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, 11, 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.
**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](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 explanations. 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 \(D\), each datapoint is of the form \((x, y) \sim D\), where \(x\) denotes the set of features, and \(y\) denotes the label. We assume that \(x \in [0, 1]^D\), yet all of the concepts we discuss can be generalized to arbitrary categorical and real-valued feature distributions. Assume we have a training set \(D_{\text{train}}\) and a test set \(D_{\text{test}}\), both drawn from \(D\). 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 \(x\), then \(g(x, f) = w \in \mathbb{R}^D\), where each output weight \(w_i\) corresponds to the relative importance of feature \(i\) when making the prediction \(f(x)\). 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 \(x\). For example, given a datapoint \(x\) and a set of feature weights \(w = g(x, f)\), then a value near or below zero indicates that \(g\) did not provide an accurate feature attribution estimate for \(x\), 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 $\{1, \ldots, D\}$. Formally, given a datapoint $x \sim D$ and a set of indices $S \subseteq \{1, \ldots, D\}$, we define $D(x_S)$ as the conditional probability distribution $x' \sim D$ such that $x'_i = x_i$ for all $i \in S$. In other words, given $x$ and $S$, we have

$$p(x' \sim D(x_S)) = p(x' \sim D \mid x'_i = x_i \text{ for all } i \in S).$$

(1)

By this definition, $D(x_S) = D$, and if we define $F = \{1, \ldots, D\}$, then $x' \sim D(x_F)$ is equal to $x$ with probability 1. Later in this section, we discuss other popular choices such as interventionist conditional distributions $\{g_i\}$. Given a datapoint $x$, a model $f$, and a weight vector $w$, the first evaluation metric, **faithfulness** [4], is defined as follows:

$$\text{faithfulness} = \text{Pearson} \left( \left[ E_{x' \sim D(x_{F \setminus S})} [f(x')] - f(x) \right]_{1 \leq i \leq D} \right),$$

(2)

Intuitively, faithfulness computes the Pearson correlation coefficient [65] between the weight vector $w$ and the approximate marginal contribution $E_{x' \sim D(x_{F \setminus S})} [f(x')] - f(x)$ 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 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 $S^+(w, i)$ as the set of $i$ most important weights, and let $S^+(w, 0) = \emptyset$. Given a datapoint $x$, a model $f$, and a weight vector $w$, **monotonicity** [33] is defined as follows:

$$\text{monotonicity} = \frac{1}{D - 1} \sum_{i=0}^{D-2} \delta_i^+,$$

(3)

where

$$\delta_i^+ = E_{x' \sim D(x_{S^+(w, i + 1)})} [f(x')] - E_{x' \sim D(x_{S^+(w, i)})} [f(x')]$$

(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 $D$, and are therefore unlikely to occur in $D_{train}$. This is especially true for structured data or data where there are large low-density regions in $D$ 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 $|E_{x' \sim D(x_{F \setminus S})} [f(x')] - f(x)|$, we would compute $|f^* (E_{x' \sim D(x_{F \setminus S})} [x']) - f(x)|$, where $f^*$ denotes a model that has been trained on a modification of $D_{train}$ 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 $i$. 

4
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 \(\mathbb{E}_{x' \sim D(x)}[f(x')]\) for a subset \(S\), datapoint \(x\), and trained model \(f\). On real-world datasets, the conditional distribution \(D(x_S)\) 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 \(x\), 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 \(y = h(x)\), with \(y\) as label and \(x\) as feature vector. The generation is split into two parts, generating features \(x\), and defining a function to generate labels \(y\) from \(x\). 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 \(X = (X_1, ..., X_D)^T\) can be written as \(X \sim \mathcal{N}(\mu, \Sigma)\), where \(\mu\) is the \(D\)-dimensional mean vector, and \(\Sigma\) is the \(D \times D\) covariance matrix. Without loss of generality, we can partition the \(D\)-dimensional vector \(x\) as \(X = (X_1, X_2)^T\). To compute the distribution of \(X_1\) conditional on \(X_2 = x_2^*\) where \(x_2^*\) is a \(K\)-dimensional vector with \(0 < K < D\), we can then partition \(\mu\) and \(\Sigma\) accordingly:

\[
\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}.
\]

Then the conditional distribution is a new multivariate normal \((X_1 | X_2 = x_2^*) \sim \mathcal{N}(\mu^*, \Sigma^*)\) where

\[
\mu^* = \mu_1 + \Sigma_{12} \Sigma_{22}^{-1}(x_2^* - \mu_2), \quad \Sigma^* = \Sigma_{11} + \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}.
\]

For any \(x_2^* \in \mathbb{R}^K\), one can compute \(\mu^*\) and \(\Sigma^*\) and then generate samples from the conditional distribution. Parameter \(\mu\) can take any value, and \(\Sigma\) must be symmetric and positive definite. Similarly, we also give the derivation for additional distribution families in Appendix 5 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. \(y_{raw} = \sum_{n=1}^{D} \Psi_n(x_n)\) where \(\Psi_n\) 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 \(y_{raw} = w^T x\). In our experiments in Section 5 we set \(w = [0, 1, ..., d - 1]\). 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 \(w = [0, 1, ..., d - 1]\) when the sum of the feature values is positive, and \(w = [d - 1, d - 2, ..., 0]\) otherwise. For piecewise constant datasets, \(\Psi_n(x_n)\) are piecewise constant functions made up of different threshold values (similar to Aas et al. 41). For nonlinear additive datasets, \(\Psi_n(x_n)\) are nonlinear functions including absolute, cosine, and exponent function adapted from Chen et al. 14. Detailed specifications can be found in Appendix 6.
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, \( \mu \) is set to 0, and the diagonal of \( \Sigma \) is set to 1). In all sections except Section 5.3, we set the non-diagonal terms of \( \Sigma \) to \( \rho \), 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 5.1 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 \( \rho = 0.5 \).

