Appendix

This appendix provides additional explanations of the evaluation settings and visualizations of the results from Section V. Appendix A provides insights into the generation process of the synthetic datasets, Appendix B elucidates the used explainers, Appendix C explains tweaks to counterfactual explanation to use the benchmarking tool and Appendix D - Appendix F visualize additional results.

A. Data Generation

The synthetic datasets described in Section IV-A were generated as described in Ismail et al. [19][9]. XTS - Bench provides the generated data for 50 time steps with a feature size of 1 and 50. The data is generated based on 6 time processes with $\epsilon_t \sim N(0, 1)$:

- Gaussian ($\mu = 0, \sigma = 0$):
  $X_t = \epsilon_t$
- Harmonic:
  $X(t) = sin(2\pi 2t) + e_t$
- Pseudo Periodic ($A_t \sim N(0, 0.5)$, $f_t \sim N(2, 0.01)$):
  $X(t) = A_t sin(2\pi f_t, t) + \epsilon_t$
- Autoregressive ($p = 1$, $\varphi = 0.9$):
  $X_t = \sum_{i=1}^{p} \varphi X_{t-i} + \epsilon_t$
- Continuous Autoregressive ($\varphi = 0.9$, $\sigma = 0.1$) :
  $X_t = \varphi X_{t-1} + \sigma(1 - \varphi)^2 * \epsilon + \epsilon_t$
- NARMA ($n = 10$, $U \sim U(0.05)$):
  $X_t = 0.3X_{t-1} + 0.05X_{t-1} \sum_{i=0}^{n-1} X_{t-1} + 1.5U(t-(n-1)) * U(t) + 0.1 + \epsilon_t$

The obtained datasets highlight predefined informative features by adding a constant to the positive class or subtracting a constant for negative classes. As visualized in Figure 3, the informative features can take various forms to replicate different ground truths

- **Middle vs. Moving vs. Positional**: denotes the location of the informative features.
- **Small vs. Normal vs. Rare Time / Feature**: refers to the size (number) of informative features. For Normal, more than 35% of all features are informative. For Small, less than 10% of all features are informative. A time or feature is rare if less than 5% of all features are informative.

Overall we obtain 60 univariate datasets and 60 multivariate datasets (6 time process × 10 informative features).

Practical Note: The function evaluate_synthetic allows filtering the synthetic datasets by providing the variable *types*, enabling the evaluation of designated informative features and time series processes. For example, providing $types = ['Rare']$ would allow the evaluation of explainers for anomaly detection.

B. Explanation Approaches

According to the taxonomy provided by Höllig et al. [18], we divided the explanation approaches into gradient-based and perturbation-based feature attribution methods and example-based approaches. Gradient-based feature attribution methods assign a relevance score to a machine learning model's inputs based on the classifier model's gradients. Perturbation-based feature attribution methods also assign relevance scores, however, they obtain the relevance scores by observing the classifier's output while masking parts of the input. In contrast to feature attribution methods, example-based methods return a manipulated version of the input instance x, e.g., to show how a counterexample looks. We evaluate all the explainers applicable to uni- and multivariate time series implemented in TSInterpret [18].

- **TSR**: Temporal Saliency Rescaling (TSR) is a wrapper for well-known perturbation (e.g., Feature Occlusion (FO) [42]) and gradient-based Feature Attribution Methods (e.g., Gradient Shap (GS) [22], Integrated Gradient (IG) [38], Saliency (GRAD) [36], Smooth Gradients (SG) [37]) developed by Ismail et al. [19]. TSR is applied after the explanation calculation and decouples the time and feature domain by computing time and feature relevance scores.
- **LEFTIST**: LEFTIST [15] adapts SHAP [22] and LIME to time series. An interpretable model is fitted locally by perturbing the input x meaningfully by segmenting the original time series into interpretable components and perturbing those components with a) linear interpolation, b) a constant, or c) a background obtained from a reference set. In this work, we make use of variant c) and segmentation of size 10.
- **TSEvo**: TSEvo [17] generated counterfactuals for uni- and multivariate time series using time series specific perturbation functions (e.g., perturbing the frequency domain). We use the authentic information transformer and run TSEvo for 100 epochs.
- **NG**: Delaney et al. [10] propose using a Native Guide (i.e., an existing instance in the data that is the nearest unlike neighbor to the original instance) to generate counterfactuals. The original time series is thereby manipulated with the Native Guide by replacing the most important features of x (obtained with, e.g., GradCam [32]) with the Native Guide.

