR-based metrics require retraining models $O(D)$ times which is computationally expensive. In Figures 6 and 7, we give experiments in the same setting as in Section 5, on a synthetic dataset with 100 dimensions, for all but the most computationally-intensive Shapley-based explainers.

H Additional results

In this section, we present additional results and experimental details.

Table 4: Time taken in seconds by explainers to explain 100 five-dimensional test datapoints from the Gaussian piecewise constant dataset for a multilayer perceptron model.

	Random	SHAP	SHAPR	BF-SHAP	MAPLE	LIME	L2X
Time (in seconds)	0.00009	3.9	323.8	0.2	3.2	28.0	6.5

Table 4 shows the time explainers take to generate explanations for 100 test datapoints. All of our experiments were run on CPUs. We report mean and standard deviation across three runs for all experiments except for Table 4. All synthetic experiments have a training size of 1000, and test size of 100.

The wine dataset contains 4898 datapoints. In Table 5, we give the mean squared error between explanations for predictions of models trained on the real vs. simulated wine dataset described in Section 5.

We conclude by presenting the comprehensive results for ten different evaluation metrics, seven different feature attribution algorithms, nine different datasets, and five different values of ρ.

Table 5: Mean squared error (MSE) between explanations for predictions of models trained on real and simulated wine dataset. Random predictions are generated from standard Gaussian distribution for every feature for each datapoint. Low MSE across ML models and explainers suggest the simulated wine dataset is a good representation of the real dataset for explainability benchmarking.

Model	SHAP	LIME	MAPLE	L2X	Random
Linear	0.028 ± 0.009	0.047 ± 0.016	0.027 ± 0.009	0.0009 ± 0.0001	
Tree	0.047 ± 0.003	0.009 ± 0.001	0.052 ± 0.012	0.0008 ± 0.0001	1.988 ± 0.001
MLP	0.028 ± 0.003	0.037 ± 0.008	0.040 ± 0.002	0.0008 ± 0.0001	

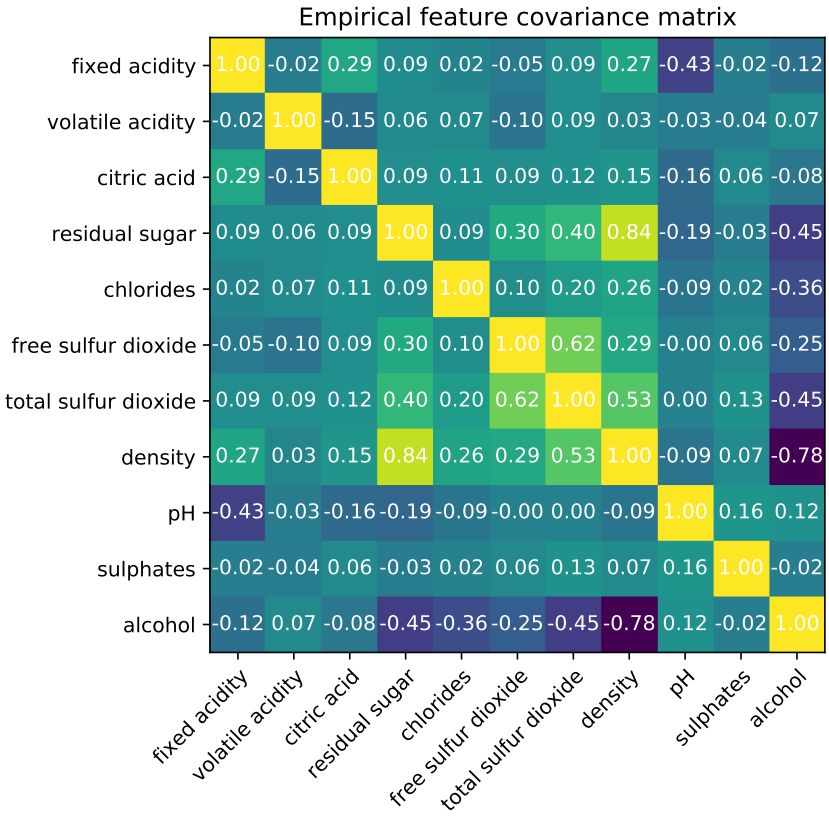


Figure 5: Empirical covariance matrix of the wine dataset. Features are normalized to have unit variance and zero mean.

