Methods for Enhancing Industrial Applications with Explainable Artificial Intelligence

Zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

von der KIT-Fakultät für
Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation
von

M.Sc. Jacqueline Höllig

Tag der mündlichen Prüfung:	06.02.2025
Referent:	Prof. Dr. York Sure-Vetter
Korreferent:	Prof. Dr. Achim Rettinger

Karlsruhe, 2025

Abstract

With the growing adoption of Artificial Intelligence (AI) and the successful application of deep learning methods in various domains, AI, particularly deep learning, is increasingly influencing people's lives. Despite the continuous performance advancements and success of AI models, many manufacturers still hesitate to deploy AI on a larger scale, as failures due to a black-box AI model in production can cause significant financial damage and potentially harm workers. The emerging topic of Explainable Artificial Intelligence (XAI) offers approaches and algorithms that introduce transparency into black-box models by producing explanations for an AI System's inner workings and decisions. Specifically, in industrial settings, where complex problems and decision-making processes are widespread, enabling transparent automation and decision support is crucial. However, while research in XAI is trending, applying XAI in industrial settings is challenging. For many data types (e.g., images or tabular data), XAI methods are well-studied. Nevertheless, support for time series, ubiquitous in industrial settings, is missing. Furthermore, to use any XAI method in deployment, understanding the explainers' quality, strengths, and weaknesses is vital to prevent ambiguous and incorrect explanations. Well-performing XAI methods can help users understand the reasons behind a deep learner's prediction and enable the recognition of spurious correlations learned by a deep learner or missing information in the collected data. However, reverting incorrect predictions and learned patterns is not in the scope of XAI. Nevertheless, reverting such is especially important in industrial settings, where only a limited amount of (often) noisy data is available. Explanations and explanation correction can potentially provide the opportunity to include domain knowledge of users and enable a deep learner to infer the missing context and close this gap.

The main contributions of this thesis are to facilitate and enable a more widespread use of XAI in industrial settings by providing methods, frameworks, and in-depth evaluations addressing the application obstacles.

Individual contributions of this thesis are summarized as follows:

1. An extensible framework enabling unified access to time series explainers, including a new counterfactual explainer considering various time series transformation mechanisms and thereby enabling plausible explanations on uni- and multivariate time series.
2. A framework and in-depth evaluation of time series explainers, a metric measuring causal coherence of counterfactual approaches and corresponding benchmark datasets.
3. A methodology combining continuous, interactive, and explanatory interactive machine learning to enable human supervised transfer learning in industrial contexts.

The contributions are evaluated in accordance with predefined requirements and to state-of-the-art setups. The outcome (i) allows facilitated access to time series explainers and the generation of more plausible counterfactual explanations, helping to bridge the gap between complex models and contrastive explanations; (ii) indicates the need for additional research specifically for multivariate time series explainers and the generation of causal coherent counterfactual explanation; and (iii) shows that explanation based *AI*-expert alignment enables continuous learning under human supervision.

Publications

Parts of this thesis have been published in the following publications:

Jacqueline Höllig et al. “XTSC-Bench: Quantitative Benchmarking for Explainers on Time Series Classification”. In: *2023 22st IEEE International Conference on Machine Learning and Applications (ICMLA)*. Jacksonville, Florida, USA: IEEE, 2023, pp. 1126–1131. DOI: `10.1109/ICMLA58977.2023.00168` [97]

Jacqueline Höllig et al. “Semantic Meaningfulness: Evaluating Counterfactual Approaches for Real-World Plausibility and Feasibility”. In: *Explainable Artificial Intelligence*. Ed. by Luca Longo. Vol. 1902. Series Title: Communications in Computer and Information Science. Cham: Springer Nature Switzerland, 2023, pp. 636–659. ISBN: 978-3-031-44066-3 978-3-031-44067-0. DOI: `10.1007/978-3-031-44067-0_32` [96]

Jacqueline Höllig et al. “TSInterpret: A Python Package for the Interpretability of Time Series Classification”. In: *Journal of Open Source Software* 8.85 [2023], p. 5220. ISSN: 2475-9066. DOI: `10.21105/joss.05220` [94]

Jacqueline Höllig et al. “TSEvo: Evolutionary Counterfactual Explanations for Time Series Classification”. In: *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*. Nassau, Bahamas: IEEE, 2022, pp. 29–36. ISBN: 978-1-66546-283-9. DOI: `10.1109/ICMLA55696.2022.00013` [93]

Jacqueline Höllig et al. *TSInterpret: A unified framework for time series interpretability*. arXiv:2208.05280. 2022 [95]

Jacqueline Höllig et al. “Evolutionary Counterfactual Visual Explanation”. In: *KI (Workshops)*. Vol. 3457. Trier, Germany: CEUR Workshop Proceedings, 2022 [98]

Contents

Part I

Foundations

This part establishes the foundations of the thesis. We begin by introducing the topic of Explainable Artificial Intelligence in Industrial Settings (Chapter 1) including its motivation, a problem statement, and the research questions and contributions that arise from it. After the introduction, we define the preliminaries needed for the subsequent content (Chapter 2). Finally, we discuss related works concerning the contributions of this thesis (Chapter 3).

1 Introduction

Given the topic "Methods for Enhancing Industrial Applications with Explainable Artificial Intelligence", we start this thesis by motivating the need for *EXplainable Artificial Intelligence* (*XAI*) in industrial settings in Section 1.1. In Section 1.2, we present challenges to the applicability of *XAI* in industrial settings. We then introduce our hypotheses and research questions in Section 1.3. In Section 1.4 and Section 1.5, we frame this work's scope and contributions by pointing out which aspects we focus on and which we neglect. In the last section of this introduction, Section 1.6, we give an overview of all parts and chapters of this thesis.

1.1 Motivation

Artificial Intelligence (*AI*), particularly its subcategories *Machine Learning* (*ML*) and *Deep Learning* (*DL*), offers unprecedented opportunities for advancement, automation, and innovation due to its ability to tackle problems traditionally requiring human intelligence. The successful implementation of *DL* techniques across various fields, from voice recognition and recommendation systems to self-driving cars, already significantly impacts our daily lives. However, depending on the use case, wrong decisions can be costly, dangerous, or have a significant social impact (e.g., pre-trial bail [62], credit risk assessment [110], health care [144, 109]). With the increasing reliance on *AI* in such critical domains, understanding how these systems make decisions becomes crucial. However, while *AI* methods, especially *DL*, are high-performing, they lack transparency to provide such understanding due to their complexity. They are so-called black-boxes.

> ➢ **Definition 1** (Black-Box Model)**.** A Black-Box Model is a system that does not reveal its internal mechanisms, i.e., a model that cannot be understood by looking at its parameters.

The emerging topic of *EXplainable Artificial Intelligence* (*XAI*) offers tools, methods, and algorithms for producing explanations of AI-based systems decisions and inner workings. Research in *XAI* is at an all-time high (see Figure 1.1), especially since the 2021 proposed and in July 2024 adopted and announced AI Act[1], an EU regulation for developing and using AI. In particular, high-risk AI systems are now subject to a set of requirements that include, among others, transparency and human oversight [171].

AI models are increasingly popular in industrial applications driven by the rise of Industry 4.0 and the accompanying advancements in digitalization, automation, and data availability [76]. However, the need for transparency and human oversight often hinders the widespread adoption of *AI* models (e.g., [254, 6, 8]). Failures due to black-box *DL* models in production can cause significant financial damage and potentially harm to workers [76]. Therefore, using *XAI* to explain *DL* decisions is crucial to enable a more widespread

[1] `https://www.europarl.europa.eu/doceo/document/TA-9-2024-0138_DE.pdf`, accessed 15.June.2024

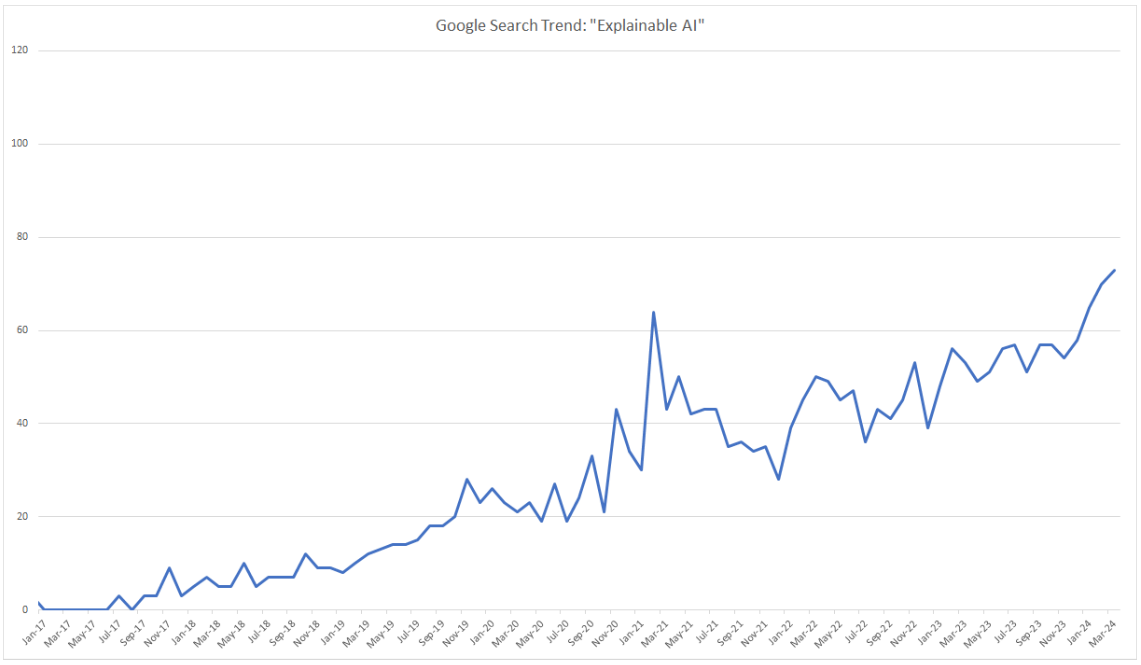


Figure 1.1: The search trend for Explainable Artificial Intelligence between 2017 and 2024. The spike in June 2023 coincides with the adoption of the draft of the Artificial Intelligence Act by the EU Parliament.

adoption of *DL* models in safety-critical aspects of manufacturing, such as part inspection and qualification [8].

Due to the heterogeneous nature of industrial systems ranging from quality control or production line monitoring to predictive maintenance, *XAI* systems need to address a variety of data types (e.g., tabular data from Product Information Management Systems, image data showing the good to be manufactured or wear, sensor data as time series giving implications on a machine state) and depending on the data type and task various architectures (e.g., *LSTM* for time series, *CNN* for images). While methods and tools are available for many data types (e.g., tabular and image data) and architectures (e.g., *CNN*s), methods for time series, which are ubiquitous in production scenarios, are still neglected.

Further, many methods include *XAI* in the model design or learning process (ante-hoc explainability), allowing for transparent and explainable models by design (e.g., Decision Trees). While those methods are transparent by definition, that transparency is frequently traded for task performance. In industrial settings, the relationships between variables and processes are often complex, requiring more powerful and accurate models to capture these complexities. As a result, models tend to be intransparent and more difficult to understand. In such cases, post-hoc *XAI* applied only after training a *DL* model becomes advantageous as no "Accuracy-Interpretability Trade-off"[2] has to be taken into account, and decision support systems can be designed with the goal of high task performance without paying attention to transparency. This allows industrial applications to benefit from the full potential of advanced AI models while still ensuring insights into the decision-making processes when needed. However, post-hoc *XAI*-methods are often criticized as another black-box heuristic above a black-box model (e.g., [171, 128]), creating a need for a thorough evaluation [171]. Specifically, evaluating the coverage of an explanation concerning the black-box [32] is essential to ensure that the behavior of the *XAI* method is consistent with the *DL* model.

[2] For details on the "Accuracy-Interpretability Trade-off" refer to [128].

Well-performing *AI* models with *XAI* enable domain experts to understand the reasons behind a model's prediction and enable the detection of model errors as well as missing information in the collected data. However, *XAI* alone does not allow for corrections. Especially in industrial settings where data is often noisy, reverting incorrect model decisions and explanations provides the opportunity to include the domain knowledge of workers.

1.2 Problem Statement and Challenges

Many industrial problems, from predictive maintenance to manufacturing decision support, can be formulated as a supervised classification problem $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the domain of the data and $\mathcal{Y}$ the target domain.[3] Using a dataset $\mathcal{D} = (X, Y)$ we want to approximate the parameters θ of a deep learning based decision support system f. Thereby, $x \in X$ denotes a single input instance from X and $y \in Y$ a single categorical output. The inference of model f for a new instance x can be described as $\hat{y} = f(x)$. The deep learning model f is assumed to be a black-box due to its large number of parameters. To shed light into the workings of f, we assume the availability of an Explainer E_f approximating and visualizing (parts of) the behavior of f to unveil parts of the decision-making process of f. Thereby, E_f provides an explanation $e = E_f(x)$ based on the classifier f, the input instance x, and the predicted classification $\hat{y}$. This problem setting is depicted in the conceptual framework visualized in Figure 1.2. The model f can be any classifier, and explainer E any post-hoc explainer compatible with the model f. To enable the usage of *XAI* in industrial settings and, therefore, the deployment of thr conceptual framework, the following observations and challenges are crucial:

Challenge 1 Industrial applications cover a wide range of data types: images representing a product to be manufactured or a tool used during the manufacturing process, text representing error messages from production machines, tabular data presenting technical parameters, and finally, time series from sensors. However, because of the properties of time series, existing and easy-to-access *XAI* methods are not usable (see Section 3.2). For one, time series usually do not fulfill the independent feature assumption. Hence, many tabular methods can not be directly applied to time series data. Further, the *XAI* for vision tasks assumes a dependency between the time and feature domain, which is not necessarily a given. Thus, to increase the usability of *XAI* in industrial settings, the **missing support and accessibility for time series data** needs to be addressed.

Challenge 2 In industrial settings, specifically, the use of post-hoc *XAI* is of interest due to the need for high-performing decision support systems, complex data relationships, and various data types. Post-hoc explainers approximate the behavior of a classifier f with a heuristic built around f that should highlight the significant behaviors of f. Therefore, f and E_f should show similar behavior. To ensure a correct approximation of f, **the quality of an explainer** needs to be quantified to ensure robust (and ideally realistic) explanations coherent with the underlying classifier behavior.

Challenge 3 Challenge 1 and Challenge 2 address the need for (post-hoc) explainers on the most common data types in industrial use cases and quantify their suitability. However, shown explanations might be still incorrect, due to incorrect patterns learned from the data or missing domain knowledge. The often available domain experts are able to detect such patterns with the help of *XAI* and usually has the expertise to correct them. While *XAI* allows the detection of incorrect patterns and decisions, it does not include their corrections. Therefore, creating a need for **model revision** by interacting with the model f based on the prediction $\hat{y}$ and the explanation e.

[3] For detailed use cases, we refer the reader to surveys for deep learning application in industrial setting and XAI. [8, 32, 113]

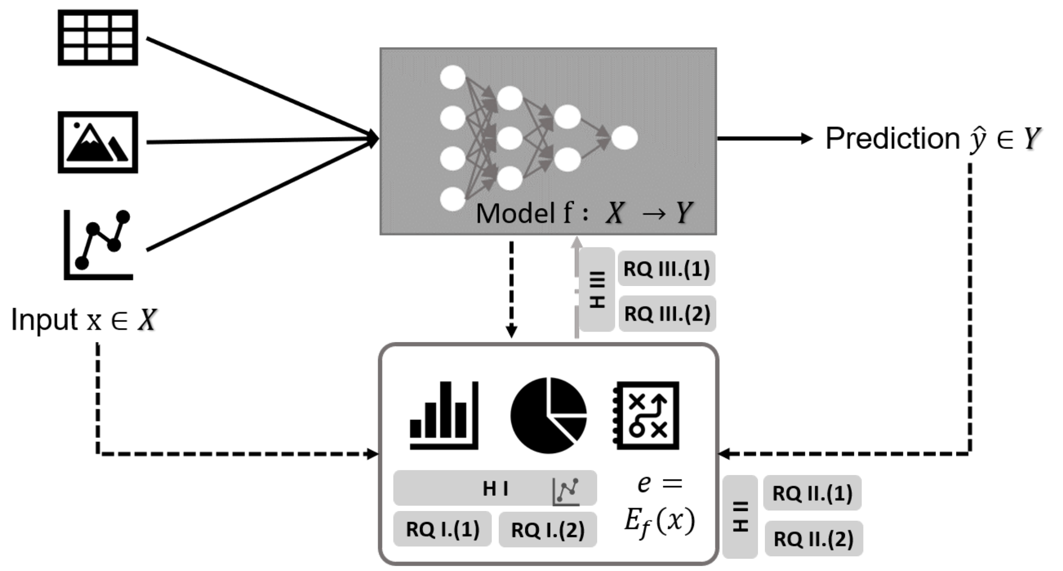


Figure 1.2: A conceptual framework for enabling (post-hoc) *XAI* in industrial settings, highlighting the problem setting, research problems, and research question location. The first hypothesis deals with the explainer component. While post-hoc explainers are readily available for various data types and use cases, this work focuses on the lack of (post-hoc) explainers in the time series domain and increasing their availability to ensure facilitated applicability like in the other data domains. The second hypothesis aims to quantify the quality of the explainer component to ensuring that its behavior aligns with the dynamics of the black-box model while also adhering to predefined explainer properties. Finally, the third research question explores how (well-performing) explainers can be leveraged to interact with black-box models in a continuous learning environment.

1.3 Hypotheses and Research Questions

Our research aims to provide novel methods that allow the application of *XAI* in industrial settings. This leads us to the following principal research question:

How can we facilitate and improve *XAI* to enable *XAI* usage in industrial settings?

Based on the conceptual framework shown in Figure 1.2 and the challenges described in the previous chapter (Section 1.2), we formulate our hypotheses and respective research questions for improving the applicability and potential of *XAI*. The research questions' locations are highlighted in the conceptual framework in Figure 1.2. We begin by dealing with Challenge 1, the missing support of *XAI* for time series, in Hypothesis I.

Hypothesis I (Methods)
The adoption of XAI *in industrial settings can be facilitated by providing time series explainers based on time series transformation mechanisms and an easier access to such explainers.*

Addressing Hypotheses I, we define the following research questions:

RQ I. How can we facilitate and enhance *XAI* on time series?

RQ I.1. How can the cross-domain application of explanation methods for deep learning based time series classification be facilitated ?

RQ I.2. How can we enhance counterfactual time series explainers by including time series based transformers?

We approach Hypothesis I with RQ I. from two perspectives. First, we focus on the missing democratization of *XAI* on time series in RQ I.(1), implying a need for a unified framework. The second research question, RQ I.(2), focuses on enhancing counterfactual explainers on time series by including more time series related transformation mechanisms.

After dealing with the missing support for *XAI* on time series data, we take a closer look at Challenge 2, the quantification of *XAI* to enable performance comparisons across different methods and properties in Hypothesis II.

Hypothesis II (Evaluation)
To ensure the correctness of explanations for further use, the quality of explanation algorithms needs to be quantifiable.

Hypothesis II tackles the following research questions:

RQ II. How can we quantify *XAI* performance on time series and industrial use cases?

RQ II.1. How can we quantify the quality of time series explainers for an easier benchmarking?

RQ II.2. How can we evaluate counterfactual explainers for real-world coherence?

Similar to RQ I.(1), RQ II.(1) explores the need for standardizing *XAI* benchmarking for time series data and the challenges of applying benchmarking metrics in this domain. Secondly, RQ II.(2) addresses the need for a new metric measuring whether the proposed explanation methods adhere to real-world relations.

Despite quantitatively well-performing *XAI* methods visualizing *DL* decisions and detecting errors, *XAI* alone cannot revise model errors ad hoc. Hypothesis III aims to enable such model corrections under the assumption of evaluated and well-performing *XAI* methods.

Hypothesis III (Revision)
By combining Explanatory Interactive Machine Learning with Continuous Learning, we can enable continuous human supervision and boost model performance.

Finally, Hypothesis III tackles the following research questions:

RQ III. How can we enable continuous, explanatory, and interactive model improvement?

RQ III.1. Which combinations of continuous, explanatory, and interactive are the most suitable?

RQ III.2. How does continuous, explanatory, and interactive model improvement perform on realistic data?

RQ III.(1) investigates which interactive and continuous learners, and combinations thereof, are suitable to be included in such a framework to tackle continuous learning with human feedback for learning new data distributions, new tasks, or simply fine-tuning *AI* models with additional domain knowledge. RQ III.(2) how the results can be transferred to a real data use case.

1.4 Contributions

Figure 1.2 shows the superordinating framework for enabling (post-hoc) explainability in industrial settings. In correspondence to the challenges (Section 1.2) currently faced in deploying such a framework, we contribute by filling the research gap along the research questions identified and detailed in Section 1.3 in three parts: Methods, Evaluation, and Revision.

Contributions to Hypothesis 1: Methods We present a novel framework that facilitates the access to state-of-the-art *XAI* methods on time series data. It consists of 6 explainers providing a variety of explanation types (see Chapter 4). Additionally, we introduce a new method for instance-based explanations that takes additional time series properties into account for more plausible explanations (see Chapter 5).

Contributions to Hypothesis 2: Evaluation Based on the developed framework for *XAI* on time series data, we developed a benchmarking framework including unified datasets and pretrained models to enable comparable evaluation of time series explainers. Further, as many specifically instance-based methods claim causal adherence, we introduce the first metric quantifying causal coherence of instance-based methods and evaluating state-of-the-art explainers, with the result that only explainers based on causal inference are able to capture relationships reliably.

Contributions to Hypothesis 3: Revision To achieve ad hoc model revisions based on *XAI*, we propose a framework combining continuous learning with explanatory interactive learning. By analyzing the algorithm combinations for their fine-tuning, class incremental, and domain incremental learning capabilities, we found that the replay strategy works best for all feedback algorithm, but the optimal feedback algorithm depends on the dataset.

Software Artifacts The conceptual contributions have been implemented as software artifacts. To demonstrate the feasibility, applicability, and reusability of our methods and frameworks and increase the democatization of *XAI*, we provide PyPi-ready software toolings and detailed documentations.

1.5 Scope of the Thesis

The most considerable restrictions apply to the prediction task, the type of explainer used, and the exclusion of human factors, both in the evaluation and the interaction component. The thesis focuses on classification tasks solved in a supervised machine-learning setting. We exclude unsupervised machine learning methods from our scope. While industrial use cases for unsupervised learning include outlier detection, the majority of unsupervised learning methods are used in industrial settings to preprocess data for the following downstream task[4]. Further, we do not take into account regression or forecasting tasks. Specifically, in the case of time series forecasting, statistical models often perform on par or better than deep learning models, eliminating the need for *XAI*. In the past decade, empirical studies have demonstrated that simple methods are as accurate as complex or statistically sophisticated methods. However, deep learning based forecasting models are improving. In the most recent M5 competition [142], a competition aiming to identify ways to improve the forecasting accuracy on real-life data, usually dominated by tree-based forecasting methods, deep forecasting methods were the second and third-placed solutions [141]. Regression problems are often more complex compared to classification due to the continuous and thereby much larger target space. While similar deep learning architectures are used for classification and regression (except for the last layer), we

[4] A survey from Bertolini et al. [27] investigates the application of machine learning in industrial settings. Only 25 out of the 144 application domain papers used unsupervised machine learning, but 112 supervised machine learning. Of the 25 unsupervised applications, 15 used unsupervised learning to preprocess a downstream task.

evade the complexity of regression tasks by focusing on classification. However, due to the similarity of base-architectures, many methods translate to regression.

In the first part of this thesis, we deal with *XAI* methods in industrial settings. In industrial settings, the focus is usually on deploying high-performing models instead of interpretable "white-box" models. During the model development phase, focus is on model prediction performance, and explainability is only a subsidiary goal. To make high-performing models explainable without prediction power interference, we focus on *post-hoc XAI*, i.e., providing explainability after model development.

The second main problem addressed in this thesis is the evaluation of *XAI* methods. Our goal is to quantify the performance of *XAI* methods without the need for human feedback. Although qualitative evaluation of *XAI* methods is an important and complementary type of evaluation analysis, specifically to evaluate how helpful explanations are perceived, human perception is highly subjective. Depending on the background and use case, *XAI* methods might yield inconsistent results, hindering reproducible and standardized method benchmarking. Additionally, many properties of explainers can hardly be evaluated with a human actor, e.g., the alignment between *XAI* methods and *ML* model. Further, human feedback is expensive and tiring to create, and studies evaluating *XAI* with human feedback have to be carefully curated to prevent bias. Even then, objective quantification of the feedback is hard. If methods should be evaluated objectively, all benchmarking baselines must utilize the same study setting. A decision on which explainer is helpful can take quite some time, and comparisons are often not objective.

Finally, the last problem focuses on *XAI* based model revision. The aim is to combine continuous learning with explanatory interactive machine learning to enable lifelong learning under human supervision. Again, we exclude the human factors from the experiment, as the aim is to get a proof of concept if such a setting would allow continuous learning and, thereby, the learning of changing data distributions and tasks under human supervision. In a first setting, we exclude human bias to evaluate which algorithm combination works best in a perfect setting - i.e., known ground truth, known labels, and perfect simulated annotations. While the second setting, the real-world dataset, utilizes real (noisy) human annotations, we simulate the feedback-giving process to eliminate pitfalls of human cognition, e.g., Influences of the graphical interface on the annotation/feedback process.

1.6 Outline

We now outline the contents of the thesis, consisting of four parts: *Foundations*, *Explainable Machine Learning Methods for Time Series*, *Evaluating Explainable Machine Learning*, *Model Revision*, and *Conclusion*.

Starting with Part I – *Foundations*, we introduce the following chapters to establish the knowledge necessary for the remaining thesis.

Chapter 2 – *Preliminaries*
This chapter defines and introduces the terminology, baseline methods, and techniques necessary for the contribution of this thesis.

Chapter 3 – *Related Work*
This chapter deals with works and publications related to the contributions of this thesis focusing on *XAI*, the quantification of *XAI*, and *XAI* based model revision.

In Part II – *Explainable Machine Learning Methods for Time Series*, we address the first research problem: the missing support and accessibility of *XAI* on time series data.

Chapter 4 – *A Framework for post-hoc* XAI *on Time Series*
This chapter creates an extensible general-purpose framework following the scikit-learn principles for *XAI* on time series.

Chapter 5 – *Counterfactual Explanations for Uni- and Multivariate Time Series*
This chapter extends the framework with a new method to time series counterfactuals for uni- and multivariate data by formulating counterfactual search as multi-objective problem and approximating a solution with a custom genetic-based algorithm that utilizes time series transformation mechanisms.

In Part III – *Evaluating Explainable Machine Learning*, we develop an evaluation framework for *XAI* on time series and a new metric for causal coherence.

Chapter 6 – *Benchmarking (post-hoc)* XAI *on Time Series*
This chapter builds on Chapter 4 and 5 by proposing and evaluating the time series algorithm with synthetic data from different time series processes and a variety of known ground truths on *XAI*-metrics adopted to the time series context.

Chapter 7 – *Measuring real-world coherence of counterfactual explanations*
This chapter proposes a new metric for the evaluation of real-world coherence of counterfactual methods utilizing causal inference. Further, the chapter introduces (semi-) synthetic data sets of various complexities to use in combination with the metrics as structural causal models are hard to obtain in the real-world.

Part IV – *Model Revision* introduces and evaluates a framework for continuously including human feedback on basis of the explanation obtained from an *XAI*-method.

Chapter 8 – *Continuous Explanatory Interactive Machine Learning*
This chapter builds on all the previous chapters by combining explanatory, interactive, and continuous machine learning to allow human supervision and model corrections.

Finally, Part IV – *Conclusion* provides a retrospective overview of the content of the thesis and an outlook on future works.

Chapter 9 – *Summary*
This chapter summarizes and discusses the thesis with respect to the challenges, research questions, hypotheses, and contributions. Further, the chapter gives an outlook on future research directions.

Throughout the thesis, we will mention the published research papers which correspond to the contributions of the respective chapters.

2 Preliminaries

The following chapters define the foundations of this thesis. It is divided into three parts: Supervised Machine Learning (Section 2.1), Explainable Machine Learning (Section 2.2), and Explainable Machine Learning for time series (Section 2.3). Thereby, Explainable Machine Learning builds on the foundations of Supervised Machine Learning and Explainable Machine Learning for time series on the foundations discussed in the section Explainable Machine Learning.

2.1 Supervised Machine Learning and Deep Learning

Machine Learning (*ML*), a subdiscipline of *AI*, focuses on giving "computers the ability to learn without explicitly being programmed" [205] by enabling computers to learn complex relationships from data. More formally, *ML* can be defined as: "A computer program is set to learn from experience E with respect to some task T and some performance measure P, if its performance at task T, as measured by P, improves with Experience E." [154] Thereby, task T is the problem to be solved quantified by some performance P, often in the form of a loss function $\mathcal{L}$, based on the experience E represented by the available data D. Supervised Machine Learning assumes the availability of a labeled dataset $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ to learn a function f to predict the outcome y.

➢ **Definition 2** (Supervised Machine Learning)**.** Given a dataset $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ consisting of n samples, supervised machine learning learns a function $f : X \rightarrow Y$, that enables the mapping of a feature vector $x_i \in X$ on a target $y_i \in Y$. To enable such a mapping, the model has to learn parameters θ by minimizing a loss $\mathcal{L}$ specifying the error over all training instances (see Equation (2.1)).

$$\theta^* = \min \mathcal{L}(f(X), Y) \tag{2.1}$$

The model approximating the function f can be anything from linear regression to neural networks. The loss $\mathcal{L}$ optimized by the model to learn f depends on the target task. Often, for classification tasks cross-entropy is used, while for regression the Mean Absolute Error is preferred. Supervised *ML*-models aim to minimize a loss $\mathcal{L}$ based on the available data D.

Depending on the data availability, *ML* differentiates between online and offline learning. If all data instances are available simultaneously, learning is called Offline or Batch learning. Online Learning refers to continuously arriving data.

➢ **Definition 3** (Offline Learning). Let D be a set of data points $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ given at training time. A machine learning model f tasked with learning a function $f : X \rightarrow Y$ minimizes the loss $\mathcal{L}$, using all labeled data points at once.

In online learning scenarios, data becomes available continuously and the model f has to update model parameters gradually for individual data points.

➢ **Definition 4** (Online Learning). Let D_t be a data point (x_t, y_t) available at time t, a machine learning model f tasked with learning a function $f : X \rightarrow Y$ only relies on D_t to learn or update the model.

In the following, we focus on Deep Learning in a supervised learning setting and its core component, Artificial Neural Networks (see Section 2.1.1). Deep Learning, a subclass of *ML*, exploit many layers of non-linear information processing for supervised or unsupervised feature extraction with unprecedented performance in many tasks [79].

2.1.1 Artificial Neural Networks

Artificial Neural Networks, from now on only referred to as neural networks, are modeled after the human brain and consist of many interconnected neurons [193]. The smallest entity in a neural network is a neuron. A neuron (see Figure 2.1), first introduced by [149], consists of N input channels, an input and an activation

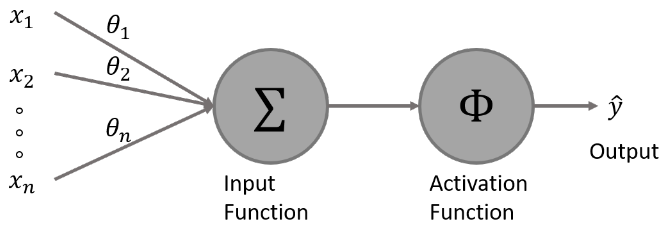
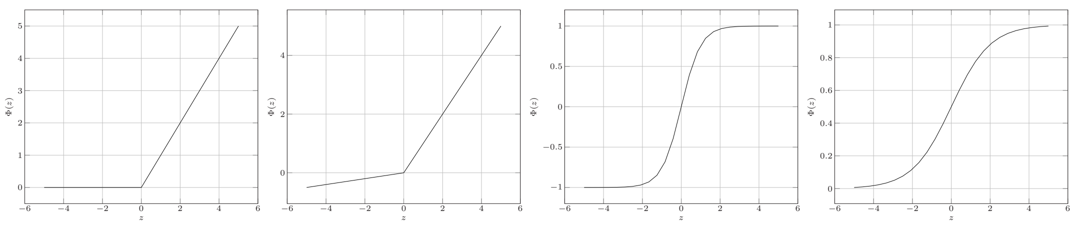


Figure 2.1: Abstract artificial neuron. Inspired by [154, p. 87]

function. Thereby, the input of the neuron consists of the weighted sum $z = \sum_j \theta_j x_j$ of input value $x = \{x_1, x_2, \dots, x_N\} \in \mathbb{R}$. The neuron's output $\hat{y}$ is calculated by applying a function Φ, called activation function, to z (Equation (2.2)). The activation function Φ enables modelling complex functions. The choice of activation function depends on the usage of a neuron. It usually contains non-linearities. Figure 2.2 shows the most common choices: Rectified Linear Units (ReLU), Leaky ReLU, TanH, and Sigmoid. The also popular softmax, an extension of the sigmoid function, calculates the probability distribution over a vector, instead of a single input. A softmax function in a vector of size two, yields the same result as applying sigmoid. More information on activation function and their specific properties can be found in many textbooks, e.g., [79, 86].

$$\hat{y} = \Phi(z) = \Phi(\sum_{j}^{N} \theta_j x_j) \tag{2.2}$$



(a) ReLU $\Phi(z) = \max(0, z)$ **(b)** Leaky ReLU: $\Phi(z) = \max(0.1z, z)$. **(c)** TanH: $\Phi(z) = \frac{(e^z - e^{-z})}{(e^z + e^{-z})}$ **(d)** Sigmoid: $\Phi(z) = \frac{1}{(1+e^{-z})}$

Figure 2.2: Visualization of activation functions.

A neural network consists of many neurons arranged into different layers and connections between those layers and neurons. One of the most simple neural architectures is a *Multilayer Percepton* consisting of neurons organized in at least three layers (an input layer, a hidden layer and an output layer). Let l be the number of layers of a neural network, n the number of input neurons $x_1, x_2, \ldots, x_n$, m the number of output neurons $y_1, y_2, \ldots, y_m$ and H_l the neurons of layer l $h_1^l, h_2^l, \ldots, h_{H_l}^l$. If the k^{th} neuron is in layer $(l-1)^{th}$ and the j^{th} neuron is in the l^{th} layer, then the connection between these two neurons has the weight θ_{jk}^l.

Let the function of each layer in an N-layered network be $f^{(1)}, f^{(2)} \ldots f^{(N)}$. In a feedforward network, these functions are connected in a chain. By that the overall function of a network is $f(x) = f^{(N)}(\ldots(f^{(3)}(f^{(2)}(f^{(1)}(x)))))$. By training the neural network, the goal is to drive $f(x)$ to match the target function $f^*(x)$. The training data consists of multiple examples. For each example x, a label y is assigned. The output layer should produce a value $\hat{y}$ that is close to y. The behavior of the other layers (hidden layers) are not specified by the training data. A learning algorithm must decide how to use those hidden layers to get an appropriate approximation of the f^*.

More details on learning the approximation f^* with artificial neural networks can be found in Section 2.1.3.

2.1.2 Architectures

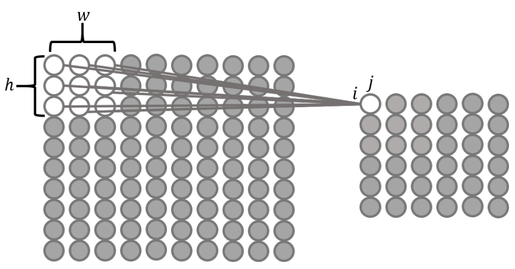
Based on the feedforward architecture, many other architectures that are more tailored to specific use cases have been proposed. In the following, we take a closer look at *Convolutional Neural Network* (*CNN*) (section 2.1.2.1) and *Recurrent Neural Network* (*RNN*) (section 2.1.2.2).