C. Feature Ranking and Mass Calculation for example-based Explainers

Replacing features to calculate the robustness (Section IV-B), reliability (Section IV-E), or faithfulness metrics (Section IV-C) relies on either ranking the most important features or calculating relevance masks. For feature attribution methods, this is straightforward, as feature attribution methods return relevance scores. However, example-based methods return a manipulated version of the original inputs. The changes made to the original input can usually not be directly interpreted as relevance scores. To be able to still use the metrics with example-based methods, we calculate the fraction of change $\Delta x = \frac{(x-E_f(x))}{x}$.

Practical Note: The synthetic data is normalized to zero and one. For non-synthetic data, a feature range needs to be provided.

[9] https://github.com/ayaabdelsalam91/TS-Interpretability-Benchmark

D. Results split on Informative Features Types

Figure 5 and Figure 6 show the results averaged over all time series processes split on the explainer and the informative feature type. Due to the availability of only one feature, all feature-based datasets are missing for univariate data.

On uni- and multivariate data across the different informative feature types, the explainers perform similarly on robustness, faithfulness and complexity. On reliability, the informative feature type has on univariate time series a huge impact on the performance of all explainers. The reliability on univariate and multivariate data is the largest for all explainers on the informative feature 'Middle' (over 30% of all time steps and features are informative) and decreases with the number of informative features ('Middle' → 'SmallMiddle' → 'Rare').

E. Results split on Classifier Models

Figure 7 shows the complexity, reliability, robustness and faithfulness averaged over all datasets and split on the classification model to be explained. If no box for a model is provided, either the model's accuracy was below 90% or the explainer was not applicable to the classifier.

F. Faithfulness: Comparison of Baselines

In Figure 8, the faithfulness metric with the generation baseline used for the synthetic data in comparison to the 'traditionally' used baselines mean and uniform. As the uniform baseline performs, on average, similar to the known generation process baseline, we advice users with non-synthetic data to make use of the uniform baseline.

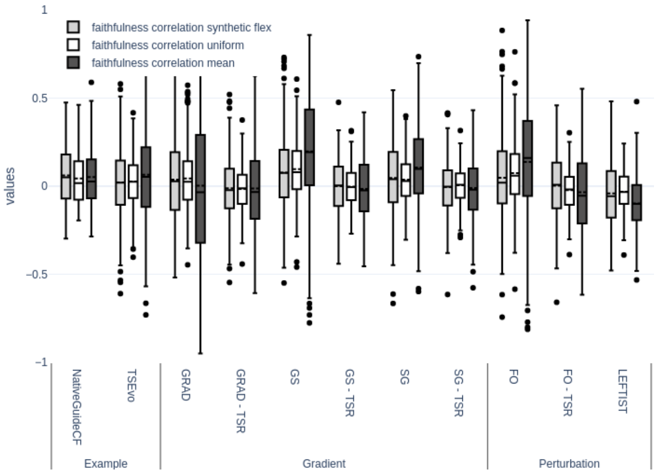
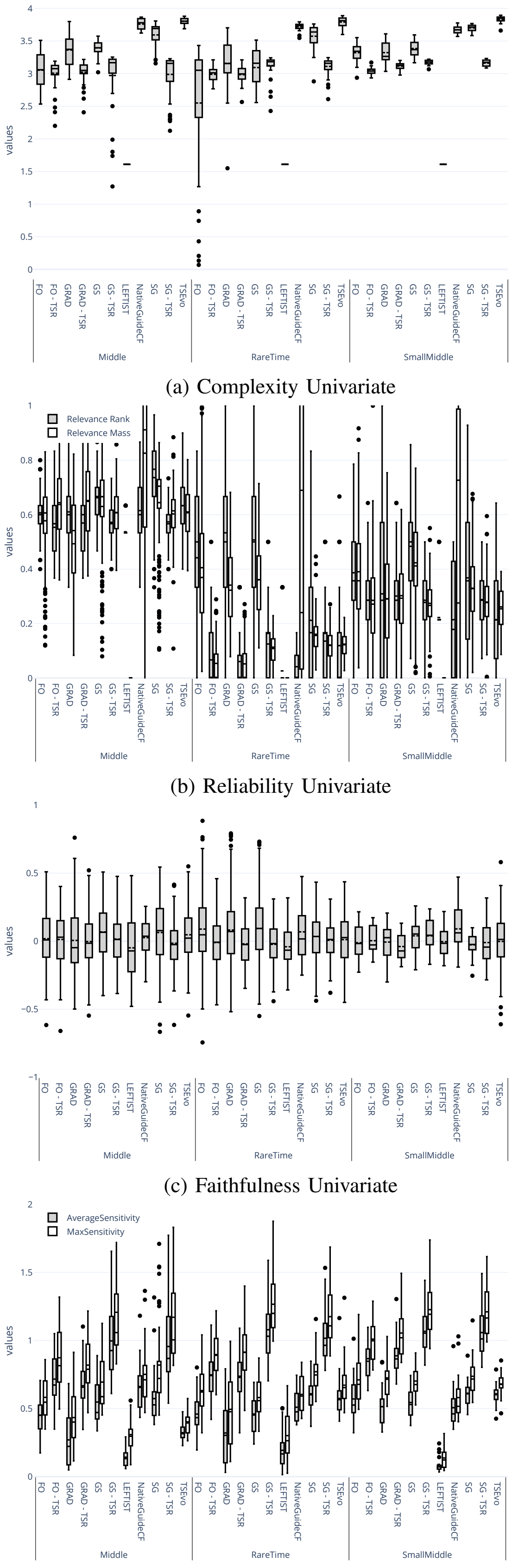