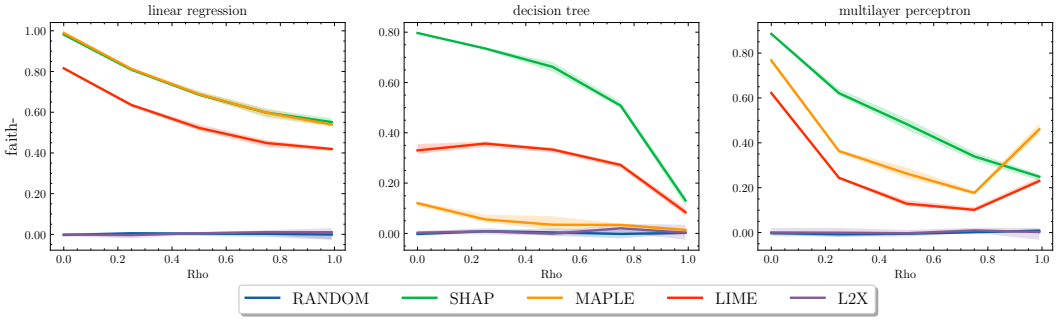


Figure 6: Results for faith- across ML models and ρs on the Gaussian linear dataset with 100 dimensions. Note that the error regions for faith- are much smaller than the error regions for mono- (Figure 7).

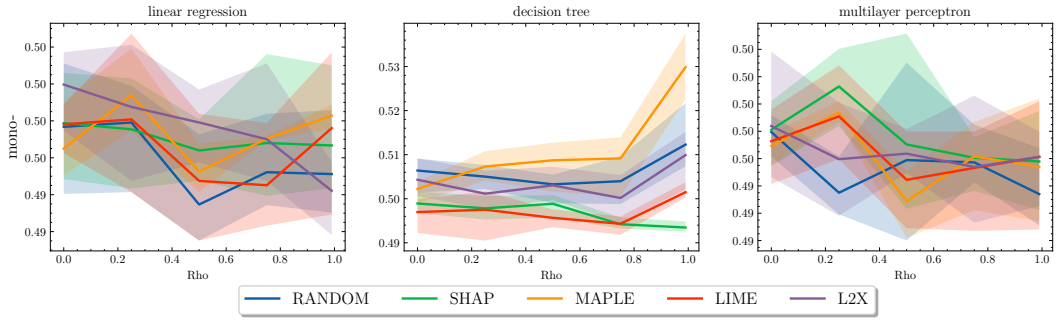


Figure 7: Results for mono- across ML models and ρs on the Gaussian linear dataset with 100 dimensions.

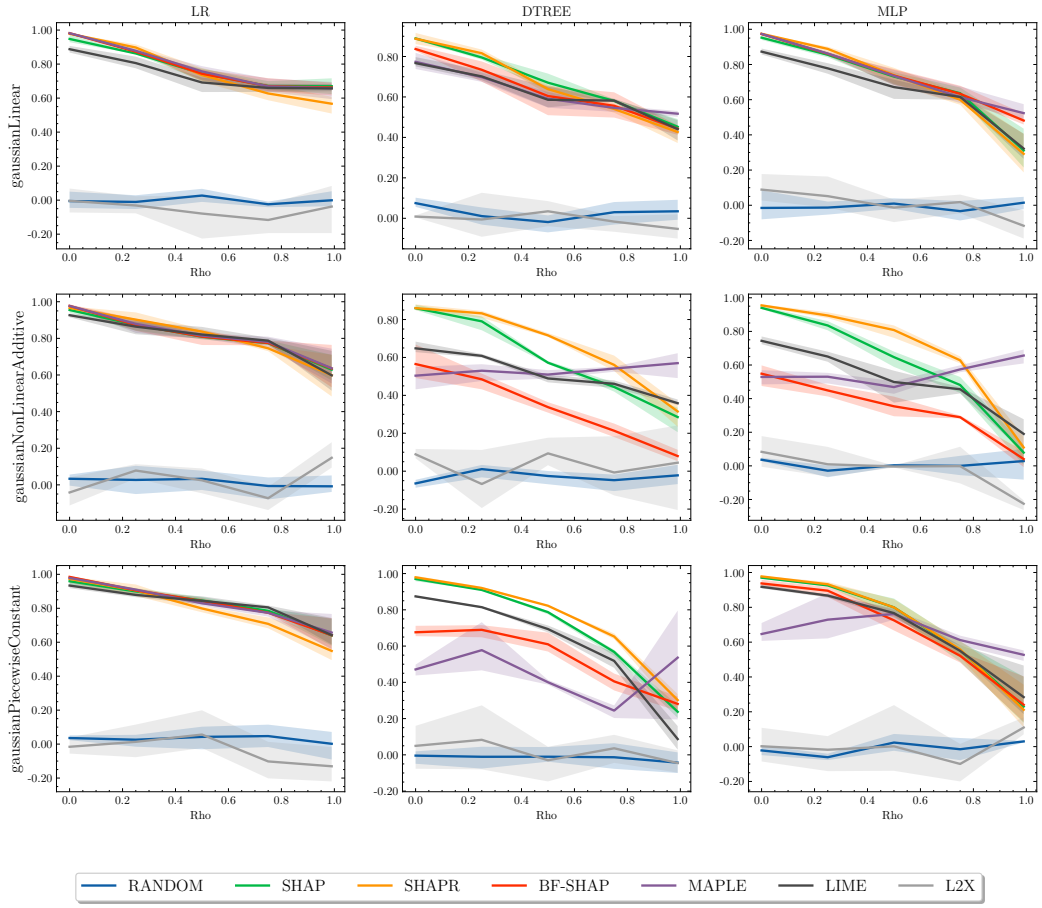


Figure 8: Results of faith- across ML models, dataset types, and ρs.

