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 across a variety of settings. 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 rapid 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 [46, 11, 47, 9]. With the widespread deployment of machine learning models in applications that impact human lives, research on model explainability has become increasingly 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 [41, 69, 7, 21]. Many different methods for explainability are actively being explored, including logic rules [26, 63, 56], hidden semantics [68], feature attribution [51, 41, 50, 15, 61], and explanation by example [38, 13]. The most common type of explainers are post-hoc, local feature attribution methods [69, 41, 1, 51, 50, 15], 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 being deployed in different use cases today, currently there are no widely adopted methods to easily

*Equal contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.

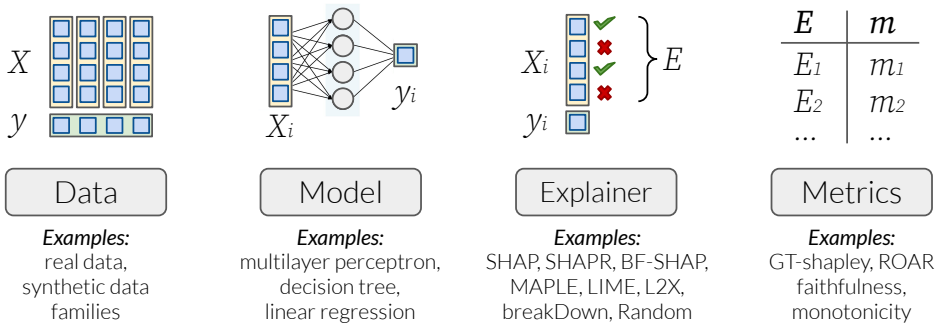


Figure 1: Overview of the main components in XAI-BENCH.

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 [41, 19, 39, 61], 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 [41], remove-and-retrain (ROAR) [31], faithfulness [4], monotonicity [43], and infidelity [67]. 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 failures, such as the deterioration of performance on datasets with high feature correlation. 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 [41], LIME [51], MAPLE [50], SHAPR [1], L2X[15], and breakDown [60], on 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 reduce the time required for researchers and practitioners to move their explainability algorithms from development to deployment. Our code, API docs, and raw experimental results 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.

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 popular evaluation metrics. Our synthetic datasets offer a number 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 [41], LIME [51], MAPLE [50], SHAPR [1], L2X[15], and breakDown [60].

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. [69] describe three dimensions of explainability techniques: passive/active, type of explanation, and local/global explainations. The types of explanations they identified are logic rules [26, 63, 56], hidden semantics [68], feature attribution [51, 41, 50, 15, 61, 1], and explanation by example [38, 13]. Other surveys on explainable AI include Arrieta et al. [6], Adadi and Berrada [2], and Došilović et al. [24].

Techniques for feature attribution include approximating Shapley values [41, 19, 39, 61], approximating the model locally with a more explainable model [51], and approximating the mutual information of each feature with the label [15]. Other work has also identified failure modes for some explanation techniques. For example, recent work has shown that explanation techniques are susceptible to adversarial feature perturbations [23, 58, 30], high feature correlations [35], and small changes in hyperparameters [27, 8].

2.1 Benchmarking Explainability Techniques

One recent work [33] gave an experimental survey of explainability methods, testing SHAP [41], LIME [51], Anchors [52], Saliency Maps [57], Grad-CAM++ [12], and their proposed ExMatchina on image, text, audio, and sensory datasets. They use human labeling via Mechanical Turk as an evaluation metric. Another work [7] 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. Other recent work gives a benchmark on explainability for time-series classification [25], or for natural language processing (NLP) [21]. Finally, concurrent work [5] releases a library with several evaluation metrics for local linear explanation methods and uses the library to compare LIME and SHAP. 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 Metrics

While the "correctness" of feature attribution methods may be subjective or application-specific [66], comparisons between methods are often based on human studies [34, 53, 55]. However, human studies are not always possible, and several empirical (non-human) evaluation metrics have been proposed. Faithfulness [4, 7, 3, 22, 36], infidelity [67, 10, 54], and monotonicity [43, 7, 18] are popular explainability metrics which measure whether each feature's susceptibility to change the model output is aligned with each feature's attribution weight. Another popular metric, remove-and-retrain (ROAR) [31, 28, 29, 44], measures these statistics by retraining the model each time relevant features are removed, in order to avoid inaccuracies due to distribution shift. In the next section, we give the formal definition and a discussion for each metric.

3 Evaluation Metrics

3.1 Preliminaries

We first 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 $\boldsymbol{x} \in [0,1]^D$, yet all of the concepts we discuss can be generalized to arbitrary categorical and real-valued feature distributions. Assume we have a training set $\mathcal{D}_{\text{train}}$ and a test set $\mathcal{D}_{\text{test}}$, both drawn from $\mathcal{D}$. For the case of regression, we train a model $f : [0,1]^D \to [0,1]$ on the training set. We also implement classification using cross-entropy loss. 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 $g(\boldsymbol{x}, f) = \boldsymbol{w} \in \mathcal{R}^D$, 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 [41] or LIME [51].

3.2 Metrics

In this section, we formally define popular evaluation metrics for explainability methods. Each evaluation metric has pros and cons and may be more or less appropriate depending on the application and problem instance. We provide a guide to choosing metrics in Section 3.3.

A *feature attribution evaluation metric* is a function that 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 or below 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, one approach is to evaluate the model when replacing the features S with their expected values conditioned on the remaining features [19, 62, 14, 32]. 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. Later in this section, we discuss other popular choices such as interventional conditional distributions [42, 1]. Given a datapoint $\boldsymbol{x}$, a model f, and a weight vector $\boldsymbol{w}$, the first evaluation metric, **faithfulness** [4], is defined as follows:

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

Intuitively, faithfulness computes the Pearson correlation coefficient [65] between the weight vector $\boldsymbol{w}$ and the approximate marginal contribution $\left|\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{F \setminus i})}[f(\boldsymbol{x}')] - f(\boldsymbol{x})\right|$ for each feature i. Faithfulness is a lightweight metric that is especially useful for comparing which feature would have the most impact on the model output when individually changed.

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 $i+1$. This makes it useful when comparing the effect of features as they are added sequentially. Formally, 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 $\boldsymbol{w}$, **monotonicity** [43] is defined as follows:

$$\text{monotonicity} = \frac{1}{D-1} \sum_{i=0}^{D-2} \mathbb{I}_{|\delta_i^+| \le |\delta_{i+1}^+|}, \tag{3}$$

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

The types of metrics discussed so far all 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 $\mathcal{D}$ that may make the evaluations on f unreliable. To help mitigate this issue, another paradigm of explainability evaluation metrics was proposed: remove-and-retrain (**ROAR**) [31]. 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 $|\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{F \setminus i})}[f(\boldsymbol{x}')] - f(\boldsymbol{x})|$, we would compute $|f^*(\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{F \setminus i})}[\boldsymbol{x}']) - f(\boldsymbol{x})|$, 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 plots the retrained model performance versus the number of features ablated [31], removing features in order of decreasing importance. Then feature attribution methods are compared by inspecting the steepness of these plots. Follow-up work has compressed the ROAR statistic into a scalar value by computing the area-under-the-curve (AUC) [28, 44]. We use this AUC version in Section 5, to be consistent with the other metrics that only output a single value. Note that to compute ROAR 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 these datasets. We give the formal definition in Appendix E.