2.1.2.1 *Convolutional Neural Network* (*CNN*)

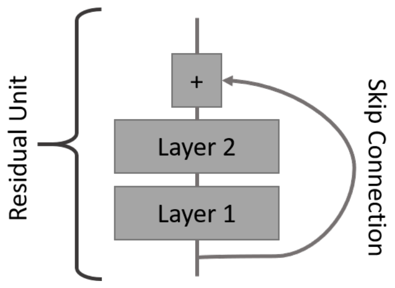
A *Convolutional Neural Network* (*CNN*) is a special kind of feedforward neural network specialized in processing data with a grid-like topology. The name *Convolutional Neural Network* arises from the mathematical operation called convolution used instead of general matrix multiplication in at least one layer.[1] Usually, convolution is used in combination with pooling layers. The ideas behind convolutional neural networks are: local receptive fields, shared weights, and pooling.

Local receptive fields Neurons in a convolutional layer are only connected to the input feature in the receptive field. Figure 2.3a visualizes the receptive field. A neuron located in row i, column j of a given layer is connected to the outputs of the neurons located in the previous layer in rows i to $i + f_h - 1$ and columns j to $j + f_w - 1$. f_w and f_h denote the width and height of the receptive field. The receptive field is slid through

[1] Convolution: mathematical operation that slides one function over another and measures their integral of pointwise multiplication. [86, p. 357] As this function is implemented differently in most of the deep learning frameworks no mathematical definition is provided.



(a) Receptive Field of a Convolutional Neuron.



(b) Residual Unit.

Figure 2.3: Visualization of the local receptive field of an *CNN* and a Residual Unit.

the whole input. Each local receptive field connects to a different hidden neuron in the next layer. The stride size, both vertical s_w and horizontal s_h, of the receptive field can vary depending on the application.

Shared weights and biases Every neuron in a convolutional layer uses the same weights and biases also often referred to as feature maps.

$$z_{i,j,k} = b_k \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f'_n-1} x_{i',j',k'} \theta_{u,v,k',k} \text{with} \begin{Bmatrix} i' = i \times s_h + u \\ j' = j \times s_w + v \end{Bmatrix} \tag{2.3}$$

$z_{i,j,k}$ denoted the output of the neuron located in row i, column j in feature map k. s_h and s_w denote the vertical and horizontal strides and f_n the number of feature maps. The output of a convolutional layer is calculated as the product of the neuron output $x_{i',j',k'}$ of the previous layer and the connection weights $\theta_{u,v,k',k}$.

Pooling layers Pooling layers are most of the time used directly after a convolutional layer. The pooling layer takes the output of each feature map and prepares a condensed feature map. Different procedures for pooling have been proposed. One of the probably most used ones is known as max-pooling. The max-pooling layer simply outputs the maximum activation of a predefined input region. Another possibility is L2 pooling which returns the square root of the sum of squared activations for a defined region.

Typical architecture Typical convolutional architectures stack a few convolutional layers (usually each followed by a ReLU layer), followed by a pooling layer and another few convolutional layers and pooling layers. On top of that, regular feedforward neural networks are stacked and a final output layer is added to make the prediction. Many other architectures have been proposed like AlexNet [120], ResNet [88], and GoogLeNet [231].

ResNet The Residual Network (ResNet) proposed by [88], introduce residual units that skip layer connections. Figure 2.3b shows a residual unit feeding the input signal both into the layer and to the output. In ResNet each residual unit is composed of two convolutional layers with Batch Normalization and ReLu activation. In the following we rely on ResNet34.

2.1.2.2 ***R**ecurrent **N**eural **N**etwork* (*RNN*)

A *Recurrent Neural Network* (*RNN*), usually used for data with a time dimension, has additionally at least one feedback loop. A feedback loop connects the output back to the neuron's input (see Figure 2.4). At each time step t, a recurrent neuron receives an input $x_{(t)}$ as well as its own output from a previous time step $y_{(t-1)}$. Figure 2.4 shows such a recurrent neuron and its forward pass through time. An *RNN* consits of many recurrent neurons. Equation (2.4) shows the output calculation of a recurrent layer for a single instance

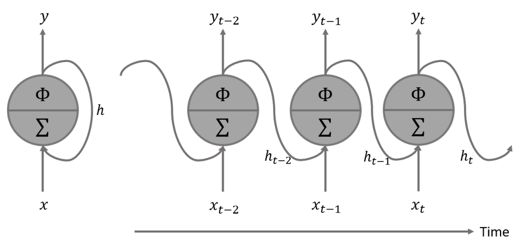


Figure 2.4: Visualization of a Recurrent Neuron (left), and a recurrent network unrolled through time (right). Adopted from [86].

x_t. Each layer has two set of weights θ_x for the inputs at time step t and θ_y for the outputs of the previous time step.

$$y_{(t)} = \Phi(\theta_x x_{(t)} + \theta_y y_{(t-1)} + b) \tag{2.4}$$

$y_{(t)}$ is now a function of x_t and y_{t-1}, therefore it preserves some state across time steps and is also referred to as memory cell. This state is often referred to as hidden state h_t and is a function of the current input and the previous output: $h_t = f(h_{t-1}, x_t)$

LSTM In 1997, Hochreiter & Schmidhuber [90] introduced the *Long Short Term Memory* (*LSTM*) consisting of three different functions called gates:

- The forget gate f_t controls which part of the long-term state should be erased.
- The input gate i_t decides which parts should be added to the long-term state.
- The output gate o_t controls which parts of the long-term state to output at time step t.

An *LSTM* cell looks similar to an *RNN* cell, except for the state split into a short-term state h_t and a long-term state c_t. Figure 2.5 shows the calculation of an LSTM cell at each time step for a single instance. The long-term state c_{t-1} traverses through the cell, going through the forget gate f_t, dropping some memories, and through the addition operation to add new memories (selected by the input gate). After adding the new memories, the short-term state $h_t = y_t$ is calculated by passing the long-term state through $\tanh$ and the output gate.

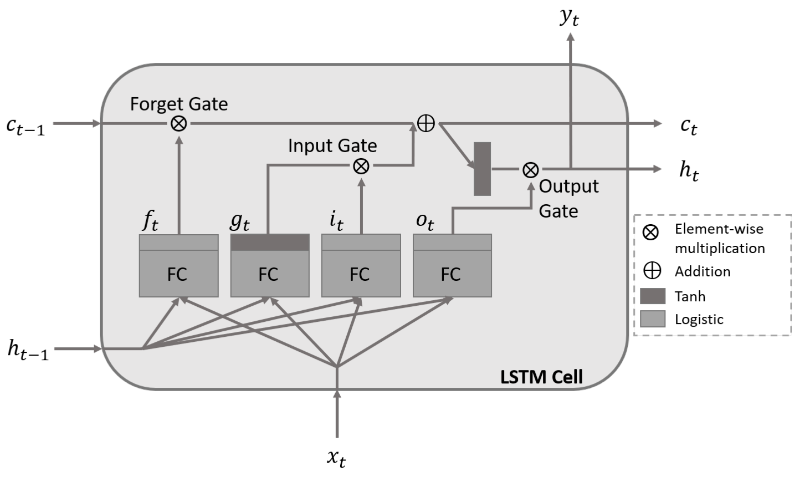


Figure 2.5: Visualization of a *Long Short Term Memory* (*LSTM*) cell. Adopted from [86].

2.1.3 Gradient-Based Training

Most machine learning algorithms involve optimization of some sort. Optimization refers to the task of either minimizing or maximizing some function $f(x)$ based on given samples (x, y) by altering x. The optimization objective in machine learning is usually to minimize a loss function L. For supervised learning the aim is to learn $\theta^* = \mathcal{L}(f(x), y)$.

Suppose we have a function $y = f(x)$, where both x and y are real numbers. The derivative of this function is denoted as $\frac{\partial y}{\partial x}$. The derivative gives the slope of $f(x)$ at the point x. In other words, it specifies how to scale a small change in the input to obtain the corresponding change in the output:$f(x+\epsilon) \sim f(x) + \epsilon\frac{\partial f(x)}{\partial x}$. The derivative is therefore useful for minimizing a function because it tells us how to change x in order to make a small improvement in y. [79]

In the following, we introduce Gradient Descent (Section 2.1.3.1), Backpropagation (Section 2.1.3.2), and Adam (Section 2.1.3.3) for learning the parameters θ of deep neural networks.

2.1.3.1 Gradient Descent

Gradient Descent requires the computation of the gradient of the loss function $\mathcal{L}$ with regard to each model parameter θ_i (see Equation (2.5)). Based on those derivatives, the model performs parameter updates with a learning rate of η (see Equation (2.6)).

$$\nabla_\theta \mathcal{L}(f(X), Y) = \begin{bmatrix} \frac{\partial}{\partial \theta_0}\mathcal{L}(f(X), Y) \\ \vdots \\ \frac{\partial}{\partial \theta_n}\mathcal{L}(f(X), Y) \end{bmatrix} \tag{2.5}$$

$$\theta^{n+1} = \theta^n - \eta \nabla_\theta L(f(x), y) \tag{2.6}$$

The smaller the learning rate, the longer gradient descent needs to converge. In some cases, a low learning rate may lead to maximum iterations before reaching the optimum point. If the learning rate is too high the algorithm may not converge to the optimal point (jump around) or even diverge completely.

In *ML*, the most common types of gradient descent are batch gradient descent, minibatch gradient descent, and *Stochastic Gradient Descent* (*SGD*).

Batch Gradient Descent utilizes the entire dataset (batch) at each iteration. While it allows for stable learning as parameter updates are based on the whole dataset, the convergence is quite slow and the requirement to hold large dataset in memory make it only feasible on small datasets.

In contrast to Batch Gradient Descent, *Stochastic Gradient Descent* picks a random instance in the training set for every iteration and computes the gradients based on this instance. Updates on a single instance fasten the convergence, however instead of gently decreasing, the loss often bounces up and down, only decreasing on average.

Minibatch gradient descent is a mixture of *SGD* and Batch Gradient Descent. Instead of only selecting one instance as in *SGD*, a batch of instances is picked from the dataset. Gradient updates are performed based on this minibatch. Unless otherwise denoted, we refer to minibatch gradient descent as batch gradient descent.

2.1.3.2 Backpropagation

Backpropagation [199], developed in 1986, is a strategy for training and thereby learning the parametrization θ of deep neural networks. In backpropagation, the algorithm makes a prediction $\hat{y} = f(x)$ for each training instance x and measures the error $L(f(x, \theta), y)$ (forward pass). Thereafter, each layer is gone through in reverse to measure the error contribution for each error (reverse pass). The goal is to understand how changes in weights or biases influence the cost function. In the last step, the gradient descent step, the weights are modified to reduce the error.

Forward pass The activation a of a neuron j in layer l is calculated as the weighted sum of all activations of neurons in the layer $(l-1)$. The number of neurons in layer $l-1$ is denoted with j. On the weighted sum, the activation function Φ is applied. The weighted sum of activations $\sum_k \theta^l_{jk} a^{l-1}_k + b^l_j$ is often referred to z^l_j. Hereby, the weights can be summarized in a weight matrix θ^l, which entries are just the weight connecting to the l^{th} layer of neurons. b^l represents the bias vector and $a^{(l-1)}$ the activations of the neurons of layer $l-1$. This would result in an activation vector a^l.

$$a^l_j = \Phi(\sum_k \theta^l_{jk} a^{l-1}_k + b^l_j) \tag{2.7}$$

Note, that the calculation of the forward pass is consistent with the output calculation of a single artificial neuron described in Equation (2.2).

Backpropagation In order to evaluate a classification done by a neural network, some kind of metric needs to be defined. Often chosen is the quadratic cost function, which measures the difference between the desired output $y(x)$ and the actual activation $a^L(x)$ for each individual training example. n denotes the number of training examples, L the number of layers in the network.

$$\mathcal{L} = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2}\sum_j (y_j - a^L_j)^2 \tag{2.8}$$

To understand how networks change if the weights and biases change the partial derivatives $\frac{\partial \mathcal{L}}{\partial \theta^L_{jk}}$ and $\frac{\partial \mathcal{L}}{\partial b^l_j}$ are calculated. The error δ^l_j is the error of the j^{th} neuron in the l^{th} layer. In function 2.9, which describes the error of the output layer, $\frac{\partial \mathcal{L}}{\partial a^l_j}$ measures how fast the equation is changing as a function of the j^{th} output activation. How fast the activation function Φ is changing at z^l_j is described in $\Phi^{'}(z^l_j)$.

$$\delta^l_j = \frac{\partial \mathcal{L}}{\partial a^l_j} \Phi^{'}(z^l_j) \tag{2.9}$$

When the error of the output function is calculated, the next step is to calculate the errors of the other layers. The error of layer l is calculated in the terms of the next layer δ^{l+1}, as can be seen in equation 2.10. The first term $((\theta^{l+t})^T \delta^{l+t})$ moves the error calculated in the next layer δ^{l+t} backwards through the network. The second term $\Phi^{'}(z^l_j)$ moves the error backwards through the activation function. In this case, $\bigodot$ denotes the Hadamard product.[2]

$$\delta^l = ((\theta^{l+t})^T \delta^{l+1}) \bigodot \Phi^{'}(z^l) \tag{2.10}$$

By combining equation 2.9 and 2.10 the error for any layer can be calculated. To move the error back through all the hidden layer, 2.10 is calculated repeatedly for all hidden layers.

[2] The Hadamard product is the element-wise product of two vector $s(\bigodot t)_j = s_j t_j$.

In Equation 2.11 the rate of change regarding the bias can be found. The rate of change is exactly equal to the error δ_j^l.

$$\frac{\partial \mathcal{L}}{\partial b_j^l} = \delta_j^l \tag{2.11}$$

In the last step, the rate of change with respect to any weights in the network is calculated. $\frac{\partial \mathcal{L}}{\partial \theta_{jk}^l}$ is the partial derivative of the weights. Hereby a_k^{l-1} is the activation of the neuron input to the weight θ and Φ_j^l is the error of the neuron output from the weight θ.

$$\frac{\partial \mathcal{L}}{\partial \theta_{jk}^l} = a_k^{l-1} \Phi_j^l \tag{2.12}$$

2.1.3.3 Adam

Training deep neural networks can be really slow resulting in the development of many different optimizers (e.g., AdaGrad [63], RMSProp[3] or Adam [114]). In the following, we focus on Adam. Adam combines the ideas of momentum optimization and RMSProp. For a more comprehensive overview of gradient descent optimization algorithms, see [79] and [197].

In 2015 Kigma et al. [114] proposed Adam (adaptive momentum estimation) optimizer for deep learning networks. Adam calculates for each parameter different adaptive learning rates from estimates of the first and second momentum of the gradients.

Algorithm 1 Adam. Reproduced from [114].

```
Require: $\alpha$: Step size
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the momentum estimates
Require: $f(\theta)$: Stochastic objective function with parameter $\theta$
Require: $\theta_0$: Initial parameter vector
 1: $m_0 \leftarrow 0$ (Initialize $1^{st}$ momentum vector)
 2: $\nu_0 \leftarrow 0$ (Initialize $2^{nd}$ momentum vector)
 3: $t \leftarrow 0$ (Initialize time step)
 4: while $\theta_t$ not converged do
 5:     $t \leftarrow t + 1$
 6:     $g_t \leftarrow \bigtriangledown_\theta f_t(\theta_{t-1})$
 7:     $m_t \leftarrow \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$
 8:     $\nu_t \leftarrow \beta_1 * \nu_{t-1} + (1 - \beta_1) * g_t^2$
 9:     $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
10:     $\hat{\nu_t} \leftarrow \frac{\nu_t}{1-\beta_2^t}$
11:     $\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m_t}}{\sqrt{\hat{\nu_t}+\epsilon}}$
12: end while
13: return $\theta_t$
```

Let the objective function $L(\theta)$ be a stochastic objective function that is differentiable in its parameters θ. Hereby $L_t(\theta)$ denotes the stochastic function at a time step $t \in [0, T]$. The evaluation of random subsamples of data points leads to the stochasticity of Adam. Let g_t be the vector of partial derivatives of f_t with respect to θ at time step t (Algorithm 1 line 6). In line 7 and 8 the algorithm updates the exponential moving average m_t which is an estimation of the 1^{st} momentum and the squared gradient (ν_t) which is an estimation of

[3] Unpublished method proposed by Geoffrey Hinton in lecture: `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`

the 2^{nd} raw momentum. The hyperparameters β_1, β_2 which control the decay rate of the moving averages influence the exponential moving average and the squared gradient are. Since m_t and ν_t are initialized to 0 at the beginning of training, they are biased towards 0. Therefore, in line 9 and 10 the estimates are corrected for the bias. After the calculations are done θ is updated in line 10.

2.2 Explainable Artificial Intelligence (XAI)

The research field of *EXplainable Artificial Intelligence* (*XAI*), a term first coined in 2004 [129], studies approaches unveiling the rationale behind automatic decision-making systems to make them more comprehensible and transparent to humans without sacrificing model performance [2]. *ML*-Models exhibiting some of their inner workings (e.g., linear or logistic regression) are often referred to as explainable and interpretable. However, there is no general agreement within the *ML*-community on the definitions of explainability and interpretability. Many authors use both terms interchangeably (e.g., [39]), while others differentiate [144, 9].

Similar to [24], we define explainability and interpretability as:

➢ **Definition 5** (Explainability)**.** Explainability is an active characteristic of a model, denoting any action or procedure taken with the intent of clarifying or detailing internal functions.

➢ **Definition 6** (Interpretability)**.** Interpretability is a passive characteristic of a model, referring to the level at which a given model makes sense for a human observer (transparency).

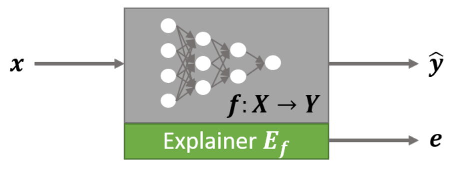
Therefore, in our case, interpretability comprises explainability. Explainability is provided with the help of an Explainer E representing any *XAI* method.

➢ **Definition 7** (Explainer E)**.** For a model $f : X \to Y$ with in input x and output prediction $\hat{y}$. An explainer E provides a heuristic generating an explanation e, that provides insights into the functioning of the model f and therefore explainability.

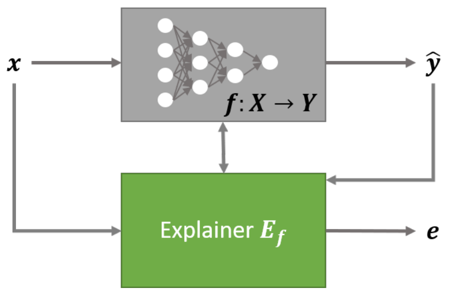
The exact definition of *XAI* and many of its related terminology is not yet fully established. Therefore, in the following sections we introduce the notions, terms and technologies used in this work. We start of by intoducing the taxonomy of *XAI* used in this work (Section 2.2.1), followed by more details on *Feature Attribution*-methods (Section 2.2.2) and *Instance-Based Methods* (Section 2.2.3).

2.2.1 Taxonomy

XAI methods can be classified based on various criteria. We follow the taxonomy of [2], [39], and [81]. Methods are classified according to the point of introduction (Section 2.2.1.1), the type of model they can explain (Section 2.2.1.2), the explanation scope they cover (Section 2.2.1.4), and the output they produce (Section 2.2.1.3).



(a) Ante-Hoc *XAI*



(b) Post-Hoc *XAI*

Figure 2.6: Post-Hoc vs. Ante-Hoc *XAI*.

2.2.1.1 Point of Introduction

Depending on the used *XAI* methods, integration steps become necessary at different stages of the *ML* models development process. Some methods allow the out-of-box usage on trained *ML* models. Others need to be introduced during the model development stage. For example, some *XAI* methods rely on adding additional regularizers to the *ML* model's loss function. Post-Hoc and Ante-Hoc describe the point of time at which an *XAI* method is applied.

Ante-Hoc *XAI* (Figure 2.6a), sometimes called instrinct explainability, refers to models interpretable by design. This can be achieved by constraining model complexity (e.g., using decision trees instead of a neural network) or including explanation components in the model or training process design (e.g., [206]).

Post-Hoc *XAI* (Figure 2.6b) refers to explanation methods applied after model training and are usually decoupled from the model. In contrast to ante-hoc methods, post-hoc *XAI* does not influence the *ML* model performance.

2.2.1.2 Predictor Constraints

Some *XAI* methods place constraints on the *ML* model due to their working mechanism. Model-specific and model-agnostic refer to constraints placed on the type of predictor f by the *XAI* approach E.

A model-agnostic explainer E can be used to explain any type of model f and rely on analyzing the connection between input x and output $\hat{y}$. Those methods cannot access the model's internal parametrization θ.

Model-specific explainers are limited to specific model classes (e.g., *CNN*) or rely on a specific model internas (e.g., the internal parametrization θ).

2.2.1.3 Explanation Output

Depending on the *XAI* method deployed, the resulting explanation can take on various forms needing different types of visualization. Generally, the output is differentiated into Instance-Based, Feature Attribution, and Rule-Based explanations.

Feature Attribution (*FA*) returns a per-feature attribution score ϕ_j based on the feature's contribution to the model's output. *FA* methods are often further divided according to the mechanism used to obtain the attributions. Perturbation-based methods are model-agnostic and obtain feature attributions by analyzing different perturbations x and the predicted outcome $\hat{y}$ of the model f. Gradient-based methods approximate feature attribution by relying on the model's internal parametrization θ.

Instance-Based Methods (*IB*) calculate a subset of relevant features that must be present to retain the prediction or removed to change the prediction of a given model. The explanation e comprises a new sample x' consisting of $\tilde{x}_i$ original, perturbed or removed features.

Rule-Based Methods (*RB*) extract a set of rules approximating the decision boundaries of the model f by fitting a rule-based surrogate or using instrinct interpretable models. In this work, this type of explanation is not considered further.

2.2.1.4 Scope of Explanation

The scope of methods refers to the context covered by an explanation. *XAI* methods can cover the whole model, or only provide explanations for a single instance x and prediction $\hat{y}$ of a model f.

Global explainers require no predictions and rely on the learned model f, often the training data D and/or some set of feature vectors. The explainer is static for all instances x.

Local explainers are instance-based – i.e., for a single test input x and the corresponding prediction $\hat{y}$, a new explanation is generated explaining only the decision of f for the specific data point x.

2.2.2 Feature Attribution Methods

Feature attribution methods measure the feature contributions to the models' output prediction.

➢ **Definition 8** (Feature-Attribution Explainer)**.** An explainer returning a feature attribution assigns an attribution ϕ_j to explain the importance of feature j, resulting in $E_f(x) = \phi = (\phi_0, \ldots, \phi_n)$, where n is the number of interpretable features.

Depending on the methodology used, *FA* methods are further divided into gradient-based and perturbation-based methods. Perturbation-based methods rely on modifying the input and concluding from the output change the attribution of features. Such a separation of black-box model and its explanation by only working with the inputs and outputs, provides better accuracy, flexibility, and usability [191]. The common idea behind gradient-only methods is that if a pixel in the input image is altered, the predicted probability of the class will either increase (positive gradient) or decrease (negative gradient). The greater the impact of an alteration to a pixel, the higher the absolute value of that gradient. In the following, we introduce only attribution methods related to the remaining sections of the thesis.

2.2.2.1 LIME

Locally Interpretable Model-Agnostic Explainer (*LIME*) [191] learns a local (white-box) surrogate model $\mathcal{F}$ to explain an individual prediction $\hat{y}$ of a black-box classifier f. The two-step approach includes extracting interpretable components and building the white-box proxy $\mathcal{F}_x$ around the instance x. To learn the local surrogate model $\mathcal{F}$, *LIME* assumes that for each x a set of interpretable components x'_i can be obtained. The function $h : X \to X'$ denotes the mapping function from the original instance x to the interpretable feature domain x'. The exact nature of h depends on the context. Tabular data features are often perturbed individually, while images use "superpixels" - i.e., interconnected pixels with similar colors are used. Algorithm 2 describes the steps for fitting the surrogate model. LIME generates a perturbed dataset $\mathcal{Z}$ consisting of sampled data points and the prediction on those data points by iteratively masking the interpretable components x' with uninformative features and observing the output $f(X') = \hat{Y}'$. The sampling function varies depending on the data type. This dataset $\mathcal{Z}$ and the loss function $\mathcal{L}$ Equation (2.13) are used to train the surrogate model $\mathcal{F}_x$. The loss function $\mathcal{L}$ measures the unfaithfulness of the fitted surrogate model $\mathcal{F}_x$ to the model f in the locality defined by π where $\pi(x, x')$ is the proximity between the original and perturbed samples.

$$\mathcal{L}(f, \mathcal{F}_x, \pi_x) = \sum_{z,z' \in \mathcal{Z}} \pi_x(z)(f(z) - \mathcal{F}_x(z'))^2 \tag{2.13}$$

Algorithm 2 *LIME*. Reproduced from [191].

```
Require: Classifier $f$, Number of Samples $N$
Require: Instance $x$, Mapping $h$
Require: Similarity Kernel $\pi_x$, Length of Explanations $K$
1: $\mathcal{Z} \leftarrow \{\}$
2: for $i$ in $n$ do
3:     $z'_i \leftarrow$ sample_around$(x')$
4:     $\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z'_i), \pi_x(z_i) \rangle$
5: end for
6: $\theta \leftarrow LinearModel.fit(\mathcal{Z}, f(\mathcal{Z}))$
7: return $\theta$
```

The obtained explanation $E_f(x)$ interprets the target sample x with the weight θ of the white-box model $\mathcal{F}_x$.

2.2.2.2 *SHAP*

SHapley Additive exPlanations (*SHAP*)[139] explains individual predictions by using shapely values aiming to explain the prediction for any instance x as a sum of contributions from its individual feature values. Thereby, *SHAP* computes individual feature contributions towards the output prediction by formulating the data features as players in a coalition game and learning to distribute the payout fairly. A player can be an individual feature value x_i (e.g., for tabular data) or a set of interpretable feature components x'_i. To extract such interpretable components from the original features a mapping function $h : X \rightarrow X'$ is used, for example grouping image pixels into superpixels. Equation (2.14) specifies the explanation of *SHAP*.

$$E(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{2.14}$$

E is the explanation model, $z' \in \{0, 1\}$ the coalition vector indicating the presence (1) or absence (0) of a feature, M the maximal coalition size, and ϕ_j the feature attribution for feature j. In the coalition vector, an entry of 1 means that the corresponding feature value is "present" and 0 that it is "absent".

To reduce the number of evaluations for large inputs, Lundberg et al. [139] introduce KernelSHAP. Algorithm 3 shows the five steps of KernelSHAP. First, *SHAP* samples coalition vectors. In the next step, the feature transformation h_x transforms z back into the original feature space to obtain the prediction by model f. The SHAP Kernal calculates the weights for each z_k, before fitting a linear model. The feature attribution ϕ corresponds to the coefficients of the Linear Model θ.

Algorithm 3 KernelSHAP Algorithm. Reproduced from [49].

```
$z_k \leftarrow$ SampleCoalitionsByRemovingFeatures($x$)
$z_k \leftarrow h_x(z_k)$
$y_k \leftarrow f(z_k)$
$W_x \leftarrow SHAP(f, z_k, y_k)$
$L \leftarrow LinearModel(W_x).fit(p, m, simscore)$
**return** $\theta$
```

2.2.2.3 Feature Occlusion

Feature Occlusion (*FO*) [268], mainly used for explaining image classifiers, computes attributions by replacing each contiguous rectangular region with a given baseline/reference, and calculating the difference in output. For features located in multiple regions (hyperrectangles), the corresponding output differences are averaged to compute the attribution for that feature. For each input x, let $\tilde{x}$ be the occluded input. For each input i we compute $\epsilon_i = a(x_i)^l - a(\tilde{x}_i)^l$, where $a(x_i)^l$ and $a(\tilde{x}_i)^l$ are the activation vectors at layer l for the original and occluded input. In this thesis, we assume using the last layer with a softmax output. *FO* then measures the consistency of this difference vector ϵ with the Hemming distance $\mathcal{H}$ between all related input pairs (i, j): $E_f^{FO}(x) = \sum_{i,j=1,i\neq j} \mathcal{H}(sign(\epsilon_i), sign(\epsilon_j))$.

2.2.2.4 Saliency

Vanilla Gradients [219], in this work referred to as Saliency, calculates the loss function's gradient with regard to the networks inputs. After performing a forward pass on model f with input instance x, Saliency calculates the gradient of class score $f_c(x)$ of interest with respect to the input pixels (Equation (2.15)).

$$E_f^{Saliency}(x) = \frac{\partial f_c(x)}{\partial x} \tag{2.15}$$

2.2.2.5 Grad-CAM

Grad-CAM [212] uses the gradient information flowing into the last convolutional layer of the *CNN* to assign importance values to each neuron for a particular decision. Grad-CAM aims at finding class discriminative localization maps $L^C \in \mathbb{R}^{u\times v}$ of width u and height v for any class c from the k feature maps in the last convolutional layer $\{A_1, ., A_k\}$. Grad-CAM approximates the importance α_k of each of the k feature map to class c (see Equation (2.16)).

$$L^C_{Grad\_CAM} = ReLU(\sum_k \phi^c_k A^k) \tag{2.16}$$

To calculate the importance ϕ, Grad-CAM computes the gradient of the score for class c, y_c (before the softmax), with respect to feature map activations A^k of a convolutional layer, i.e. $\frac{\partial y^c}{\partial A^k}$. These gradients flowing back are global-average-pooled over the width and height dimensions (indexed by i and j) to obtain the neuron importance weights ϕ^c_k.

$$\phi^c_k = \frac{1}{Z}\sum_i\sum_j \frac{\partial f_c(x)}{\partial A^k_{i,j}} \tag{2.17}$$

Given the class of interest c, the explanation $E_f^{GradCAM}$, consist of the neuron importance weights α: $E_f^{GradCAM} = \phi^c$.

2.2.2.6 *Integrated Gradients* (*IG*)

Integrated Gradients (*IG*) [229] observes the average gradient while input changes from a non-informative reference point $\bar{x}$ to x. The explanation $e = E_f^{IG}(x)$ will depend upon the choice of the reference point $\bar{x}$ (which is often set to zero). The explanation along the i^{th} dimension of x_i is defined as $E_f^{IG}(x)_i$

$$E_f^{IG}(x)_i = (x_i, \bar{x}_i) \times \int_{\alpha=0}^{1} \frac{\partial f(\bar{x} + \alpha(x - \bar{x}))}{\partial x_i} d\alpha \tag{2.18}$$

2.2.2.7 Smooth Gradient

Smooth Gradients (*SG*) [221] computes the gradient n-times adding Gaussian noise $\epsilon \approx \mathcal{N}(0, \sigma^2)$ with standard deviation σ to the input at each time to make gradient-based methods less noisy.

Smooth Gradient follows three steps:

1. Generate n versions of the input of interest by adding noise: $\bar{x} = x + \epsilon$.
2. Create attribution maps for all inputs $\bar{x}$.
3. Average the pixel attribution maps. (see Equation (2.19))

$$E_f^{SG} = \frac{1}{n} \sum_{1}^{n} \frac{\partial f(x + \mathcal{N}(0, \sigma^2))}{\partial x_i} \tag{2.19}$$

2.2.3 Instance-Based Methods

Instance-Based Methods (*IB*) select or modify data instances of the dataset to explain the behavior or distribution of a predictor [156]. Methods include Counterfactual Explanation (e.g., [252]), Prototypes (e.g., [112]), and Influential Instances (e.g., [1]). Thereby, instance-based explanations help humans to construct mental models of the machine learning model and the associated data [156].

➢ **Definition 9** (Instance-Based Explainer)**.** An instance-based explainer E_f provides an (perturbed) example with the same or a counter prediction within the data distribution trained resulting in $E_f(X) = (\bar{x}_1, \ldots, \bar{x}_N)$, where $\bar{x}_i$ denotes the original or perturbed feature value for instance x in feature i.

In the following, we focus on counterfactual explanations, referred to as *Counterfactuals* (*CF*). The notion of counterfactual stems from causal reasoning. In causal inference, *CF*s make causal assumptions about interdependencies among variables and use these assumptions to incorporate consequential adjustments when particular variables are set to new values [175]. Counterfactual explanations in *XAI* explore how input variables must be modified to change a model's output without making explicit causal assumptions [252].

➢ **Definition 10** (*Counterfactuals* (*CF*)). A counterfactual explanation $e = x^{cf}$ of a prediction $\hat{y}$ describes the smallest change to the feature values that changes the prediction to a predefined output y'.

Most methods follow Definition 10 loosely by adding additional constraints and only differ in the approach taken to solve the optimization problem [246]. Closely related to the notion of *CF*s is actionable recourse. Actionable Recourse focuses on finding a counterfactual by performing a feasible and minimal action a^* on the original instance $x^{cf} = x + a^*$. A feasible action is an action that can be performed in the real-world to obtain the counterfactual outcome. In case of a counterfactual x^{cf} and original instance x, the action can be calculated as $a^* = x^{cf} - x$. Therefore, we use actionable recourse and counterfactual interchangeably in this work. In the following sections, we introduce the counterfactual methods used in this work as a basis or benchmark. For a list of more methods refer to [246].

2.2.3.1 *Wachter Counterfactual* (*W-CF*)

Wachter et al. [252] find counterfactual explanations by formulating Definition 10 as an optimization problem and solving it with gradient descent (see Section 2.1.3.1). Equation (2.20) shows the proposed loss consisting of two terms. The first term measures how far the counterfactual's predicted outcome $f(x^{cf})$ is from the desirable outcome y'. The second term minimizes the change (distance) relative to the original data point $d(x, x^{cf})$. λ is a weighting factor balancing both objectives.

$$L(x, x^{cf}, y', \lambda) = \lambda(f(x^{cf}) - y')^2 + d(x, x^{cf}) \tag{2.20}$$

2.2.3.2 *Growing Spheres* (*GS*)

Growing Spheres (*GS*) [123] is a two-step greedy method minimizing Equation (2.21) that generates samples around the original data point by growing hyperspheres until a data point with desired predicted class is found.

$$L(x, x^{cf}) = ||x - x^{cf}||_2 + \gamma||x - x^{cf}||_0 \tag{2.21}$$

In the first step, *GS* explores the input space by generating instances in all possible directions until the decision boundary of the classifier f is crossed, thus minimizing the l2-component of the metric (see Equation (2.21)). In a second step, *GS* minimizes the L_0-component of Equation (2.21) by substituting original feature values x back into x^{cf} iteratively in the most proximate areas $\min ||x - x^{cf}||_2$, while $f(x) \neq f(x^{cf})$ still holds.

2.2.3.3 *Actionable Recourse* (*AR*)

Actionable Recourse (*AR*) [243] is based on integer programming and only applicable to linear models (e.g., logistic regression models, linear support vector machines). Instead of directly finding a counterfactual x^{cf}, this method aims at finding an action a reverting the prediction outcome y from an undesirable outcome -1 to a desirable outcome $+1$. In such a case, the counterfactual can be formulated as: $x^{cf} = x + a$. The optimization problem aiming to find an action a such that $f(x + a) = f(x^{cf}) = +1$ concludes in the desirable outcome. Compared to the other introduced methods so far, Ustun et al. [243] employ a strict constraint in their optimization framework (Equation (2.22)). They require each proposed action a to be in the set of feasible actions $A(x)$.

$$\begin{aligned} &\min \operatorname{cost}(a, x) \\ \text{s.t. } &f(x+a) = +1 \\ &a \in A(x) \end{aligned} \tag{2.22}$$

For details on the discretizing the problem, we refer the reader to [243].

2.2.3.4 ***Multi-Objective Counterfactuals* (*MOC*)**

Multi-Objective Counterfactuals (*MOC*) [48] builds upon the loss formulation of *W-CF* [252]. The method considers additional constraints and trade-offs between the proposed objective by formulating the search for counterfactuals as a multi-objective problem tailored to tabular data. Equation (2.23) shows the formulation and objectives of the counterfactual search. O_1 quantifies the difference between the counterfactual's predicted outcome $f(x^{cf})$ and the predefined outcome y'. The second criterion, tailored to mixed features, quantifies the difference between x^{cf} and x using the gower distance. Objective O_3, often called sparsity, calculates the number of features changed in x to obtain x^{cf} utilizing the L_0-norm. The fourth objective O_4, sometimes referred as plausibility, is an approximation for the likelihood of x^{cf} emerging from the input distribution $\mathcal{X}$.