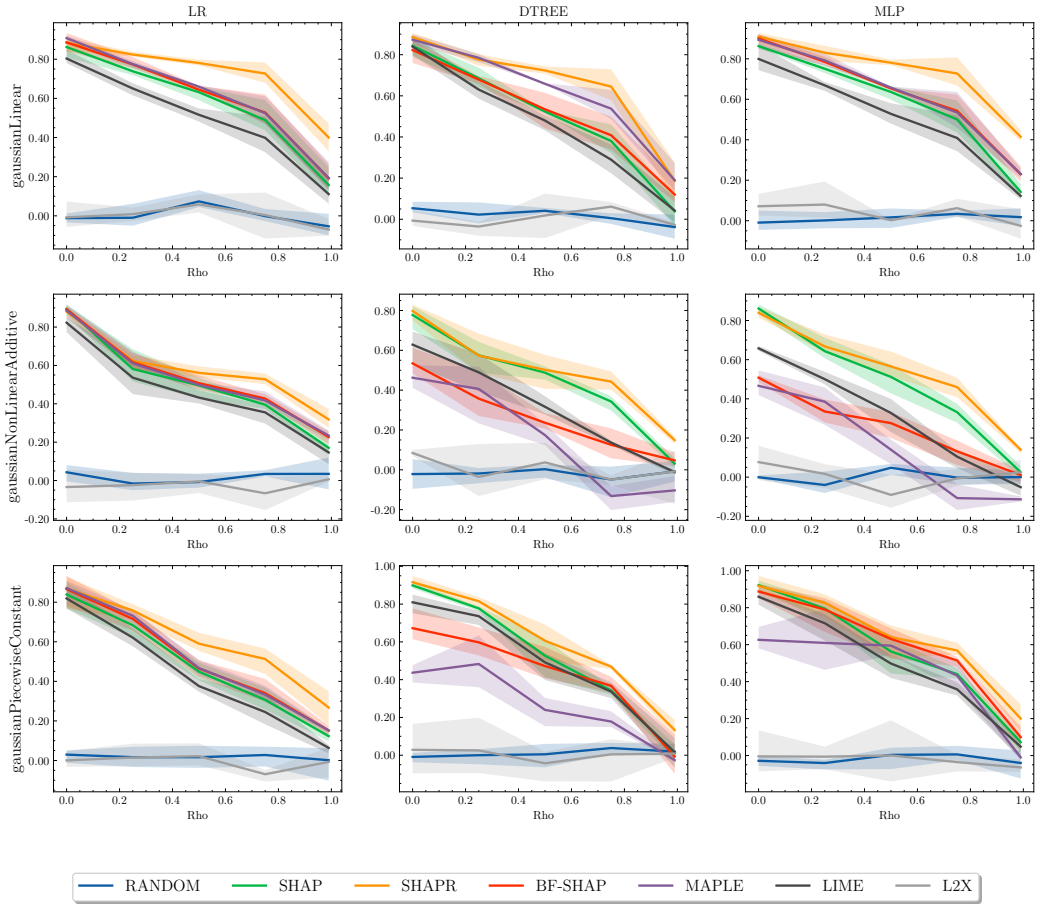


Figure 9: Results of faith+ across ML models, dataset types, and ρs.