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 in many cases, it may lead to unreliable measurements for e.g. highly nonlinear models. Furthermore, the explicit goal of a popular line of explainability methods is to obtain fast and accurate approximations of *Shapley values* [41, 1, 40, 19, 39, 61]. To address this, we consider a metric based on Shapley values, **GT-Shapley**, which computes the Pearson correlation coefficient [65] of the feature weights to the ground-truth Shapley values. Shapley values take into account the marginal improvement of a feature i across *all possible* exponentially many sets with and without i.

Next, we consider the **infidelity** metric [67]. This metric is computed by considering the effects of replacing each feature with a *noisy* baseline conditional expectation. Instead of computing the correlation between the feature importances and the change in function values (as in faithfulness and GT-Shapley), infidelity computes the difference between the change in function value and the dot product of the change in feature value with the feature importance vector, in expectation over the noise. Note that if we were to only add noise to one feature at a time, this would be similar in spirit to faithfulness (since the dot product would be equal to the weight of the feature which had noise added). Similar to prior work [67], we consider perturbations based on Gaussian noise. Therefore, infidelity can pick up nonlinear trends in feature importances better than faithfulness or monotonicity.

Finally, while Equation (1) defines "observational" conditional expectations [41, 1], we also implement "interventional" conditional expectations [19, 62], which are defined by assuming the features in S are independent of the remaining features. This can be applied to all metrics defined in this section. The best choice of conditional expectations depends on the application [14], and we discuss the tradeoffs in the next section.

3.3 A guide to choosing metrics

All of the metrics listed above may be used for evaluating and comparing different feature attribution techniques. However, each metric has strengths and weaknesses, and choosing the most useful metric for a given situation depends on the use case, dataset, feature attribution technique, and computational constraints. We discuss strengths, weaknesses, and example use cases of each metric type.

For the ROAR paradigm, retraining the model with the most important features removed is especially important when the original model is not calibrated for out-of-distribution predictions [31], such as in high-dimensional applications like computer vision [44, 29, 59]. However, retraining might fail to give an accurate evaluation in the presence of high feature correlations [48]. Furthermore, retraining the model incurs a much larger computational cost.

For some feature attribution algorithms, the explicit goal is to efficiently approximate the Shapley values [41, 1, 40, 19, 39, 61], and the GT-Shapley metric is the best choice to determine which technique gives the best approximations to the true Shapley values. However, evaluating the ground-truth Shapley values has a computational cost that is exponential in the number of features. Therefore, the GT-Shapley metric is slow to evaluate on high-dimensional datasets.

Faithfulness, monotonicity, and infidelity are far less computationally intensive compared to ROAR and GT-Shapley. The main difference between faithfulness and monotonicity is that faithfulness considers subsets of features by iteratively removing the most important features *with replacement*, while monotonicity does this *without replacement*. Therefore, the former is better for applications where the main question is which features would individually change the output of the model on a given datapoint (and therefore may be better on datasets with less correlated features). The latter is better for applications where the main goal is to see the cumulative effect of adding features (and therefore performs comparatively better in the presence of correlated features).

The main difference between infidelity and faithfulness (as well as monotonicity) is that infidelity considers ablations of subsets of features, while faithfulness only considers ablating a single feature at a time. Therefore, infidelity may be more appropriate for models with highly nonlinear feature interactions, compared to faithfulness and monotonicity.

Finally, we discuss using interventional versus observational conditional expectations. As pointed out in prior work [14], interventional conditional expectations are better for applications that require being "true to the model", while observational conditional expectations are better for applications that require being "true to the data", because observational conditional expectations tend to spread out importance among correlated features (even features that are not used by the model). For example, interventional conditional expectations are more appropriate in explaining why a model caused a loan to be denied, while observational conditional expectations are more appropriate in explaining the causal features in the drug response to RNA sequences [14].

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, and then describe the feature distributions implemented for these datasets.

4.1 The case for synthetic data

As shown in Section 3.2, for multiple metrics it is key to compute 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 $\mathcal{D}(\boldsymbol{x}_S)$ can only be approximated, and the approximation may be very poor when the conditional distribution defines low-density regions of the feature space. Since all evaluation metrics require computing $\Theta(D)$ or $\Theta(2^D)$ 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 [41] 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 Synthetic feature distributions

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 implement multiple families of synthetic distributions in our library, including multivariate Gaussian, mixture of Gaussians, and multinomial feature distributions.

To give a concrete example, we describe here how to generate and use multivariate Gaussian synthetic features. The multivariate normal distribution of a 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 D-dimensional mean vector, and $\boldsymbol{\Sigma}$ is the $D \times D$ covariance matrix. Without loss of generality, we can partition the 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 K-dimensional vector with $0 < K < D$, 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

$$\boldsymbol{\mu}^* = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2^* - \boldsymbol{\mu}_2), \quad \boldsymbol{\Sigma}^* = \boldsymbol{\Sigma}_{11} + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}. \tag{5}$$

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. Parameter $\boldsymbol{\mu}$ can take any value, and $\boldsymbol{\Sigma}$ must be symmetric and positive definite. Similarly, we also give the derivation for additional distribution families in Appendix D, including mixtures of multivariate Gaussians, and multinomial features.

4.3 Labels

After defining a distribution of features via one of the above distribution families, we can then define a distribution over labels. The distributions we implement are `linear`, `piecewise constant`, `nonlinear additive`, and `piecewise linear`.

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 $y_{raw} = \boldsymbol{w}^T\boldsymbol{x}$. In our experiments in Section 5, we set $\boldsymbol{w} = [0, 1, \ldots, d-1]$. `piecewise linear` datasets are similar to `linear`, but a different weight vector is used in different parts of the feature space. In our experiments in Section 5, on the datasets with continuous features, we set $\boldsymbol{w} = [0, 1, \ldots, d-1]$ when the sum of the feature values is positive, and $\boldsymbol{w} = [d-1, d-2, \ldots, 0]$ otherwise. 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. [15]. Detailed specifications can be found in Appendix F.

5 Experiments

We show experiments on several popular feature attribution methods across synthetic datasets.

5.1 Feature attribution methods

We compare eight different feature attribution methods: SHAP [41], SHAPR [1], brute-force Kernel SHAP (BF-SHAP) [41], LIME [51], MAPLE [50], L2X [15], breakDown [60], and the baseline RANDOM, which outputs random weights drawn from a standard normal distribution. We ran light hyperparameter tuning on all datasets. See Appendix E for details and descriptions for all methods. We report the mean and standard deviation from ten trials for all experiments.