The multi-objective problem is solved with *Non-dominated Sorting Genetic Algorithm* (*NSGA-II*) [52].

$$\begin{aligned} L(x, x^{cf}, y', X^{obs}) &= (O_1(f(x^{cf}), y'), O_2(x, x^{cf}), O_3(x, x^{cf}), O_4(x^{cf}, X^{obs})) \\ O_1(f(x^{cf}), y') &= \begin{Bmatrix} 0 & \text{if} f(x^{cf}) \in y' \\ inf_{y' \in} & \text{else} \end{Bmatrix} \\ O_2(x, x^{cf}) &= \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j^{cf}, x_j) \\ O_3(x, x^{cf}) &= ||x - x^{cf}||_0 = \sum_{j=1}^{p} \mathbb{1}_{x_j^{cf} \neq x_j} \\ O_4(x^{cf}, X^{obs}) &= \sum_{i=1}^{k} w^{[i]} \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j^{cf}, x^[i]_j) \end{aligned} \tag{2.23}$$

2.2.3.5 ***Counterfactual Conditional Heterogeneous Autoencoder* (*C-CHVAE*)**

Counterfactual Conditional Heterogeneous Autoencoder (*C-CHVAE*) [173] generates faithful counterfactuals by ensuring that the produced counterfactuals are proximate (i.e., not local outliers) and connected to regions with substantial data density (i.e., close to correctly classified observation). The counterfactual search is thereby included in a data density approximator, in this case, a *Variational Autoencoder* (*VAE*), and counterfactuals are sampled from the latent space of the *VAE*. The learned encoder m, maps x to its latent representation z, and the decoder g maps z back into the original data space. *C-CHVAE* generates the counterfactuals for x in the latent space of the encoder by perturbing $z^{cf} = z + \delta$. The decoder projects the latent representation z^{cf} back into the original feature space. The objective is to find the closest minimal perturbation of z, that fulfills $f(g(m(x) + \delta)) \neq f(x)$.

2.2.3.6 ***Counterfactual Recourse Using Disentangled Subspaces* (*CRUDS*)**

Counterfactual Recourse Using Disentangled Subspaces (*CRUDS*) [61] creates counterfactuals by using a conditional subspace *VAE*. In a conditional subspace *VAE*, the latent space is partitioned into two parts: one for learning label representations w, and one for learning the remaining latent representations necessary for generating the data z. The restricted subspace w allows the generation of counterfactuals by only changing relevant latent features. From a data point x, the approximated latent representation $w^s \sim m(w|x, y = 0)$ and $z^s \sim m(z|x)$, *CRUDS* draws N samples from the data distribution posteriori $m(z|x)$ and the posteriori encoding for a positive outcome $m(w|x, y = 1)$.

2.2.3.7 ***Causal Recourse* (*CR*)**

Causal Recourse (*CR*) [107] assumes an available *Structural Causal Model* (*SCM*) $\mathcal{M} = (U, V, F)$ to the problem solved by model f, where U is a set of latent background variables, V is a set of observed variables, and F is a set of functions showing the relations between V capturing the inter-variable causal dependencies in the real-world. *CR* aims to find minimal structural interventions that change the output of a model f. A structural intervention denotes an intervention $A = \mathrm{do}(V_i := a_i)$ on the *SCM*, instead directly on the model input x. Equation (2.24) describes the problem of finding the cost minimal (causal) action set. $x^{SCF} = F_A(F^{-1}(x))$ denotes the values obtained from the SCM after abducting x, performing the action a on the *SCM* and recursively determines the values of all endogenous variables. Thereby, the occuring structural counterfactual x^{SCF} does not necessarily correspond to the cost-minimal counterfactuals as it is obtained from the *SCM*.

$$\begin{aligned} A^* &\in \mathrm{argmin}_A \, \mathrm{cost}(A, x) \\ \text{s.t. } & f(x^{SCF}) \neq f(x) \\ & x^{SCF} = F_A(F^{-1}(x)) \\ & A \in F \end{aligned} \tag{2.24}$$

2.2.3.8 ***Counterfactual Latent Uncertainty Explanations* (*CLUE*)**

Counterfactual Latent Uncertainty Explanations (*CLUE*) [16] uses a generative model (variational autoencoder with arbitrary conditioning) that considers the classifier's uncertainty and generates counterfactual explanations likely to occur under a data distribution. CLUE aims to find points in latent space z that generate inputs similar to an original observation x but are assigned low uncertainty. The encoding of x to the latent space z is denoted by $\mu_\theta(z|x)$, the decoding by $\mu_\theta(x|z)$. Equation (2.1) shows the loss. $\mathcal{H}$ denotes a differentiable estimate of uncertainty. The pairwise distance metric takes the form $d(x, x^{cf}) = \lambda_x d_x(x, x^{cf}) + \lambda_y d_y(f(x), f(x^{cf}))$ such that we can enforce similarity between uncertain points and CLUEs in both input and prediction space. Algorithm 4 illustrates the counterfactual generation process.

$$L(z) = \mathcal{H}(y|\mu_\theta(x|z)) + d(\mu_\theta(x|z), x) \tag{2.25}$$

Algorithm 4 CLUE. Reproduced from [16]

```
Set initial value of $z = \mu_\phi(z|x_0)$
while loss $\mathcal{L}$ is not converged do
    Decode $x = \mu_\phi(x|z)$
    Use Predictor to obtain $\mathcal{H}(y|x)$
    $\mathcal{L} = \mathcal{H}(y|x) + d(x, x_0)$
    Update $z$ with $\Delta_z \mathcal{L}$
end while
return Decode explanation: $x_{CLUE} = \mu_\phi(x|z)$
```

2.2.3.9 ***F**lexible **O**ptimizable **C**ounterfactuals for **T**ree **E**nsembles* (*FOCUS*)

Flexible Optimizable Counterfactual Explanations for Tree Ensembles (FOCUS) [138] finds counterfactuals for non-differentiable models, e.g., tree ensembles. The method uses a differentiable and probabilistic approximation $\hat{f}$ of the original tree ensemble f. With the probabilistic approximation $\hat{f}$ and a data point x the loss is formulated in Equation (2.26). The distance loss $\mathcal{L}_{dist}$ denotes a user defined metric specifying the distance between the original and the counterfactual instance. The prediction loss $\mathcal{L}_{pred}$ modifies the prediction loss of *W-CF* with a regularizer punishing diverging behavior of the approximation $\hat{f}$ from f. The loss only stays active if the prediction of f has not changed but the gradient based on the differentiable $\hat{f}$: $\mathcal{L}_{pred}(x, x^{cf}|f, \hat{f}) = \infty_{\text{argmax}_y f(x) = \text{argmax}_{y'} f(x^{cf}) \times \hat{f}(x^{cf})}$

$$\mathcal{L} = (x, x^{cf}|f, \hat{f}, d) = \mathcal{L}_{pred}(x, x^{cf}|f, \hat{f}) + \beta \mathcal{L}_{dist}(x, x^{cf}|d) \tag{2.26}$$

For building the differential approximation $\hat{f}$, we refer the reader to [138].

2.2.3.10 ***F**eature **T**weaking* (*FT*)

Feature Tweaking (*FT*) [241] exploits the internals of a tree-based ensemble classifier f. *FT* identifies the leaf nodes where the prediction do not match the original prediction y. Based on those leafes, *FT* identifies the leafes to be activated to obtain the counterfactual prediction (see Equation (2.27)).

$$T_{change} = \left\{ j | i \in T_{leaf} \land \hat{y} \neq \arg\max_y T(y|i) \right\} \tag{2.27}$$

For every T_{change} in a tree ensemble f, *FT* generates a perturbed example per node with an activation difference of least ϵ difference. For every feature threshold ϵ_j involved, the corresponding feature is changed accordingly: $x_j^{cf} = \epsilon_j \pm \epsilon$. Then, *FT* selects the best performing example (i.e., the one closest to the original instance). The result is a perturbed example that was changed minimally to activate a leaf node in T_{change}.

2.3 XAI for Time Series

Most *XAI* methods introduced in the previous section (Section 2.2) are not directly applicable to time series. Some approaches build on the independent feature assumption (e.g., *W-CF* Section 2.2.3.1), while others couple the feature and time domain (e.g., *FO* Section 2.2.2.6), or find limited applicability on time series data as the model architectures utilized on time series differ from imaged or tabular data (e.g., Grad-CAM utilizing 2-D Convolutions). Further, some methods utilize perturbation or replacement functions (e.g., *IG* replacing with a baseline of 0), leading to incorrect assumptions on time series data. Therefore, this section introduces the *XAI* methods and adaptions tailored to the time series domain. We introduce the two feature attribution methods (Section 2.3.1, Section 2.3.4) and the two counterfactual methods (Section 2.3.3, Section 2.3.2) used in the following work as basis or baselines. For more detailes on *XAI* on time series, we refer the reader to the surveys [194, 238].

2.3.1 Temporal Saliency Rescaling

Temporal Saliency Rescaling (*TSR*) [101] is a wrapper around established feature attribution methods (e.g., *SG*, *FO*, *SHAP*) that decouples time and feature importance using a two-step rescaling. TSR relies on calculating the change in feature attribution by iteratively masking time steps $x_{:,t}$ with an uninformative value r (see Algorithm 5 line 1-4) and calculating the feature-relevance score for each feature by computing the total change in saliency values if that feature $x_{i,:}$ is masked (see Algorithm 5 line 5-13). The final attribution is obtained by calculating the product of both scores.

Algorithm 5 Temporal Saliency Rescaling. Adapted from [101].

```
Require: input $x,$ a baseline feature attribution method E(.), a replacement value $r$
 1: for $t \leftarrow 0$ to $T$ do
 2:     Mask all features at time $t$:$\bar{x}_{:,t} = r$, otherwise $\bar{x} = x$
 3:     Compute Time-Relevance Score $\Delta_i^{time} = \sum_{i,t} |E_{i,t}(x) - E_{i,t}(\bar{x})|$
 4: end for
 5: for $t \leftarrow 0$ to $T$ do
 6:     for $i \leftarrow 0$ to $N$ do
 7:         if $\Delta_i^{time} > \alpha$ then
 8:             Mask feature $i$ at time $t$:$\bar{x}_{i,:} = r$, otherwise $\bar{x} = x$
 9:             Compute Feature-Relevance Score $\Delta_i^{feat} = \sum_{i,t} |E_{i,t}(x) - E_{i,t}(\bar{x})|$
10:         else
11:             Feature-Relevance Score $\Delta_i^{feat} = 0$
12:         end if
13:     end for
14:     Compute (time,feature) importance score: $E_{i,t}^{TSR} = \Delta_i^{feat} \times \Delta_i^{time}$
15: end for
16: return $E^{TSR}$
```

2.3.2 Native Guide

Native Guide (*NG*), developed by Delaney et al. [55] proposes using the k-nearest neighbors from the dataset belonging to a different class as native guide ng to generate counterfactuals. Algorithm 6 shows the heuristic used to generate x^{cf}. To perturb the original instance x, *NG* proposes utilizing the semantic meaningful positions of x. The semantic meaningful positions idx are approximated by the feature weight vector of a deep learner f (given by 1D-GradCam). Thereby, x is iteratively perturbed at the positions idx by replacing $x[idx]$ with $ng[idx]$. The maximal window size of the semantic meaningful positions idx are iteratively increased till the prediction switched to the desired counterfactual class y'

Algorithm 6 Native Guide Counterfactual.

```
Require: input $x$, black-box classifier $f$, dataset $D$, desired CF class $y'$
 1: $x^{cf} = x$
 2: $ng \leftarrow$ getNativeGuide$(x, D)$
 3: $w \leftarrow$ getWeights$(f, x)$
 4: $subarray\_length = 1$
 5: while $f(x^{cf}) \neq y'$ do
 6:     $idx \leftarrow$ findMostInfluentialSubarray$(w, subarray\_length)$
 7:     $x^{cf}[idx] = ng[idx]$
 8:     $subarray\_length += 1$
 9: end while
10: return $x^{cf}$
```

Delany et al. [55] report three types of explanations: the plain native guide ng representing the k-nearest-neighbor from the dataset belonging to a different class (see line 2 in Algorithm 6), the native guide with bary centering mixing the original instance x with the native guide ng, and transformation based on the native guide and class activation mapping x^{cf} (see Algorithm 6).

2.3.3 COMTE

Counterfactual Explanations for Machine Learning on Multivariate Time Series Data (*COMTE*), developed by Ates et al. [19], builds counterfactuals in a multivariate setting by perturbing the features of a time series with the help of a heuristic. They adapted the original formulation of Wachter et al. [252] (see Section 2.2.3.1) by replacing the point-wise distance function d with $||A||_1$, where A is a binary matrix indicating if a time series feature is swapped. The distractor time series x_{dist} used to replace a feature in x is, similar to *NG* (Section 2.3.2), chosen via k-nearest neighbors. To solve the problem, Algorithm 7 introduces the random restart hill climbing used to generate *COMTE* counterfactuals throughout this work.

Algorithm 7 Random Restart Hill Climbing. Reproduced from [19]

```
Require: input $x$, black-box classifier $f$, distractor $x_{dist}$, desired CF class $y'$
 1: for $i \in [o, num_{restarts}]$ do
 2:     Randomly initialize $A$
 3:     attempts $\leftarrow 0$
 4:     iters $\leftarrow 0$
 5:     $x^{cf} \leftarrow (I_m - A)x + Ax_{dist}$
 6:     $l \leftarrow L(f, y', A, x^{cf})$
 7:     while attempts $<$ max attempts $\&$ iters $<$ max ters do
 8:         iters ++
 9:         $A_{temp} \leftarrow$ RandomNeighbor$(A)$
10:         $x^{cf} \leftarrow (I_m - A_{temp})x + A_{temp}x_{dist}$
11:         if $L(f, y', A_{temp}, x^{cf}) < l$ then
12:             attempts $\leftarrow 0$
13:             $A \leftarrow A_{temp}$
14:             $l \leftarrow L(f, y', A_{temp}, x^{cf})$
15:         else
16:             attempts $++$
17:         end if
18:     end while
19: end for
20: return $x^{cf}$
```

2.3.4 LEFTIST

Local Explainer For TIme Series classificaTion (*LEFTIST*) by Guilleme et al. (2019) [84] extends LIME (see Section 2.2.2.1 and [191]) for time series classification by proposing a mapping function h segmenting time series into interpretable features and customized perturbation functions (see Algorithm 2 sample_around). The mapping function h assumes prefixed (both the length and the position) shapelets, dividing a time series x into segments $S(x) = \{S_{1:w}, \ldots, S_{t-w:t}\}$ as the interpretable components. Based on these segments, LEFTIST proposes three types of perturbation functions replacing the original function sample_around (see Algorithm 2):

- Linear Interpolation: replace segment $S_i(x)$ with a line $d_{t,i}(x) = a_{t,i}x + b_{t,i}$ going through the last point before $S_i(x)$ and the first point after $S_i(x)$.
- Constant: replaces the removed segment $S_i(x)$ with a constant c. The constant should be a parameter or computed from the time series (average).
- Random Background: assumes the availability of a reference dataset R. An example $r \in R$ is randomly chosen and the removed segment $S_i(x)$ is replaced by $S_i(r)$ (the ith segment of r).

Note, that *LEFTIST* returns the feature importance of each shapelet, instead of each time step.

3 Related Work

In this chapter, we provide research work related to the topic of this work. Section 3.1 addresses *XAI* in Industrial Applications. The following sections address related work to the research problem identified in Section 1.2: (i) Post Hoc Explainable AI (Section 3.2), (ii) the evaluation of XAI methods (Section 3.3), and (iii) the model improvement via human interaction, i.e., Human-in-the-Loop Machine Learning (Section 3.4).

The following chapters are based on:

> "TSEvo: Evolutionary Counterfactual Explanations for Time Series Classification" (2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA) - 2022)
>
> "TSInterpret: A unified framework for time series interpretability" (arXiv - 2022)
>
> "XTSC-Bench: Quantitative Benchmarking for Explainers on Time Series Classification" (2023 22st IEEE International Conference on Machine Learning and Applications (ICMLA) - 2023)
>
> "TSInterpret: A Python Package for the Interpretability of Time Series Classification" (Journal of Open Source Software - 2023)
>
> "Semantic Meaningfulness: Evaluating Counterfactual Approaches for Real-World Plausibility and Feasibility" (Explainable Artificial Intelligence - 2023)

3.1 Industrial Applications of AI

The increasing interest in smart manufacturing [137], combined with the increasing power and accuracy of *AI*-models, resulted in a growing number of non-interpretable ML models to address various industrial problems (e.g., in manufacturing [69], logistics [161] or predictive maintenance [39, 238]). Although application-driven research is popular, many manufacturers still hesitate to deploy *AI* on a larger scale into the manufacturing cycle. Recently, many researchers began surveying the obstacles to the use of *AI* by analyzing existing research with literature reviews from different angles (e.g., predictive maintenance [31, 251], customized manufacturing [254], Industry 4.0 [6, 8], sensor-based sorting [15], manufacturing [43, 113]).

For one, *AI* methods are black-boxes. This lack of transparency is one of the biggest reasons to question the adoption of *AI* models in manufacturing applications, as a failure due to a black-box *ML* model in production can cause significant financial damage and potential harm to workers [6, 32, 43, 76, 102, 113, 223, 164, 254]. While the paradigm of *XAI* allows the opening of such black boxes, the application of *XAI* in industrial use cases (e.g., [32]) has been reluctant. Many *XAI*-algorithms, especially on time series [15, 251], are still an active part of research. For an overview, we refer the reader to the next section, Section 3.2.

While *XAI* is still an active part of research, a wide variety of *XAI* methods is already available; however, finding a good fit for the *AI* model and the domain use case is crucial. To enable such an evaluation, standardized metrics, and benchmarking datasets are necessary [251, 47]. For an overview on *XAI* evaluation, we refer the reader to Section 3.3.

Further, despite opportunities for automation systems, many companies still rely on human workers due to their cognitive and motor skills as well as their domain knowledge [214, 247]. In today's fast-paced environment, production must keep up with environmental changes, leading to higher complexity and more difficult quality control [254]. Obtaining enough data to train models is often difficult due to the complexity of industrial processes, machine varieties, and technical incompatibilities [148]. Performance degradation under data shortage [113] hinders a broader application of *AI*. As workers often obtain domain knowledge with which such tasks are easily solved, integrating such domain knowledge into *AI* models is still an important active research problem [8, 32, 47, 164, 251]. For a full overview, we refer the reader to Section 3.4.

3.2 Post-Hoc XAI for Time Series

Advancements in the performance of neural networks on various tasks are significant drivers of the development of *XAI* methods. Specifically, the development of post-hoc methods received increasing attention as they do not interfere with the predictor's design choices and predictive capabilities. While research on *XAI* from both a theoretical [2, 60, 81, 91] and application perspective [144, 239, 251] has been reviewed in many surveys and their usage facilitated in unified implementations (see Table 3.1), most research on *XAI* focuses on image, tabular, and textual data. While for *XAI* on tabular, image, and textual data, unified implementation facilitates the applicability and comparability of methods (see Table 3.1), time series remain neglected. Apart from the libraries wildboar [204], OmniXAI [265], and H2O [87], no libraries provide explainability methods for time series. The support of OmniXAI [265] is restricted to anomaly detection, univariate time series and includes only non-time-series-specific explainability methods. Wildboar [204] focuses on temporal machine learning (classification, regression, and anomaly detection) and provides a limited number of (counterfactual) explainability methods as additional features.

For one, this limited accessibility of *XAI* for time series is due to the lack of need for research on the topic until recently. A first detailed survey for *XAI* on time series only dates back to 2021 [194], as time series classification has been considered one of the most challenging problems in data mining for the last two decades [264, 68]. Only with the rising data availability and accessibility (e.g., provided by the UCR / UEA archive [21, 51]), hundreds of (deep) time series classification algorithms were proposed, sparking interest in *XAI* for time series. However, the time component impedes the usage of existing *XAI* methods as they fail to produce reliable and accurate feature importance and do not consider time-ordered input [101]. Often, *XAI* methods assume independent features [252], the definition of “interpretable features” [139, 84, 268, 229] or a spatial-feature relationship [212] that is not necessarily given on time series. Therefore, a significant fraction of research focuses on the adaptability of post-hoc saliency and instance-based methods to the time series domain (e.g., LEFTIST based on SHAP / Lime [84], Temporal Saliency Rescaling for Saliency Methods[101]). Often, methods are adapted by introducing time series transformers (e.g., slidings windows or fourier transformation on the frequency domain) to define “interpretable” features and manipulate the original time series with a distractor time series [84, 158]. Table 3.2 summarizes post-hoc *XAI* methods for times series, their code availability, and evaluation settings.

Table 3.1: Overview of available *XAI* libraries with respect to the data types, covered tasks, explanations scopes and explanation outputs. Only libraries with a commit in the last year were taken into account. Inspired and updated from [95].

		AIX360 [18]	ALIBI [117]	Anchor [190]	Captum [118]	Carla [174]	DALEX [23]	DeepExplain [13]	Dice [157]	ELI5 [169]	H2O 12	iNNvestigate 13	InterpretML [167]	Lucid [233]	OmniXAI [265]	Pytorch-cnn-vizualizations [170]	Shap [139]	Skater [168]	Tf-Explain [151]	TorchRay [71]	What-if [260]	Wildboar [204]
Data	Tabular	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓				✓	✓			✓	
	Text	✓	✓	✓	✓			✓		✓							✓	✓			✓	
	Image	✓	✓		✓			✓		✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	
	Time										✓											✓
Task	Classification	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Regression	✓	✓		✓	✓	✓	✓	✓		✓		✓	✓	✓		✓	✓			✓	✓

Table 3.2: Overview of *XAI* for time series with code availability and evaluation setting. Extended and updated from [97].

		Availability				Method Type				Evaluation Settings															
		Available	GitHub	API	PyPI Ready	Gradient Based	Perturbation Based	Instance Based	Other	UCR/UEA	Synthetic	Other	Faithfulness	Robustnes	Reliability	Complexity	Other	Integreted Gradients	GradCam	Feature Occlusion	Shap	Lime	Wachter CF	Other	TS-Baseline
Explainer	AB-CF [131]	✓						✓		✓							✓						✓		✓
	COMTE [19]	✓	✓					✓				✓	✓	✓		✓							✓		
	DEMUX [59]	✓	✓				✓			✓															
	DynaMask [46]	✓	✓	✓			✓				✓	✓					✓			✓				✓	✓
	FIT [242]	✓	✓	✓							✓	✓	✓		✓			✓						✓	
	GLACIER [257]	✓	✓					✓		✓							✓								✓
	LASTS [225]	✓	✓	✓		✓	✓	✓	✓	✓			✓	✓			✓				✓				✓
	LEFTIST [84]	✓					✓			✓			✓			✓									✓
	LimeSegment [220]	✓	✓		✓		✓			✓			✓	✓											
	NG [55]	✓	✓					✓		✓							✓				✓		✓		
	MP [150]	✓	✓				✓	✓		✓				✓								✓		✓	
	Neves [162]						✓					✓	✓				✓				✓	✓			
	SETS [22]	✓						✓				✓					✓								✓
	TimeShap [26]	✓	✓	✓			✓					✓												✓	
	TimeXPlain [158]	✓	✓				✓			✓			✓				✓							✓	
	TSInsight [217]						✓	✓		✓			✓					✓	✓	✓					
	TSR [101]	✓	✓			✓	✓				✓				✓			✓	✓	✓	✓	✓		✓	
	TSViz [218]	✓	✓	✓		✓	✓					✓	✓	✓		✓									

3.3 Quantifying XAI

As seen in the previous section, an increasing number of methods providing a variety of explanation types (e.g., example-based methods like counterfactuals [252], or feature attribution methods like SHAP [139]) on different data types (e.g., images [221], tabular data [191]) are available. However, measuring the performance of such explanation methods quantitatively is still challenging. Metrics for the evaluation of *XAI* depend on the scope of the method, the desired properties to be evaluated, and the underlying data type. Only some metrics are applicable independent of the scope. Amongst others, this is the case for faithfulness [11, 29, 47, 89], robustness[11, 29, 47, 89], complexity [29, 89, 163, 174] and reliability [17]. Faithfulness quantifies the consistency between the behavior of the classifier and the explainer. Robustness measures how sensitive an explainer is to minor (irrelevant) input changes. Complexity measures the number of features used in an explanation under the assumption that a smaller number of used features is better understood by human cognition, and reliability quantifies the explanation coverage with respect to a given ground truth. While most authors propose and utilize some or similar evaluation properties (see Table 3.2), there are no generally agreed upon metric definitions measuring the quality of explanations, and comparisons between different implementations and metrics are difficult (e.g., [39, 134, 209]). Furthermore, depending on the explanation type, different additional properties are essential.

For *IB* methods often the distance to the original instances is added as a measure for proximity and sometimes also the complexity of the explanation. Following the literature on counterfactuals, the proposed explanations should also be sparse (i.e., entails limited features [105, 246]), realistic (i.e., is near the training data [246] - also called data manifold adherence), actionable (i.e., contains mutable features [246]), feasible (i.e., adhering to real-world relations [140]) and plausible (i.e., perceived as sensible by human users [222]) to be useful in practice. Counterfactuals are mostly evaluated by quantifying the desirable properties of a counterfactual explanation (e.g.,[174]): validity measuring whether a counterfactual explanation was able to flip the classifier's decision [61, 106, 140, 138, 174], proximity as the user-specified distance (e.g., mean absolute error) between the original instance and the counterfactual [106, 138, 140, 174], sparsity quantifying the number of changes made to the input [106, 174] and diversity, a measure for the similarity of the different counterfactual explanations when multiple explanations are generate [246]. Sometimes constraint feasibility is also measured by checking user-given constraints [140, 174]. Laugel et al. [122] suggest two measures that are applicable to most counterfactuals to quantify feasibility proximity (i.e., whether a counterfactual is a local outlier) and concreteness (i.e., whether a counterfactual is connected to other correctly classified observations from the same class). However, their measures are data-driven and therefore not able to test coherence with real-world relations. Other feasibility metrics are only applicable to specific counterfactual generation methods (e.g., confidence lower bound for probabilistic recourse [106], interpretability score [140, 245], causal-edge score in case the true SCM is known [140]). There exists no agnostic metric that is applicable to all counterfactual generation methods to evaluate the fulfillment of real-world relations.

While benchmarking frameworks for standardizing the quantification of *XAI* methods [89] and especially counterfactual methods [174] have been proposed, the evaluation of the causality of explanations and time series explainer is still an open issue. Using metrics from traditional frameworks (e.g., [89]) can lead to erroneous assumptions in the time series domain. Similar to the explanation methods implemented in the different explainability frameworks, transferring metrics to the time series domain is complex. Often, metrics are directly transferred from image classification [217]. However, many metrics rely on replacing input parts with uninformative information (e.g., to measure if the explanation method shows the same behavior as the classifier) or on comparisons to segmentation masks (e.g., to measure if the explanation method was able to localize relevant features). While providing uninformative features is trivial for images by replacing parts of an image with black or white pixels [29], replacing features with standard techniques (class means or zeros) might be relevant information in time series. Table 3.2 shows the various evaluation settings used in the time series domain. Most methods are evaluated with specific metrics and benchmarked against non-time series specific explainers.

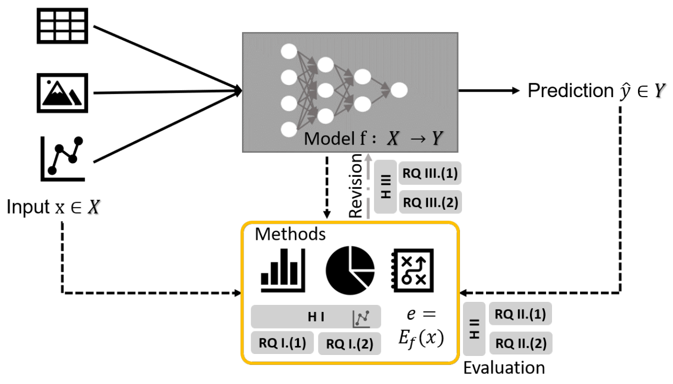
3.4 Interactive Machine Learning and Continuous Learning in Applied Research

The need to involve human experts in the model training process for industrial use cases has been around for over a decade (e.g., [65, 67, 196, 258] and Section 3.1). It has sparked research on interactive and active learning [111]. While in traditional machine learning, users have no control over the systems behavior, interactive and active learning focus on incorporating human feedback into the training process by letting humans interact with the model's output. While such a paradigm allows the machine learning model to learn from human inputs, the understandability of such models is still an open problem [67]. Patterns used to classify a problem are not accessible or understandable and cannot be used to evaluate or improve the model. The emerging paradigm of *XAI* tackles opening such black-boxes by including mechanisms to visualize the behaviors of machine learning models (see Section 2.2). While XAI offers the end user explanations on why a model came to a specific conclusion, the interaction or improvement of such conclusion is not in the scope of previous work. To enable such interactions, Teso and Kersting [237] propose *EXplanatory Interactive Learning* (*XIL*), which enables richer interactions with humans by allowing user feedback on visual explanations. *XIL* combines user supervision (active learning) with model explanation to interactively revise the model's learning process. The first methods dating back to 2017 [195] can generally be divided into loss function-based [195, 213] and data augmentation-based [237] methods. Loss function-based methods allow users to penalize incorrect feature attributions by adding a regularization term. Augmentation-based methods use the obtained explanation feedback to resample the input, aiming to make *AI* models invariant at specific features. The ability to revise confounded models with *XIL* has already been shown in multiple works [73, 192, 195, 210, 237] and successful applications (e.g., scientific datasets [210] or quality inspection [196]). However, in all applications, the model is expected to learn interactively from scratch; pre-training and improving models, continuous learning, or enhancement of models are not considered. For full surveys on explanations in interactive machine learning, we refer the reader to [236] and [73].

Continuous learning with human interaction has not been addressed so far, as training with new information (possibly from a different data distribution) often leads to forgetting the previously learned knowledge and a significant performance decrease on the previously learned tasks. *Continuous Learning* (*CL*) proposes adapting prediction models to possible distribution changes by constantly incorporating new knowledge after deployment by a) regularizing network parameters or activations (e.g., [132]), b) increasing the model capacity for each new task (e.g., [201]), or c) keeping in training data examples from previous tasks (e.g., [136]). *CL* has been evaluated in applied research areas like medicine for image segmentation or classification and pattern recognition in time series data [116, 179]. Work on application to manufacturing and production is rare and has just begun recently (e.g., [147, 148, 234, 235]). However, *CL* is highly relevant, especially in manufacturing, due to fast-changing environments (e.g., manufacturing a new product [234]). Limitations include the insufficient performance of continuous learners in the task-free learning context [270] and debugging models in real-time without accessing production data and logs [25].

Part II

Explainable Machine Learning Methods for Time Series



This part deals with challenge 1, **the missing support and accessibility of *XAI* for time series data**, where we deal with research questions about the facilitation and improvement of *XAI* in the time series domain. The first section (Chapter 4) develops a unified framework for standardizing and facilitating access to existing time series explainers. The second section (Chapter 5) builds on the first section and develops a new and improved counterfactual explainer for uni- and multivariate time series classification based on time series transformation.

4 A Framework for post-hoc XAI on Time Series

While a wide selection of explainability methods with easy-to-adopt implementation are available on image and tabular data (see Table 3.1), *XAI* on time series is still a relatively new research field and, therefore, lacks standardization (see Table 3.2). Various *XAI* methods for time series classification have been developed only recently ([194]). However, the proposed methods often lack (a) open code (e.g., [217]), (b) an easy-to-use interface (e.g., [101]), or (c) visualization (e.g., [84]), making the application of those methods inconvenient and thereby hindering the usage of deep learning methods in safety-critical scenarios ranging from health care [144] to industrial scenarios like predictive maintenance [251], where time series data are ubiquitous.

This chapter addresses these obstacles by answering RQ I.(1):

RQ I.(1) How can the cross-domain application of explanation methods for deep learning based time series classification be facilitated?

This chapter proposes *TSInterpret*, a framework allowing the implementation of various time series algorithms by providing a unified framework for time series interpretability, including unified visualizations for the implemented algorithms with the ultimate goal of "interpretability" – i.e., the ability to support users' understanding and comprehension of the model decision-making process and predictions – for time series classification. Explainability algorithms, implemented in *TSInterpret*, provide users with a (fractal) understanding of model decisions. For users to understand the whole decision process, often multiple explainability algorithms providing different views on the model and its prediction are necessary. *TSInterpret* facilitates access to various types of explanation methods to provide model interpretability.

All following sections are based on "TSInterpret: A Python Package for the Interpretability of Time Series Classification" (Journal of Open Source Software – 2023) and "TSInterpret: A unified framework for time series interpretability" (arXiv – 2022). To answer RQ I.(1), we first specify the problem in more detail (Section 4.1), then we describe the framework (Section 4.2). Section 4.3 describes the implemented algorithms and Section 4.4 provides an outlook.

4.1 Problem Formalization

Imagine a decision support system for engine damage diagnostics where motor behavior is supposed to be classified as "normal" or "abnormal" based on engine noise. Such a classification task can be a sensitive field due to incurring costs, and, in the worst case, health risks to human operators arise in cases of misclassification. For example, if abnormal motor behavior is classified incorrectly as normal, motor operators might be exposed to health risks, e.g., due to excessive heat from a nearly broken machine. Therefore, decisions need to be taken carefully. Explaining the utilized models is crucial to allow operators to make such data-driven decisions with the help of machine learning. It is relevant not only if the data show some significant behavior, but also why or which indications exist for such behavior.

Such a problem can be formulated in line with the proposed conceptual framework (Figure 1.2) as a supervised classification problem $f : \mathcal{X} \rightarrow \mathcal{Y}$ with parametrization θ, where $\mathcal{Y} \in \{\text{normal}, \text{ abnormal}\}$. To shed light on the inner workings of a model f, we utilize an explainer E_f. Depending on the target audience of the explanation obtained from the explainer E_f, different explainer types are necessary. For example, an explainer producing simplified visualizations of the model's functionality, such as marking relevant time steps for classification, might be particularly relevant for an AI expert for error evaluation of a model. However, during operations, an explainer generating simple and intuitive explanations without technical understanding might be more helpful for the *AI* systems' end users. While explainers E are already available for most use cases, e.g., counterfactuals [55] to provide contrastive explanations for the domain expert or feature attribution methods [101] for model debugging, they are not easily used because of missing standardization and lack of applicability (see Table 3.2).

To improve adaptability, enable the usage of *XAI* on a broader range, and encourage the use of post-hoc *XAI* throughout the data science lifecycle from model design to deployment and production, we derive the following requirements:

- $\mathbf{R_{4.1}}$: The access to the explainers E need to be unified by a common API, ideally orientated at known Design Principles for easy usage.
- $\mathbf{R_{4.2}}$: The output, the explanation e, should always take on the same structure to enable unified post-processing.
- $\mathbf{R_{4.3}}$: The explanation e should be presented with appropriate default visualizations. Similar explanation types should be visualized similarly for better comparability and consistency.

4.2 Framework Design

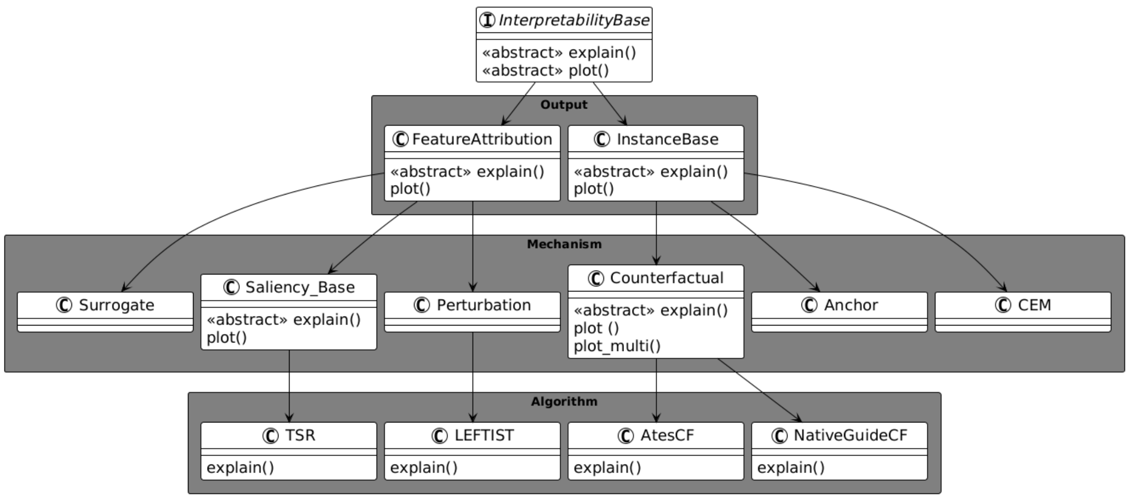


Figure 4.1: Structure of TSInterpret.