Fig. 8: Comparison of baselines used in the calculation of the faithfulness metric.



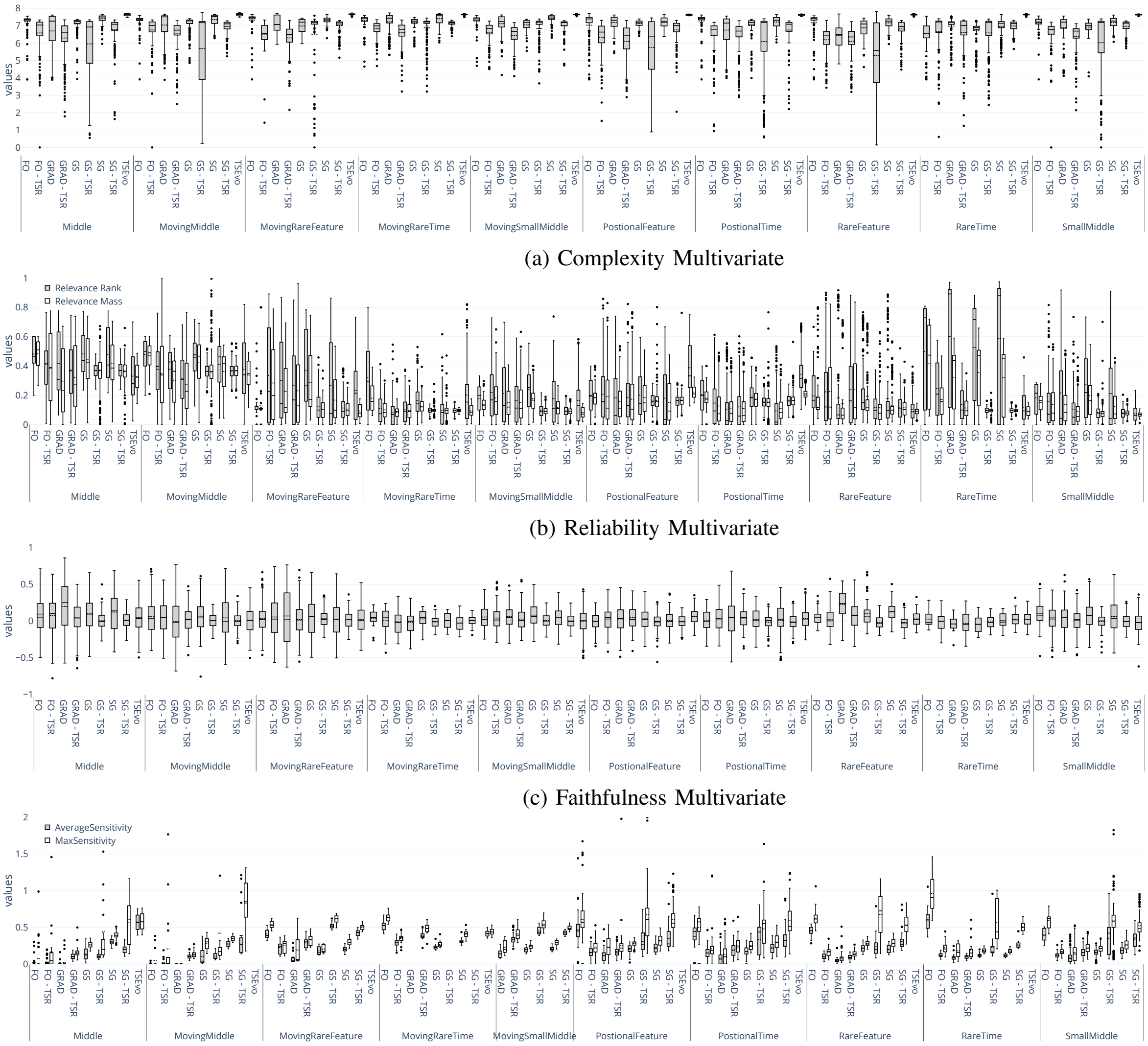
(a) Complexity Univariate

(b) Reliability Univariate

(c) Faithfulness Univariate

(d) Robustness Univariate

Fig. 5: Informative-feature-wise explainer performance on complexity, reliability, faithfulness, and robustness averaged over all generation processes.