5.2 Parameterized synthetic data experiments

We first show 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 the convenient parameterization of a global level of feature dependence [1].

We run experiments that compare eight feature attribution methods on the five evaluation metrics defined in Section 3.1 across several datasets and ML models. 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 G, we give the exhaustive set of experiments. Throughout this section, we will identify different types of failure modes, for example, failures for some explainability techniques over specific metrics (Table 1) or failures for some techniques on datasets with high levels of feature correlation (Figures 2 and 3).

Performance across metrics As shown in Table 1, the relative performance of explainers varies dramatically across metrics for a fixed multilayer perceptron trained on a `nonlinear additive` dataset with $\rho = 0.5$. Since $\rho = 0.5$ implies that the features are fairly correlated, we find that SHAPR outperforms SHAP on GT-Shapley, which is consistent with the fact that SHAPR was designed to outperform SHAP in the presence of dependent features [1]. SHAPR achieved the top performance for three metrics, but MAPLE had the most consistent performance across all five metrics. One possible explanation for this is that MAPLE draws on ideas from three different areas of explainability: example-based, local, and global explanations [50], which helps it achieve steady performance across many metrics. Finally, while breakDown achieves the worst score for GT-Shapley, it achieves the best score for monotonicity. Note that breakDown works by greedily choosing the features with the greatest effect on the model output, *with replacement*, making it particularly well-suited for the monotonicity metric, which checks whether replacing features sorted by importance with their background value with replacement monotonically decreases the change in model output.

Table 1: Explainer performance across metrics. All performance numbers are from explaining a multilayer perceptron trained on the Gaussian nonlinear additive dataset with $\rho = 0.5$.

	RANDOM	SHAP	SHAPR	LIME	MAPLE	L2X	BREAKDOWN
faithfulness(↑)	$0.002_{\pm 0.034}$	$0.651_{\pm 0.051}$	$\mathbf{0.799}_{\pm 0.036}$	$0.524_{\pm 0.06}$	$0.478_{\pm 0.061}$	$0.000_{\pm 0.075}$	$0.110_{\pm 0.049}$
monotonicity(↑)	$0.525_{\pm 0.017}$	$0.537_{\pm 0.014}$	$0.550_{\pm 0.025}$	$0.517_{\pm 0.022}$	$0.543_{\pm 0.026}$	$0.535_{\pm 0.022}$	$\mathbf{0.562}_{\pm 0.021}$
ROAR(↑)	$0.380_{\pm 0.051}$	$0.455_{\pm 0.054}$	$\mathbf{0.465}_{\pm 0.054}$	$0.432_{\pm 0.051}$	$0.432_{\pm 0.059}$	$0.365_{\pm 0.053}$	$0.329_{\pm 0.057}$
GT-Shapley(↑)	$0.004_{\pm 0.049}$	$0.810_{\pm 0.023}$	$\mathbf{0.930}_{\pm 0.012}$	$0.711_{\pm 0.032}$	$0.530_{\pm 0.128}$	$-0.014_{\pm 0.068}$	$-0.127_{\pm 0.066}$
infidelity(↓)	$0.114_{\pm 0.058}$	$0.050_{\pm 0.023}$	$0.036_{\pm 0.013}$	$0.053_{\pm 0.016}$	$\mathbf{0.019}_{\pm 0.011}$	$0.025_{\pm 0.010}$	$0.126_{\pm 0.057}$

Performance across dataset types and feature correlations Next, we explore how the type of dataset and feature correlation affects performance of explainers on a multilayer perceptron 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 Kernel SHAP assume feature independence [1, 45] and tend to perform well when features are indeed independent ($\rho = 0$). This is especially apparent with the `linear` dataset, where the performance of most methods cluster above 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. For experiments on the `piecewise linear` dataset, see Appendix G.

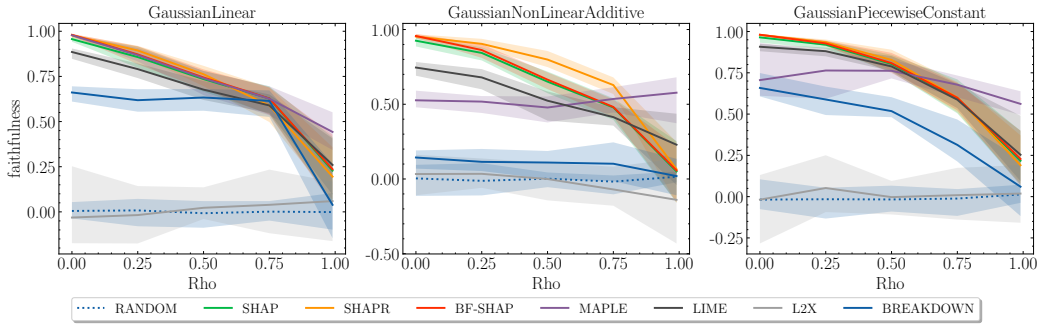


Figure 2: Results for faithfulness on a multilayer perceptron trained on three different datasets.

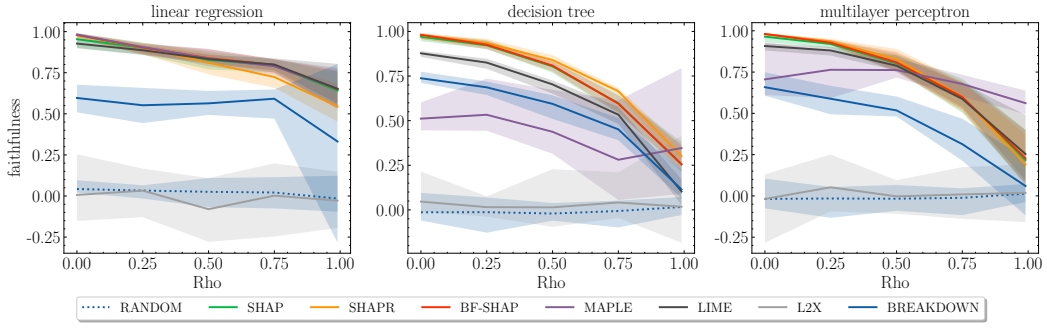


Figure 3: Results for faithfulness for three types of ML models—linear regression, decision tree, and multilayer perceptron—trained on a Gaussian piecewise constant dataset.

Performance across ML models Next, we train three ML models: linear regression, decision tree, and multilayer perceptron, with a `piecewise constant` dataset and compare faithfulness. 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, MAPLE performs significantly worse on the decision tree model.

5.3 Simulating real datasets

In this section, we demonstrate the power and flexibility of synthetic datasets by simulating two popular datasets: the wine quality dataset [16, 60] and the forest fire dataset [17] with synthetic features so that they can be used to efficiently benchmark feature attribution methods.