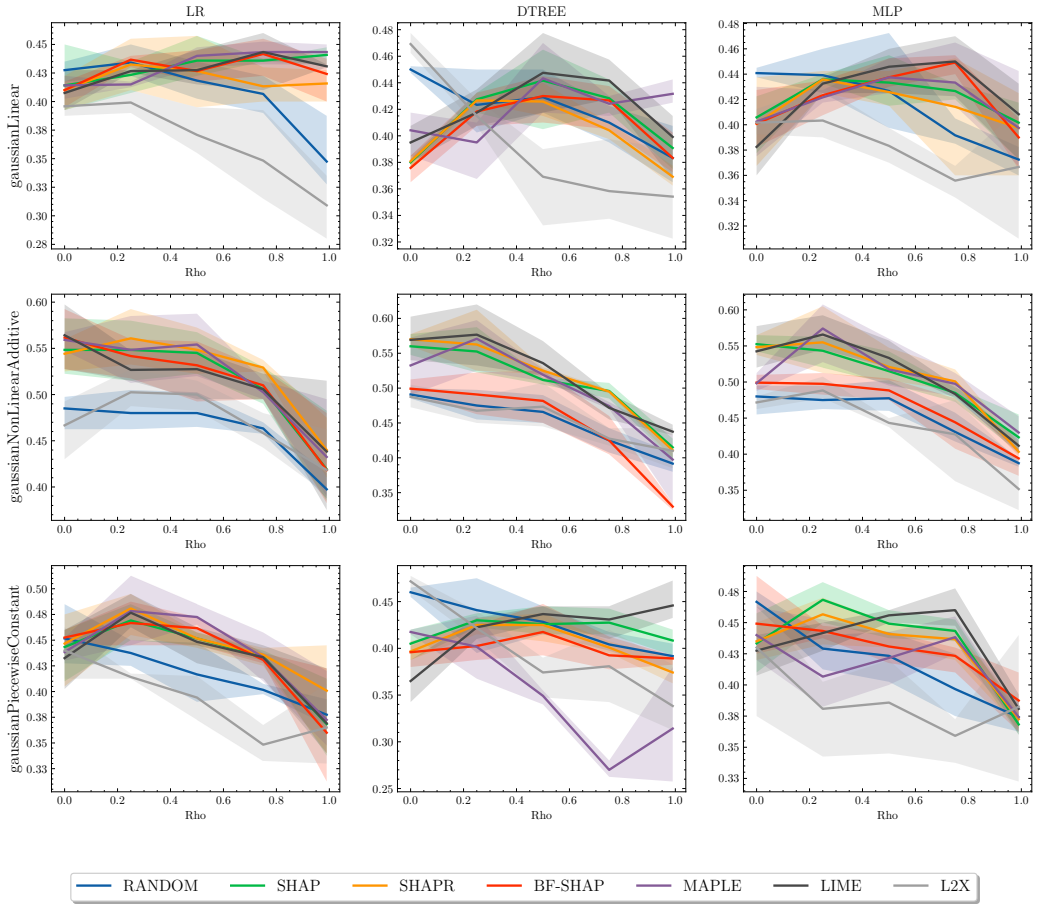


Figure 10: Results of mono- across ML models, dataset types, and ρs.

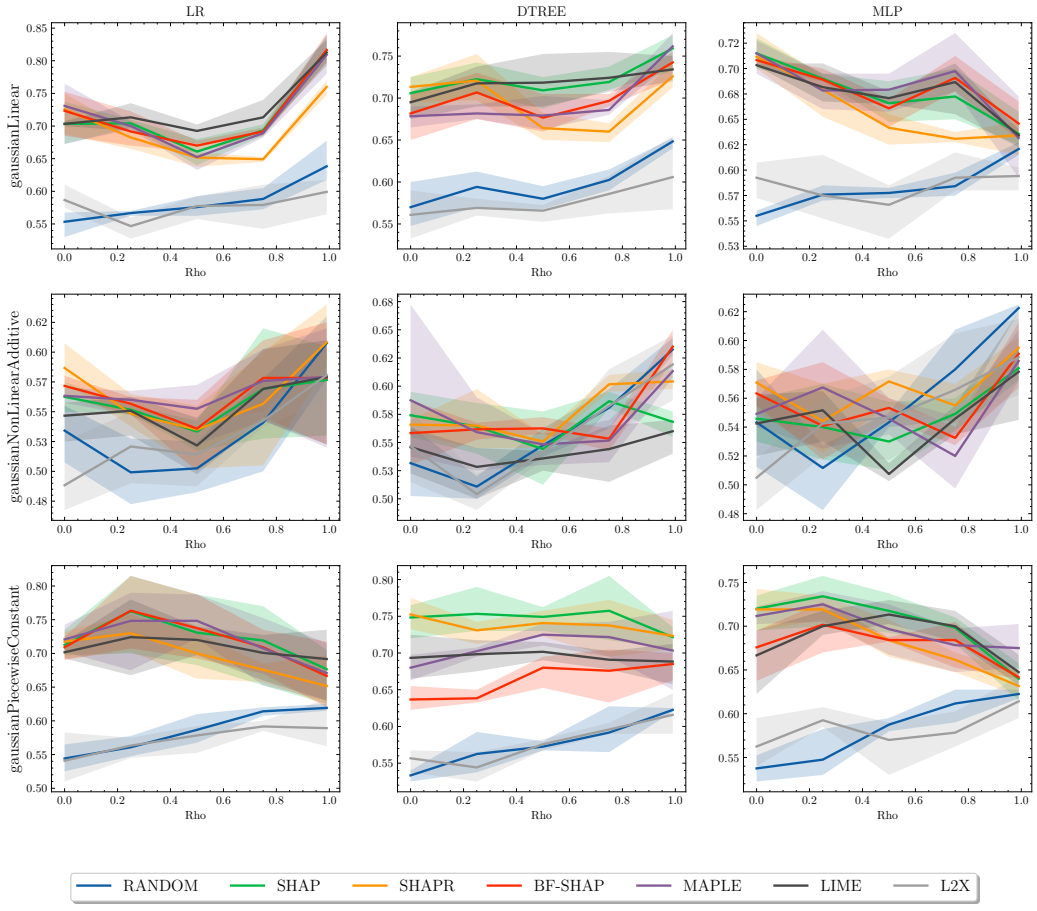


Figure 11: Results of mono+ across ML models, dataset types, and ρs.