Figure 4.1 shows the structure of *TSInterpret*. Based on the taxonomy of *XAI*, the tool is divided into layers. The first layer, the "Output Layer", focuses on the different explanation output types (see Section 2.2.1.3). Depending on the explanation type, different visualizations and outputs are necessary (e.g., a feature attribution method might be visualized as a heatmap; a counterfactual is a second time series to plot). The "Mechanism Layer" refines the "Output Layer" by considering the working mechanisms of an *XAI* algorithm. Many *XAI* algorithms are built around similar approaches, therefore, this layer implements multi-use mechanisms to prevent code duplication and diverging implementation (e.g., most counterfactual methods have a similar basis; see Section 2.2.3). In the "Algorithm Layer", the final *XAI* algorithm is implemented.

Addressing $\mathbf{R_{4.1}}$, the whole architecture and API design are orientated on the scikit design paradigms consistency, sensible defaults, composition, nonprofilarity of classes, and inspection ([33]).

Consistency All implemented objects share a consistent interface. Every explanation method inherits from the interface `InterpretabilityBase` to ensure that all methods contain a method `explain`, a `plot` function, and the corresponding minimal method signature ($\mathbf{R_{4.2}}$). The `plot` function is implemented on the level below based on the output structure provided by the explanation algorithm to provide a unified visualization experience (e.g., in the case of *FA*, the plot function visualizes a heatmap on the original sample). If necessary, those plots are refined by the "Mechanism Layer". This is necessary to ensure suitable representation (see $\mathbf{R_{4.3}}$) as the default visualization can sometimes be misinterpreted (e.g., the heatmap used in the plot function of `InterpretabilityBase` allows positive and negative values, while *`TSR`* is scaled to $[0, 1]$. Using the same color pattern for both scales would lead to a high risk of misinterpreting results while comparing *`TSR`* with `LEFTIST`). The `explain` function is implemented on the method level.

Sensible Defaults *TSInterpret* provides reasonable defaults for most parameters by providing the default parameterization for each method from the designated papers. Those parameters can easily be changed if needed by providing alternative values during model instantiation.

Composition Many explanation methods for time series classification are based on already existing methods for tabular, image, or text data (e.g., amongst others *TSR* [101] is based on *SHAP* [139]). Whenever feasible, existing implementations of such algorithms are used (e.g., SHAP [139], captum [118], or tf-explain [151]).

Nonprofilarity of classes *TSInterpret* implements the explanation algorithms as custom classes. Datasets, instances, and results are represented as NumPy arrays, Lists, or Tuples, instead of classes. For instance, the counterfactual method returns a Tuple of the counterfactual time series and the label (list, int). Hyperparameters are regular strings or numbers.

Inspection *TSInterpret* stores and exposes the parameters of the explanation algorithms as public attributes. In some methods, parameters have a significant impact on the obtained results. Making those parameters publicly available through attributes facilitates experimenting with hyperparameters.

Listing 4.1 shows the workflow of the library in a coding sample. Given a trained machine learning model and an instance to be classified: First, the desired interpretability method is imported (line 1) and instantiated (line 2), followed by explaining the instance (line 3), and finally, the generation of the plot (line 4).

Code Listing 4.1: API Usage Example.

```python
from TSInterpret.InterpretabilityModels.Saliency.TSR import TSR

xai_model = TSR(ml_model, item.shape[-1], item.shape[-2])
exp = xai_model.explain(item, labels = pred_label, TSR = True)
xai_model.plot(item,exp)
```

4.3 Algorithm Overview

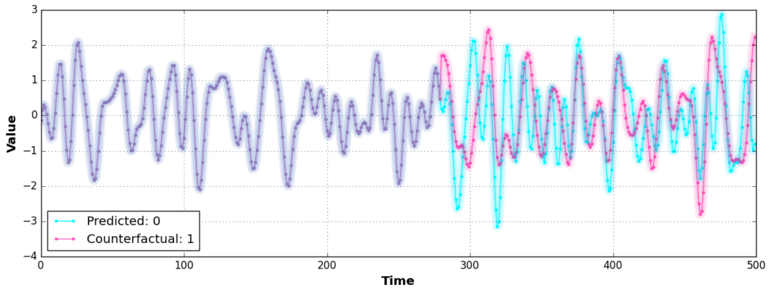
TSInterpret supports various time series explainability types, resulting in explanations and visualizations for various use cases. The current version of the library includes various types of explainability algorithm covering the variety of output types and working principles introduced in Section 2.2.1. *TSInterpret* includes the citation-wise most popular, foundational (post-hoc) explainability algorithms for time series listed in Table 4.1[1]. Their implementation in *TSInterpret* is based on code provided by the algorithms' authors, which is adapted to the framework and extended to ensure the availability of multiple model backends (e.g., PyTorch or TensorFlow).

	Method	**Explanations**	**Uni**	**Multi**	**Dataset**
Original	*NG* ([55])	*IB*	✓	✗	✓
	COMTE ([19])	*IB*	✗	✓	✓
	LEFTIST ([84])	*FA*	✓	✗	✓
	TSR ([101])	*FA*	✓[2]	✓	✗
Added	*TSEvo* ([93])	*IB*	✓	✓	✓
	SETS ([22])	*IB*	✗	✓	✓

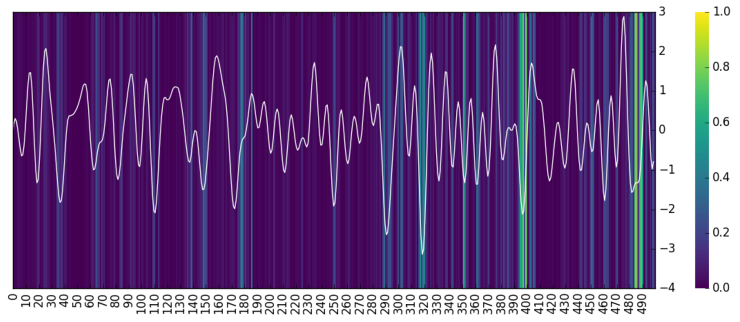
Table 4.1: Explainability Methods implemented in *TSInterpret*. For more details on the working principles of those methods, refer to Section 2.3.

Returning to the use case described in Section 4.1, we train a 1D-conv ResNet on FordA [51] to differentiate between two motor settings. Further, to showcase the multivariate algorithms, we use the CNC-Machines [240] dataset to detect anomalies based on vibration sensors. Figure 4.2 and Figure 4.3 show explanations obtained with *TSInterpret* for the 1D-conv ResNet on the first item of the test set for the univariate FordA dataset (Figure 4.2) and the multivariate CNC-dataset (Figure 4.3). With the help of the explanations shown in Figure 4.2, domain experts can gather evidence on the functionality of the ResNet classification on FordA. The feature attribution method TSR (Figure 4.2b) identifies sections with positive / negative influence on the current prediction. In this case, the last third of the time series has the most considerable impact on the current prediction being a normal motor behavior. If a domain expert wants to know why the motor shows a normal instance and not an abnormal one, conclusions can be drawn from the counterfactuals produced by *NG*. Compared to the original time series (in blue in Figure 4.2a) classified as normal, the classification is changed if the fluctuations in the last part change (pink line in Figure 4.2a). Figure 4.3 shows explanations for the CNC-machine vibrations dataset. Suppose a worker wants to know how the original instance is predicted as normal (blue) instead of abnormal. In such a case, the practitioner applies *COMTE* to generate a counterfactual in the class direction of abnormal, resulting in a different vibration behavior on the y- and z-axis (see Figure 4.3a). If the domain expert is only interested in seeing the most important features, *TSR* (Figure 4.3b) can be applied. With the help of explanations, domain experts can gather evidence for the predicted classification and gain trust in the system ([12]). In general, the visualization of the counterfactuals generated with *COMTE* and NUN-CF have a similar style, although NUN-CF visualizes univariate and *COMTE* multivariate data. Note that *COMTE* only visualizes the changed features. For the feature attribution methods LEFTIST and *TSR*, the visualizations have similar styles but different color maps. This is necessary as *LEFTIST* returns a positive and negative influence (range $[-1, 1]$), while *TSR* returns normalized time slices and feature importances (range $[0, 1]$).

[1] Details on the methods are given in Section 2.3.

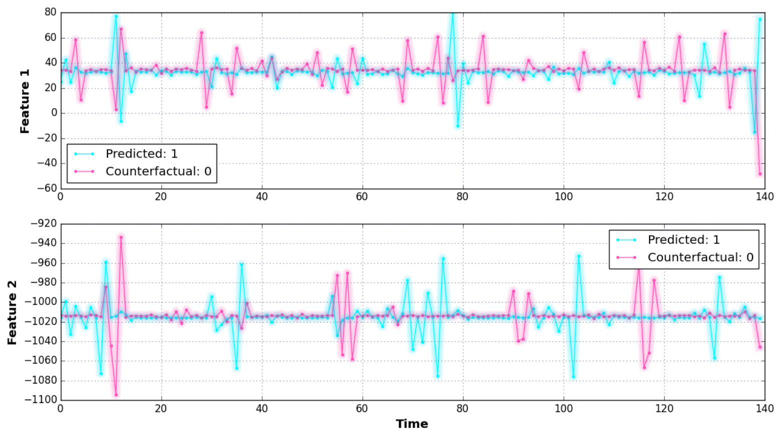


(a) NUN-CF.

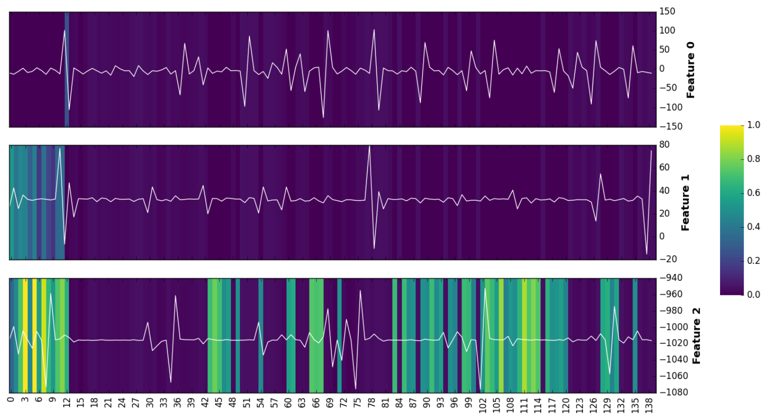


(b) TSR.

Figure 4.2: Explanations obtained with TSInterpret on the univariate FordA dataset.



(a) CoMTE.



(b) TSR.

Figure 4.3: Explanations obtained with *TSInterpret* on the multivariate CNC Machines dataset.

The easy-to-use interface of *TSInterpret* allows the out-of-box usage of explainability models on time series data for different use cases, time series types, and classification models. Further, the framework can be easily extended by inheriting from either `InterpretabilityBase` or one of the more customized classes (e.g., `Feature Attribution` or `InstanceBased`). All classes below the Output layer also come with a default plot function to match the visualizations obtained by the already implemented algorithms.

4.4 Summary and Discussion

This section focused on answering RQ I.(1) facilitating the cross-domain access to explainers for (deep) time series classification. We proposed *TSInterpret*, a framework standardizing the access of time series explainers. *TSInterpret* provides a cross-backend unified API for the explainability of time series classifiers, enabling explanation generation with just three lines of code. The design of *TSInterpret*, according to the scikit-learn principles in Section 4.2, allows the framework to be easily extensible by inheritance and provides a simple API design. Through the open source availability of *TSInterpret*, *TSInterpret* advances the democratization of *XAI* methods in the time series domain by providing accessibility, offering a publicly available code basis and allowing changes, improvements, and advances in the framework from the community.

In the first release of *TSInterpret*, the research gap in counterfactual generation for both uni- and multivariate time series data became apparent. No existing method was able to cater to both time series types. Additionally, the developed methods relied heavily on perturbing original instances with a reference dataset, missing out on frequency changes and concept changes between classes. This problem will be revisited in RQ I.(2) and addressed in Chapter 5.

While *TSInterpret* provides standardized access to various explainers, quantifying the explainer's capabilities and making them comparable to create an easily usable and standardized benchmarking standard for the research community is still an open issue. Utilizing the *XAI*-method standardization provided by *TSInterpret* to enable a thorough benchmarking of existing methods will be revisited in Chapter 6.

Further, *TSInterpret* omits ante-hoc explainability. However, explainability on recurrent neural networks primarily relies on ante-hoc mechanisms like attention mechanisms, which include explainability in the model design ([194]). Therefore, for now, no explainability mechanisms tailored to recurrent neural networks are implemented. Also, the current framework version focuses on the explainability of the time series classifier. Research on explainability for time series forecasting and regression is still limited due to their higher complexity. Depending on the context, training well-performing (deep) forecasting models is still an open problem, as argued in Section 1.5 statistic based models not in need for *XAI* still dominate in forecasting. Explainability for regressional problems is still an open issue independent of the data type that is often addressed by reformulating regression tasks as a multiclass classification problem or by utilizing thresholds (e.g., [130]). Specific *XAI* methods for regressional tasks are still an open issue. However, the framework can be easily extended if reliable explainability algorithms become available.

5 Counterfactual Explanations for Uni- and Multivariate Time Series

While a wide selection of explainability methods on image and tabular data is available (see Section 3.2), most advances in time series classification explainability are restricted to attention-based and model-specific methods [194]. Only highlighting the time step or feature importance can be hard to grasp, especially on time series, since it is often unclear what behavior at a specific time step led to the prediction. Providing a contrastive explanation showing the minimal time series change to achieve a different prediction is, according to Miller [153], besides selectivity (i.e., only some causes of the prediction are shown), sociability (i.e., interactiveness), and exclusion of probability, an essential factor for human-understandable explanations. A specific class of algorithms that can provide contrastive explanations are counterfactuals (see Section 2.2.3). Inspired by counterfactuals for tabular data (Section 2.2.3.1-Section 2.2.3.9), previous work on time series focuses on adapting the *W-CF* optimization problem and approach to the time series domain by providing custom perturbation functions. On univariate time series, Delany et al. [55] propose greedy or gradient-based time slice perturbations (see Section 2.3.2). In a multivariate setting, Ates et al. [19] use full feature perturbations (see Section 2.3.3). Both proposed perturbation functions only consider perturbations with an unmodified distractor from a reference dataset, missing out on potential frequency changes. Further, the proposed perturbation functions are specific to uni- or multivariate data. None of these approaches combines various perturbation and manipulation techniques for time series.

Based on these observations, we aim to answer RQ I.(2):

RQ I.(2) How can we enhance counterfactual time series explainers by including time series based transformers?

To answer this question, this chapter presents *TSEvo*, an approach to counterfactual generation on uni- and multivariate time series. *TSEvo* combines the time series transformation approaches proposed in [84] and [158] with a genetic algorithm for perturbation, enabling the generation of model-agnostic counterfactual explanations in both uni- and multivariate classification. Thereby, in contrast to [55], [84], and [158], no advanced decision regarding the transformer has to be made.

This section mainly builds on "TSEvo: Evolutionary Counterfactual Explanations for Time Series Classification" (2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA) - 2022). First, Section 5.1 formalizes the problem, then Section 5.2 introduces our approach *TSEvo* by reformulating the problem as Multi-objective Problem and approximating a solution with an *NSGA-II*-based search. Section 5.3 evaluates and compares *TSEvo* to other counterfactual approaches for time series. Finally, Section 5.4 provides a summary and discussion.

5.1 Problem Formalization

We study a supervised time series classification problem in line with the overarching problem setting described in Section 1.2. Let x be a uni- or multivariate time series, T is the number of time steps, and N is the number of features. Then, $x_{i,t}$ denotes the input feature i at time t. Similarly, let $X_{:,t} \in R^N$ and $X_{i,:} \in R^T$ be the feature vector at time t and the time vector for feature i, respectively. y denotes the output, and $f : X \rightarrow Y$ a classification model returning a probability distribution vector over classes $y = [y_1, \ldots, y_C]$, where C is the total number of classes (i.e., outputs) and y_j the probability of x belonging to class j. The classification model f is seen as a black-box – i.e., no access to the inner workings of a model is available. Only the result $\hat{y}$ is observable.

The goal of counterfactual methods is, given a time series x and a classifier f, to provide an explanation via counter examples, allowing humans to understand why classifier f chose class y for data point x and not a counterfactual class y^{cf} [252]. To allow such understanding, we assume that for each x, a counterfactual sample x^{cf} can be computed, that is close to x, but classified differently $y \neq y^{cf}$. The resulting x^{cf} is supposed to be a proximate ($\mathbf{R}$) [157], sparse ($\mathbf{R}$) [157], and plausible ($\mathbf{R}$) [124] adaption of x. Proximity refers to the distance between the query instance x and the counterfactual instance x_{cf}, calculated as a distance measure d between x and x^{cf}. Sparsity refers to the number of feature changes between x and x_{cf}. A plausible adaption indicates that the resulting x^{cf} is in distribution with the available data D.

$\mathbf{R_{5.1}}$: $min(d(x, x^{cf})), s.t. f(x) \neq f(x^{cf})$

$\mathbf{R_{5.2}}$: $min(\sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{1}_{|x_{i,t} - x_{i,t}^{cf}| != 0}), s.t. f(x) \neq f(x^{cf})$

$\mathbf{R_{5.3}}$: $x^{cf} \sim D, s.t. f(x) \neq f(x^{cf})$

5.2 Approach

Around the requirements specified in Section 5.1, we formulate a multi-objective counterfactual search problem (see Section 5.2.1). We solve the problem by using a to counterfactual search adapting *Non-dominated Sorting Genetic Algorithm* (*NSGA-II*) (see Section 5.2.2).

5.2.1 Multi-objective Optimization Problem

Combining the desired properties $\mathbf{R_{5.1}}$ and $\mathbf{R_{5.2}}$ with a function for guiding the output distance away from the original classification leads to multi-objective problem O. Equation 5.1 shows the minimization problem. O_1 is derived from $\mathbf{R_{5.1}}$ by applying mean absolute error as distance function d [252, 157]. O_2 is consistent with $\mathbf{R_{5.2}}$ and O_3 denotes the output distance, maximizing the output distance on a target class l. If no target class is chosen, the second-highest class probability is designated as the target.

$$\begin{aligned}
min\, O(x) &:= (O_1(x, x^{cf}), O_2(x, x^{cf}), O_3(x^{cf})) \\
&s.t. f(x) \neq f(x^{cf}) \\
O_1(x, x^{cf}) &= \frac{1}{N * T} \sum_{i=1}^{N} \sum_{t=1}^{T} \left| x_{i,t} - x_{i,t}^{cf} \right| \\
O_2(x, x^{cf}) &= \frac{1}{N * T} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{1}_{|x_{i,t} - x_{i,t}^{cf}| \neq 0} \\
O_3(x^{cf}) &= 1 - f(x^{cf})_l
\end{aligned} \tag{5.1}$$

5.2.2 Counterfactual Search

TSEvo is based on the Non-Dominated Sorting Genetic Algorithm (NSGA-II) [52], a well-established algorithm for multi-objective optimization. Most other multi-objective algorithms either rely on the additional computation of an indicator called indicator-based methods (e.g., SMS-EMA [66], IBEA [274]) or are highly dependent on the shape of the pareto front like decomposition-based methods (e.g., MOEA/D [184], NSGA-III [54]) [100].[1]

Algorithm 8 shows our optimization algorithm for the Multi-Objective Counterfactual Search Problem. The inputs of our approach are a query time series x, a classification model f, and a data basis D. Based on the input instance x, a population p consisting of P individuals $p = \{I_1, \ldots, I_P\}$ is initialized. Each individual I is a candidate solution for the optimization problem O and, therefore, a possible counterfactual x^{cf} to x based on the model f. This population p is evaluated according to the multi-objective problem stated in Section 5.2.1 and ranked with Non-Dominated Sorting according to NSGA-II [52]. The assigned ranks are used as the primary criterion in the tournament selection. Thereby, two individuals are compared according to their rank. If they have the same rank, the crowding distance is used as a secondary criterion to retain the individual lying in the less crowded region to maintain the population's diversity. The selected individuals λ are recombined with a crossover probability cx to exploit well-performing individuals and mutated with a probability of $mutpb$ to explore the search space. The resulting offsprings λ differ from the original population p. The offsprings λ are combined with the population p and evaluated, selected, recombined, and mutated again. The algorithm stops if the desired number of generations is met.

5.2.3 Initialization

Each individual $I_i \in p$ denotes a possible counterfactual time series. In the first population, these possible counterfactual solutions are set to the original input x. Further, every individual I_i has a time slice window size w_i assigned. In the initial population p, an individual's window size w_i is a randomly drawn integer from the interval $[1, 0.5 * |x|]$ and is optimized throughout. $|x|$ denotes the length of the time series x. The window size w_i is necessary to divide a candidate solution I_i into its segments S_i. By keeping a different window size for each individual, we explore various dimensions of possibly discriminative information.

[1] For details on the algorithm selection, we refer the reader to Appendix A.1.1.

Algorithm 8 Basic Optimization Algorithm.

Input Population Size P, Generation G , Original x, Dataset D
Output Best Counterfactual x^{cf}

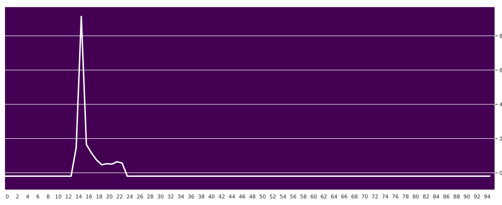
```
1: $p \leftarrow$ initializePopulation$(x)$
2: $R \leftarrow$ calculateReferenceSet$(D)$
3: $G \leftarrow$ maximal number of generations
4: evaluate$(p)$
5: $p \leftarrow$ selectNSGA$(p)$
6: for $g \in \{0, 1, \ldots, G\}$ do
7:     $\lambda \leftarrow$ selTournament$(p)$
8:     for $j$ in 1,...,$(|\lambda| - 1)$) do
9:         if $random() < cx$ then
10:            $\lambda_{j-1},\lambda_j \leftarrow$ crossover$(\lambda_{j-1},\lambda_j)$
11:        end if
12:    end for
13:    for $j$ in 0,...,$(|\lambda| - 1)$ do
14:        if $random() < mutpb$ then
15:            $\lambda_j \leftarrow$ mutate$(\lambda_j)$
16:        end if
17:    end for
18:    evaluate$(\lambda + p)$
19:    $p \leftarrow$ selectNSGA$(\lambda + p)$
20: end for
21: return $x^{cf} \leftarrow \min(\text{evaluate}(p))$
```

5.2.4 Segmented individual

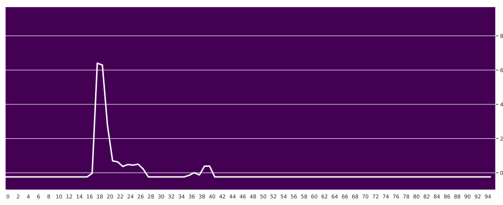
Time series interpretation of single data points is complex. Therefore, it has been proposed that discriminative information is contained in contiguous subsequences of the series [267, 127]. Our operators define contiguous subsequences of an individual I_i by segmenting according to window size $I_i.w$. We denote this segmentation by $S(I_i)$, with $S(I_i) = \{S_{1:w}, \ldots, S_{t-w:t}\}$, and $S_{1:w} = I_{:,1:w}$ where w is the window size and t the time series length.

5.2.5 Crossover

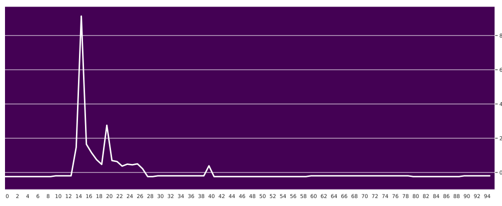
The intuition of crossover is to exploit the search space by combining high-performing individuals [53]. The crossover operator is inspired by uniform crossover and operates on the segmented individuals $S(\lambda)$. Two previously selected individuals, λ_i and λ_j are segmented according to Section 5.2.4 with window size w of the better-performing individual. The segmented individuals $S(\lambda_i)$ and $S(\lambda_j)$ are recombined by uniformly swapping segments in place, resulting in two offsprings. Figure 5.1 shows a crossover between two individuals and the possible offsprings. Figure 5.1a and Figure 5.1b show the selected individuals λ_i and λ_j before crossover performance, Figure 5.1c and Figure 5.1d the outcoming individuals after the crossover – i.e., the swapping of segments in place. Note, for example, in Figure 5.1c changed course between 20-28, showing introduced segments of Figure 5.1a in Figure 5.1b.



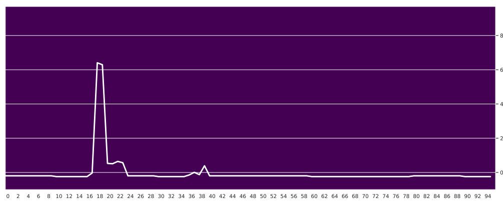
(a) Individual A.



(b) Individual B.



(c) Offspring A.



(d) Offspring B.

Figure 5.1: Visualization of Recombination on one Sample of UCR Electric Devices.

5.2.6 Mutation

Traditionally, individuals are mutated to produce new offsprings that differ from their parents. Using the crossover operator alone leads to decreasing diversity and often results in local optima, as only the good parts of the parents survive in each generation (premature convergence) [53].

We present four different mutation types, focusing on different significant time series behaviors to introduce new relevant information: Authentic Opposing Information includes semantically relevant time slices, Frequency Mutation copes with frequency changes that might influence the classification, Gaussian mutation accounts for distribution shifts and a combination of all to balance the different behaviors. Figure 5.2 visualizes the effects of the different mutation types on a data sample. Figure 5.2a shows the original candidate λ, Figure 5.2b the candidate after applying Opposing Information, Figure 5.2c the candidate after mutating the frequency bands, and Figure 5.2d the candidate after Gaussian Mutation. Additionally, a hyperparameter mutation is included that changes the individual's window size.

For generating plausible counterfactuals ($\mathbf{R_{5.3}}$), we introduce a reference set R. This reference set R is necessary for the mutation types Opposing Information, Frequency Mutation, and Gaussian Mutation. The reference set $R = \{z \in D : D_y \neq f(x)\}$ is a subset of all known data D with a prediction other than the original class $f(x)$.

5.2.6.1 Opposing Information

Opposing Information, first introduced by Guilleme et al. [84], is based on the assumption that interpretable values of time series can exhibit shapes (e.g., peaks) that are easily understandable by humans. To use those shapes included in a reference set R, we draw a random sample $r \in R$. Both r and the selected individual λ_i are segmented according to Section 5.2.4 with window size w_i, resulting in $S(r)$ and $S(\lambda_i)$. The mutation then draws a random segment index $s \in [0, |S(r)| - 1]$ and replaces the drawn slice $S(\lambda_i)[s]$ with the slice $S_r[s]$ from the replacement time series.

5.2.6.2 Frequency Mutation

Some time series classifiers may not only pay attention to the temporal structure of a time series but also its frequency. To be able to build counterfactuals for such types of time series, we introduce Frequency Mutation. Frequency Mutation converses a candidate time series λ_i and the reference set r from the time domain to the frequency domain via discrete Fourier transformation (see Equation (5.2), [228]). The resulting coefficients $a = (a_0, \ldots, a_{N-1})$ are the amplitudes of the decomposition components. x denotes the time series to transform, N the length of x, and k the current frequency.

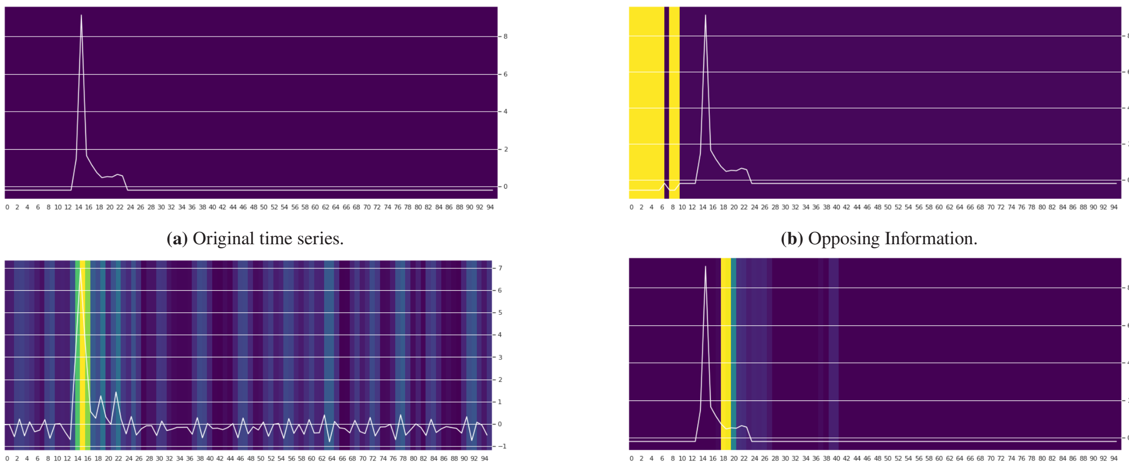
$$a_k(x) = \sum_{n=0}^{N-1} e^{-2\pi j \frac{kn}{N}} x[n] \tag{5.2}$$

$$x[k](a_k) = \sum_{k=0}^{N-1} e^{-2\pi j \frac{kn}{N}} a_k \tag{5.3}$$

The resulting frequency bands $a(\lambda_i) = (a_0(\lambda_i), \ldots, a_{N-1}(\lambda_i))$ and $a(r) = (a_0(\lambda_i), \ldots, a_{N-1}(r))$ are split into segments with quadratically increasing bandwidths. Analog to Opposing Information, one segment $a_:(\lambda_i)$ is selected randomly and replaced with the same segment of a randomly chosen item in the reference set $a_:(r)$. On the manipulated time series, inverse fourier transformation (Equation (5.3)) is applied to return the mutated individual λ_i.

5.2.6.3 Gaussian Mutation

The idea behind Gaussian Mutation is that some classifications are based on shifts in time series distribution. To account for such changes, the reference set R is used to calculate the mean $\mu_{f,t}$ and standard deviation $\sigma_{f,t}$ of each time step t and feature f. Let $x_{f,t}$ be a random time series point from $x_{f,t} \in \lambda_i$, Gaussian mutation replaces $x_{f,t}$ with a point $p \sim N(\mu_{f,t}, \sigma_{f,t})$.



(a) Original time series.

(b) Opposing Information.

(c) Frequency Mutation.

(d) Gaussian Mutation.

Figure 5.2: Visualization of Mutation Types on one Sample of UCR Electric Devices. Colored are the changes made to the original time series. The brighter the color, the larger the change.

5.2.6.4 Combination Mutation

The combined mutation mixes Opposing Information, Frequency Mutation, and Gaussian Mutation. Each individual I_i specifies their own mutation type in the variable $mut_i \in \{opposing, freq, gaussian\}$. This has the advantage that no decision regarding the mutation type has to be made in advance. The selection of the mutation type is optimized during the run of *TSEvo* by the selection step. The selection step chooses high-performing individuals. As each individual I_i, has its own mutation type mut_i the selection step thereby also selects the mutation type leading to these high performances.

5.2.6.5 Hyperparameter Mutation

The individuals' hyperparameters are mutated randomly after the use of any mutation by redrawing the window size w from the initialized range $[1, 0.5 * |x|]$.

5.3 Empirical Evaluation

This section evaluates the performance of our counterfactual method on a variety of uni- and multivariate datasets. We show that *TSEvo* generates well-performing counterfactuals by answering the following questions:

- How do the proposed mutation operators influence the performance of *TSEvo*? $\rightarrow$ Section 5.3.2
- How does *TSEvo* perform compared to other time series counterfactual methods regarding established metrics?$\rightarrow$ Section 5.3.3

The code for our implementation, experiments, and results can be found on GitHub[2].

5.3.1 Experimental Setting

We select 10 diverse datasets from the UCR [20] and the UEA Archive [21]. Both are well-known databases for time series classification tasks for uni- and multivariate data. The selected datasets contain different fields, lengths, and features to get a variety of time series shapes and their significant behavior. Table 5.1 shows an overview of the selected datasets.

For each dataset, we use the original train and test split and trained a 1D-convolutional ResNet as our black-box classifier. We chose ResNet because Fawaz et al. [70] found that ResNet significantly outperforms the other methods on univariate data and dominates in multivariate time series classification. However, any other architecture that return a probability distribution for the classification task is a suitable classifier for *TSEvo*. The test set accuracy for each dataset can be found in Table 5.1.

We run *TSEvo* on the first 20 time series of each test set and evaluate the resulting counterfactuals according to their distances, sparsity, yNN, and validity for both research questions. The metrics were adapted and fitted to this context from Pawelczyk et al. [174]. In the following, we depict the main metrics:

[2] `https://github.com/JHoelli/TSEvo`

Table 5.1: Dataset Description and Classification Results.

	CBF	**Coffee**	**CharacterTrajectories**	**ECG5000**	**ElectricDevices**	**FordA**	**GunPoint**	**NATOPS**	**Heartbeat**	**UWaveGestures**
Field	Simulated	Spectro	Motion	ECG	Devices	Sensor	Motion	Human Activity	EEG/MEG	Human Activity
Features	1	1	3	1	1	1	1	21	61	3
Length	128	286	109-205	140	96	500	150	51	405	315
Classes	3	2	20	5	7	2	2	6	2	8
Accuracy	0.99	0.96	0.98	0.94	0.7	0.93	0.93	0.81	0.61	0.77

- **Distances:** measure the distance between a counterfactual x^{cf} and the original time series x with the l_0- and the l_1- norm. The l_0 norm calculates the number of pixels changed between the original and counterfactual instance, and is identical to the sparsity from the optimization problem ($\mathbf{R_{5.2}}$). The l_1 norm is a measure for proximity ($\mathbf{R_{5.1}}$) that calculates the mean absolute error.
- **yNN:** evaluates a counterfactual's data support based on instances from the training set. It is an indication on how plausible ($\mathbf{R_{5.3}}$) a counterfactual is. Ideally, a counterfactual should be close to a factual time series from the same target class t. yNN is calculated by measuring how different neighborhood points around the counterfactual x^{cf} are classified. A higher value for yNN indicates better support from the data. kNN is the k-nearest neighbors of the original time series x. As distance measure to calculate kNN we use dynamic time warping. We use a value of $k = 5$.

$$\text{yNN} = 1 - \frac{1}{k} \sum_{j \in kNN(x^{cf})} \mathbb{1}_{f(x^{cf}) = f(x_j)} \tag{5.4}$$

- **validity:** indicates the fraction of counterfactuals that fulfill the requirement of $f(x) \neq f(x^{cf})$.

5.3.2 Mutation Types

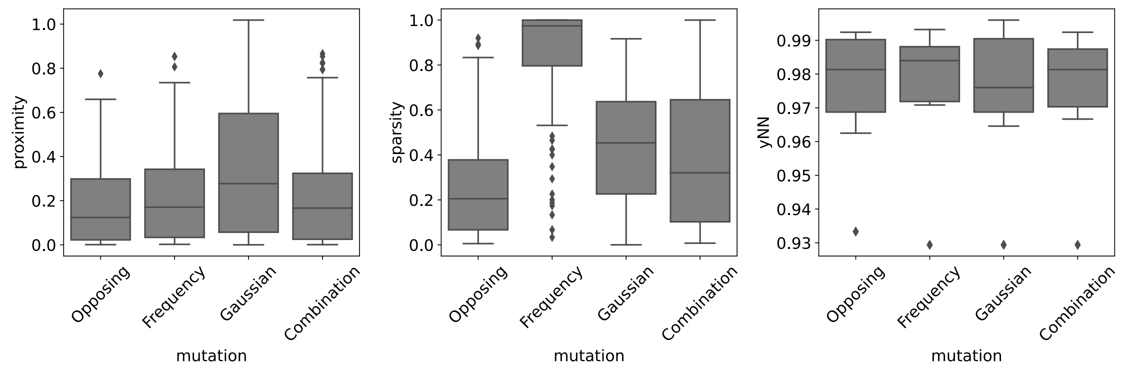


Figure 5.3: Boxplot of proximity ($\mathbf{R_{5.1}}$), sparsity ($\mathbf{R_{5.2}}$) and yNN ($\mathbf{R_{5.3}}$) for the different mutation types.