Wine quality dataset The 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) [64] 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 seven 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 2, SHAPR performs well on GT-Shapley, consistent with Table 1. SHAP and SHAPR both outperform LIME and MAPLE on faithfulness.

Forest fire dataset The forest fire dataset has 12 continuous features and one real-valued label indicating the area of burned forest. Again, we normalize the features to have zero mean and unit variance, and then we compute the covariance matrix, which is used to generate the synthetic dataset (the same way as the wine quality dataset above).

For the forest fire dataset, the JSD of marginal distributions between the real empirical features and the synthetic Gaussian features has a mean of 0.17, and the JSD of real and synthetic targets is 0.15, suggesting a good fit. We compute evaluation metrics for six different explainers on the synthetic forest fire dataset. See Table 3. SHAP achieved top performance on three of the five metrics.

Table 2: Explainer performance on the simulated wine dataset across metrics. All performance numbers are from explainers for a decision tree.

	RANDOM	SHAP	SHAPR	LIME	MAPLE	L2X	BREAKDOWN
faithfulness (↑)	$-0.007_{\pm 0.005}$	$\mathbf{0.534}_{\pm 0.045}$	$0.528_{\pm 0.032}$	$0.368_{\pm 0.031}$	$0.034_{\pm 0.033}$	$-0.030_{\pm 0.018}$	$-0.042_{\pm 0.011}$
monotonicity (↑)	$0.529_{\pm 0.008}$	$0.549_{\pm 0.009}$	$\mathbf{0.551}_{\pm 0.009}$	$0.547_{\pm 0.007}$	$0.520_{\pm 0.014}$	$0.522_{\pm 0.005}$	$0.493_{\pm 0.014}$
ROAR (↑)	$0.698_{\pm 0.031}$	$0.780_{\pm 0.016}$	$0.549_{\pm 0.031}$	$0.738_{\pm 0.026}$	$\mathbf{0.818}_{\pm 0.022}$	$0.664_{\pm 0.02}$	$0.625_{\pm 0.002}$
GT-Shapley (↑)	$0.004_{\pm 0.013}$	$0.825_{\pm 0.006}$	$\mathbf{0.945}_{\pm 0.002}$	$0.745_{\pm 0.015}$	$0.685_{\pm 0.008}$	$-0.108_{\pm 0.029}$	$-0.064_{\pm 0.02}$
infidelity (↓)	$0.353_{\pm 0.174}$	$0.234_{\pm 0.124}$	$\mathbf{0.212}_{\pm 0.146}$	$0.234_{\pm 0.126}$	$0.234_{\pm 0.132}$	$0.285_{\pm 0.115}$	$0.365_{\pm 0.133}$

Table 3: Explainer performance on the simulated forest fires dataset across metrics. All performance numbers are from explainers for a decision tree.

	RANDOM	SHAP	LIME	MAPLE	L2X	BREAKDOWN
faithfulness (↑)	$0.022_{\pm 0.034}$	$\mathbf{0.571}_{\pm 0.023}$	$0.449_{\pm 0.007}$	$0.080_{\pm 0.056}$	$0.001_{\pm 0.008}$	$0.158_{\pm 0.032}$
monotonicity (↑)	$0.537_{\pm 0.02}$	$0.591_{\pm 0.007}$	$\mathbf{0.598}_{\pm 0.002}$	$0.561_{\pm 0.002}$	$0.527_{\pm 0.01}$	$0.575_{\pm 0.012}$
ROAR (↑)	$0.575_{\pm 0.002}$	$0.615_{\pm 0.011}$	$0.616_{\pm 0.008}$	$\mathbf{0.696}_{\pm 0.024}$	$0.534_{\pm 0.018}$	$0.604_{\pm 0.019}$
GT-Shapley (↑)	$0.012_{\pm 0.06}$	$\mathbf{0.870}_{\pm 0.005}$	$0.779_{\pm 0.027}$	$0.804_{\pm 0.011}$	$0.031_{\pm 0.12}$	$0.105_{\pm 0.013}$
infidelity (↓)	$0.207_{\pm 0.125}$	$\mathbf{0.075}_{\pm 0.074}$	$0.077_{\pm 0.075}$	$0.077_{\pm 0.079}$	$0.091_{\pm 0.07}$	$0.117_{\pm 0.076}$

5.4 Recommended usage

In Section 5, we gave a sample of the types of experiments that can be performed with our library (recall that comprehensive experiments are in Appendix G). For researchers looking to develop new explainability techniques, we recommend benchmarking new algorithms across all metrics using our synthetic 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. See Section 3.3 for a guide to choosing the best metric based on the application.

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 important for the purposes of debugging, legal obligations, and mitigating bias [41, 69, 7, 21]. Given the importance of high-quality explanations, it is essential that 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 [37]. Frameworks for evaluating and comparing explainability methods are an important part of creating inclusive and

unbiased technology. As pointed out in prior work [20], 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 accurate 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, for example, techniques whose performance is negatively correlated with dataset feature correlation. We showcase the power of our library by benchmarking several popular explainers with respect to five evaluation metrics across a variety of settings.

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 failures, 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 process of moving 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.

Acknowledgments and Disclosure of Funding

Work done while the first three authors were working at Abacus.AI. WN was supported by U.S. Department of Energy Office of Science under Contract No. DE-AC02-76SF00515.

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] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

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

[5] Elvio Amparore, Alan Perotti, and Paolo Bajardi. To trust or not to trust an explanation: using leaf to evaluate local linear xai methods. *PeerJ Computer Science*, 7:e479, 2021.

[6] 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.

[7] 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.

[8] Naman Bansal, Chirag Agarwal, and Anh Nguyen. Sam: The sensitivity of attribution methods to hyperparameters. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 8673–8683, 2020.

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

[10] Umang Bhatt, Adrian Weller, and José MF Moura. Evaluating and aggregating feature-based model explanations. *arXiv preprint arXiv:2005.00631*, 2020.

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

[12] 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.

[13] 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.

[14] Hugh Chen, Joseph D Janizek, Scott Lundberg, and Su-In Lee. True to the model or true to the data? *arXiv preprint arXiv:2006.16234*, 2020.

[15] 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.

[16] 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.

[17] Paulo Cortez and Aníbal de Jesus Raimundo Morais. A data mining approach to predict forest fires using meteorological data. 2007.

[18] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[19] 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.

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

[21] 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.

[22] Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. Model agnostic contrastive explanations for structured data. *arXiv preprint arXiv:1906.00117*, 2019.

[23] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Advances in Neural Information Processing Systems*, 32:13589–13600, 2019.

[24] 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.

[25] 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.

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

[27] Damien Garreau and Ulrike Luxburg. Explaining the explainer: A first theoretical analysis of lime. In *International Conference on Artificial Intelligence and Statistics*, pages 1287–1296. PMLR, 2020.

[28] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Explaining failure: Investigation of surprise and expectation in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 12–13, 2020.