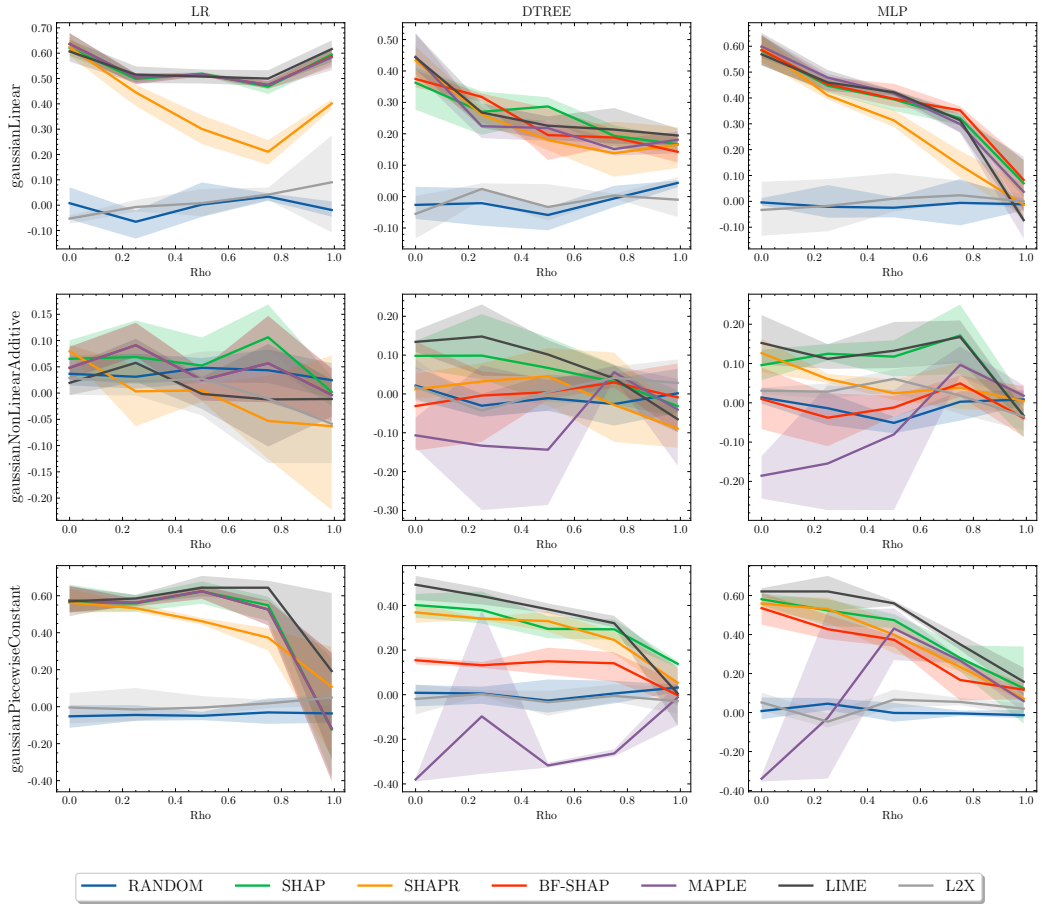


Figure 12: Results of roar-faith- across ML models, dataset types, and ρs.