We validate our mutation choices by running *TSEvo* with Opposing Information, Frequency Mutation, Gaussian Mutation, and Combination Mutation for each dataset. The generated counterfactuals are evaluated on the requirements $\mathbf{R_{5.1}}$ - $\mathbf{R_{5.3}}$.

Figure 5.3 shows the proximity ($\mathbf{R_{5.1}}$), sparsity ($\mathbf{R_{5.2}}$), and plausibility ($\mathbf{R_{5.3}}$). Opposing Information, followed by Combination Mutation, achieved the lowest sparsity and proximity on average. Both mutation types generated slight changes (proximity) in a limited number of time steps (sparsity). The in general higher sparsity for Frequency Mutation shows the trade-off of working in the frequency domain. If a frequency band is exchanged, multiple changes occur in the whole time series. Those changes are usually smaller but come at the cost of additional changed time steps. Multiple small changes can make a time series harder to understand, as small changes are often not recognizable.

Comparing the mutation types on yNN, most of our mutations created counterfactuals with data support. The means of Frequency Mutation and Combination Mutation are slightly higher and less deviating; however, Opposing and Gaussian Mutation seem to offer higher-performing outliers, indicating that in specific situations, a specific mutation type performs better. The dataset-wise evaluation of the plausibility metric (see Table 5.4) supports this assumption by showing that different mutation types offer the highest plausibility for different datasets. On CharacterTrajecorties, Coffee, and NATOPS, Opposing Information offers the best results on $\mathbf{R_{5.3}}$, while the Frequency Mutation dominates for CBF, ECG5000, ElectricDevices, and Heartbeat. Gaussian Mutation performs best on FordA and UWaveGestureLibrary. The variety of best-performing mutations on $\mathbf{R_{5.3}}$ indicates that the data content affects plausibility. The best plausibility values for each dataset are distributed between the three basic mutation types Opposing, Frequency, and Gaussian. Note that on $\mathbf{R_{5.3}}$ Frequency Mutation performs exceptionally well on sensor-related datasets, indicating that Authentic Opposing Information might not provide a good solution for all types of datasets. Combination Mutation is ranked second for most datasets and metrics, indicating that different mutations might be necessary for different dataset types to achieve plausible results.

Based on sparsity, proximity, plausibility, and the fluctuation of the results on all metrics, no clear decision can be made on which mutation should be used. From the perspective of proximity and sparsity, Opposing Information is the preferred choice. However, on plausibility, the performance varies largely, indicating the relevance of the dataset content. This requires domain experts to choose a suitable mutation type based on the dataset content. As Combination Mutation combines Frequency, Gaussian, and Opposing Information and performs second-best on all metrics, we proceed with Combination Mutation in our benchmark study.

Table 5.2: Proximity ($\mathbf{R_{5.1}}$) for each dataset and mutation type. The lower, the smaller the changes made to the original instance.

	Opposing	**Frequency**	**Gaussian**	**Combination**
CBF	$\mathbf{0.22 \pm 0.13}$	0.28 ± 0.14	0.59 ± 0.17	0.29 ± 0.14
CharacterTrajectories	$\mathbf{0.21 \pm 0.07}$	1.0 ± 0.01	0.66 ± 0.14	0.26 ± 0.13
Coffee	$\mathbf{0.03 \pm 0.02}$	0.04 ± 0.01	0.05 ± 0.02	$\mathbf{0.03 \pm 0.02}$
ECG5000	0.34 ± 0.12	0.39 ± 0.16	0.45 ± 0.14	$\mathbf{0.33 \pm 0.11}$
ElectricDevices	0.12 ± 0.09	$\mathbf{0.2 \pm 0.17}$	0.58 ± 0.16	$\mathbf{0.2 \pm 0.14}$
FordA	$\mathbf{0.12 \pm 0.11}$	0.15 ± 0.13	0.24 ± 0.27	0.14 ± 0.12
GunPoint	0.25 ± 0.3	0.22 ± 0.28	$\mathbf{0.16 \pm 0.12}$	0.25 ± 0.3
Heartbeat	$\mathbf{0.0 \pm 0.01}$	0.01 ± 0.03	0.02 ± 0.03	0.01 ± 0.03
NATOPS	$\mathbf{0.15 \pm 0.09}$	0.17 ± 0.07	0.19 ± 0.1	0.17 ± 0.1
UWaveGestureLibrary	$\mathbf{0.35 \pm 0.15}$	0.43 ± 0.19	0.83 ± 0.12	0.57 ± 0.24

Table 5.3: Sparsity ($\mathbf{R_{5.2}}$) for each dataset and mutation type. The lower, the smaller the number of changes made to the original instance.

	Opposing	**Frequency**	**Gaussian**	**Combination**
CBF	$\mathbf{0.26 \pm 0.15}$	0.88 ± 0.14	0.62 ± 0.16	0.36 ± 0.15
CharacterTrajectories	$\mathbf{0.21 \pm 0.07}$	1.0 ± 0.01	0.66 ± 0.14	0.26 ± 0.13
Coffee	$\mathbf{0.12 \pm 0.08}$	0.78 ± 0.21	0.37 ± 0.1	0.15 ± 0.1
ECG5000	$\mathbf{0.38 \pm 0.13}$	0.98 ± 0.05	0.56 ± 0.14	0.45 ± 0.19
ElectricDevices	$\mathbf{0.32 \pm 0.18}$	0.8 ± 0.21	0.64 ± 0.17	0.4 ± 0.24
FordA	$\mathbf{0.21 \pm 0.19}$	0.82 ± 0.25	0.22 ± 0.24	0.31 ± 0.29
Heartbeat	$\mathbf{0.03 \pm 0.03}$	0.88 ± 0.18	0.06 ± 0.08	0.11 ± 0.21
GunPoint	0.35 ± 0.38	0.66 ± 0.39	$\mathbf{0.33 \pm 0.2}$	0.37 ± 0.38
NATOPS	$\mathbf{0.25 \pm 0.15}$	0.96 ± 0.04	0.32 ± 0.15	0.57 ± 0.33
UWaveGestureLibrary	$\mathbf{0.29 \pm 0.12}$	0.96 ± 0.1	0.72 ± 0.1	0.85 ± 0.24

Table 5.4: Plausibility ($\mathbf{R_{5.3}}$) of the generated counterfactuals for each dataset and mutation type. The higher, the better – i.e., the more data support from the training data.

	Opposing	**Frequency**	**Gaussian**	**Combination**
CBF	0.9688	**0.9719**	0.9688	0.9703
CharacterTrajectories	**0.9813**	0.9791	0.9791	0.9791
Coffee	**0.9902**	0.9881	0.9881	0.9874
ECG5000	0.9714	**0.9743**	0.9729	0.9729
ElectricDevices	0.9625	**0.9708**	0.9646	0.9667
FordA	0.9924	0.9932	**0.996**	0.9924
GunPoint	0.9813	0.984	0.976	0.9813
Heartbeat	0.9906	**0.9916**	0.9911	0.9906
NATOPS	**0.9333**	0.9294	0.9294	0.9294
UWaveGestureLibrary	0.9873	0.9873	**0.9905**	0.9873

5.3.3 Benchmark Study

We compare *TSEvo* with existing counterfactual methods for uni- and multivariate time series on the properties of a good counterfactual: proximity ($\mathbf{R_{5.1}}$), sparsity ($\mathbf{R_{5.2}}$), and data support ($\mathbf{R_{5.3}}$). We also add validity as a relevant metric, indicating the fraction of successfully generated counterfactuals. As a baseline for uni- and multivariate time series, we use the method of Wachter et al. [252] (*W-CF*), a traditional gradient-based method with no restriction on the input structure or classification model. Further, we employ two methods from Delaney et al. [55] for univariate data, namely the Native Guide Counterfactual based on dynamic time warping (*NG*-CF) and the Native Guide Counterfactual based on GradCam (*NG*-Grad). The authors applied *NG*-CF and *NG*-Grad only to univariate classification tasks. Therefore, for the multivariate classification task, we include the method of Ates et al. (*COMTE*-CF) [19], a method based on perturbing a time series feature as a whole.

Table 5.5 shows the fraction of successfully generated counterfactuals. All generated counterfactuals by *TSEvo* fulfill the counterfactual restriction. For *NG*-CF and Nun-Grad, no counterfactuals were found on Electric Devices and for *W-CF* on ECG5000.

Table 5.5: Validity. Fraction of correctly generated counterfactuals, where $f(x) \neq f(x^{cf})$

Dataset	TSEvo	*W-CF*	*NG*-CF	*NG*-Grad	*COMTE*
GunPoint	**1.00**	0.40	0.60	0.60	–
Coffee	**1.00**	0.30	0.70	0.70	–
CBF	**1.00**	0.80	0.15	0.05	–
ElectricDevices	**1.00**	0.85	0.00	0.00	–
ECG5000	**1.00**	0.00	0.90	0.30	–
FordA	**1.00**	0.40	0.25	0.60	–
CharacterTrajectories	**1.00**	0.00	–	–	**1.00**
Heartbeat	**1.00**	0.25	–	–	**1.00**
UWaveGestureLibrary	**1.00**	0.05	–	–	**1.00**
NATOPS	**1.00**	0.30	–	–	0.70

On proximity ($\mathbf{R_{5.1}}$, Table 5.6), *TSEvo* produced the best results on most datasets (except GunPoint). On sparsity ($\mathbf{R_{5.2}}$, Table 5.7), *TSEvo* produced fewer changes than the other baselines. Indicating that compared to *W-CF*, *NG*-CF, and *NG*-Grad, *TSEvo* produced the least number of small changes. Overall, Wachter-CF produces the largest sparsity, as the method is not time series related and operates without additional data on a point basis. Surprisingly, *NG*-Grad, a method that perturbs the time series based on gradient information from the classifier model, obtains a large sparsity too. A reason might be the increasing time slice window with every unsuccessful perturbation. Even on multivariate data, we outperform the results from *COMTE*-CF since *COMTE*-CF perturbs the whole feature sequence while we select a feature sequence and treat it as a univariate time series.

Table 5.6: Proximity ($\mathbf{R_{5.1}}$) in comparison to the baselines. The results show the mean absolute error without normalizing the data values to $[0, 1]$, therefore a $proximity > 1$ is possible.

Dataset	TSEvo	*W-CF*	*NG*-CF	*NG*-Grad	*COMTE*
CBF	$\mathbf{0.29 \pm 0.14}$	0.86 ± 0.32	0.91 ± 0.19	0.96 ± 0.00	–
Coffee	$\mathbf{0.03 \pm 0.02}$	0.11 ± 0.03	0.08 ± 0.05	0.05 ± 0.04	–
ECG5000	$\mathbf{0.33 \pm 0.11}$	12.38 ± 9.49	$4e10 \pm 1e11$	0.39 ± 0.16	–
ElectricDevices	$\mathbf{0.20 \pm 0.14}$	0.31 ± 0.21	–	–	–
FordA	$\mathbf{0.14 \pm 0.12}$	1.08 ± 0.18	1.38 ± 0.18	1.04 ± 0.25	–
GunPoint	0.25 ± 0.30	$\mathbf{0.07 \pm 0.05}$	$8e7 \pm 2e8$	0.42 ± 0.28	–
CharacterTraj.	$\mathbf{0.19 \pm 0.10}$	0.42 ± 0.10	–	–	0.41 ± 0.05
Heartbeat	$\mathbf{0.01 \pm 0.03}$	0.02 ± 0.02	–	–	$\mathbf{0.01 \pm 0.01}$
UWaveGesture	$\mathbf{0.57 \pm 0.24}$	0.72 ± 0.25	–	–	0.97 ± 0.15
NATOPS	$\mathbf{0.17 \pm 0.10}$	8.01 ± 17.57	–	–	0.24 ± 0.19

Table 5.8 shows the results for the data support. The results on plausibility ($\mathbf{R_{5.3}}$) are mixed. For 4 out of 10 datasets, *TSEvo* jas the highest data support. However, especially on the multivariate datasets Character Trajectories, UWaveGestureLibrary, and NATOPS, *COMTE*-CF outperformed *TSEvo*.

Figure 5.4 visually compares the counterfactuals for the univariate datasets generated with *TSEvo* and the baselines[3]. Especially compared to *W-CF*, *TSEvo* shows fewer time points are marked, and fewer marked brightly (the brighter the color, the more significant the change) for *TSEvo*. Generally, *TSEvo* makes visually

[3] Due to the data dimensionality, only univariate data is visualized. Code for visualizing multivariate datasets can be found on Github.

Table 5.7: Sparsity ($\mathbf{R_{5.2}}$) in comparison to the baselines.

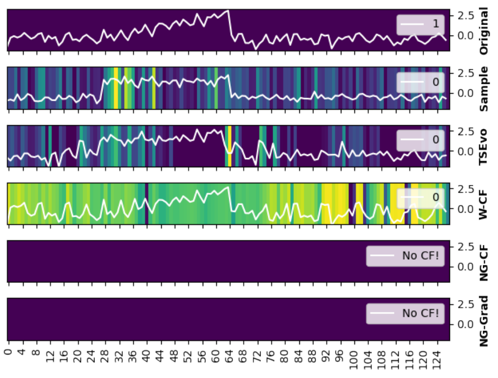
Dataset	**TSEvo**	***W-CF***	**Nun-CF**	**Nun-Grad**	***COMTE***
CBF	$\mathbf{0.36 \pm 0.15}$	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	–
Coffee	$\mathbf{0.15 \pm 0.10}$	1.00 ± 0.00	0.97 ± 0.12	0.58 ± 0.48	–
ECG5000	$\mathbf{0.45 \pm 0.19}$	1.00 ± 0.00	0.99 ± 0.03	0.87 ± 0.29	–
ElectricDevices	$\mathbf{0.40 \pm 0.24}$	1.00 ± 0.00	–	–	–
FordA	$\mathbf{0.31 \pm 0.29}$	1.00 ± 0.00	1.00 ± 0.00	0.95 ± 0.11	–
GunPoint	$\mathbf{0.37 \pm 0.38}$	1.00 ± 0.00	1.00 ± 0.00	0.92 ± 0.27	–
CharacterTraj.	$\mathbf{0.26 \pm 0.13}$	1.00 ± 0.00	–	–	0.62 ± 0.07
Heartbeat	0.11 ± 0.21	1.00 ± 0.00	–	–	$\mathbf{0.06 \pm 0.05}$
UWaveGesture	$\mathbf{0.85 \pm 0.24}$	1.00 ± 0.00	––	–	0.93 ± 0.13
NATOPS	0.57 ± 0.33	1.00 ± 0.00	–	–	$\mathbf{0.37 \pm 0.31}$

more focused changes compared to the benchmarks, fewer time points are highlighted and only some are highlighted brightly. In some cases (e.g., Coffee, Electric Devices) the changes provided by *TSEvo* are similar to the changes necessary to transform the original time series into the closest sample time series from another class (denoted in the visualization as sample).

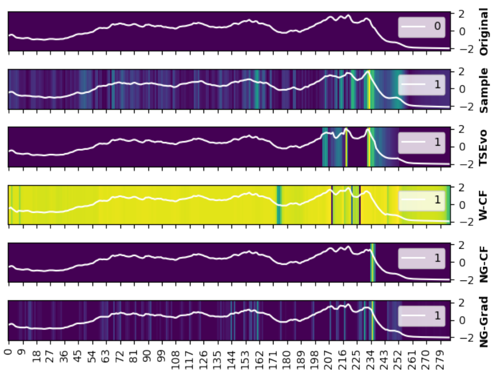
Overall, *TSEvo* was the only method to generate valid counterfactuals for all uni- and multivariate settings by changing only a fraction of time slices and features. The sample visualizations show focused and recognizable changes. We rank the performance on each dataset based on the results obtained for $\mathbf{R_{5.1}}$, $\mathbf{R_{5.2}}$, and $\mathbf{R_{5.3}}$ and average those ranks over all datasets and metrics. Looking at the overall average ranks of *TSEvo* and the other baselines, *TSEvo* (1.43) outperforms the other baselines (*W-CF*: 2.7, *NG*-CF: 2.88, *NG*-GRAD: 2.33, *COMTE*-CF: 1.58).

Table 5.8: Plausibility ($\mathbf{R_{5.3}}$) of the baselines and *TSEvo*.

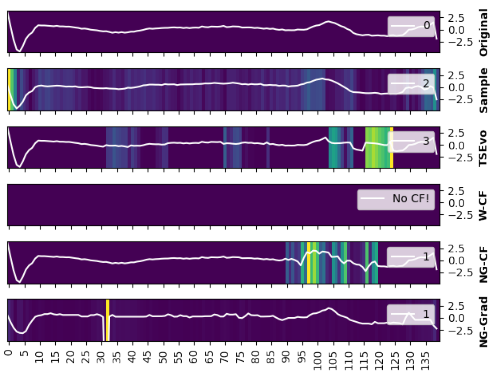
Dataset	**TSEvo**	***W-CF***	***NG*-CF**	***NG*-Grad**	***COMTE***
CBF	0.9688	0.9750	**0.9953**	0.8000	-
Coffee	**0.9993**	0.9888	0.9923	0.9909	-
ECG5000	0.9714	0.9886	0.9871	**0.9914**	-
ElectricDevices	0.9604	**0.9812**	-	-	-
FordA	**1.0000**	0.9928	0.9992	0.9956	-
GunPoint	**0.9907**	0.9747	0.9880	0.9840	-
CharacterTraj.	0.978	0.9879	-	-	**0.9978**
Heartbeat	**1.0000**	0.9901	-	-	0.9946
UWaveGesture	0.9886	**0.9930**	-	-	**0.9930**
NATOPS	0.9216	**0.9569**	-	-	**0.9569**



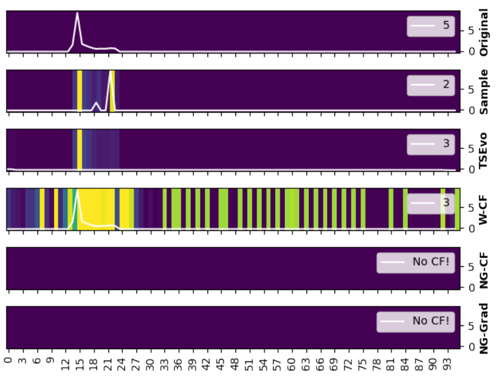
(a) CBF.



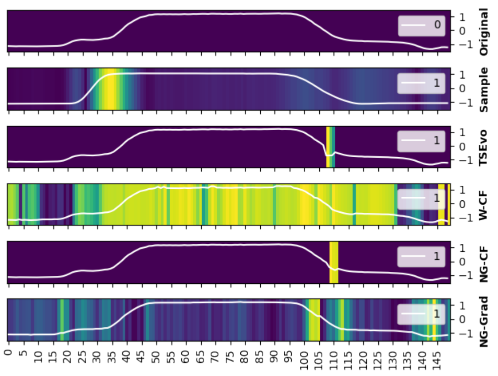
(b) Coffee.



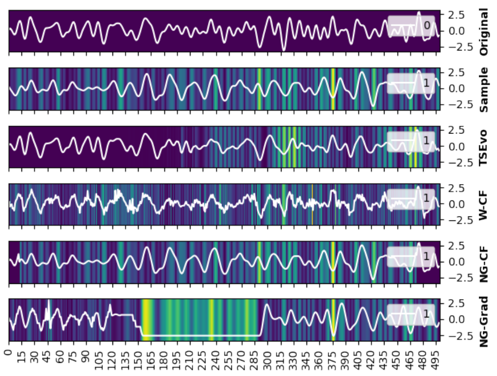
(c) ECG5000.



(d) ElectricDevices.



(e) GunPoint.



(f) FordA.

Figure 5.4: Visualization of the first original time series from the test set, the time series with the smallest mean squared error sampled from the training data, and the counterfactual time series for *TSEvo*, Wachter-CF, *NG*-CF, *NG*-Grad, and *COMTE*-CF. The brighter the color, the more significant the change. If the counterfactual time series label is consistent with the original time series label, the shown counterfactual is invalid. Further on ElectricDevices, *NG*-CF, and *NG*-Grad could not find a valid counterfactual at all, while for CBF, none were found on the first sample.

5.4 Summary and Discussion

This section answers RQ I.(2) by introducing *TSEvo*, an evolutionary approach to generating plausible counterfactuals for time series classifiers. *TSEvo* incorporates a variety of time series transformation mechanisms to cope with different types and structures of time series in both uni- and multivariate classification. In the ablation study, we were able to show that none of the transformers were significantly dominant, indicating that the transformer selection depends on the data content. Furthermore, *TSEvo* outperformed the baselines on both uni- and multivariate data.

The results indicate that depending on the dataset, different mutation types are optimal; investigating how to select the best mutation type for a given dataset based on time series properties and patterns might improve the results even more. While we were able to show that *TSEvo* outperforms all tested counterfactual methods for counterfactual-specific metrics, significant properties of a good explanation, such as the consistency between the behavior of the back-box classifier and the counterfactual explainer, have yet to be investigated. For one, this is because those metrics are the state-of-the-art evaluation for counterfactual explainers (see [174]). In addition, general metrics often miss adaption for both the time series domain and counterfactual explainers. This issue will be revisited and evaluated in the next section, Chapter 6.

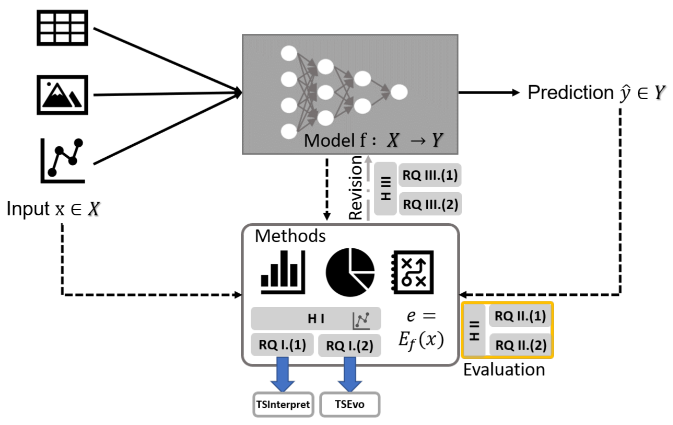
Counterfactuals aim at providing contrastive and realistic explanations (– i.e., counterfactual explanations should be observable in the real world). While many newer methods take this into account by utilizing the data distribution, similar to *TSEvo*, user-given constraints or partly given causal models [58, 48, 107, 140] the proposed evaluation of "realistic" counterfactuals focuses on correlation in the collected data, rather than realistic causal relationship. Quantifying the fulfillment degree of such causal relationships in order to evaluate the "realisticness" of a counterfactual is still an open issue. This problem will be revisited in Chapter 7.

Besides quantifying and benchmarking counterfactuals, an investigation of its practical usefulness to humans is of interest. We leave that open to future work.

Like *TSInterpret*, introduced in the previous chapter, *TSEvo* is currently only applicable to classification tasks. For regressional tasks, *XAI* methods are often translated to multi-class classification by building buckets [202, 30] or for counterfactual explanations by providing a minimal threshold for a mock class change [226]. Both approaches could be used to make *TSEvo* applicable to regression tasks.

Part III

Evaluating Explainable Machine Learning



This part deals with challenge 2, **the need to ensure the quality of an explainer quantitatively** by developing new approaches for measuring the performance of *XAI*. The first section (Chapter 6) builds on Chapter 4 and Chapter 5 by developing a benchmarking framework for standardizing the evaluation of time series explainers on faithfulness, robustness, reliability, and complexity. It provides a thorough comparison of existing time series explainers. The second section (Chapter 7) extends the evaluation of counterfactual explainers by proposing a new metric to measure the causal coherence of counterfactual approaches.

6 Benchmarking (post-hoc) XAI for Time Series

While the first step to standardize the explanation benchmarking process for the time series domain has been taken by *TSInterpret* [94] – a framework implementing explanation methods for time series classification in a unified interface – in the last part (see Chapter 4), standardized metrics for evaluating the quality of explanation methods specifically for time series are still missing [238]. Measuring the performance of explanation methods is still challenging as there are no generally agreed-upon metrics measuring the quality of explanations, and comparisons between different implementations and metrics are difficult (see Section 3.3). Similar to adapting explanation methods to the time series domain, transferring metrics to the time series domain is complex. Using metrics from traditional frameworks (e.g., [89]) can lead to erroneous assumptions in the time series domain. This lack of specific and standardized metrics leads to a wide variety of proposed metrics, metric implementations, proposed baselines, and baseline implementations. This chapter addresses these obstacles by answering RQ II.(1):

RQ II.(1) How can we quantify the quality of time series explainers for an easier benchmarking?

To address these issues, this section proposes *XTSC-Bench*, a benchmarking tool implementing a variety of metrics for a standardized and systematic evaluation of explainers for time series classifiers. Its connection to TSInterpret [94] ensures a unified implementation of benchmarking algorithms. We utilize *XTSC-Bench* to evaluate 3 gradient- and 6 perturbation-based feature importance methods and 2 example-based methods.

The following sections are based on “XTSC-Bench: Quantitative Benchmarking for Explainers on Time Series Classification” (2023 22st IEEE International Conference on Machine Learning and Applications (ICMLA) - 2023). First, we formalize the problem setting in Section 6.1. Section 6.2 describes the benchmarking tool including the metrics, used models and utilized synthetic data, followed by the evaluation in Section 6.3. Finally, Section 6.4 provides a summary and a discussion.

6.1 Problem Formalization

In line with Section 1.2 , we study a supervised time series classification problem. Let $x = [x_{11}, ..., x_{NT}] \in \mathrm{R}^{N \times T}$ be a uni- or multivariate time series, where T is the number of time steps, and N is the number of features. Let $x_{i,t}$ be the input feature i at time t. Similarly, let $X_{:,t} \in \mathrm{R}^N$ and $X_{i,:} \in \mathrm{R}^T$ be the feature vector at time t, and the time vector for feature i, respectively. Y denotes the output, and $f : X \rightarrow Y$ is a classification model returning a probability distribution vector over classes $y = [y_1, \ldots, y_C]$, where C is the total number of classes (i.e., outputs) and y_i the probability of x belonging to class i. An explanation method E_f finds an explanation $E_f(x) \in \mathrm{R}^{N \times T}$. In the case of feature attribution methods, the explainer E_f assigns an attribution a_{it} to explain the importance of a feature i a time step t, resulting in $E_f(x) = (a_{11}, \ldots, a_{NT})$. For example-based methods E_f provides an example with the same prediction or a counterexample resulting in $E_f(x) = (x'_{11}, \ldots, x'_{NT})$.

For an Explainer E_f to provide good explanations, those explanations need to be:

- Reliable[1]: An explanation should be centered around the region of interest, the ground truth GT.

$$E_f(x) \cong GT$$

- Faithful[2]: The explanation algorithm E_f should replicate the model's f behavior.

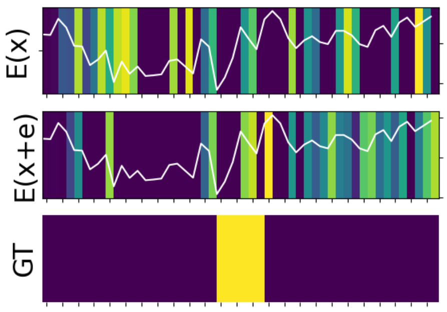
$$E_f(x) \sim f(x)$$

- Robust[3]: Similar inputs should result in similar explanations.

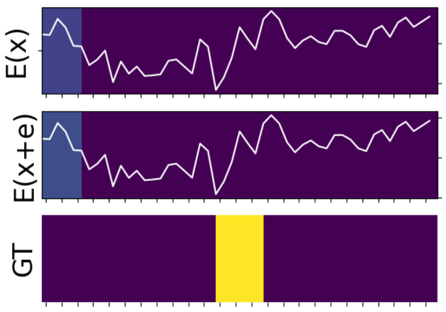
$$E_f(x) \approx E_f(x + \epsilon)$$

- Complex[4]: Explanations using a smaller number of features are preferred. It is assumed that explanations using many features are difficult for the user to understand [153].

$$\min \mathbb{1}_{E_f(x)>0}$$



(a) Gradient-Based E



(b) Perturbation-Based E

Figure 6.1: Visualization of metric implications on a sample explanation $E(x)$.

Figure 6.1 visualizes the implications of the requirements above on explanations obtained from a gradient-based explainer (an explanation based on the classifiers gradient estimations) and a perturbation-based explainer (an explanation based on observing the influence of input modifications). The top images show the original time series x with an explanation $E(x)$ visualized as a heatmap. The middle image shows the perturbed time series $x + \epsilon$ with the explanation $E(x + \epsilon)$. The bottom image shows the known ground truth GT. In case of this specific time series: The complexity is high for Figure 6.1a, resulting from many

[1] Also referred to as Localization (e.g., [89]).
[2] Often referred to as Fidelity (e.g., [84]).
[3] Often referred to as Stability or Sensitivity (e.g., [29]).
[4] In context of example-based explanations often called Sparsity (e.g., [174]).

attributions (highlights). For Figure 6.1b, the complexity is low. Although, Figure 6.1b performs better on complexity taking the ground truth GT into account, the attributions obtained on the sample are inconsistent with GT. The explanation in Figure 6.1b is more robust than Figure 6.1a, as the explanations $E(x)$ and $E(x+\epsilon)$ are identical. Faithfulness quantifies the consistency between the decision-making process of f and the explanations E. The consistency of Figure 6.1a is higher than the one from Figure 6.1b as Figure 6.1a relies on the gradients of f while Figure 6.1b fits a surrogate model. Overall, in this case, although Figure 6.1b performs better on complexity and robustness than Figure 6.1a, due to the limited reliability (i.e., consistency with GT), Figure 6.1a should be the preferred explainer.

6.2 Approach

XTSC-Bench provides a standardized framework to allow users to apply and evaluate different state-of-the-art explanation models in a replicable manner on the notions of complexity, reliability, robustness, and faithfulness. Figure 6.2 visualizes the architecture. The benchmarking tool is split according to Section 6.1 into different classes for benchmarking robustness, faithfulness, reliability, and complexity. As some notions (e.g., reliability) rely on a fairly accurate definition of an explanation ground truth GT or the iterative masking of parts of the original input with known uninformative features, we include uni- and multivariate synthetic data and pre-trained models in the benchmarking tool (see Section 6.2.1). Each class follows the evaluation interface, providing a method `evaluate` and a method `evaluate_synthetic`. The function `evaluate` allows the usage of non-synthetic data and models as well as the evaluation of a single explanation on-the-fly. For all metrics, we use a wrapper build around Quantus [89] and added some time-specific tweaks.

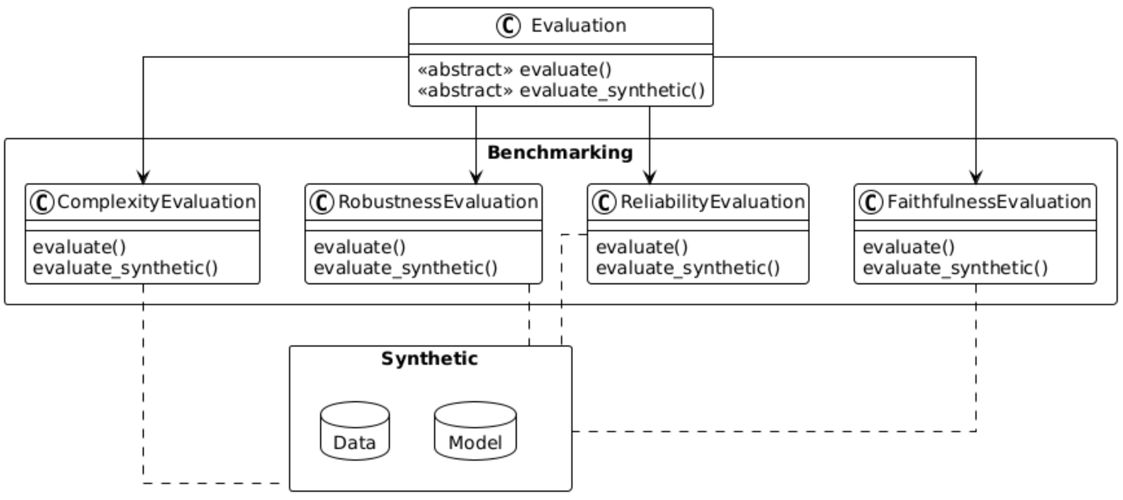


Figure 6.2: Architecture of *XTSC-Bench*.

6.2.1 Synthetic Data and Pretrained Models

XTSC-Bench provides 60 uni- and 60 multivariate synthetic datasets with 50 time steps generated according to Ismail et al. [101]. The 'base' dataset is generated based on various time series processes with $\epsilon_t \sim N(0,1)$:

- Gaussian ($\mu = 0, \sigma = 0$):
  $X_t = \epsilon_t$

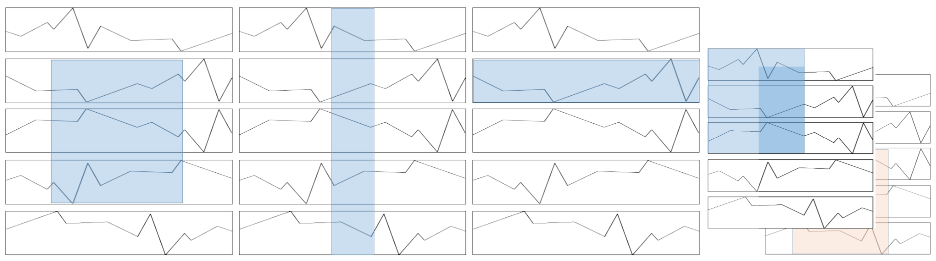


Figure 6.3: Visualization of Informative Features types. The rectangle indicates the informative features.

- 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 ($n = 10$, $U \sim U(0.05)$):
  $X_t = 0.3X_{t-1} + 0.05X_{t-1} \sum_{i=0}^{n-1} X_{t-1} + 1.5U(t-(n-1)) * U(t) + 0.1 + \epsilon_t$

For each 'base' dataset obtained from the time series process, multiple synthetic datasets are obtained by adding various Informative Features ranging from Rare Features (less than 5% of features) and time steps (less than 5% of time steps) mimicking an anomaly detection task to boxes covering over 30% of features and time steps (see Figure 6.3) and the placement of those informative features.

- **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.

A binary label is added for each dataset (time process $\times$ informative feature) by highlighting informative features with the addition of a constant for positive classes and subtraction for negative classes. Overall we obtain 60 univariate datasets and 60 multivariate datasets (6 time processes $\times$ 10 informative features). [5]

For all synthetic uni- and multivariate datasets, we train a 1D-Convolutional Network with ResNet Architecture (CNN) and Long Short Term Memory (LSTM) with a hidden layer of size 10. We train the networks with a cross-entropy loss for 500 epochs with patience of 20 and Adam with a learning rate of 0.001. The trained networks are also provided in *XTSC-Bench*.

[5] *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.