[29] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Swag: Superpixels weighted by average gradients for explanations of cnns. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 423–432, 2021.

[30] Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. *Advances in Neural Information Processing Systems*, 32:2925–2936, 2019.

[31] 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.

[32] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable ai: A causal problem. In *International Conference on Artificial Intelligence and Statistics*, pages 2907–2916. PMLR, 2020.

[33] 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.

[34] 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.

[35] Himabindu Lakkaraju and Osbert Bastani. " how do i fool you?" manipulating user trust via misleading black box explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 79–85, 2020.

[36] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 131–138, 2019.

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

[38] 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.

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

[40] Scott M Lundberg and Su-In Lee. Consistent feature attribution for tree ensembles. *arXiv preprint arXiv:1706.06060*, 2017.

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

[42] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

[43] 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.

[44] Chuizheng Meng, Loc Trinh, Nan Xu, and Yan Liu. Mimic-if: Interpretability and fairness evaluation of deep learning models on mimic-iv dataset. *arXiv preprint arXiv:2102.06761*, 2021.

[45] Christoph Molnar. *Interpretable Machine Learning*. 2019.

[46] 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.

[47] 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.

[48] An-phi Nguyen and María Rodríguez Martínez. On quantitative aspects of model interpretability. *arXiv preprint arXiv:2007.07584*, 2020.

[49] 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.

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

[51] 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.

[52] 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.

[53] 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.

[54] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[55] 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.

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

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

[58] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[59] Suraj Srinivas and François Fleuret. Full-gradient representation for neural network visualization. *arXiv preprint arXiv:1905.00780*, 2019.

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

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

[62] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *International Conference on Machine Learning*, pages 9269–9278. PMLR, 2020.

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

[64] 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.

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

[66] Mengjiao Yang and Been Kim. Benchmarking attribution methods with relative feature importance. *arXiv preprint arXiv:1907.09701*, 2019.

[67] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Suggala, David I Inouye, and Pradeep K Ravikumar. On the (in) fidelity and sensitivity of explanations. *Advances in Neural Information Processing Systems*, 32:10967–10978, 2019.

[68] 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.

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

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 [49]. An earlier verision of this checklist (v1.2) was used for NeurIPS 2019 [49].

- 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 reported 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 evaluation 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, which we discuss in Appendix E.2. Our repository allows setting the hyperparameters to other values set by the user.
  - **The exact number of training and evaluation runs.** We reported that we ran ten trials for each experiment.
  - **A clear definition of the specific measure or statistics used to report results.** We define our metrics in Section 3.2.
  - **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 G.
  - **A description of the computing infrastructure used.** We use CPUs for all experiments. We give details of our experiments in Appendix Section G.

C Multivariate Gaussian distribution

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

$$f_x(x_1, ..., x_D) = \frac{\exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}))}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}}, \tag{6}$$

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

D Additional Synthetic Feature Distributions

Mixture of multivariate Gaussians features We first describe mixture of multivariate Gaussians features. Suppose now that $\boldsymbol{X} = (X_1, ..., X_D)^T$ is a 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 D-dimensional mean vector for the j^{th} mixture component, and $\boldsymbol{\Sigma}_j$ is the $D \times D$ 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

$$\boldsymbol{\mu}_{\boldsymbol{j}}^* = \boldsymbol{\mu}_{j,1} + \boldsymbol{\Sigma}_{j,12}\boldsymbol{\Sigma}_{j,22}^{-1}(\boldsymbol{x}_2^* - \boldsymbol{\mu}_{j,2}) \tag{7}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{j}}^* = \boldsymbol{\Sigma}_{j,11} + \boldsymbol{\Sigma}_{j,12}\boldsymbol{\Sigma}_{j,22}^{-1}\boldsymbol{\Sigma}_{j,21} \tag{8}$$

$$\pi_j^* = \frac{\pi_j f_{j,2}(\boldsymbol{x}_2^*)}{\sum_{\ell=1}^k \pi_\ell f_{\ell,2}(\boldsymbol{x}_2^*)} \tag{9}$$

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

Multinomial features We follow a similar derivation for the conditional distribution of a multinomial distribution. Suppose now that $\boldsymbol{X} = (X_1, ..., X_D)^T$ is a D-dimensional random vector following a multinomial distribution, where $X_i \in \{0, \ldots, m\}$, and $\sum_{i=1}^D X_i = m$. We write this as $\boldsymbol{X} \sim \text{Multinomial}(m, p_1, \ldots, p_D)$, where the parameter $m > 0$ denotes the number of trials, and the parameters $p_1, \ldots, p_D$ denote the D event probabilities.

Suppose, as before we use the partition defined by $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \end{bmatrix}$. Then, given $X_2 = \boldsymbol{x}_2^* \in \{0, \ldots, m\}^k$, the conditional distribution is also distributed as a multinomial, written $(\boldsymbol{X}_1 | \boldsymbol{X}_2 = \boldsymbol{x}_2^*) \sim \text{Multinomial}(m^*, p_1^*, \ldots, p_{D-k}^*)$, where the parameters of of this multinomial can be written $m^* = m - \sum_{j=1}^k x_{2,j}^*$, and $p_i^* = p_i / \left(1 - \sum_{j-1}^k p_j\right)$.

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 $\boldsymbol{x}$:

$$\bar{\boldsymbol{x}}_S = \begin{cases} x_i \text{ for indices } i \in S \\ \mathbb{E}\left[x_i' \mid \boldsymbol{x'} \sim \mathcal{D} \text{ s.t. } x_j' = x_j \text{ for } j \in S\right] \text{ for indices } i \notin S \end{cases} \tag{10}$$

Recall from Section 3 that $S^+(\boldsymbol{w}, i)$ 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 most important features for each datapoint. Let $f^{S(k)+}$ denote the model f retrained on $\mathcal{D}_{\text{train}}^{S(k)+}$ instead of $\mathcal{D}_{\text{train}}$. Then **ROAR** is defined as follows:

$$\bar{\delta}_i^+ = f^{S(k)+}(\bar{\boldsymbol{x}}_{S_{i+1}^+(\boldsymbol{w})}) - f^{S(k)+}(\bar{\boldsymbol{x}}_{S_i^+(\boldsymbol{w})}), \tag{11}$$

$$\text{ROAR} = \frac{1}{D-1} \sum_{i=0}^{D-2} \mathbb{I}_{|\delta_i^+| \leq |\bar{\delta}_{i+1}^+|} \tag{12}$$

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

$$v_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} (\mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_{S \cup \{i\}})}[f(\boldsymbol{x}')] - \mathbb{E}_{\boldsymbol{x}' \sim \mathcal{D}(\boldsymbol{x}_S)}[f(\boldsymbol{x}')]), \tag{13}$$

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

$$\text{GT-Shapley} = \texttt{Pearson}\left([v_i]_{1 \le i \le D}, [w_i]_{1 \le i \le D}\right). \tag{14}$$

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. [41] proposed a few methods such as BF-SHAP to estimate Shapley values defined by Equation 13. 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) [51] 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 [50] 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 [15] 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:

$$\boldsymbol{w} = \frac{2}{k(k+1)} \sum_{k=1}^{D} I_k, \tag{15}$$

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 BREAKDOWN

BREAKDOWN [60] is another technique to decompose model predictions into parts that can be attributed to particular variables. We use the official python implementation from `https://github.com/MI2DataLab/pyBreakDown`.