<table>
<thead>
<tr>
<th>Explainer</th>
<th>RANDOM</th>
<th>SHAP</th>
<th>SHAPR</th>
<th>LIME</th>
<th>MAPLE</th>
<th>L2X</th>
<th>BREAKDOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faithfulness()</td>
<td>0.002±0.034</td>
<td>0.694±0.051</td>
<td><strong>0.799±0.006</strong></td>
<td>0.524±0.061</td>
<td>0.478±0.061</td>
<td>0.000±0.075</td>
<td>0.110±0.049</td>
</tr>
<tr>
<td>Monotonicity()</td>
<td>0.525±0.017</td>
<td>0.547±0.014</td>
<td>0.509±0.026</td>
<td>0.517±0.022</td>
<td>0.543±0.026</td>
<td>0.535±0.022</td>
<td><strong>0.562±0.021</strong></td>
</tr>
<tr>
<td>ROAR(↑)</td>
<td>0.380±0.051</td>
<td>0.455±0.054</td>
<td><strong>0.465±0.004</strong></td>
<td>0.322±0.051</td>
<td>0.432±0.059</td>
<td>0.365±0.053</td>
<td>0.328±0.057</td>
</tr>
<tr>
<td>GT-Shapley(↑)</td>
<td>0.004±0.049</td>
<td>0.810±0.023</td>
<td><strong>0.930±0.012</strong></td>
<td>0.711±0.032</td>
<td>0.530±0.128</td>
<td>-0.014±0.068</td>
<td>-0.127±0.066</td>
</tr>
<tr>
<td>Infidelity(↓)</td>
<td>0.114±0.058</td>
<td>0.050±0.025</td>
<td>0.036±0.013</td>
<td>0.053±0.016</td>
<td><strong>0.019±0.011</strong></td>
<td>0.025±0.016</td>
<td>0.129±0.057</td>
</tr>
</tbody>
</table>

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 \( \rho \). 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 ($x_{\text{real}}$) and one integer quality rating ($y_{\text{real}}$) 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 ($x_{\text{sim}}$). Simulated wine quality ($y_{\text{sim}}$) is labeled by a $k$-nearest neighbor model based on real datapoints ($x_{\text{real}}, y_{\text{real}}$).

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.

<table>
<thead>
<tr>
<th></th>
<th>RANDOM</th>
<th>SHAP</th>
<th>SHAPR</th>
<th>LIME</th>
<th>MAPLE</th>
<th>L2X</th>
<th>BREAKDOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>faithfulness (↑)</td>
<td>-0.001</td>
<td>0.554</td>
<td>0.533</td>
<td>0.529</td>
<td>0.365</td>
<td>0.034</td>
<td>-0.030</td>
</tr>
<tr>
<td>monotonicity (↑)</td>
<td>0.529</td>
<td>0.549</td>
<td>0.551</td>
<td>0.547</td>
<td>0.520</td>
<td>0.522</td>
<td>0.493</td>
</tr>
<tr>
<td>ROAR (↑)</td>
<td>0.004</td>
<td>0.825</td>
<td>0.945</td>
<td>0.745</td>
<td>0.689</td>
<td>0.108</td>
<td>0.064</td>
</tr>
<tr>
<td>GT-Shapley (↑)</td>
<td>0.353</td>
<td>0.234</td>
<td>0.212</td>
<td>0.234</td>
<td>0.234</td>
<td>0.285</td>
<td>0.365</td>
</tr>
<tr>
<td>infidelity (↓)</td>
<td>0.075</td>
<td>0.077</td>
<td>0.077</td>
<td>0.077</td>
<td>0.090</td>
<td>0.008</td>
<td>0.002</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>RANDOM</th>
<th>SHAP</th>
<th>SHAPR</th>
<th>LIME</th>
<th>MAPLE</th>
<th>L2X</th>
<th>BREAKDOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>faithfulness (↑)</td>
<td>0.022</td>
<td>0.571</td>
<td>0.479</td>
<td>0.499</td>
<td>0.481</td>
<td>0.481</td>
<td>0.158</td>
</tr>
<tr>
<td>monotonicity (↑)</td>
<td>0.537</td>
<td>0.591</td>
<td>0.598</td>
<td>0.561</td>
<td>0.527</td>
<td>0.575</td>
<td></td>
</tr>
<tr>
<td>ROAR (↑)</td>
<td>0.575</td>
<td>0.615</td>
<td>0.616</td>
<td>0.696</td>
<td>0.534</td>
<td>0.604</td>
<td></td>
</tr>
<tr>
<td>GT-Shapley (↑)</td>
<td>0.012</td>
<td>0.870</td>
<td>0.779</td>
<td>0.804</td>
<td>0.031</td>
<td>0.105</td>
<td></td>
</tr>
<tr>
<td>infidelity (↓)</td>
<td>0.207</td>
<td>0.075</td>
<td>0.077</td>
<td>0.077</td>
<td>0.091</td>
<td>0.117</td>
<td></td>
</tr>
</tbody>
</table>

5.4 **Recommended usage**

In Section 5.3, we gave a sample of the types of experiments that can be performed with our library (recall that comprehensive experiments are in Appendix A). For researchers looking to develop new explainability techniques, we recommend benchmarking new algorithms across all metrics using our synthetic datasets with different values of \( \rho \). 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