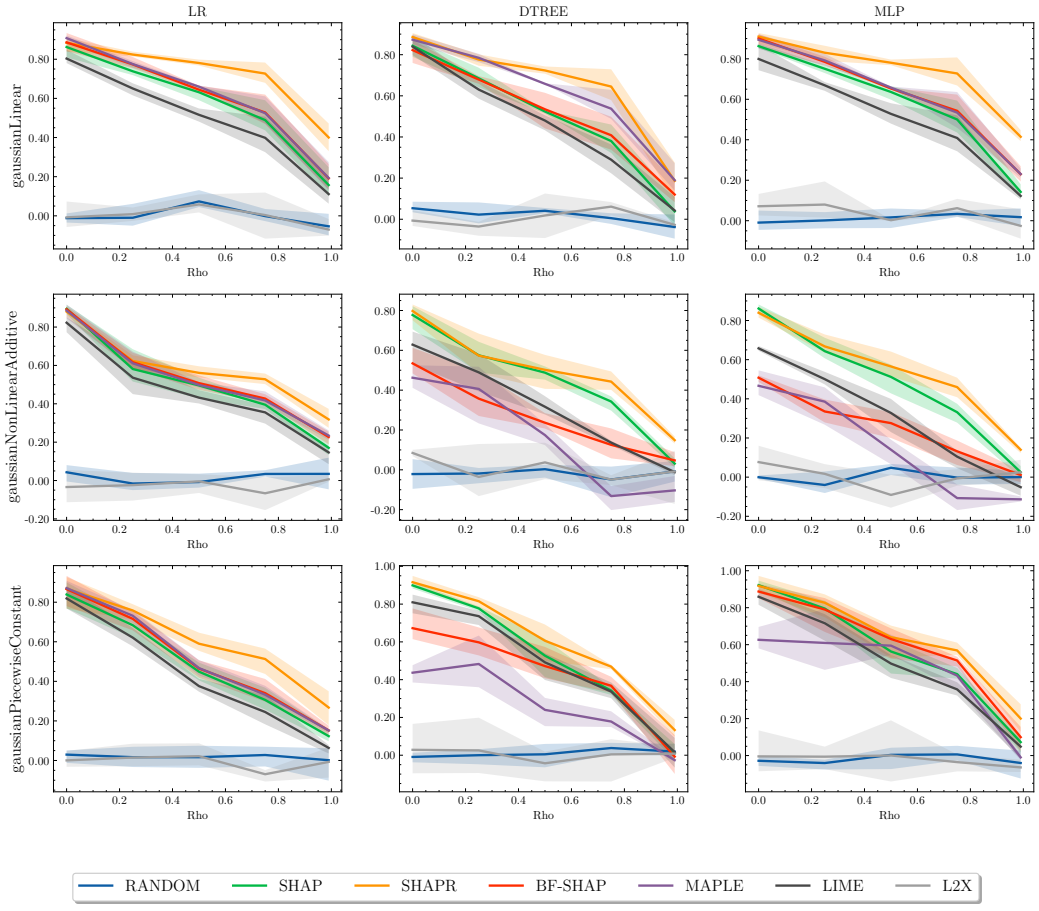


Figure 13: Results of roar-faith+ across ML models, dataset types, and ρs.

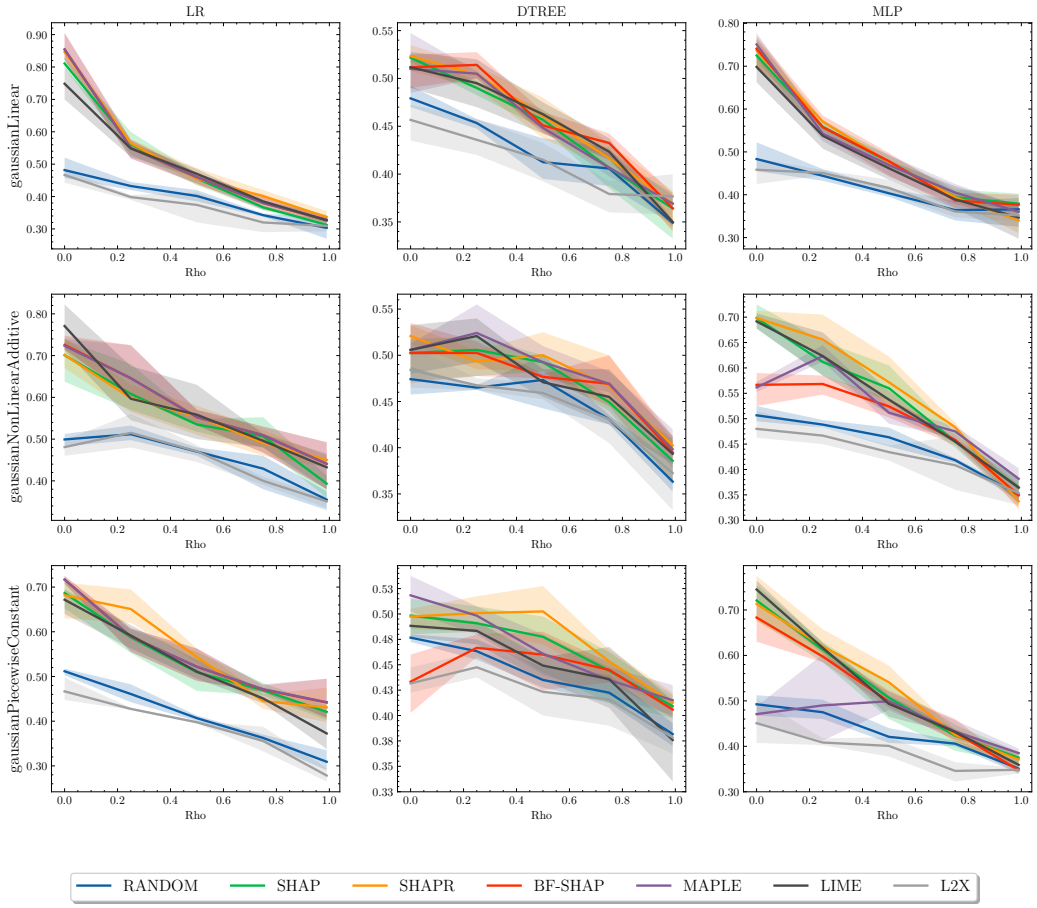


Figure 14: Results of roar-mono- across ML models, dataset types, and ρs.