E.2.7 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:

$$\Psi_1(x_1) = \begin{cases} 1, \ x_1 >= 0 \\ -1, \ x_1 < 0 \end{cases} \tag{16}$$

$$\Psi_2(x_2) = \begin{cases} -2, \ x_2 < -0.5 \\ -1, \ -0.5 \leq x_2 < 0 \\ 1, \ -0 \leq x_2 < 0.5 \\ 2, \ x_2 \geq 0.5 \end{cases} \tag{17}$$

$$\Psi_3(x_3) = floor(2cos(\pi x_3)) \tag{18}$$

$$\Psi_i(x_i) = 0, \ i = 4, 5 \tag{19}$$

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

Nonlinear additive:

$$\Psi_1(x_1) = sin(x_1) \tag{20}$$

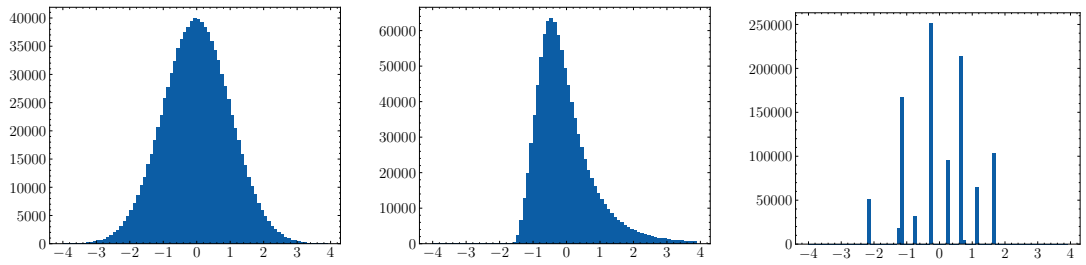
$$\Psi_2(x_2) = |x_2| \tag{21}$$

$$\Psi_3(x_3) = x_3^2 \tag{22}$$

$$\Psi_4(x_4) = e^{x_4} \tag{23}$$

$$\Psi_5(x_5) = 0 \tag{24}$$

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.

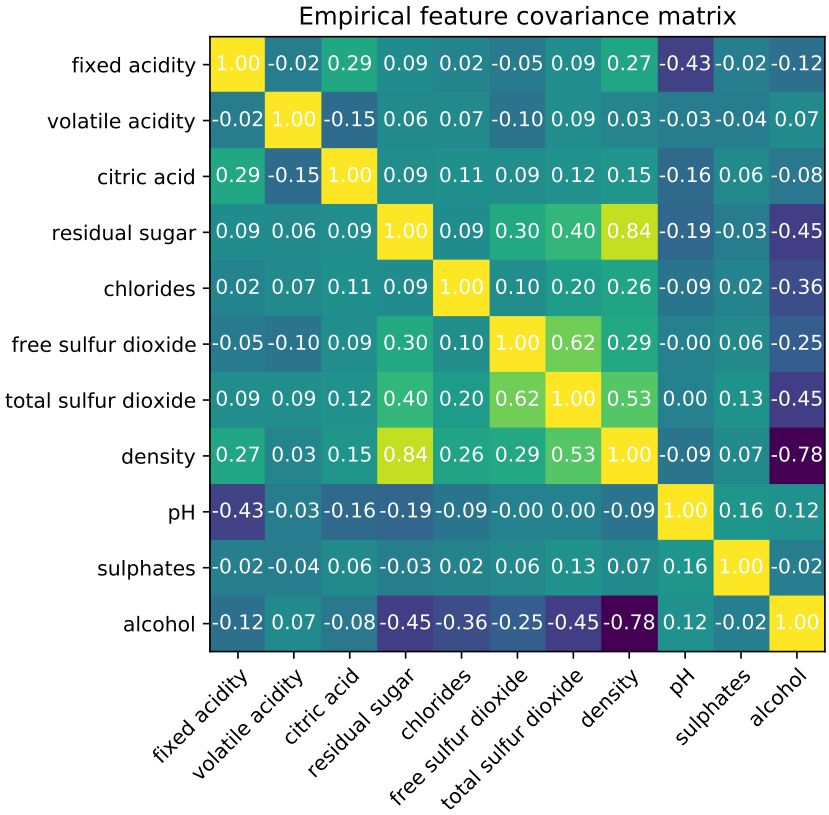


Figure 5: Empirical covariance matrix of the wine dataset. Features are normalized to have unit variance and zero mean.

G Additional results

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

Table 4: Time taken in seconds by explainers to explain 100 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 five different evaluation metrics, eight 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	