6.2.2 Robustness

Robustness measures the stability of an explanation method's output subjected to a slight input perturbation $\bar{x} = x + \epsilon$ under the assumption that the model's output approximately stays the same $f(x) \approx f(\bar{x})$. Small, unmeaningful changes around x should lead to a consistent explanation. *XTSC-Bench* employs two metrics measuring the robustness of an explanation algorithm E:

- Max Sensitivity [29] measures the maximum change in the explanation with a small perturbation of the input x. r denotes the input neighborhood ratio.

$$Sens_{max}(E, f, x, r) = max_{\bar{x}-x \leq r} ||E_f(\bar{x}) - E_f(x)|| \tag{6.1}$$

- Average Sensitivity [29] denotes the average sensitivity in the neighborhood of x with $\bar{x} - x \leq r$.

$$Sens_{mean}(E, f, x, r) = \frac{1}{|x|} \sum ||E_f(\bar{x}) - E_f(x)|| \tag{6.2}$$

6.2.3 Faithfulness

Faithfulness quantifies the consistency between the prediction model f and explanation model E. Most faithfulness metrics rely on so-called reference baselines consisting of non-informative features. In literature, those reference baselines are often training data means or zeros (e.g., [**sundararajan2017axiomatic**]). However, for time series data those baselines might contain information (e.g., 0 might be an informative anomaly). Therefore, on the proposed synthetic data, the reference baseline $\tilde{x}$ is sampled from the generation process. *XTSC-Bench* employs faithfulness correlation [29] to measure the correlation between the sum of attributions $\sum_{s \in S} E_f(x_{x_s = \tilde{x}_s})$ and the difference in output $f(x) - f(x_{x_s = \tilde{x}_s})$ when setting those features to a reference baseline $x_{x_s = \tilde{x}_s}$. S is a subset of input features, $\tilde{x}_S$ denotes a subset of the reference baseline $\tilde{x}$ and x_s is the corresponding subset for the original instance x.[6]

$$Faith(f, E, x) = corr(\sum_{s \in S} E_f(x_{x_s = \tilde{x}_s}), f(x) - f(x_{x_s = \tilde{x}_s})) \tag{6.3}$$

6.2.4 Complexity

Complexity [29] measures the number of features used in an explanation with a fractional contribution distribution $\mathbb{P}_g$: the fractional contribution of feature x_i to the total magnitude of the attribution: $\mathbb{P}_g(i) = \frac{E_f(x)_i}{\sum |E_f(x)|}; \mathbb{P}_g \in \{\mathbb{P}_g(1) \ldots, \mathbb{P}_g(d)\}$. The maximum value of complexity is $log(|E_f(x)|)$, where $|.|$ denotes the vector length.

$$cpx(f; E; x) = -\sum_{i=1}^{d} \mathbb{P}_g(i) ln(\mathbb{P}_g(i)) \tag{6.4}$$

[6] In case of using our benchmarking tool with non-synthetic data we provide the possibility of custom baselines. As default, baselining is done uniformly.

6.2.5 Reliability

Explanation methods should distinguish important from unimportant features at each time step and note changes over time. “Major” parts of an explanation should lie inside the ground truth mask $GT(x)$. *XTSC-Bench* includes the ground truth based measures relevance rank accuracy and relevance mask accuracy from [17].

- Relevance Rank Accuracy [17]: The relevance rank accuracy measures how much of the high intensity relevance lies within the ground truth. We sort the top K values of $E_f(x)$ in decreasing order $X_{topK} = \{x_1, \ldots, x_k | E_f(x)_1 > \ldots > E_f(x)_K\}$.

$$RACC = \frac{|X_{topK} \cap GT(x)|}{|GT(x)|} \tag{6.5}$$

- Relevance Mass Accuracy [17]: The relevance mass accuracy is computed as the ratio of the sum of the Explanation values lying within the ground truth mask over the sum of all values.

$$MACC = \frac{\sum_{E_f(x)_i \in GT(x)} E_f(x)_i}{\sum E_f(x)} \tag{6.6}$$

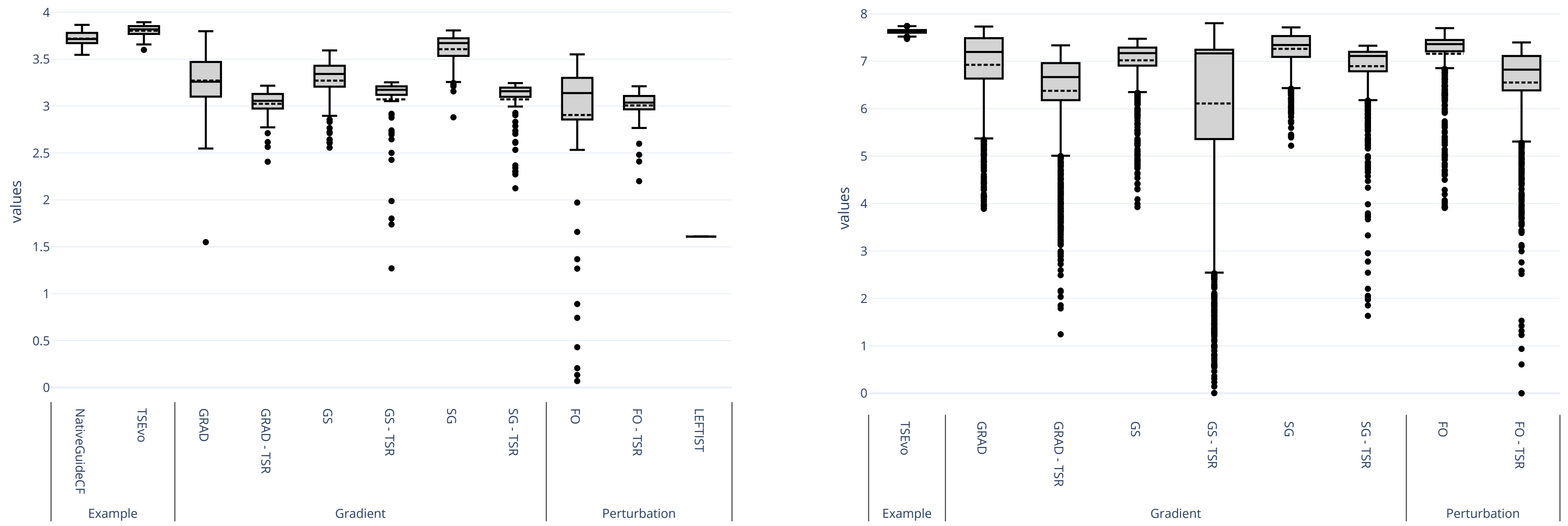
6.2.6 Application to instance-based methods

Replacing features to calculate robustness (Section 6.2.2), reliability (Section 6.2.5), or the faithfulness metrics (Section 6.2.3) 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, instance-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 instance-based methods, we calculate the fraction of change $\Delta x = \frac{(x - E_f(x))}{x}$.[7]

6.3 Empirical Evaluation

This section compares the performance of 6 gradient- with 3 perturbation-based feature attribution methods and 2 instance-based methods across Recurrent Neural Networks and Temporal Convolutional Networks for both the multi- and univariate synthetic time series (Section 6.2.1). The results are reported on a before unseen test set. As gradient-based methods, we include Saliency (GRAD) [219], *Growing Spheres* (*GS*) [139], and *Smooth Gradients* (*SG*) [221] with and without *Temporal Saliency Rescaling* (*TSR*) [101]. As perturbation-based, we include *Feature Occlusion* (*FO*) [268] with and without *Temporal Saliency Rescaling* (*TSR*) [101] and *Local Explainer For TIme Series classificaTion* (*LEFTIST*) [84], a method based on Lime adapted to time series. *TSEvo* [93], and *Native Guide* (*NG*) [55] represent the instance-based methods. For all methods, we use the implementation in TSInterpret [94].[8] By employing *XTSC-Bench*, we evaluate the explainers' capabilities on complexity, reliability, robustness, and faithfulness for all classifiers with an accuracy of over 90%[9].

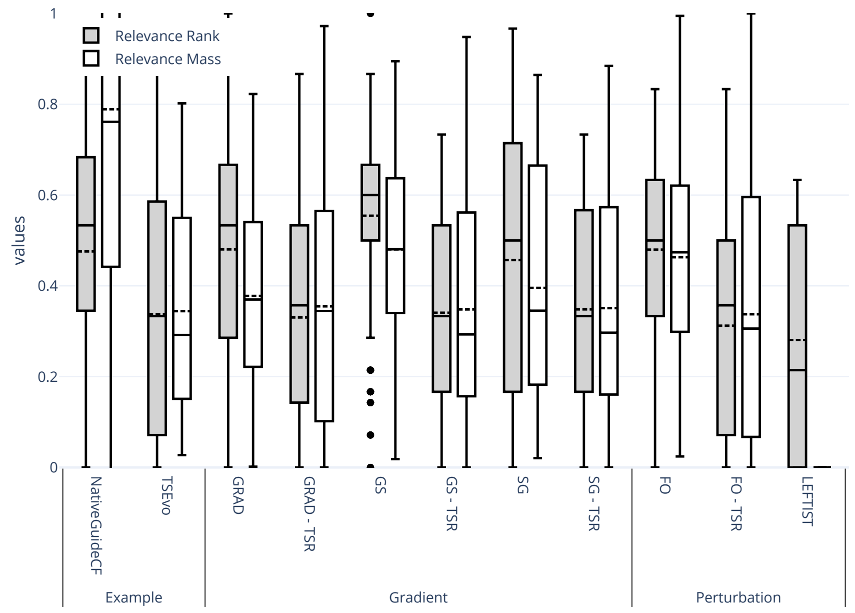
[7] *Practical Note*: The synthetic data is normalized to zero and one. For non-synthetic data, a feature range needs to be provided.
[8] More details on the methods can be found in Section 2.2, Section 2.3 and Chapter 5.
[9] Explainers are usually used to validate the inner-workings of well-performing classifiers. Classifiers with low accuracy cannot be expected to learn sufficient features to ensure an explainer's reliability and classifier consistency.



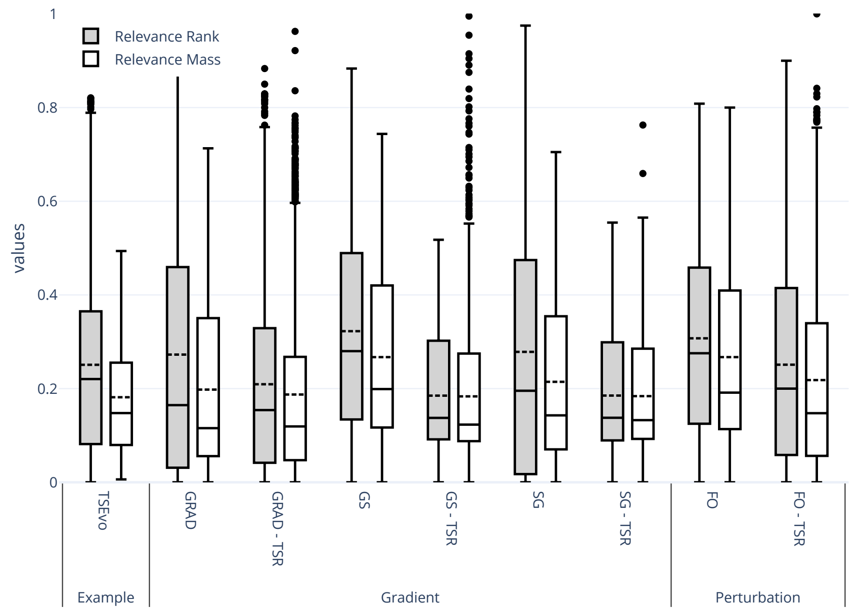
(a) Complexity Univariate

(b) Complexity Multivariate

Figure 6.4: Explainer Performance on complexity, reliability, faithfulness, and robustness averaged over all datasets. The line denotes the median and the dotted line the mean. The start and end of the boxes are the first and third quartiles. Note, that *NG* and *LEFTIST* only apply to univariate data and are therefore missing in the multivariate evaluation.

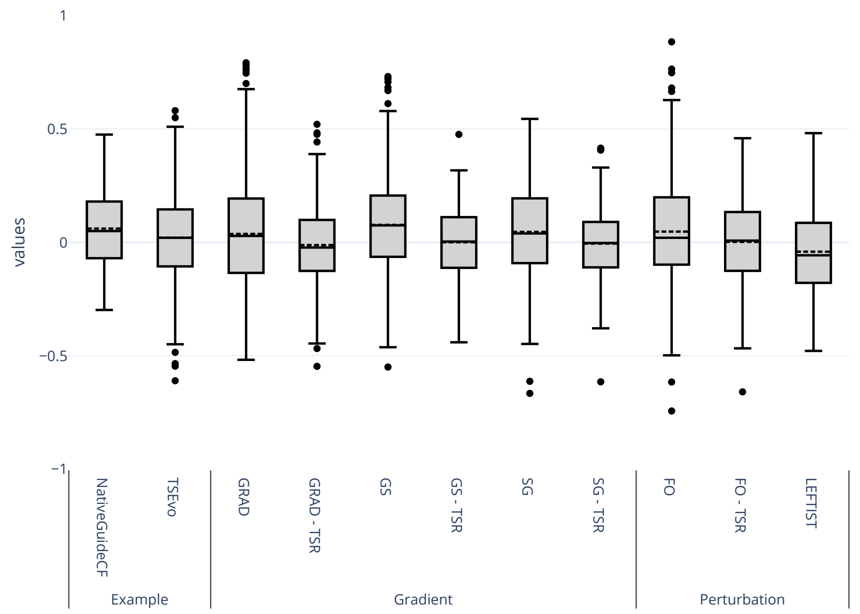


(c) Reliability Univariate

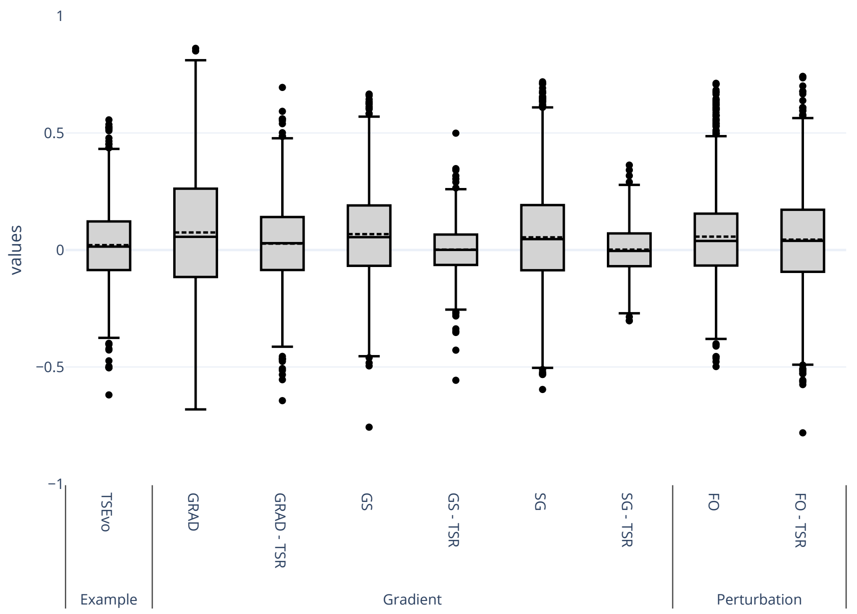


(d) Reliability Multivariate

Figure 6.4: Explainer Performance on complexity, reliability, faithfulness, and robustness averaged over all datasets. The line denotes the median and the dotted line the mean. The start and end of the boxes are the first and third quartiles. Note, that *NG* and *LEFTIST* only apply to univariate data and are therefore missing in the multivariate evaluation.

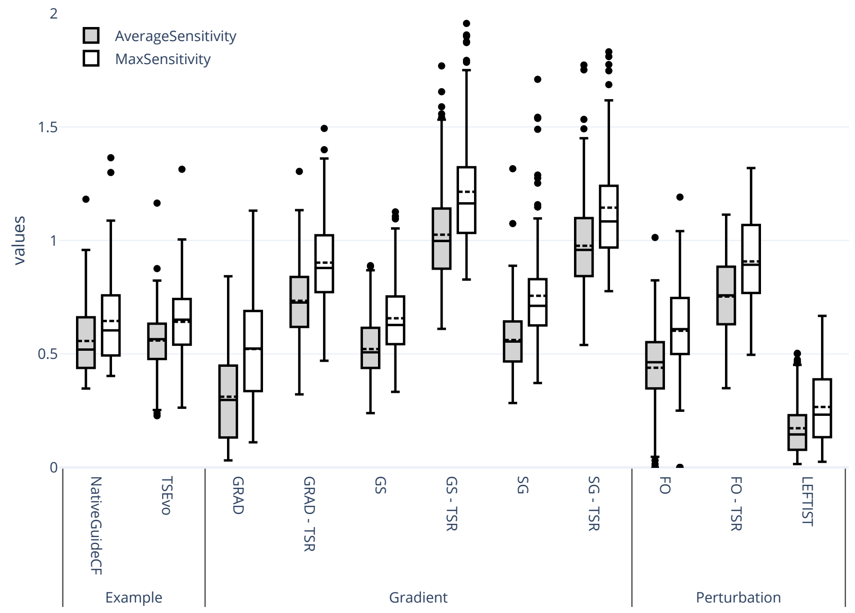


(e) Faithfulness Univariate

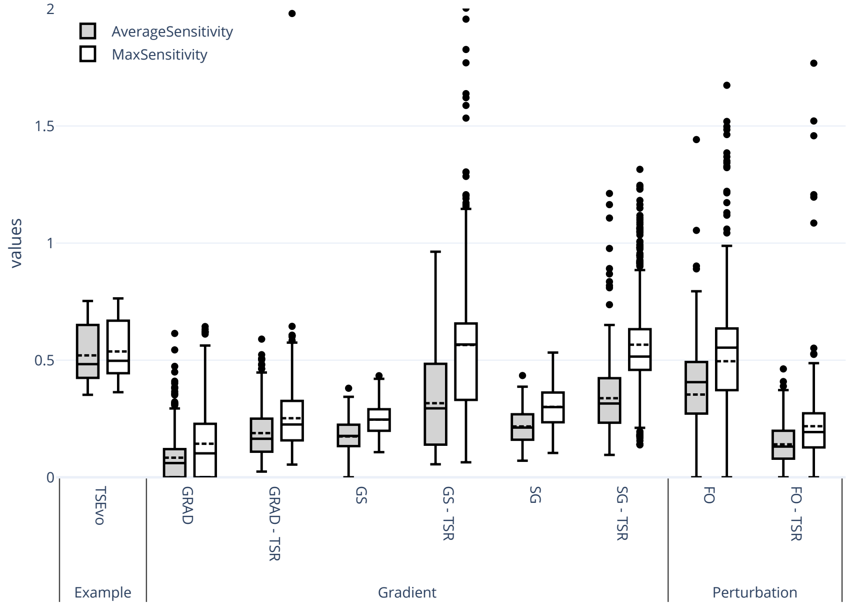


(f) Faithfulness Multivariate

Figure 6.4: Explainer Performance on complexity, reliability, faithfulness, and robustness averaged over all datasets. The line denotes the median and the dotted line the mean. The start and end of the boxes are the first and third quartiles. Note, that *NG* and *LEFTIST* only apply to univariate data and are therefore missing in the multivariate evaluation.



(g) Robustness Univariate



(h) Robustness Multivariate

Figure 6.4: Explainer Performance on complexity, reliability, faithfulness, and robustness averaged over all datasets. The line denotes the median and the dotted line the mean. The start and end of the boxes are the first and third quartiles. Note, that *NG* and *LEFTIST* only apply to univariate data and are therefore missing in the multivariate evaluation.

Figure 6.4 summarizes the explainer-wise results on complexity, reliability, faithfulness, and robustness, averaged over all datasets and classifier models. On complexity (Figure 6.4a and Figure 6.4b), gradient- and perturbation-based methods provide less complex explanations than instance-based methods. The results obtained by *TSR* contain slightly fewer attributions than the plain gradient- and perturbation-based methods, indicating that the explanations obtained after *TSR* are slightly easier to grasp. Averaging the obtained relevance scores on both, the feature and time domain with *TSR*, leads to a complexity decrease by eliminating areas with less relevance (e.g., single and small relevance scores on certain time steps).

The reliability (Figure 6.4c and Figure 6.4d) on univariate data is higher than on multivariate data showing a decreasing capability of centering the explanation around the, in this case, known ground truth of all explainers with increasing data complexity. The on average lower relevance mask than rank indicates that, while relevant features are found, the contribution of the found informative features to the overall relevance is low. Interestingly, for both dataset types, the plain gradient- and perturbation-based methods (without *TSR*) perform slightly better on the Relevance Rank. On Relevance Mass, the difference between *TSR* and the plain methods diverges (e.g., on univariate GRAD, *TSR* results in an improvement, on univariate GS, *TSR* results in a deterioration).

The faithfulness (Figure 6.4e and Figure 6.4f) of the explainers to the classification models' behavior is similar for most explainers on the uni- and multivariate data. The least faithful is *LEFTIST*, as *LEFTIST* is the only method relying on a local surrogate model instead of frequent classifier calls or the classifiers' inner workings (i.e., gradients).

The results on robustness (Figure 6.4g and Figure 6.4h) indicate that on univariate data, perturbation-based methods are less sensitive to small changes than instance-based methods. This results from perturbation-based methods only relying on the perturbation function (which is constantly the same) and the classification model's output, while gradient-based methods rely on a model's inner workings that possibly change with varying the input.

Summarizing the results, no clear indication can be given on which explanation methods should be preferred. No method was able to dominate the plain gradient, and perturbation-based methods, which are included as baselines. Both, traditional and time-series specific explainers, show potential for improvement in all aspects. With increasing data complexity (univariate vs. multivariate), the metric performances diverge further, indicating a need for less complex, more reliable, and robust explainers, especially for multivariate time series classification.

6.4 Summary and Discussion

This section focused on RQ II.(1), the performance quantification of time series explainers. We proposed *XTSC-Bench*, a benchmarking tool for explainers for time series classifiers that dissolves existing ambiguities and enables more comparability. *XTSC-Bench* consists of metrics adapted to the time series domain and provides synthetic datasets with informative features, from analogies to anomaly detection to moving features, trained models for the synthetic data, and options to evaluate custom data. A first empirical evaluation of the explainers implemented in TSInterpret [94] showed that the current time series explainers leave potential for improvement, especially in providing reliable explanations for multivariate time series classification. The results indicate that methods adapted to time series are slightly less faithful than the original non-time specific method, indicating that only adopting methods to time series by feature suppression and averaging the non-time methods (*TSR*) are insufficient. While this is already the case for counterfactual explainers, new research for time series should instead focus on including time-specific properties in the gradient and perturbation-based methods. On multivariate data, the tested methods were less reliable but more robust, indicating an additional need for research on high-dimensional data. Overall, the results indicate a strong need for future research on *XAI* for time series. New methods should focus on providing more faithful and reliable methods, specifically for time series patterns.

The proposed framework evaluates reliability, faithfulness, robustness and complexity, applicable to most explanation methods. However, depending on the explainer used, more specific metrics are necessary to capture the explainer's performance as additional explanation properties become essential. For instance, *IB* methods rely on generating realistic examples. An *IB* explanation is only good if the (perturbed) instance shown is represented in the data distribution and not a mere anomaly. Therefore, enhancing *XTSC-Bench* with more explanation scope-specific metrics becomes crucial for a more detailed explanation.

Further, the employed synthetic data generation approach currently only covers binary classification and modeling informative features by adding a constant. While we do cover data generated from a variety of time processes, time series often contain a wide variety of potential informative featured, relevant to classification tasks, e.g., time series patterns, often called shapelets 2011, statistical features like mean and variance, or the frequency. Further, there are many more ways to generate synthetic data that include additional time series properties (e.g., from neural networks). However, how to use such data generation for downstream tasks like classification with known ground truth masks is still open.

The used metrics only capture objective quantitative aspects, human-grounded evaluation was not part of this work. Nevertheless, evaluating how helpful human users find the produced explanations is crucial, specifically in the time series domain, where humans have poor intuitions. First steps in evaluating the perception of model inspection on time series have already been taken in [92]. However, their study mostly focused on instance-based explanations for unvariate data in binary classification. To get a broader overview, a more in-depth human grounded evaluation would be necessary.

7 Measuring the real-world coherence of counterfactual explanations

In the previous chapter (Chapter 6), we enhanced the model-agnostic evaluation of *XAI* methods in terms of complexity, reliability, faithfulness, and robustness for time series by refining the original notation and employing synthetic datasets with known generation mechanisms. However, these evaluation criteria alone are insufficient for certain *XAI* method types. For evaluating *IB* methods which generate perturbed instances of the original input highlighting feature changes necessary to achieve a different outcome (counterfactual) or an increased likelihood of the current outcome (prototype), metrics like proximity and sparsity – quantifying the distance between the explanation and the original instance – are essential. Moreover, the obtained instance changes should be plausible, ensuring closeness to the data manifold and valid, confirming that the intended output is achieved (see Section 3.3)[1]. As *IB* explanations are often seen as guidance to recover from unfavorable predictions or strengthen favorable predictions in the real world, the realism of these explanations becomes crucial. A 'realistic' guidance should not only lie within the data manifold but also satisfy causal relations (e.g., additional machine operating time accompanied by an increase in machine age) and lead to changes in the real world when acted upon (i.e., changes translate to a causal effect). The importance of causality for generating meaningful explanations has been discussed in various works from social sciences (e.g., [35, 153]). Therefore, recent research, specifically on counterfactual explanations, focuses on enabling 'realistic' guidance by incorporating causal constraints [106, 107, 140] or learning real-world relationships directly from data [58, 182]. However, *S̲tructural C̲ausal M̲odel* (*SCM*) describing causal relationships, used within causal-based couterfactual methods [107], are often (fully or partially) missing for problems of interest, making it impossible to apply causal-based counterfactual explanation methods and/or evaluate explanations for a given problem. Therefore, real-world applications often rely on counterfactuals generated from learned real-world relationships. Especially for those non-causal approaches, it is hard to ensure and quantify to what extent causal relations and effects are satisfied due to a lack of ground truth. Hence, causal capabilities of counterfactual approaches are often approximated by evaluating user constraints or data distributions (e.g., [122, 173]). However, this type of evaluation only states if a counterfactual follows a certain distribution or meets some predefined constraint. No evidence is given as to whether the changes follow a causal chain.

This chapter addresses this obstacle by answering RQ II.(2):

RQ II.(2) How can we evaluate counterfactual explainers for real-world coherence?

To evaluate whether the explanations from different counterfactual methods indeed fulfill known causal relationships, this chapter introduces the notion of semantic meaningfulness for counterfactual generation methods. The semantic meaningfulness of a counterfactual generation algorithm describes the fraction of

[1] For the calculation of those metrics, please refer to Section 5.3. Also, note that those metrics are accessible in *XTSC-Bench*.

generated counterfactuals that are coherent with real-world relations. We propose two measures quantifying semantic meaningfulness: the fraction of counterfactual explanations that lead to the same outcome in the real world ('*Semantic Meaningful Output*') and the fraction of known causal relationships fulfilled by the counterfactual explanation ('*Semantic Meaningful Relations*').

This section is based on "Semantic Meaningfulness: Evaluating Counterfactual Approaches for Real-World Plausibility and Feasibility" (Explainable Artificial Intelligence - 2023). First, we formalize the problem (Section 7.1), then we provide details on the proposed metrics (Section 7.2). Lastly, in Section 7.3, we use the metrics to evaluate current counterfactual state-of-the-art methods.

7.1 Problem Formalization

Aligned with Figure 1.2, consider a classifier f that predicts some instances $X = \{x_0, \dots, x_n\}$ as class $Y = \{y_0, \dots, y_n\}$. A counterfactual method h generates, based on the observable input instances X and classifier g, explanations via counter-examples $X^{cf} = \{x_0^{cf}, \dots, x_n^{cf}\}$ for each $x \in X$. Each counterfactual explanation x^{cf} consists of features $x^{cf} = \{v_1^{cf}, \dots, v_m^{cf}\}$ and shows why a classifier f predicted class y for a data point x instead of counterfactual class y^{cf}. Depending on the used method h, the generated counterfactuals may or may not be coherent with real-world relations. To benchmark a counterfactual method h for its capabilities to depict real-world relations, we introduce the notion of semantic meaningfulness. Semantic meaningfulness is calculated as the fraction of counterfactual changes $\Delta x = x - x^{cf}$ that,

- $\mathbf{R_{7.1}}$: lead to the same outcome in the real world (*Semantic Meaningful Output*)
- $\mathbf{R_{7.2}}$: fulfill the known causal relationships (*Semantic Meaningful Relations*).

7.2 Approach

To quantify semantic meaningfulness and address $\mathbf{R_{7.1}}$ and $\mathbf{R_{7.2}}$, we need a way of modeling causal relationships. Causal models are a way of describing real-world relations. While it is often possible to draw a causal graph based on expert knowledge, assumptions about the structural equations – i.e., the relationship between variables, are generally not testable and may not hold in practice [178]. However, especially the modeling of variable interplay is essential to measure $\mathbf{R_{7.1}}$, as it creates the need to conclude the output y^{real} from the causal model based on x^{cf}. Fully specified *SCM*s – including the causal graph and structural equations – are hard to obtain. We evade the issue of missing *SCM*s by providing various datasets with causal graphs that can be used to benchmark new and existing methods. Therefore, the approach to measuring semantic meaningfulness consists of the metrics and proposed datasets of various difficulties derived from known *SCM*s.

In the following sections, we first define Semantic Meaningful Output and Relationship (Section 7.2.1), then we illustrate the metric implications on an example (Section 7.2.2). Section 7.2.3 highlights the relationship to previous work in more detail. Finally, Section 7.2.4 introduces the benchmarking datasets.

7.2.1 Semantic Meaningful Output and Semantic Meaningful Relations

We assume that one can capture real-world relations in the form of *SCM*s. Relying on the work of Pearl [175], we define a causal model as a triple (U, V, G) of sets where U is a set of latent background variables, V is a set of observed variables $V = \{v_1, \dots, v_n\}$, and G is a set of functions $\{g_1, \dots, g_n\}$ showing the relations between V. We assume observable relations $g_i = g_i(PA_i, U_i)$ with a deterministic function g_i depending on $V_i \subseteq V$ parents in the graph (denoted by PA_i) and a stochastic unexplainable (exogenous) variable $U_i \subseteq U$ for $i = \{1, \ldots, n\}$. To intervene on variable V_i in the *SCM*, one substitutes the equation $V_i = g_i(PA_i, U_i)$ with some constant $V_i = v$ for some v.

Given a counterfactual method h, a number of generated explanations $X^{cf} = \{x_1^{cf}, \ldots, x_n^{cf}\}$ and a (fully or partially defined) *SCM* (U, V, G), we postulate the following metrics:

> ➢ **Definition 11** (*S̲emantic M̲eaningful O̲utput*)**.** The counterfactual explanation method has *S̲emantic M̲eaningful O̲utput*, if intervening on the *SCM* (U, V, G) with $X^{cf} = \{x_1^{cf}, \ldots, x_n^{cf}\}$ yields the results $Y^{cf} = \{y_1^{cf}, \ldots, y_n^{cf}\}$ obtained by the classifier f.

Quantifying *S̲emantic M̲eaningful O̲utput* (*SMO*) presented in Definition 11 leads to the fraction of plausible counterfactual explanations, i.e., that lead to the same outcome in the real world (captured in the *SCM*).

$$\mathit{SMO} \overset{\text{def}}{=} \frac{1}{|X^{cf}|} \sum_{e \in X^{cf}} \mathbb{1}_{f(e)=\mathit{SCM}(e)} \tag{7.1}$$

Definition 11 only indicates if the counterfactual method generates plausible outcomes in the real world. Therefore, we propose a second metric to measure the fulfillment of known causal relationships. Although the output of an method might be semantically meaningful, the changes Δx made to obtain the counterfactual do not necessarily fulfill real-world relations. Therefore, we also introduce *S̲emantic M̲eaningful R̲elations* (*SMR*) in Definition 12, quantifying the fraction of causal relationships fulfilled.

> ➢ **Definition 12** (*S̲emantic M̲eaningful R̲elations*)**.** A counterfactual explanation method has *S̲emantic M̲eaningful R̲elations*, if intervening on the *SCM* (U, V, G) with $X^{cf} = \{x_1^{cf}, \ldots, x_n^{cf}\}$ yields for the features v^{cf} of all counterfactual explanations $v_i \neq v_i^{cf}$ to fulfill $f_i(PA_i, U_i)$ after intervention on the parents of a feature with $PA_i = v_{PA_i}^{cf}$ for each counterfactual explanation $\forall n$.

Quantifying Definition 12, leads to the fraction of fulfilled causal relations for an explanation x^{cf}.

$$\mathit{SMR} \overset{\text{def}}{=} \frac{1}{|X^{cf}|} \sum_{e \in X^{cf}} \left(\frac{1}{|e|} \sum_{i \in e} \mathbb{1}_{e_i = g_i(e_{PA_i}, U_i)}\right) \tag{7.2}$$

While Equation (7.1) can only be calculated with fully defined *SCM*s, Equation (7.2) allows making statements about relationship coherence for partially defined *SCM*s and thereby estimation of semantic meaningfulness.

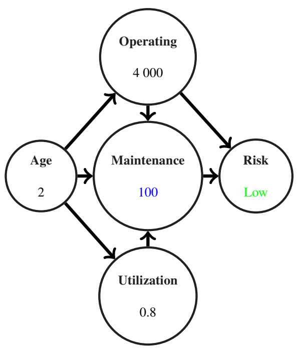
7.2.2 Illustration of metrics

Consider an *ML* model predicting machine failure risk based on machine characteristics (e.g., the age of the machine, its maintenance intervals). If the machine failure risk is high, an affected person might want to know why the risk of failure is high and how this risk can be lowered. A counterfactual explanation can be used to provide such an explanation as a "what-if"- scenario: e.g., "if the last machine maintenance had been 100 hours ago, the failure risk would be low". Consider an individual (input instance x) with { Age: 2, Maintenance: 2 000h, Utilization: 0.8, Operating Time: 4 000h} for which the black-box model f recommends a high risk of machine failure {Risk: High}. The objective of counterfactual generation algorithms h is to provide an instance x^{cf} for which the failure risk is low according to *ML* model f (validity).

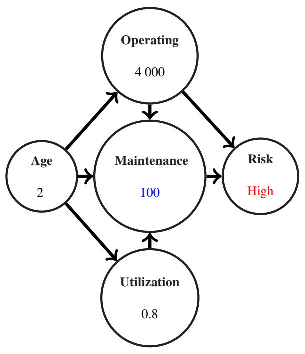
The proposed counterfactual explanation might look similar, e.g., x^{cf}= { Age: 2, Maintenance: 100h, Utilization: 0.8, Operating Time: 4 000h }. Assuming this explanation is valid ($y \neq y^{cf}$), it might or might not satisfy semantic meaningful output and relations. Figure 7.1 shows the possible scenarios of (dis)alignment with real-world relations (represented by the *SCM*).

First, the proposed counterfactual x^{cf} may satisfy both semantic meaningful output and relations (Figure 7.1a). For instance, we expect the operating hours to increase with age and the Utilization to say similar under the assumption of similar environmental factors. Since the counterfactual instance fulfills these expectations (i.e. age is increased as well as operating hours, but utilization is unchanged), we consider it feasible. Furthermore, suppose the counterfactual x^{cf} (that flipped the outcome of the classifier), also leads to a change in the outcome in the *SCM* (i.e., risk changes to low when the explanation is acted upon), then we consider it plausible. Second, the causal relations may be satisfied, but may not lead to the expected change in outcome according to the *SCM*. Figure 7.1b visualizes this case; Operating hours increase with age (as expected). However, as opposed to before, the proposed counterfactual does not lead to a change in outcome in the *SCM*.

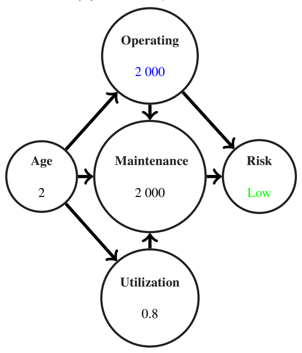
Consider now another counterfactual explanation, e.g., $x^{cf'}$ = { Operating Hours: 2 000h }. This counterfactual explanation $x^{cf'}$ might not cohere with causal relations, even though it might be valid. Operating hours is the only changing input, which does not align with the expectation that the machine need to be older in order to obtain such an operating hours increases. Furthermore, we expect an increase in income to be related to other factors, such as a higher machine utilization. If this is not reflected in the proposed counterfactual (as in Figure 7.1c), the semantic meaningful relations are not satisfied. However, it can still happen that the counterfactual explanation $x^{cf'}$ does lead to a change in the outcome in the *SCM* and therefore has semantic meaningful output due to an erroneous classification by f. Finally, it can be that both semantic meaningful output and relations are violated (Figure 7.1d).



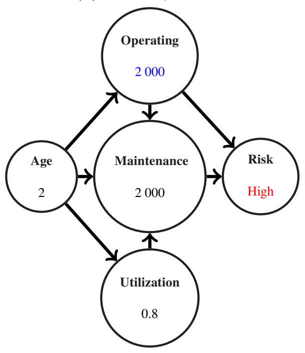
(a) SMO=1, *SMR*=1



(b) SMO=0, *SMR*=1



(c) SMO=1, *SMR*=0



(d) SMO=0, *SMR*=0

Figure 7.1: Example of *Semantic Meaningful Output* (*SMO*) and *Relation SMR* metrics on a simple machine failure example. Blue: the values changed in the proposed counterfactual explanation. Green: denotes *SMO*=1, which means that the *SCM* and *ML* model predict the same outcome. Red: indicates *SMO*=0, which means the predictions of the *SCM* and *ML* model are different.