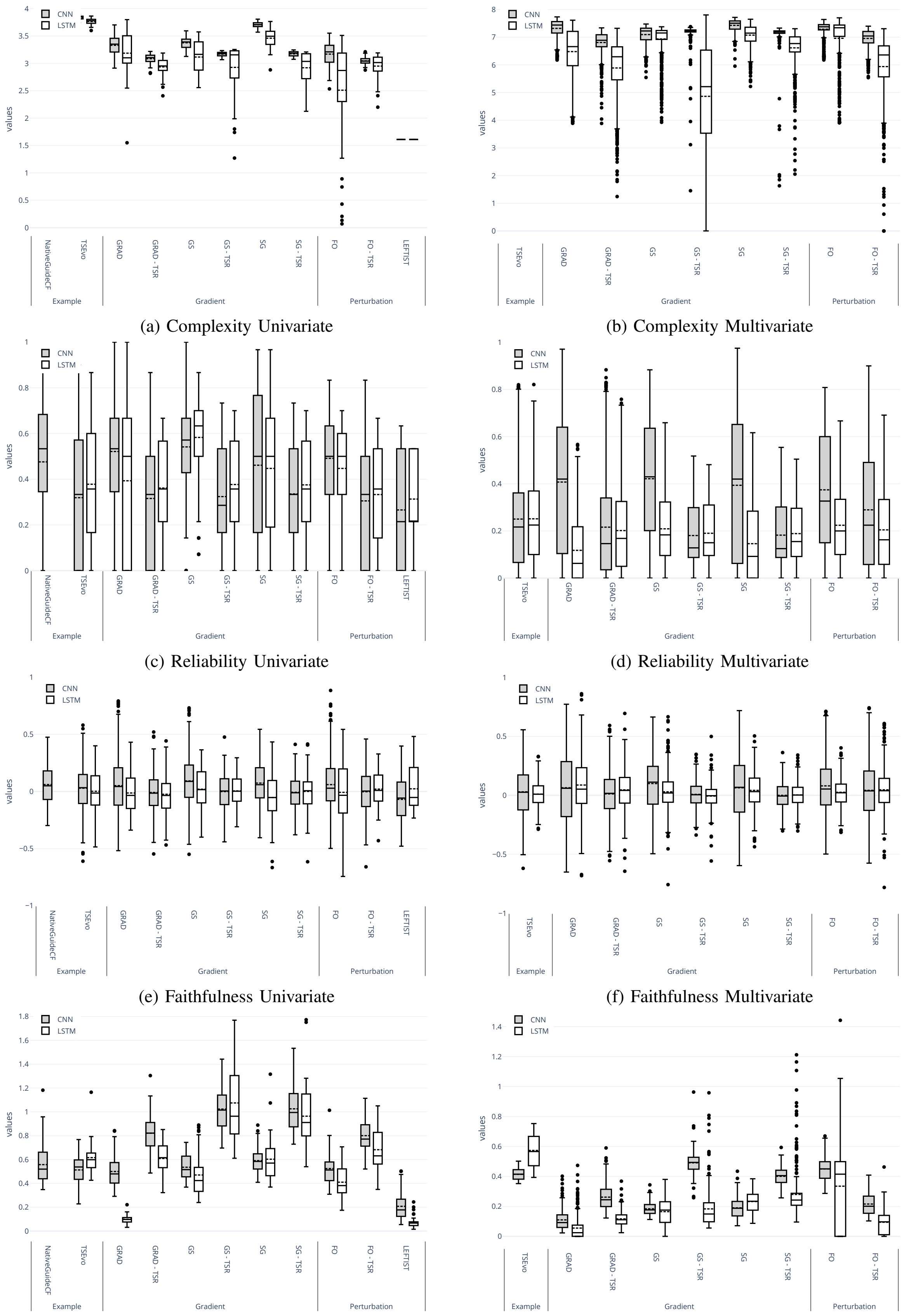
(a) Complexity Multivariate

(b) Reliability Multivariate

(c) Faithfulness Multivariate

(d) Robustness Multivariate

Fig. 6: Informative-feature-wise explainer performance on complexity, reliability, faithfulness, and robustness averaged over all generation processes.



(a) Complexity Univariate

(b) Complexity Multivariate

(c) Reliability Univariate

(d) Reliability Multivariate

(e) Faithfulness Univariate

(f) Faithfulness Multivariate

(g) Robustness Univariate

(h) Robustness Multivariate

Fig. 7: Explainer Performance on complexity, reliability, faithfulness, and robustness averaged over all datasets and split on the used classifier.