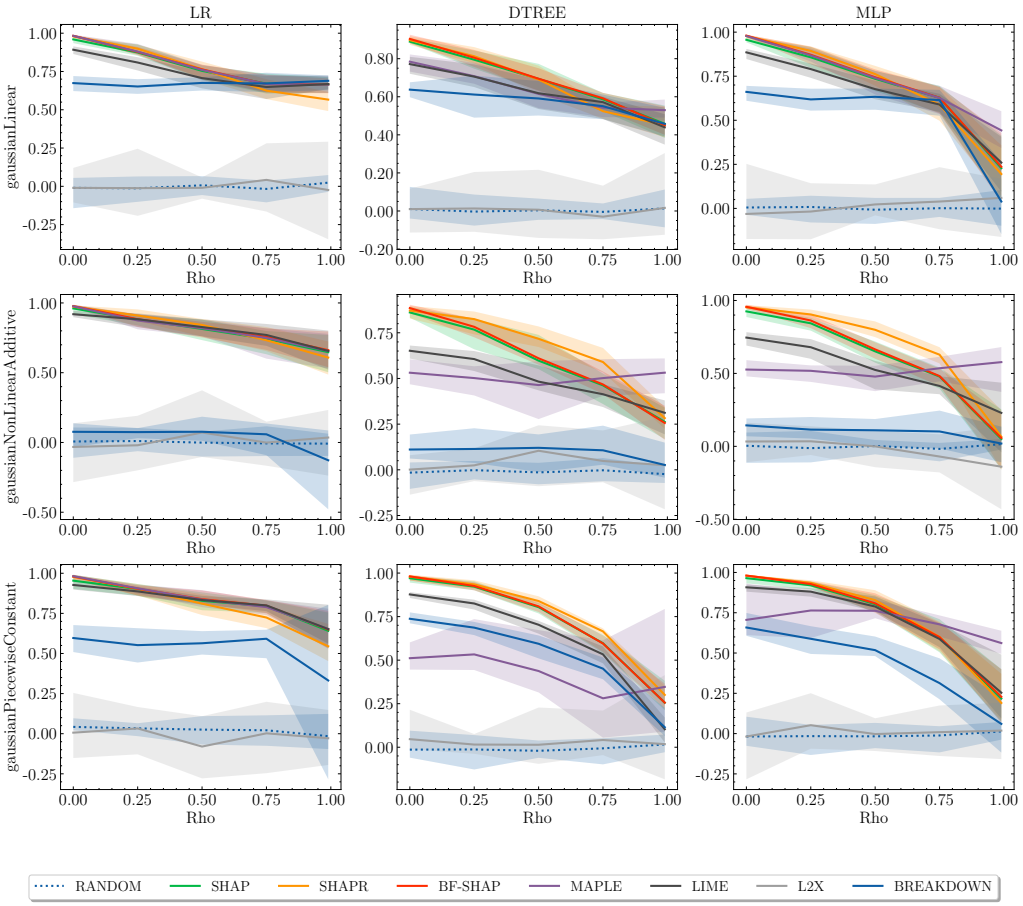


Figure 6: Results of faithfulness across ML models, dataset types, and ρs.

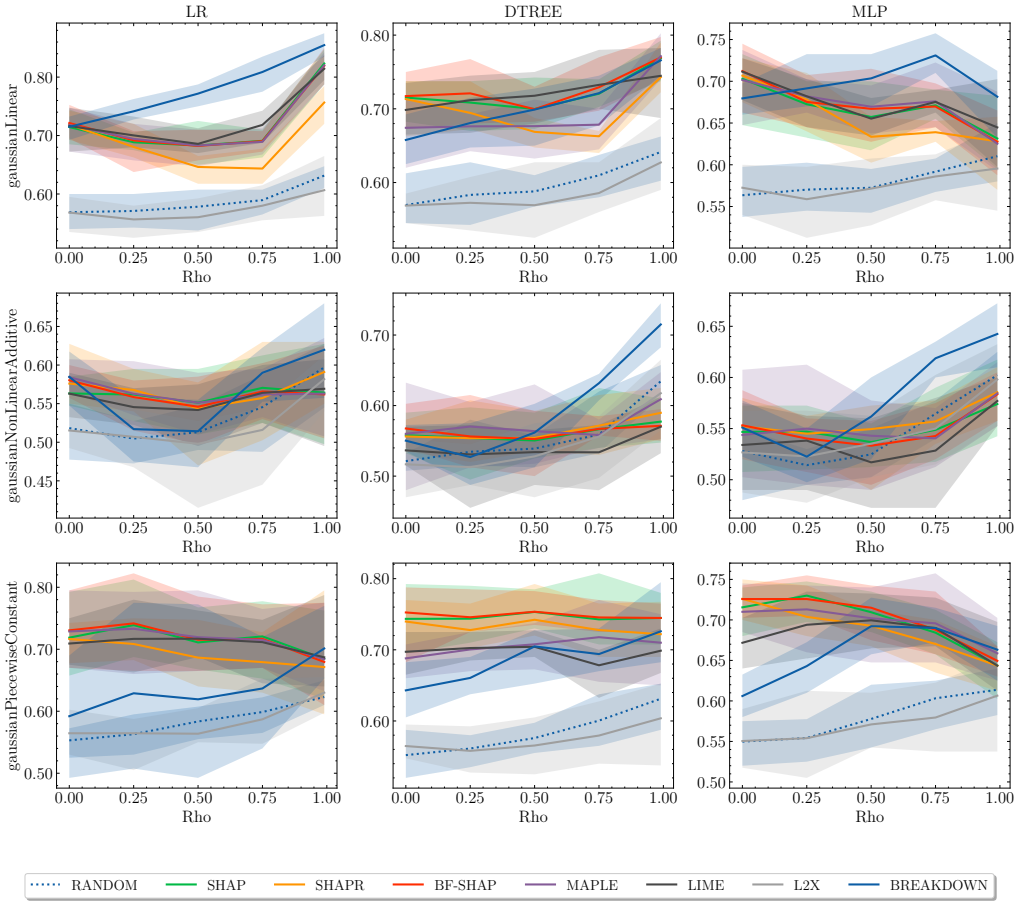


Figure 7: Results of monotonicity across ML models, dataset types, and ρs.

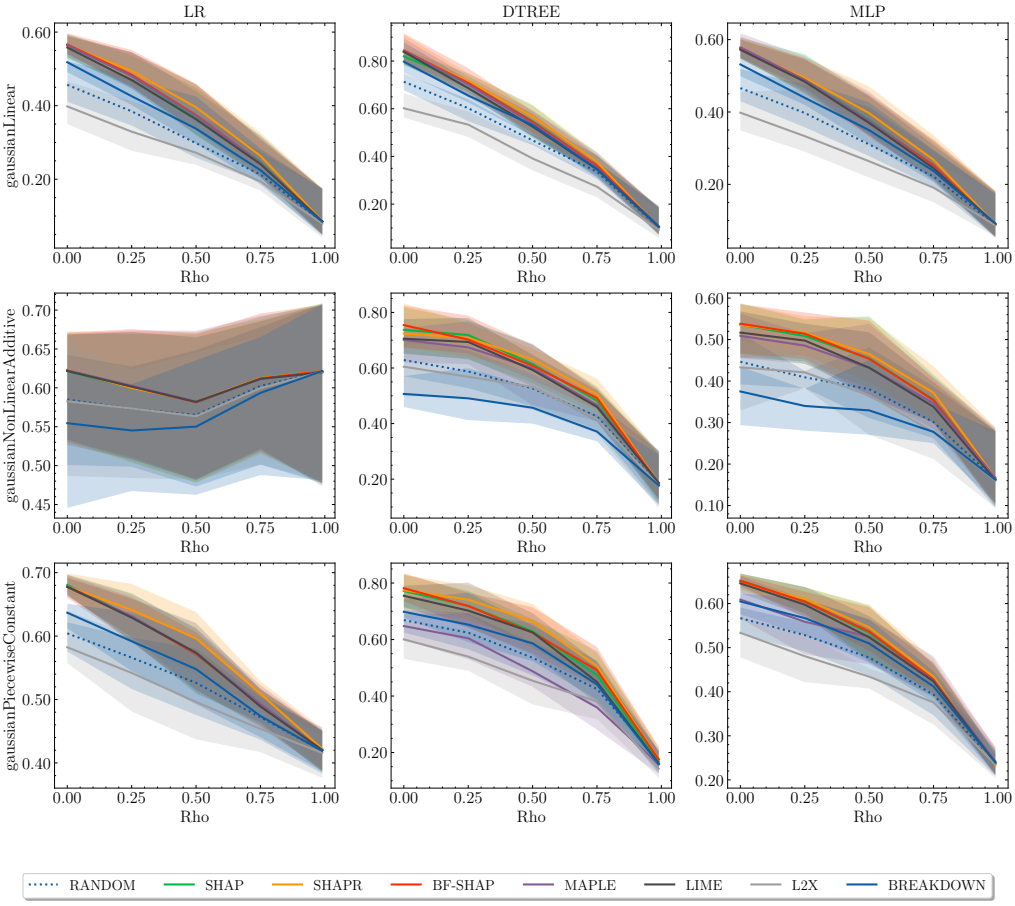


Figure 8: Results of ROAR across ML models, dataset types, and ρs.