7.2.3 Relation to prior work

In this section, we reflect on other related concepts, notations, and metrics proposed in the literature and clarify the relation to our work.

- **Karimi et al. [106]** propose a probabilistic approach to counterfactual generation based on fully or partially defined *SCM*s that takes uncertainty into account. In their evaluation, they introduce the notion of validity, which is, in their case, defined as the percentage of individuals for which the recommended action results in a favorable prediction under the true (oracle) *SCM*. This notion is closely related to our notion of *SMO*. In contrast to Karimi et al. [106], *SMO* expects a full counterfactual explanation as input (i.e., intervention on all endogenous variables) instead of a minimal action. Further, they do not consider the fraction of fulfilled relations (like *SMR*). We argue that checking if parent-child relationships are still fulfilled for generating meaningful counterfactuals is essential. A counterfactual output might be meaningful under a given *SCM* without fulfilling any given causal relationships.
- **Mahjan et al. [140]** propose the causal-edge score, which is the ratio of the log-likelihood of a counterfactual with respect to the likelihood of the original data given a causal-edge distribution. This is related to our metric *SMR*, but their proposed constraint feasibility measure only calculates the faction of counterfactuals that satisfy a given user-level constraint and not how well a method is able to capture semantic meaningful relations.
- **Afrabandpey et al. [3]** propose the notion of global (domain expert constraints) and local feasibility (end-user constraints) closely related to the constraint feasibility proposed by Mahjan et al. [140]. Both notions are highly dependent on human input and can, therefore, not fully capture semantic meaningful relations. However, their assumption that a domain expert can give feedback on the causal relationship between at least some features can be exploited to calculate *SMR* at least partly.
- **Karimi et al. [104]** formulate plausibility constraints with logic formulas to account for semantics such as immutable features. In contrast to our notion of plausibility (adhering to a realistic counterfactual), their notion of plausibility is restricted to consistency with training data (e.g., same data type and range) and the detainment of immutable features. As argued in Section 3.3, the reliance on training data is often insufficient.
- Finally, the **validity metric** is closely related to the *SMO* metric proposed in this work. In fact, where validity quantifies whether a counterfactual explanation can flip the classifier's decision, *SMO* quantifies whether a counterfactual explanation can flip the outcome of the *SCM* (which we assume represents the user's mental model of the world).

7.2.4 Benchmark datasets

To show the operationalizability of the metrics, we selected six datasets with known *SCM*s that we can use to measure semantic meaningfulness. The datasets have different complexities; the synthetic datasets contain only 3 variables (with varying relations), whereas the semi-synthetic datasets have a larger number of endogenous variables (8, 7 and 11) with a wider range of values. We use the following datasets for benchmarking (for causal graphs, see Appendix A.3):

Synthetic datasets. As a first example, we base ourselves on the synthetic toy dataset consisting of 3 variables used by Karimi et al. [106]. We consider all three variants proposed to test the semantic capabilities of the counterfactual methods on different types of relationships.

(a) Linear *SCM*

$$\begin{array}{ll} X_1 := U_1 & U_1 \sim MoG(0.5\mathcal{N}(-2, 1.5) + 0.5\mathcal{N}(1, 1) \\ X2 := -X_1 + U_2 & U_2 \sim \mathcal{N}(0, 1) \\ X3 := 0.05X_1 + 0.025X_1 + U_3 & U_3 \sim \mathcal{N}(0, 1) \\ Y := Bernoulli((1 + e^{-2.5(X_1+X_2+X_3)})^{-1}) & \end{array}$$

(b) Non-linear *SCM*

$$\begin{array}{ll} X_1 := U_1 & U_1 \sim MoG(0.5\mathcal{N}(-2, 1.5) + 0.5\mathcal{N}(1, 1)) \\ X2 := -1 + \frac{3}{1+e^{-2X_1}} + U_2 & U_2 \sim \mathcal{N}(0, 1) \\ X3 := 0.05X_1 + 0.025X_1 + U_3 & U_3 \sim \mathcal{N}(0, 1) \\ Y := Bernoulli((1 + e^{-2.5(X_1+X_2+X_3)})^{-1}) & \end{array}$$

(c) Non-additive *SCM*

$$\begin{array}{ll} X_1 := U_1 & U_1 \sim MoG(0.5\mathcal{N}(-2, 1.5) + 0.5\mathcal{N}(1, 1)) \\ X2 := 0.25\,\mathrm{sgn}(U_2)X_1^2 + (1 + U_2^2) & U_2 \sim \mathcal{N}(0, 0.25) \\ X3 := -1 + 0.1\,\mathrm{sgn}(U_3)(X_1^2 + X_2^2) + U_3 & U_3 \sim \mathcal{N}(0, 1) \\ Y := Bernoulli((1 + e^{-2.5(X_1+X_2+X_3)})^{-1}) & \end{array}$$

Nutrition dataset. For the next example, the aim is to predict survival based on demographic and laboratory measurements from the National Health and Nutrition Examination Survey [45]. We created an *SCM* for this first semi-synthetic dataset with the help of ShapleyFlow [256] and approximated the distribution with the help of the original dataset.

Age	$X_1 := U_1$	$U_1 \sim \mathcal{N}(25, 1)$
Sex	$X_2 := U_2$	$U_2 \sim \mathcal{N}(0, 1)$
Blood Pressure	$X_3 := 0.02X_1 + U_3$	$U_3 \sim \mathcal{N}(0, 1)$
SBP.	$X_4 := 0.12X_3 + U_4$	$U_4 \sim \mathcal{N}(80, 1)$
Pulse Pressure	$X_5 := 0.02X_4 + U_5$	$U_5 \sim \mathcal{N}(10, 1)$
Inflammation	$X_6 := U_6$	$U_6 \sim \mathcal{N}(0, 1))$
Poverty Index	$X_7 := U_7$	$U_7 \sim \mathcal{N}(0, 1)$
Sedimentation RAE	$X_8 := 0.03X_7 + U_8$	$U_8 \sim \mathcal{N}(0, 1)$
Risk	$y := (-0.21X_2 - 0.59X_1 + 0.03X_8 - 0.04X_7 + 0.02X_5 + 0.1X_4) > -6$	

Credit dataset. The second semi-synthetic dataset is a modification of the German Credit data and also extracted from the work of Karimi et al. [106]. The dataset contains information on personal loan applications, e.g., demographics and financial attributes, with the goal to distinguish people with good (i.e., approve loan) or bad (i.e., decline loan) credit risks.

Gender	$X_1 := U_1$	$U_1 \sim Bernoulli(0.5)$
Age	$X_2 := -35 + U_2$	$U_2 \sim Gamma(10, 3.5)$
Education	$X_3 := -0.5 + (1 + e^{-(-1+0.5X_1+(1+e^{-0.1X_2})+U3)})^{-1}$	$U_4 \sim \mathcal{N}(0, 0.25)$
Loan Amount	$X_4 := 1 + 0.01(X_2 - 5)(5 - X_2) + X_1 + U_4$	$U_4 \sim \mathcal{N}(0, 4)$
Loan Duration	$X_5 := -1 + 0.01X_2 + 2X_1 + X_4$	$U_5 \sim \mathcal{N}(0, 9)$
Income	$X_6 := -4 + 0.1(X_2 + 35) + 2X_1 + X_1X_3 + U_6$	$U_6 \sim \mathcal{N}(0, 4))$
Savings	$X_7 := -4 + 1.5 \cdot 1_{X_6>0}X_6 + U_7$	$U_7 \sim \mathcal{N}(0, 25)$
Output	$Y := Bernoulli((1 + e^{-0.3(-X_4-X_5+X_6+X_7+X_6X_7})^{-1})$	

Economic dataset. As last example, we consider a dataset with information on economic growth [263]. Xu et al. [263] modeled the relationship between economic growth and (factors related to) electricity consumption for China using data from the National Bureau of Statistics of China[2]. We adopted the *SCM* from Xu et al. [263] and approximate the distribution from the data. We further transformed the original regression problem into a classification problem by dividing the outcome (i.e., economic growth) into two classes, by setting a threshold equal to the mean.

Energy Source Struct.	$X_1 := U_1$	$U_1 \sim \mathcal{N}(0, 1)$
Informatization Level	$X_2 := 0.836X_4 + 0.464X_3 + U_2$	$U_2 \sim \mathcal{N}(0, 11)$
Ecological Awareness	$X_3 := 0.889X_4 + U_4$	$U_4 \sim \mathcal{N}(17, 90)$
Electricity Cons.	$X_4 := U_4$	$U_4 \sim \mathcal{N}(0, 100000)$
Electricity Investment	$X_5 := 0.898X_4 + U_5$	$U_5 \sim \mathcal{N}(0, 99999)$
Investment Other	$X_6 := 0.783X_5 + U_6$	$U_6 \sim \mathcal{N}(0, 15))$
Employment	$X_7 := 0.789X_4 + U_7$	$U_7 \sim \mathcal{N}(0, 70)$
Secondary Industry	$X_8 := 0.566X_4 + 0.561X_2 + U_8$	$U_8 \sim \mathcal{N}(0, 2000)$
Tertiary Industry	$X_9 := 0.537X_4 + 0.712X_2 + U_9$	$U_9 \sim \mathcal{N}(0, 2000)$
Prop. non-agriculture	$X_{10} := 0.731X_8 + 0.612X_9 + 0.662X_6 + 0.605X_2 + U_{10}$	$U_{10} \sim \mathcal{N}(0, 100)$
Labor Productivity	$X_{11} := 0.918X_4 + U_{11}$	$U_{11} \sim \mathcal{N}(0, 500000)$
Output	$Y := (0.538X_6 + 0.426X_7 + 0.826X_{11} + 0.293X_2 + 0.527X_{10} + 0.169X_4 + 0.411X_1) > 500000$	

7.3 Empirical Evaluation

With the help of the proposed metrics, we evaluate the capabilities of nine well-established counterfactual explanation methods for three *ML* models trained on three synthetic and three semi-synthetic datasets. We used the CARLA Recourse library [174] for the implementation of the counterfactual approaches and classifiers. The implementation of our metrics follows the CARLA implementation structure and can, therefore, easily be used in combination with the CARLA Benchmarking Tool. The code for our experiments can be found on GitHub[3].

Figure 7.2 shows a visualization of our experimental flow. First, we generate data based on the *SCM* to ensure data compliance. This is important to avoid any noise influencing the generation of counterfactuals. Next, we

[2] `http://www.stats.gov.cn/english/Statisticaldata/AnnualData/`
[3] `https://github.com/JHoelli/Semantic-Meaningfulness`

draw 10 000 samples and divide those samples into a 75/25 train/test split. We train our classifier with the training data as described in Section 7.3.1. Throughout the experiments, the (generated) datasets and trained classifiers remain the same. For each combination of dataset, classifier, and counterfactual method described in Section 7.3.2, we try to generate 250 counterfactual explanations. We evaluate the semantic meaningfulness of the resulting explanations with our proposed metrics and analyze the results in Section 7.3.3.

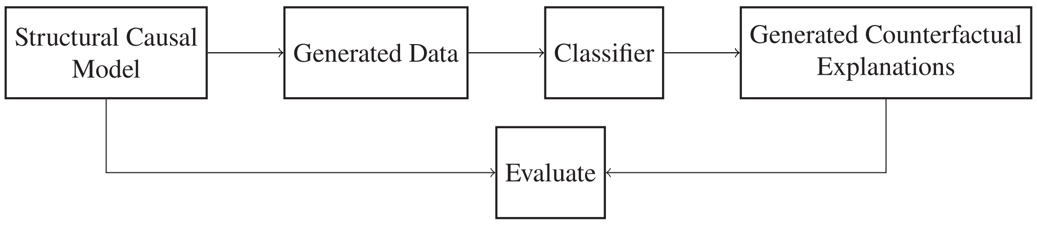


Figure 7.2: Flow of data in experiments.

7.3.1 Machine learning models

For each dataset, we train a Linear Model, a *Random Forest* (*RF*) with 5 estimators and a maximum depth of 5, and a *Multilayer Percepton* (*MLP*) with three hidden layers of size 18, 9 and 3. The *MLP* is trained for ten epochs with a batch size of 16 and a learning rate of 0.001. The performance of the classifiers are measured with the *Area Under the Curve* (*AUC*) and can be found in Table 7.1. Note that the classifiers achieve high or almost perfect (Linear, *MLP*) discrimination between classes for the three simple, synthetic datasets (i.e., Linear, Non-linear, and Non-additive *SCM*).

	Nutrition	Credit	Economic	Linear *SCM*	Non-linear *SCM*	Non-additive *SCM*
Linear	0.88	0.76	0.83	1.0	1.0	1.0
Random Forest	0.85	0.82	0.81	0.81	0.97	0.73
MLP	0.9	0.69	0.82	1.0	1.0	0.99

Table 7.1: Classifier performance as measured by *AUC* on each dataset.

7.3.2 Counterfactual methods

We compare 9 counterfactual generation methods that we categorized in causal-based (1x), constraint-based (4x), surrogate-based (2x) and model-specific (2x) methods, based on the taxonomy presented by Verma et al. [246]. In this section, we give a short description of each method. For a more detailed description, refer to Section 2.2.3. All methods were applied to the Linear and *MLP* models except for the model-specific methods, which were (only) applied to the *RF* models.

Causal-based:

- *Causal Recourse* (*CR*) [107] aims to find the minimal cost set of actions that results in a counterfactual instance favorable to the classifier g. A^* specifies the actions to be performed for a minimal causal recourse. Thereby, x^{SCF} is not a structural counterfactual obtained by intervening the *SCM*, but the minimal action needed to obtain x^{SCF}. As this method incorporates causal relations, we use it as a sanity check for our proposed metrics.

Constraint-based:

- *Wachter Counterfactual* (*W-CF*) [252] finds a counterfactual explanation with the smallest change (distance/cost) relative to the original data point that is classified differently. This method assumes independent features. As hyperparameters, the default parameters proposed by [174] are used.
- *Growing Spheres* (*GS*) [123] is a method that generates samples around the original data point by growing hyperspheres until the desired class label is found.

Surrogate-based:

- *Counterfactual Latent Uncertainty Explanations* (*CLUE*) [16] uses a generative model (variational autoencoder with arbitrary conditioning) that takes the classifiers' uncertainty into account and generates counterfactual explanations that are likely to occur under a data distribution. As hyperparameters, the default parameters proposed by [16] are used.
- *Counterfactual Conditional Heterogeneous Autoencoder* (*C-CHVAE*) [173] generates faithful counterfactuals by ensuring that the produced counterfactuals are proximate (i.e., not local outliers) and connected to regions with substantial data density (i.e., close to correctly classified observation). The counterfactual search is thereby included into a data density approximator, in this case a *Variational Autoencoder* (*VAE*). Counterfactuals are sampled from the latent space of the *VAE*.
- *Counterfactual Recourse Using Disentangled Subspaces* (*CRUDS*) [61] creates counterfactuals by using a conditional subspace *VAE* with the goal to satisfying the underlying structure of the data.
- *AR* [243] is based on integer programming and only applicable to linear models (e.g., logistic regression models, linear support vector machines). For non-linear models, coefficients are approximated with LIME [191]. As we do not only apply AR to linear models, but also need to approximate coefficients with LIME for *MLP*, we categorize AR as a surrogate-based method.

Model-specific:

- *FOCUS* [138] is an method for finding counterfactuals for non-differentiable models, e.g., tree ensembles. The method uses a probabilistic approximation of the original tree ensemble.
- *FT* [241] exploits the internals of a tree-based ensemble classifier by tweaking the feature-based representation of a true negative instance such that the modified instances result in a positive classification when re-inputted to the classifier.

7.3.3 Results

We present results evaluating nine counterfactual explanation methods using the proposed *SMO* and *SMR* metrics across six datasets. In particular, we investigate the semantic meaningfulness of different counterfactual methods (Section 7.3.3.1), the influence of the classifier (performance) on the semantic meaningfulness of the counterfactual methods (Section 7.3.3.2), and the relationship between the two semantic meaningfulness metrics (Section 7.3.3.3).

7.3.3.1 Existing (non-causal) counterfactual methods vary in semantic meaningfulness.

Figure 7.3 shows the results of evaluating the semantic meaningfulness of different counterfactual methods. As a sanity check, we first applied a causal-based method. Due to the large increase in computation time for an increasing number of endogenous variables and relations, only results for the synthetic datasets could be obtained (see Figure 7.3a - 7.3c). Using *CR* on these datasets, *SMO* has a median of 1 and a mean slightly below 1, indicating that our proposed metric aligns with the *SCM* and works as intended. Erroneous classifications of the *ML* classifier can explain why the mean is slightly less than 1 for *SMO*. *SMR* is equal to 1 for both the Linear and Non-linear *SCM*, showing causal relations are satisfied. For the Non-additive *SCM*, on average, only 2 out of 3 possible relationships could be fulfilled. This is because the counterfactuals returned by *CR* are not supposed to fulfill all constraints of the graph, but rather include a minimal change that would conclude in the desired result by iterating through the graph (i.e., *CR* returns a minimal action set). Overall, the nearly perfect values for *SMO* and *SMR* found for the causal method indicate that our method works as intended.

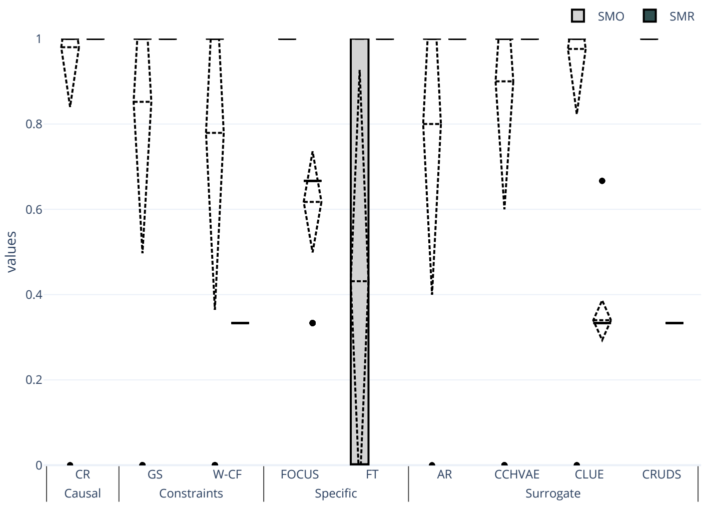
We further find that the constraint-based method *W-CF* scores very low on *SMR*, indicating that this method (assuming independence between features) is non-compliant with known causal relations. Combined with the slightly higher *SMO*, the method generates counterfactuals that are likely to have a realistic output but with a combination of features that is unlikely to be observed in the real world. *GS* offers a higher *SMR* compared to *W-CF*. The better performance compared to *W-CF* might result from the optimization function that includes a sparsity constraint (a penalization on feature changes) and their optimization heuristic.

Next, we observe that the surrogate-based methods *C-CHVAE*, *CLUE*, and *CRUDS* score relatively high on *SMO*. However, the performance on *SMR* varies; whereas *CLUE* and *CRUDS* score very low across all datasets, *C-CHVAE* scores reasonably well. This occurs despite all *VAE*-based methods being trained with the same parameter settings. A reason for the worse performance of *CRUDS* and *CLUE* might be the optimization function based on the decoded latent (similar to the *W-CF* function with additional restrictions), which can result in counterfactuals changing the outcome but not fulfilling the causal relations. Meanwhile, the nearest neighbor style search in the latent space provided by *C-CHVAE* leads to possibly further distance counterfactuals by sampling in the latent space for distance to a higher likelihood of obtaining learned constraints. For AR the linear coefficients were approximated by applying LIME [191]. The quality of these linear models can largely differ, leading to higher variability and worse average values for *SMO* and *SMR* compared to deep learning based VAEs. Note that for the Economic dataset, *CLUE* and AR are missing; *CLUE* was not able to find valid counterfactuals and the integer programming problem size of AR is too large to be calculated with the open source version of CPLEX.

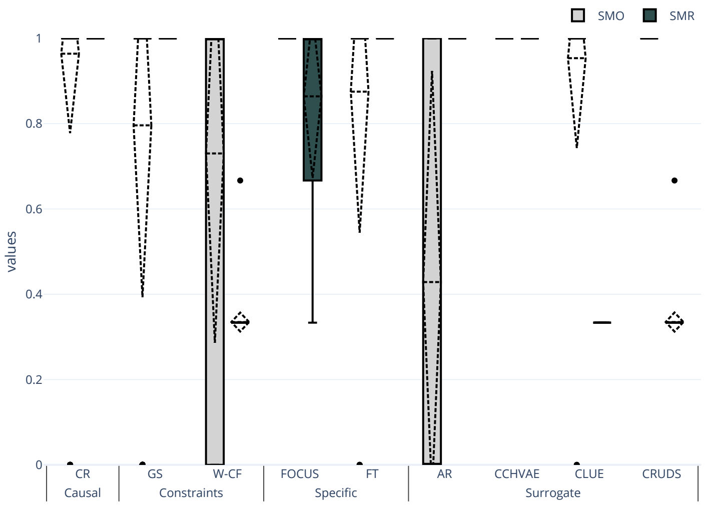
Finally, for the model-specific methods (FT and FOCUS), we observe a high *SMR*, but a lower *SMO*. These methods could only be applied to Forest Classifiers, but the high *SMR* shows that they could capture the causal relations from the underlying forest well.

From the model-agnostic and non-causal counterfactual methods, we conclude that surrogate-based methods perform slightly better than constraint-based methods. This is expected as these methods consider the data manifold. From the constraint-based methods, only *C-CHVAE* and *CRUDS* were applicable to all datasets. *C-CHVAE* performed consistently best on *SMR* across all datasets, while having a similar *SMO* compared to other methods (e.g., *CRUDS*). Surprisingly, model-specific methods also captured causal relations well and even outperformed *C-CHVAE* on the Non-additive *SCM* and Economic datasets. The weak performance of all methods on *SMR* for the Economic dataset can be explained by the small number of exogenous variables (#2) in combination with a high number of endogenous variables (#9)[4]. Overall, the results show the performance of the counterfactual methods differs between datasets and diverges for the most complex dataset (Figure 7.3f) compared to the simpler datasets (Figure 7.3a-7.3e).

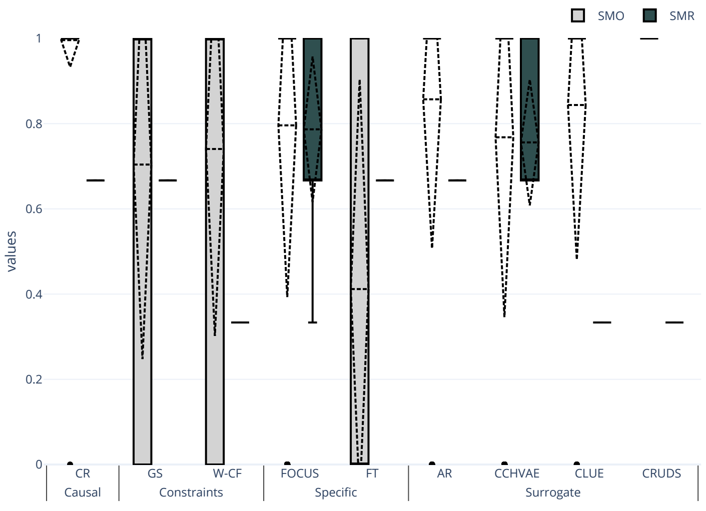
[4] While 1 of 3 relations are always fulfilled for the synthetic dataset, this is only the case for 2 out of 11 are in the Economic dataset. This leads to a significantly lower minimum performance (and can explain the worse results).



(a) Linear *SCM*.

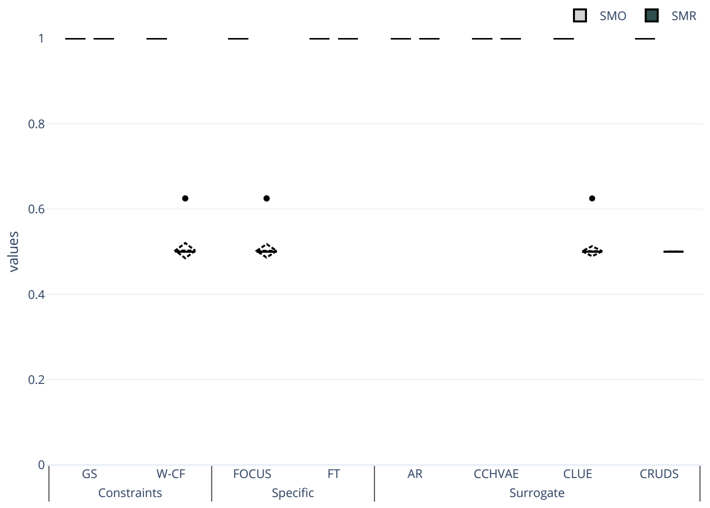


(b) Non-linear *SCM*.

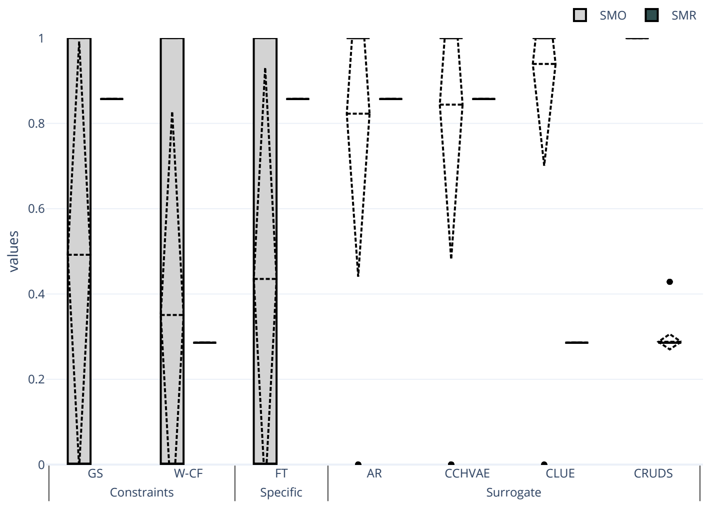


(c) Non-additive *SCM*.

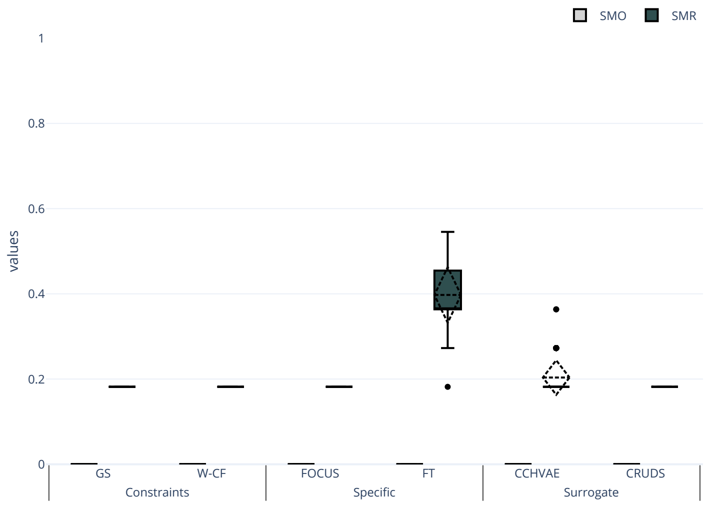
Figure 7.3: Semantic Meaningful Output (SMO) and Relation (SMR) metrics averaged over *ML* models. The bars show the median and the interquartile distance. The dotted lines show the mean and the standard deviation. When only one horizontal line is shown for a given counterfactual method, the mean, standard deviation, and median are the same for *SMO* and/or *SMR*.



(**d**) Nutrition.

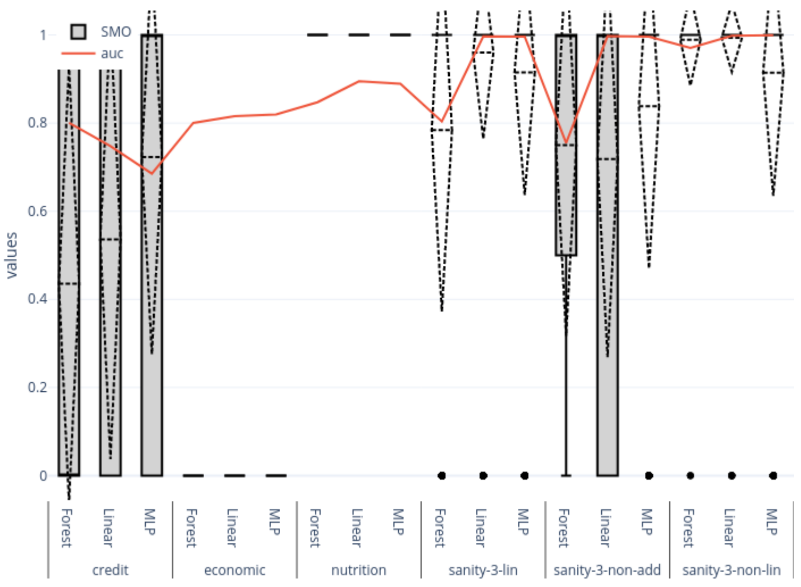


(**e**) Credit.

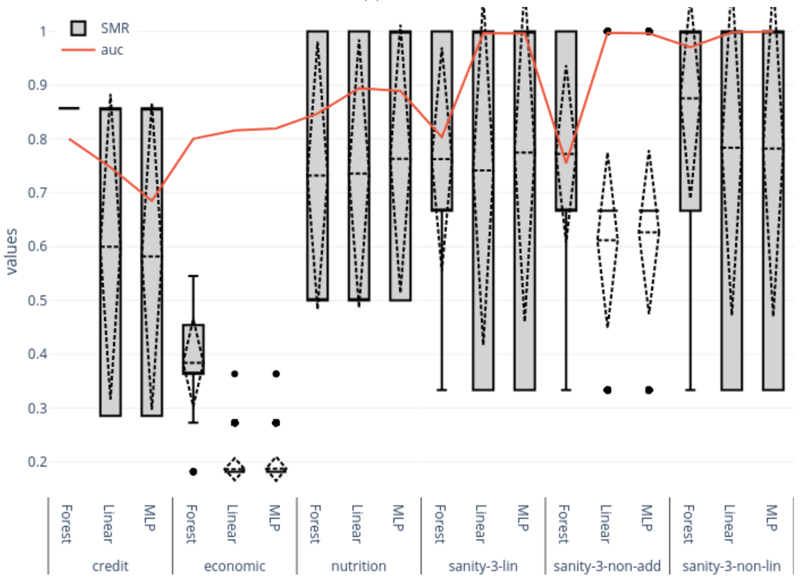


(**f**) Economic.

Figure 7.3: Semantic Meaningful Output (SMO) and Relation (SMR) metrics averaged over *ML* models. The bars show the median and the interquartile distance. The dotted lines show the mean and the standard deviation. When only one horizontal line is shown for a given counterfactual method, the mean, standard deviation, and median are the same for *SMO* and/or *SMR*.



(a) SMO.



(b) SMR.

Figure 7.4: Semantic Meaningful Output (SMO) and Relation (SMR) metrics performance averaged over counterfactual methods. The red line denotes the classifier performance in terms of AUC.

7.3.3.2 The importance of a good classifier for meaningful counterfactual explanations.

Next, we investigate between-classifier differences. Figure 7.4 shows the results for different datasets and *ML* models, averaged over counterfactual methods. The classifier performance is visualized with the red line.

In Figure 7.4a, we see that the performance of *SMO* is not directly related to the quality of the classifiers. For instance, *MLP* has the highest *SMO* on the Credit dataset, even though it achieves the lowest AUC. However, *SMO* is at least indirectly related to performance via data complexity. The *SMO* and classifier performance are both highest for the simple synthetic datasets. In Figure 7.4b, we observe better classifier performance roughly corresponds to better *SMR* – except for Non-additive *SCM*. For this dataset, both the Linear and *MLP* model, show a promising classifier performance, even though the counterfactual method based *SMR* results in a worse score than for Random Forest. It can be concluded that Linear and *MLP* based counterfactual methods have issues creating semantic meaningful counterfactuals for non-additive relations. The model-specific methods evade this issue, using the features extracted by the classifier models.

Although the results indicate that the counterfactual explanations generated on classifiers with higher AUC were more coherent with causal relations (independent of the counterfactual generation method), we conclude good classifier performance is necessary but not sufficient.

7.3.3.3 Semantic Meaningful Output and Relations measures are both needed.

Finally, we evaluate the connection between *SMO* and *SMR*. Figure 7.5 shows the connection between *SMO*, *SMR*, and the notion of validity (i.e., a counterfactual that changes the class predicted by the *ML* model from y to y^{cf}). We found that, out of the total 23,750 counterfactuals we tried to generate, 71% were valid. From those valid counterfactuals, 71% achieved a semantic meaningful output (SMO=1), and only 42% of those also fulfilled complete semantic meaningful relations (SMR=1). This means that 58% of the counterfactual explanations had semantic meaningful output without satisfying semantic meaningful relations (SMR$<$1). As semantic meaningful output can be caused by an erroneous classification of the desired class by the *ML* model or achieved with unrealistic feature combinations, evaluating feature combinations for causal relations is important. This underscores the need for the combined use of the two metrics. On the other hand, 7% of the counterfactuals with perfect semantic relationships (SMR=1) did not satisfy semantic meaningful output, which could be caused by imperfect classifications of the *ML* classifier.

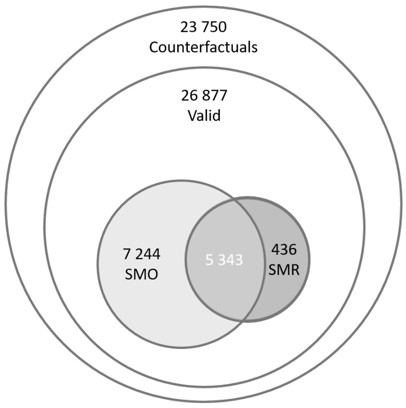


Figure 7.5: Venn Diagram visualizing overlap between Semantic Meaningful Output (SMO), Semantic Meaningful Relations (SMR), and Validity metrics for counterfactuals. Note that this plot only includes perfect semantic relationships (SMR=1).

7.4 Practical Implications

In this section, we highlight how the proposed metrics and benchmark datasets are useful for different target audiences:

- Developers of counterfactual methods can use the metrics to quantify the causal capabilities of their methods on the provided set of benchmark datasets, enabling direct comparisons of existing and new methods in a transparent, replicable, and unified way. The datasets (and *SCM*s) with varying levels of complexity allow developers to understand limitations and identify directions for improvement.

- Practitioners and users of counterfactual explanations can use the metrics with the benchmark datasets to get insight into the causal capabilities of different methods. When used in combination with other potentially important metrics (e.g., sparsity [105] and actionability [246]), this can be used to guide the choice between different explanation methods by examining several metrics (and potential arising trade-offs) for datasets (and *SCM*s) with varying levels of complexity. For example, for classification model debugging, explanations via edge cases might be interesting (acceptance of poorer causal capabilities for lower sparsity and proximity) to understand the classifier's inner workings. However, if counterfactual explanations are intended for the use in consequential decision making (i.e., to provide an explanation to individuals affected by models), counterfactual methods replicating the real world are preferred (acceptance of higher sparsity and proximity for better causal capabilities).

7.5 Summary and Discussion

In this section, we proposed two metrics in combination with (semi-)synthetic datasets to answer RQ II.(2). *Semantic Meaningful Output* (*SMO*) and *Semantic Meaningful Relations* (*SMR*), the two proposed metrics, measure the ability of counterfactual generation methods to depict real-world relations, both with respect to the final output and the relationship fulfillment. This allows for benchmarking new and existing counterfactual methods based on the fraction of explanations that lead to the same outcome in the real world (*SMO*) and the fraction of fulfilled causal relationships *SMR*. The – typically existing – lack of ground truth when measuring to what extent causal relations and effects are satisfied for (non-causal) counterfactual generation methods is overcome by providing six (semi-) synthetic datasets of various complexity. Based on the six (semi-) synthetic datasets evaluated in this work, we conclude that nine well-established counterfactual methods differ in semantic meaningfulness, with drastically decreasing overall performance for more complex datasets. We found surrogate methods work well for simple datasets, but could perform better on datasets with a larger number of variables (e.g., Economic dataset). Further, we show that the *ML* model must sufficiently capture causal relations for the counterfactuals to align with the *SCM*. Although this might seem straightforward, there is little work considering classifier performance when evaluating counterfactual desiderata (e.g., realisticness or feasibility). Finally, the proposed metrics can only capture the notion of semantic meaningfulness when used in combination; observing semantic meaningful output (i.e., plausible explanations) for a counterfactual generation method does not necessarily imply semantic meaningful relations (i.e., feasible explanations), and vice versa.