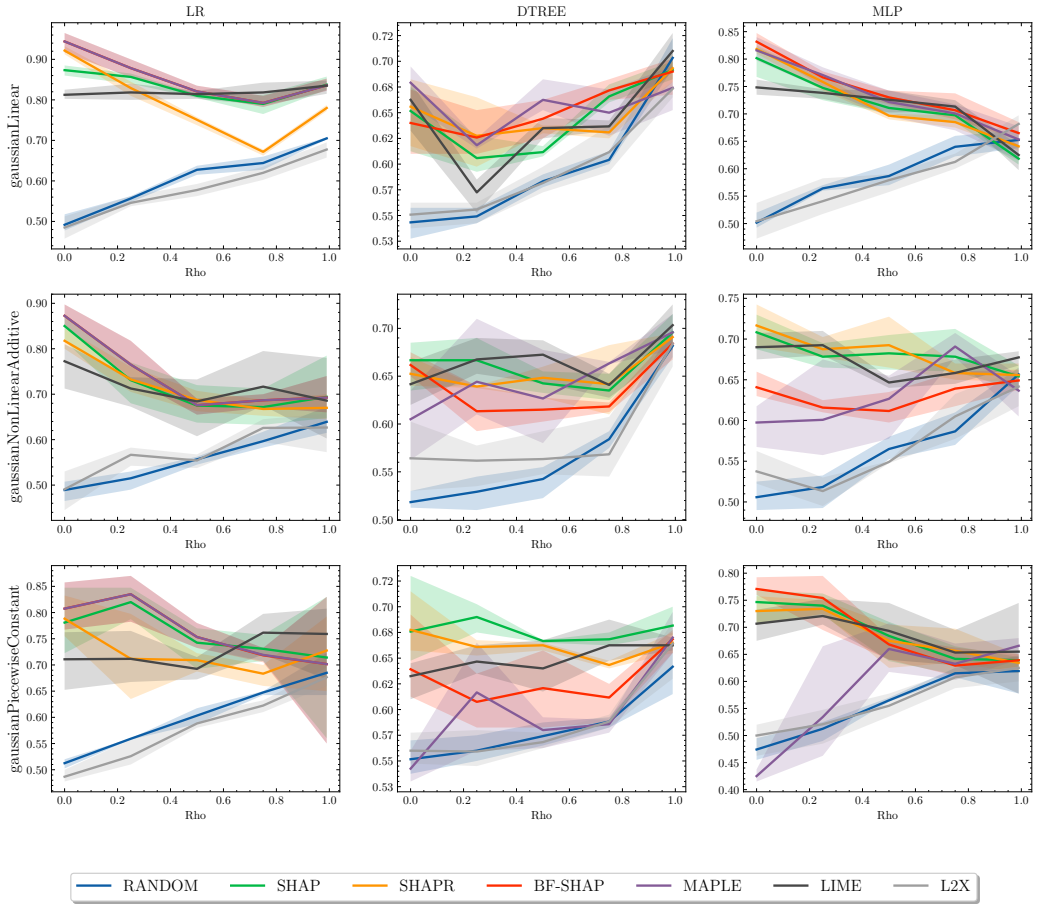


Figure 15: Results of roar-mono+ across ML models, dataset types, and ρs.