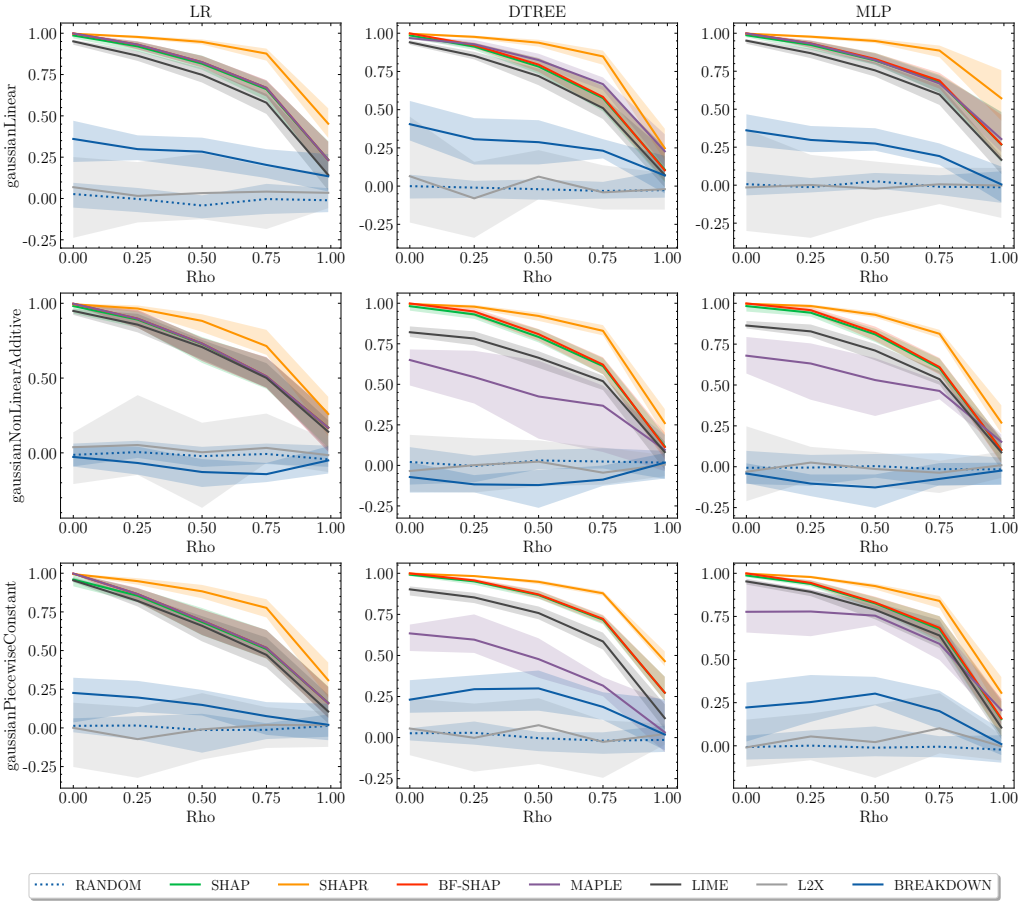


Figure 9: Results of GT-Shapley across ML models, dataset types, and ρs.

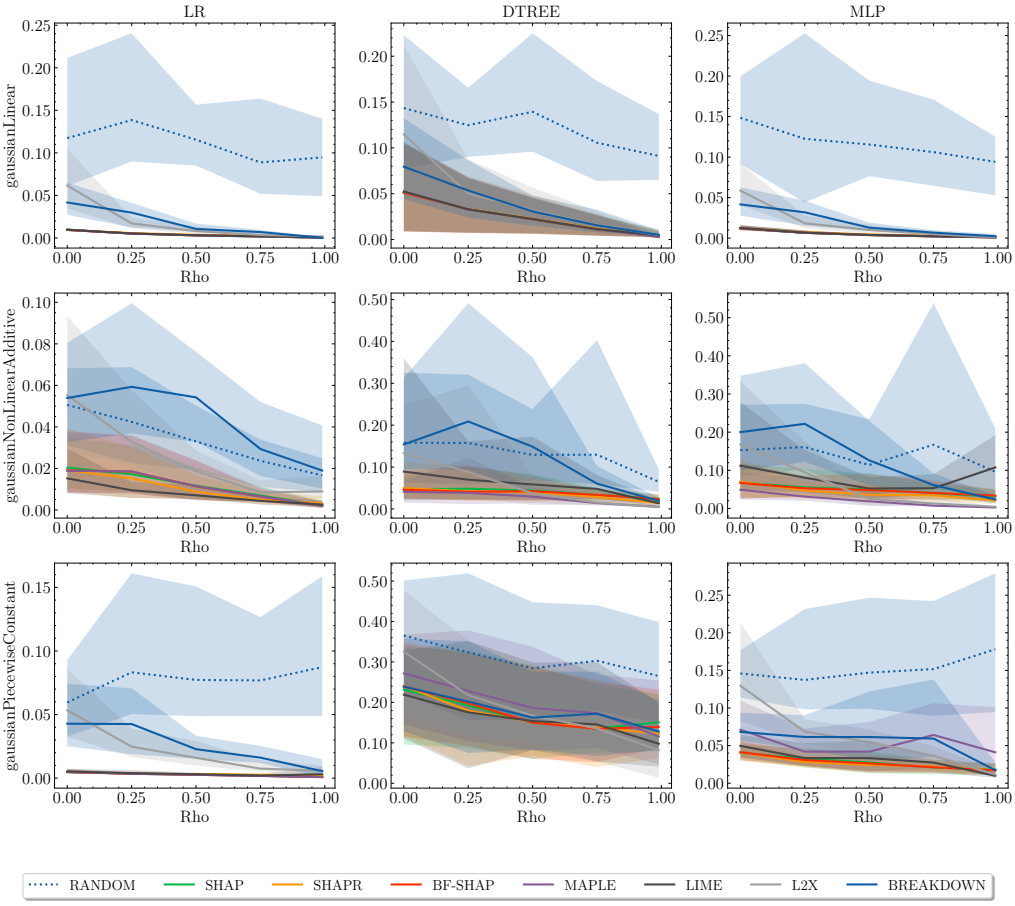


Figure 10: Results of infidelity across ML models, dataset types, and ρs.