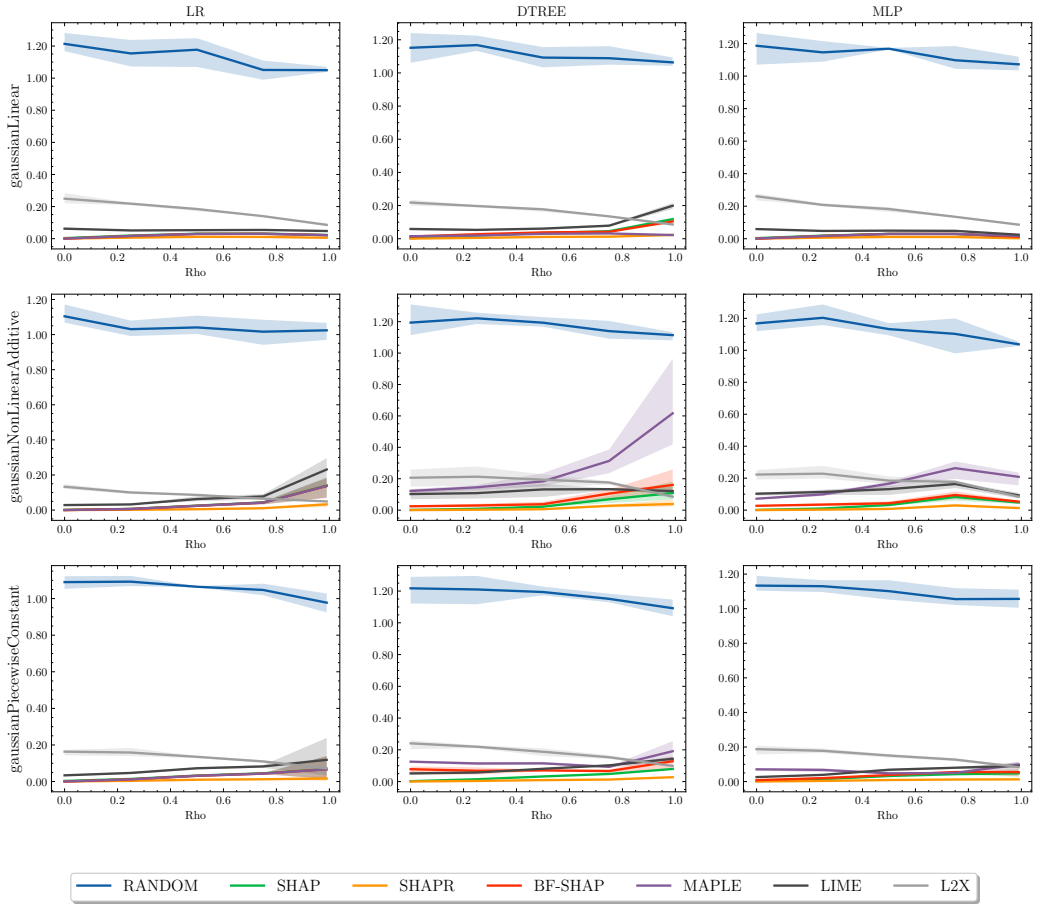


Figure 16: Results of shapley-mse across ML models, dataset types, and ρs.

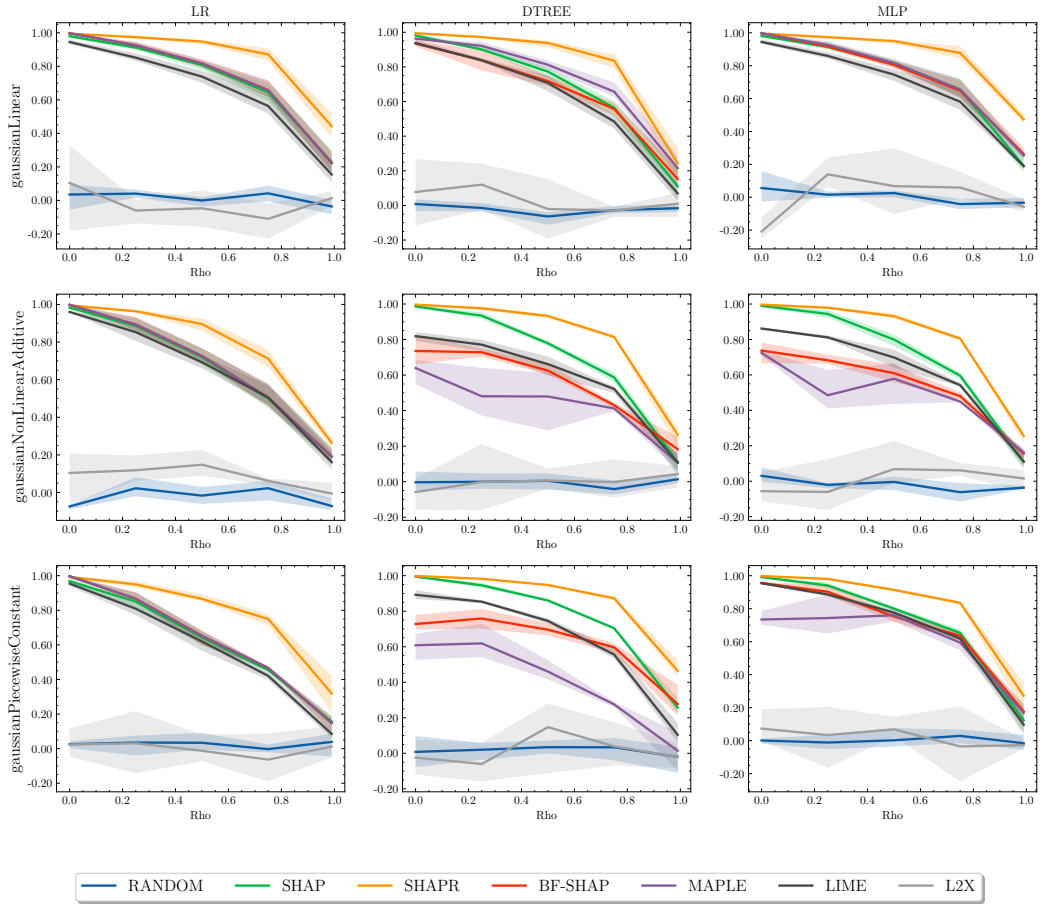


Figure 17: Results of shapley-corr across ML models, dataset types, and ρs.