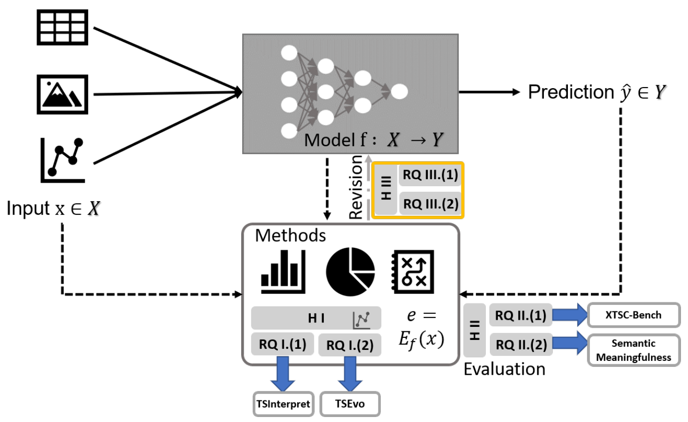
Note that our notion of semantic meaningfulness only evaluates if a counterfactual is causally consistent. Whether a counterfactual is actionable (i.e., a user can change the output of the model by doing an action), is not part of this work. Usually, actionability is considered when generating counterfactual explanations by setting features as mutable or immutable (e.g., [246]). Furthermore, evaluating semantic meaningfulness on any real-world dataset is still an open issue due to the complexity of obtaining a fully specified *SCM*.

We only evaluated the counterfactual methods on a small set of models (Linear, Random Forest, Multilayer Perceptron) that were trained on relatively simple (semi-)synthetic datasets. Even though the proposed metrics and benchmarks work independently of the model and the counterfactual method chosen, not all counterfactual methods could be applied to all models and datasets. First, we could not apply the causal-based methods to the three semi-synthetic datasets (due to the large computation power needed), which could have given a better insight into the relative performance of these methods in relation to non-causal methods. Second, only the model-specific methods were usable for Random Forest, as the remaining counterfactual methods (currently) only work for gradient-based models in the CARLA Recourse library.

In future work, it would be interesting to analyze more classifiers (e.g., Convolutional Neural Networks) with varying performance, more complex *SCM*s (e.g., containing more than 10 endogenous variables and more complex relations), and extend the notion of semantic meaningfulness to different data types (e.g., images [188], time series [126]) to expand the applicability of our approach. Our study showed that capturing causal relations while generating counterfactuals is still an open problem. For one, counterfactual generation methods based on causal relations lack real-world applicability. Moreover, none of the non-causal methods were able to consistently create semantic meaningful counterfactuals, resulting in unreliable explanations not necessarily coherent with known causal relationships. Although there might not be a direct relationship between the performance of *SMO*/SMR on the benchmark datasets and the dataset of interest, the metrics can be used to develop an understanding of the limitations of explanation generating methods, which is crucial for the adequate application and interpretation of counterfactual explanation methods. Quantifying semantic capabilities is just the first step to developing counterfactual methods with better causal capabilities.

Part IV

Model Revision



Part II and **Part III** focused on making *XAI* more applicable in the industrial domain by providing explainers for time series and approaches for evaluating an explainer's performance. This part focuses on how explanations and explainers can be leveraged to include domain knowledge into *AI* models and thereby deals with challenge 3 **model revision**. The following section, Chapter 8, introduces and evaluates Continuous Explanatory Interactive Machine Learning, allowing continuous model feedback on both the explanation and the model prediction.

8

Continuous Explanatory Interactive Machine Learning

While **Part II** and **Part III** focused on increasing the applicability of *XAI* in the industrial domain by proposing new algorithms, metrics, and democratized frameworks, this section leverages *XAI* methods and feedback provided on the explanations generated by *XAI* algorithm to improve the performance of the underlying *AI* model. To enhance the application of *AI* and *XAI* in industrial settings, models need to adapt to fast-changing conditions and cope with new information, ideally under human oversight, to prevent errors (see Section 3.1 and Section 3.4). *XAI* allows the recognition of model shortcomings, however, it does not allow model and error revision under human supervision. *EXplanatory Interactive Learning* (*XIL*) resolves those errors by interacting with the explanations. However, the paradigm of *XIL* is built around an active learning setting. In industrial settings, this inability to enhance the model on the fly leads to additional costs due to the longer latency of affected industrial processes. *Continuous Learning* allows *DL* models to learn an increasing number of prediction tasks (class learning) and adapt to changed input data distributions (domain learning) iteratively, however, without human supervision.

While there are already paradigms that allow model improvement with explanation interaction and iterative learning through time, combining both paradigms and thereby putting continuous learning under human supervision is still an open problem. The promising direction of combining both paradigms aims at solving the lack of (a) consecutive learning, (b) transferability, (c) human supervision, and (d) the inclusion of domain knowledge that hinders the usage of *DL* models in many applications. This chapter addresses RQ III.:

RQ III. How can we enable continuous, explanatory and interactive model improvement?

RQ III.(1) Which combinations of continuous, explanatory and interactive is the most suitable?

RQ III.(2) How does continuous, explanatory and interactive model improvement perform on realistic data?

To answer these questions, this section proposes a combination of Explanatory, Interactive and continuous Machine Learning, combined in a framework called Continuous *EXplanatory Interactive Learning* (*CXIL*) to enable lifelong learning by including domain knowledge. Thereby, *CXIL* focuses on fine-tuning, domain, and task learning under human supervision to enable model training in production scenarios.

First, we formalize the problem at hand (Section 8.1). Then, we lay out the approach in Section 8.2. Section 8.3 evaluates the settings in a domain adaption, task learning, and fine-tuning scenario. Then, we transfer the results on a real-world data set (Section 8.4). Finally, Section 8.5 summarizes and discusses the results.

8.1 Problem Formalization

In correspondance to Section 1.2, consider a supervised classification problem $f : X \rightarrow Y$, where $X = \{x_1, \ldots, x_n\}$ is the data and $Y = \{y_1, \ldots, y_n\} \in \mathbf{C}$ the target to be predicted. $\mathbf{C}$ denotes the finite space of all known classes. Most supervised learning methods assume that each example (x_i, y_i) is an identically and independently distributed (iid) sample from a fixed probability distribution P. The prediction of y can be performed with different types of predictors f. In this context, we assume a single shared neural network with parameters θ, where $\{\theta_{ij}\}$ corresponds to the weight of the connection between the neurons n_i and n_j in two consecutive layers. We assume that we already have historically available data $D_B = (X_B, Y_B)$ with which we can pre-train a machine learner f_B in traditional batch mode. During the usage of such a traditionally supervised machine learning model f_B, new data x_t, y_t becomes available iteratively. Due to the changes in production scenarios, available data can include previous unseen classes $y_t \notin \mathbf{C}$ (e.g., due to a new manufacturing option), a diverging data distribution $P_B(x, y) \neq P_t(x, y)$ (e.g., due to the increasing age of the machine) or the application of the model to a newly built production line.

So, we are building a framework, that can cope with

- new domains $f : X \rightarrow Y$, where B denotes the original domain and $B + 1$ the new domain with $P_B(x, y) \neq P_{B+1}(x, y)$
- new classes $f : X \rightarrow \mathcal{G}$ with $\mathcal{G}$ being the global label and $\mathcal{C} \subset \mathcal{G}$

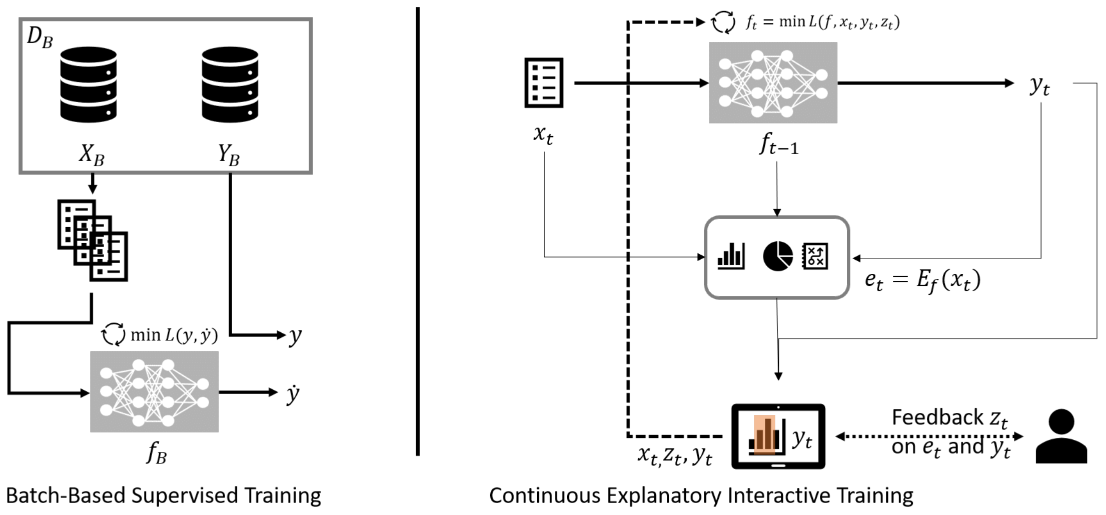
while still being under human supervision to ensure that the model does not pick up confounders.

8.2 Approach

Given the pre-trained model f_B, we want to consecutively update the model weights θ_{f_B} with respect to new data points (x_t, y_t) as time t progresses under human supervision. For that, we assume the availability of an explainer E_f for the machine learning model f that is able to visualize the model's inner workings to a human agent. Starting with $f_0 = f_B$, we provide a human agent with the result $\hat{y}_t$ of model f_{t-1} on a new item x_t with an explanation e_t. The human interacts with the explanation e_t by providing feedback z_t or a new target y_t. The new data triple x_t, y_t, z_t is instantly played back into the predictor f_{t-1} with the help of an interactive updating algorithm G and a continuous learner C, resulting in a new machine learning model f_t as shown in Figure 8.1.

8.2.1 Explainer E_f

In the context of *CXIL*, any explainer returning a featurewise importance (feature attribution) is suitable. Feature attribution explains individual predictions by attributing each input feature according to how much it changed the prediction (negatively or positively). More formally, given that model f makes a prediction $\hat{y}_t = f(x_t)$ for input x_t, an explanation method E_f finds an explanation $e_t = E_f(x_t) \in \mathcal{R}^d$, where d denotes the number of features. In the case of feature attribution methods, the explainer E_f assigns an attribution ϕ_i to explain the importance of a feature i, resulting in $E_f(X) = (\phi_1, \ldots, \phi_d)$. In general, any explainer returning a feature-wise relevance is suitable (e.g., Input X Gradient [216], *SHAP* [139], GRAD-CAM [212]).



D_B
X_B
Y_B
$\min L(y, \dot{y})$
y
$\dot{y}$
f_B
Batch-Based Supervised Training
$f_t = \min L(f, x_t, y_t, z_t)$
x_t
f_{t-1}
y_t
$e_t = E_f(x_t)$
x_t, z_t, y_t
Feedback z_t on e_t and y_t
Continuous Explanatory Interactive Training

Figure 8.1: Visualization of the Framework.

In this case, we use Input X Gradient, as it is applicable to all types of neural networks while still taking the gradients into account[1].

8.2.2 Continuous Learner C

Incremental learning of new information from non-stationary data streams, is a key feature of natural intelligence but a challenging problem for deep neural networks. Deep neural networks are prone to catastrophic forgetting, which can result in an abrupt performance decrease or, in the worst case, a complete loss of knowledge from previous tasks. *CL* methods focus on preventing the loss of knowledge learned from previous distributions. Algorithms for continuous learning are divided into three categories: regularization-based algorithms that regularize activations, outputs of networks (e.g., [132]), or the parameters of networks (e.g., [115]), architectural algorithms that modify the neural network's architecture (e.g., [201]), and memory-based algorithms that keep a subset of the previous data (e.g., [201]). The following sections describe the continuous learner used in this framework in more detail. Architectural-based methods are excluded in this framework as we assume that architectural changes in production environments are not practicable.

8.2.2.1 Regularization-Based Methods

Regularization-based methods combine the task-specific loss L_C with an extra loss term L_{Reg} to avoid forgetting. Usually, the regularization term L_{Reg} is added to the loss L_C with a scaling factor λ:

$$L_{total} = L_C(x, y; \theta) + \lambda L_{Reg}(\theta). \tag{8.1}$$

[1] GradCam is only applicable to Convolutional Neural Networks, and *SHAP* is a perturbation-based method that approximates the influence of each feature by fitting an interpretable model instead of relying on the model's gradients.

The regularization term aims to keep the model's weights θ from diverging too far from the weights of the previous tasks $\hat{\theta}^{K-1}$. In general, the regularization term consists of the product of the change in network parameter values from their original values and their estimated importance Ω.

$$L_{reg}(\theta) = \sum_i \Omega_i^{K-1}(\theta_i - \hat{\theta}_i^{K-1})^2 \tag{8.2}$$

Most methods differ only in how they calculate the parameter importance factor Ω. In the following, we describe how this importance factor is defined in different methods.

EWC *E̲lastic W̲eight C̲onsolidation* (*EWC*) [115] measures the importance via the Fisher Information Matrix. The Fisher Information Matrix approximates the curvature of the loss function, giving insights into how sensitive the network is to weight changes. *EWC* computes the importance weight Ω_i^{K-1} of parameter i as the squared gradient of loss function L_C w.r.t. the parameter of the previous task $\hat{\theta}_i^{K-1}$:

$$\Omega_i^{K-1} = E_{(x,y)}\left[\left(\frac{\partial L_C}{\partial \hat{\theta}_i^{K-1}}\right)^2\right] \tag{8.3}$$

SI *S̲ynaptic I̲ntelligence* (*SI*) [269], in contrast to *EWC*, calculates the weight importance on the fly. The strength of each parameter's penalty depends on how important that parameter ($w_i^{(k)}$) is thought to be for the tasks learned so far. The estimated importance of a parameter Ω_i^{K-1} for the first $K-1$ tasks is given by:

$$\Omega_i^{K-1} = \sum_{k=1}^{K-1} \frac{w_i^{(k)}}{(\Delta_i^{(k)})^2 + \epsilon} \tag{8.4}$$

$\Delta_i^{(k)}$ denotes the difference of parameter i between tasks and $w_i^{(k)}$ is estimated as the per-parameter contribution to the change of the loss with:

$$w_i^{(k)} = \sum_{t=1}^{N_{iters}} (\theta_i[t^{(k)}] - \theta_i[(t-1)^{(k)}]) \frac{\partial L_{total}[t^{(k)}]}{\partial \theta_i} \tag{8.5}$$

MAS *M̲emory A̲ware S̲ynapses* (*MAS*) [10] derives the parameter importance from the respective output layers' sensitivity to parameter changes. O_k is the network output for the data sample x_k on task k and 0 is a zero vector of the same size. l_2 denotes the l_2-norm of the output layer O.

$$\Omega_i^{K-1} = \frac{1}{K} \sum_{k=1}^{K-1} \left| \frac{\partial l_2^2(O_k(x_k), 0)}{\partial \theta_i^k} \right| \tag{8.6}$$

8.2.2.2 Memory-Based Methods

Memory-based methods try to prevent catastrophic forgetting by holding onto a subset of elements from previous distributions in a buffer $\mathcal{B}$ that can be added to the training set to mitigate the drift. With this methods, the model can learn from the new distribution while retaining information from the past. Thereby, memory-based methods either save raw examples of past experiences (e.g., [201]), or generate examples from previous tasks (e.g., [262]). Algorithm 9 shows the generalized memory-based training method. Before updating the model f, data samples from previous tasks are drawn, and the model update is followed by the buffer update (e.g., always store the newest samples of the previous task). The working principles of the data sampling, the model update, and the buffer update depend on the memory-based method deployed.

Algorithm 9 Memory-Based Training Method

```
$\mathcal{B} \leftarrow \{\}$
**for** $(x, y)$ **do**:
    **if** $\mathcal{B} \neq \{\}$ **then**
        $D_{Train} = SAMPLE(\mathcal{B}) \cup (x, y)$
    **else**
        $D_{Train} = (x, y)$
    **end if**
    Update $f$ with $D_{Train}$
    $\mathcal{B} \leftarrow \mathcal{B} \cup (x, y)$
    **if** $|\mathcal{B}| > M$ **then**
        $\mathcal{B}$ = DISCARD_SAMPLES($\mathcal{B}$)
    **end if**
**end for**
```

In a basic setting, we use buffer $\mathcal{B}$ with a memory allocation of size $m = M/K$ for each task K. M denotes the maximal number of samples to be stored. In cases where no task label is provided, K is specified by the number of unique labels. If the memory m is fully allocated, the samples in the buffer are replaced with the newest samples obtained for each task k. The model updates are obtained with Stochastic Gradient Descent (SGD) and a cross-entropy loss.

8.2.2.3 ***Label Trick*** **(*LT*)**

All regularization-based methods described above need a task ID to inform the algorithm that the distribution switched. In reality, this is often not feasible. To allow continuous learning without known task IDs, [270] introduced the label trick. The trick is to train only the heads that correspond to labels included in the current batch (i.e., heads of the current task), in contrast to the common practice of training all heads in such a scenario. For example, if a batch consists of samples only with labels $(3, 4)$, we calculate the loss only over the heads corresponding to $(3, 4)$, instead of over all heads $(0, 1, \dots, \mathcal{C} - 1)$.

8.2.3 Explainable Interactive Machine Learner G

For interacting with explanations, there are four scenarios to be considered: a) Right for the Right Reasons, b) Right for the Wrong Reasons, c) Wrong for the Right Reasons, and d) Wrong for the Wrong Reasons. In scenario a), the model is updated by the continuous learner without human feedback. In cases c) and d), the model is only updated based on the prediction target. In instance b), where the prediction is correct, but the explanation is wrong, the user is asked for feedback. The feedback is provided as an annotation matrix $A \in \{0, 1\}^{N \times D}$, where N is the number of annotated observations, D is the number of features, and incorrect parts of an explanation are denoted by 1. We include two strategies for revising a model based on the feedback obtained in A: Dataset Augmentation (CAIPI [237]) and Loss Augmentation (Right for the Right Reasons [195]).

8.2.3.1 CAIPI

CAIPI [237] includes the explanation feedback with the help of data augmentation. New data samples, specifically tailored to remove the wrong features, are added to the training dataset. The explanation corrections contained in the annotation matrix A are used to transform the original input x_t into counter-examples. A counterexample is defined as $(\hat{x}_t, y_t)$, where y_t is, if needed, the corrected label and $\hat{x}_t$ is the identical input, except for the features annotated with $A > 0$. These features are either randomized, changed to an alternative value, or substituted with the value of the same feature appearing in other training examples of the same class.

8.2.3.2 *RRR*

Right for the Right Reasons (*RRR*) [195] provides a loss function for differential models that penalizes the irrelevant parts of explanations based on the annotation matrix A.

$$L_{exp} = \sum_{n=1}^{N} (A_n \times E_{f_\theta}(x_n))^2 \tag{8.7}$$

The explanation penalty L_{exp} is thereby added onto the prediction loss with a regularization factor λ.

$$\min_\theta L_C(f_\theta(X), y) + \lambda L_{exp}(E_{f_\theta}(X), A) \tag{8.8}$$

8.3 Empirical Evaluation

Before applying *CXIL* to our real-world use case, we evaluate the suitability of *CXIL* on synthetic datasets of the most common data types in industrial settings with known ground truths. Utilizing synthetic data enables the simulation of user feedback without human bias and, therefore, the evaluation of the technical feasibility of the proposed method. This section evaluates the capabilities of *CXIL* in hindsight to:

- Fine-tuning (Section 8.3.1): How do pre-trained neural networks correspond to fine-tuning with explanation interaction?
- Continuous Learning (Section 8.3.2): How do continuous learners react to learning with explanation feedback?

To enable the evaluation of continuous learning with human feedback, we utilize three synthetic datasets. Thereby, the synthetic datasets need to include multiple tasks to allow the evaluation of continuous learning and a ground truth that highlights important and irrelevant parts of the input data for the prediction task for model feedback simulation. Using synthetic datasets allows the simulation of human feedback without the need for real human interaction and eliminates aspects related to human cognition. Based on those requirements, we generate synthetic data for an image, tabular, and time series classification task.

Inspired by the popular continuous learning benchmarking dataset Split MNIST [244], we split Decoy MNIST into consecutive tasks to generate ground truth data and a rule we can apply to interact with the model. On Decoy MNIST [195], images have 4×4 gray swatches in randomly chosen corners. The swatches shades are functions of their digits y in training (in particular, $255 - 25y$) and random in the test set. In correspondence to Split MNIST, we split the items of Decoy MNIST into five tasks containing two consecutive classes (see Figure 8.2).

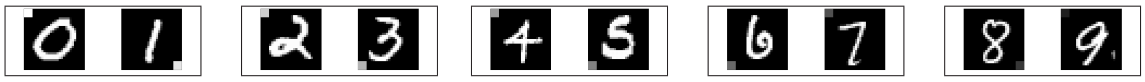


Figure 8.2: In the Split Decoy MNIST 5 tasks need to be learned consecutively without forgetting the previously learned tasks.

For the tabular task, we generate, similar to Guyon et al. [85], a dataset with 10 classes and 20 features, of which 10 are informative, using the scikit-learn implementation[2]. Therefore, the remaining features are irrelevant and should not be used in the classifier. The method generates clusters of normally distributed points around the vertices of a 10-dimensional hypercube with sides of length 20 and assigns an equal number of clusters to each class. It introduces interdependence between these features and adds various noise to the data. Like Decoy MNIST, we split the tabular data into 5 consecutive tasks based on the label for the continuous experiments.

The time series dataset is inspired by the approach of Ismail et al. [101] introduced in Chapter 6. Based on the 5 time series processes, we generate data for 50 time steps with a feature size of 1. For each process, we divide the generated data into two classes by adding or subtracting a constant ϵ in the area of the desired informative features, resulting in a 10-class classification problem (5 processes x 2 classes per process). The informative features are in the middle of the time series, masking 35% of the time steps.

For all datasets, a simple neural network with one hidden layer and 512 neurons is used. We employ a 0.8/0.2 data split on all datasets. The first 80% of data D_B is used to pre-train a model f_B, and the remaining data D_T is the held-out test set. Throughout all experiments, we report the F1-Score as a measure of the classification capabilities of the trained model.

8.3.1 Fine-tuning

For all datasets, the model is pre-trained on either the complete training data D_B or not pre-trained at all. After pre-training, we finetune all models with SGD and a learning rate of 0.001, corresponding to $\frac{1}{10}th$ of the initial pre-training rate [77]. We report the results for fine-tuning incrementally and with replay – i.e., we collect the incrementally arriving data and update the model every 32 data instances. All results are reported on the test set. As the upper bound for the classification performance, we use the models trained on D_B in batch mode with Adam, a learning rate of 0.001, and a batch size of 32, and as the lower bound, we use the incremental trained models (batch size 1).

Table 8.1 shows the F1-Score for the upper and lower bounds and the models trained and finetuned with CAIPI and RRR. All models trained iteratively with CAIPI beat the iterative lower bound, indicating that the provided feedback and mechanism to include that feedback influences the classification capabilities positively. Especially when training a new model iteratively from scratch, CAIPI has an advantage, indicating that using CAIPI can help mitigate catastrophic forgetting even without using replay or another continuous learning strategy. For RRR, only pre-trained models and models trained with replay are able to outperform the lower limit. The results suggest that pre-training a model, followed by iteratively including feedback on the explanation via CAIPI or *RRR*, improves previously trained models. While no trend in the feedback can be seen, the overall best results are achieved by training with replay.

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html`

Table 8.1: Mean and standard deviation of the F1 averaged over 10 Runs.

	Baselines		CAIPI				*RRR*			
			Empty		Full		Empty		Full	
	lower	upper	iterative	replay	iterative	replay	iterative	replay	iterative	replay
Decoy	0.02 ± 0.00	0.69 ± 0.07	0.69 ± 0.05	$\mathbf{0.89 \pm 0.01}$	0.6 ± 0.08	0.71 ± 0.04	0.75 ± 0.03	0.68 ± 0.01	0.7 ± 0.06	0.68 ± 0.09
Tabular	0.51 ± 0.01	0.75 ± 0.01	0.59 ± 0.02	0.75 ± 0.01	0.69 ± 0.01	0.75 ± 0.01	0.15 ± 0.03	0.71 ± 0.01	0.75 ± 0.01	$\mathbf{0.77 \pm 0.01}$
Time	0.54 ± 0.01	0.86 ± 0.01	0.68 ± 0.02	0.86 ± 0.00	0.8 ± 0.02	$\mathbf{0.87 \pm 0.01}$	0.15 ± 0.03	0.85 ± 0.00	0.86 ± 0.01	0.86 ± 0.01

8.3.2 Continuous Learning

To test the continuous learning capabilities of the included continuous learner in combination with the interactive learner, the dataset containing 10 classes is split into 5 consecutive tasks according to the classes.

Table 8.2 shows the F1-Score over all datasets and algorithm combinations. In addition to the continuous learning described in Chapter 8, continuous learning of the tasks with Adam and SGD is reported. The Upper Bound is consistent with the upper bound described in Section 8.3.1 – i.e., a model trained in batch mode with fully available data.

For all feedback types, the replay strategy results in the best overall F1-Score. While none of the continuous learners except replay outperform the lower bounds, the replay strategy combined with CAIPI even outperforms the upper bound. Using the label trick yields inconsistent results, as the best results were achieved without the label trick or on par with the label trick. While it is not clear which feedback interaction is the best choice, the experiments show that continuous learning with human feedback in replay mode enables neural networks to learn consecutively multiple tasks with limited forgetting. Using explanation interaction can thereby even result in better performance than learning in traditional batch mode with complete available data. Further, as found in multiple studies before [244], using the replay strategy is still the most reliant way to prevent catastrophic forgetting. Therefore, we only apply the replay strategy for our real-world use case.

8.4 Real-World Data

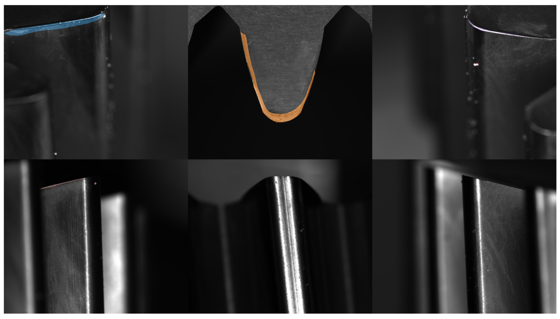
As the real-world use case, we predict the wear on skiving wheels. Thereby, we differentiate based on the tooth condition between light wear, strong wear, and no wear. Overall, we have 363 manually labeled tooth images from 6 different views (see Figure 8.3). Due to the limited number of samples available, we use ResNet pre-trained on ImageNet for all experiments. Similar to the empirical evaluation, we investigate the method's capabilities to fine-tune and continuously classify additional aspects. In the fine-tuning scenario, we only utilize the top center image as domain experts recognize most types of wear from this top view. In the first setting, we differentiate between no, light, and strong wear. Going from there, we enhance the model by

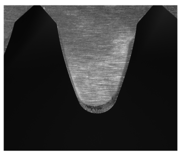
Table 8.2: F1 for continuously learning additional classes a) feedback included via CAIPI, b) feedback included via RRR, and c) without feedback.

		continuous tabular		continuous time		split decoy mnist	
			LT		LT		LT
CAIPI	Adam	0.08 ± 0.02	0.06 ± 0.00	0.07 ± 0.02	0.06 ± 0.00	0.06 ± 0.02	0.08 ± 0.02
	SGD	0.07 ± 0.01	0.06 ± 0.00	0.07 ± 0.04	0.06 ± 0.00	0.06 ± 0.01	0.02 ± 0.00
	EWC	0.06 ± 0.0	0.07 ± 0.03	0.06 ± 0.0	0.08 ± 0.03	0.09 ± 0.02	0.06 ± 0.02
	SI	0.06 ± 0.0	0.07 ± 0.01	0.06 ± 0.0	0.07 ± 0.03	0.09 ± 0.02	0.06 ± 0.04
	MAS	0.06 ± 0.0	0.08 ± 0.01	0.06 ± 0.0	0.07 ± 0.03	0.02 ± 0.01	0.05 ± 0.01
	replay	$\mathbf{0.74 \pm 0.02}$	$\mathbf{0.74 \pm 0.02}$	$\mathbf{0.87 \pm 0.01}$	$\mathbf{0.87 \pm 0.01}$	0.77 ± 0.05	0.82 ± 0.03
RRR	Adam	0.14 ± 0.03	0.14 ± 0.02	0.15 ± 0.02	0.15 ± 0.02	0.09 ± 0.01	0.08 ± 0.01
	SGD	0.14 ± 0.03	0.13 ± 0.02	0.14 ± 0.04	0.13 ± 0.04	0.09 ± 0.01	0.09 ± 0.01
	EWC	0.06 ± 0.0	0.02 ± 0.00	0.06 ± 0.0	0.02 ± 0.00	0.07 ± 0.01	0.02 ± 0.00
	SI	0.06 ± 0.0	0.02 ± 0.00	0.06 ± 0.0	0.02 ± 0.00	0.07 ± 0.0	0.02 ± 0.00
	MAS	0.06 ± 0.0	0.02 ± 0.00	0.06 ± 0.0	0.02 ± 0.00	0.02 ± 0.0	0.02 ± 0.00
	replay	0.58 ± 0.01	0.61 ± 0.01	0.64 ± 0.02	0.77 ± 0.02	$\mathbf{0.9 \pm 0.0}$	0.84 ± 0.01
Sanity	Adam	0.22 ± 0.04	0.12 ± 0.02	0.16 ± 0.03	0.19 ± 0.03	0.36 ± 0.04	0.07 ± 0.00
	SGD	0.1 ± 0.02	0.13 ± 0.02	0.12 ± 0.02	0.13 ± 0.03	0.49 ± 0.05	0.12 ± 0.03
	Upper Bound	0.63 ± 0.01	–	0.75 ± 0.01	–	0.68 ± 0.03	–

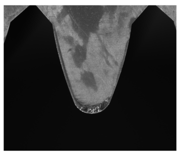
continuously learning the additional views, which can be seen as incremental domain learning. In industrial settings, data acquisition is often developed and enhanced iteratively, therefore new views are available consecutively. Further, many use cases are similar: Instead of classifying the wear on skiving wheels teeth, the wear of another tool (e.g., cutting tools) might need to be classified. The task of the model is the same, however, in different domains. In the last experiment, we want to add another class of wear for completely broken teeth to the existing three classes to get a more fine-grained classification and also an indication of the need for an urgent switch of the skiving wheel. The use case shows several real-world constraints:

- Limited Data: We have a limited amount of data to train our classifier and therefore use a pre-trained model that needs to be fine-tuned.
- Domain Knowledge: We have a task that's easily done with domain knowledge and human cognition, but a challenge for DL due to the limited data.
- Domain Learning: We have different views on the same problem, and therefore different domains.
- Task Learning: Instead of only recognizing strong wear, we want to differentiate between strong wear and broken skiving wheel teeth.

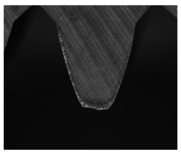




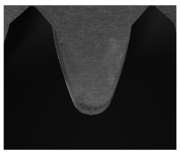
(a) Light



(b) Strong



(c) Broken



(d) OK

Figure 8.3: Visualization of the real world dataset. The left side shows a skiving wheel tooth from all 6 views with annotations from the domain expert. The right side visualizes one sample per class from the top view.

8.4.1 Fine-tuning

Table 8.3 shows the F1-Score of the pre-trained ResNet fine-tuned without human annotations (baselines), with feedback via CAIPI and feedback via RRR. The baseline's lower and upper bounds refer to the ResNet fine-tuned iteration-wise and batch-based (without interaction), respectively. The empty base model is the ResNet trained on the ImageNet only, while the full base model refers to the fine-tuned upper bound.

In general, replaying parts of the data positively influences all settings, except interactively fine-tuning the fully trained model with RRR. All methods based on the fully trained ResNet achieve a performance increase from the domain knowledge-based interaction and, therefore, beat the upper bound with an F1 increase of at least 0.05. The best results are achieved by interacting with the fully trained model via CAIPI in replay mode. This might be due to the additional data generated by CAIPI, which can act as data augmentation in cases of little data. The results show, that fine-tuning of pre-trained and pre-fine-tuned models with human knowledge via explanation interaction can help deal with limited and incomplete data.

Table 8.3: Mean and standard deviation of the F1 averaged over 10 runs.

Baselines		CAIPI				RRR			
base model		empty		full		empty		full	
lower	upper	iterative	replay	iterative	replay	iterative	replay	iterative	replay
0.28 ± 0.00	0.46 ± 0.07	0.28 ± 0.00	0.51 ± 0.07	0.51 ± 0.06	$\mathbf{0.55 \pm 0.06}$	0.22 ± 0.08	0.46 ± 0.05	0.51 ± 0.03	0.51 ± 0.09

Table 8.4: F1 Score for continuously learning to classify wear on different views on the teeth. All results are reported on the test set.

	CAIPI			RRR			Baselines		
	Adam	SGD	replay	Adam	SGD	replay	Adam	SGD	Upper Bound
	0.2 ± 0.08	0.28 ± 0.05	0.5 ± 0.11	0.31 ± 0.03	0.28 ± 0.06	$\mathbf{0.58 \pm 0.04}$	0.29 ± 0.04	0.29 ± 0.05	0.43 ± 0.04
LT	0.09 ± 0.00	0.3 ± 0.07	0.56 ± 0.03	0.29 ± 0.04	0.22 ± 0.04	0.57 ± 0.04	0.3 ± 0.06	0.28 ± 0.02	–

8.4.2 Domain Learning

In the next step, we enhance the model for the classification of skiving wheel teeth described in Section 8.4.1 by learning the classification for different views on the teeth.

Table 8.4 shows the results for incrementally learning the 6 views. Including different views on the teeth leads to an F1 increase from on average 0.55 in Section 8.4.1 to 0.58 in the best setting. While for fine-tuning model interaction via CAIPI was the best choice, RRR in replay mode is preferred for learning additional domains. Overall, all interaction strategies in combination with replay outperform the batch-based upper bound, indicating that learning new domains by including human domain knowledge provides a helpful asset.

8.4.3 Task Learning

Sometimes, models are required to make more fine-grained decisions as the original classes are not enough to make decisions. For the skiving wheel task, the decision of strong wear should include a differentiation between strong wear and a broken tooth as the tools with strong wear can still be reused and maintained while a destroyed tooth, depicted in Figure 8.3, leads to the necessity of a new tool. To showcase the suitability of the interactive methods, we use the previous best model (trained with RRR & Replay) and utilize CAIPI and RRR in a replay setting to learn the difference between strong wear and the complete failing of the tool.

Table 8.5 shows the F1 score on a test set labeled with the four classes OK, light wear, strong wear, and broken. The starting model, trained on the three classes, implies the lower bound. Again, learning to distinguish between new classes derived from the old superclass by interacting via CAIPI in combination with replay outperforms all other combinations.

Table 8.5: F1 for continuously learning the difference between strong wear and broken teeth in addition to the previous three classes. All results are reported on the test set.

	CAIPI			RRR			Baselines		
	Adam	SGD	replay	Adam	SGD	replay	Adam	SGD	Starting Model
	0.32 ± 0.08	0.34 ± 0.08	$\mathbf{0.66 \pm 0.04}$	0.38 ± 0.03	0.37 ± 0.04	0.61 ± 0.03	0.34 ± 0.00	0.33 ± 0.03	0.22 ± 0.00
LT	0.3 ± 0.01	0.4 ± 0.03	0.62 ± 0.01	0.38 ± 0.1	0.42 ± 0.04	0.6 ± 0.03	0.38 ± 0.05	0.3 ± 0.05	–

8.5 Summary and Discussion

This section, addressed RQ III. by proposing *CXIL*, a framework for Continuous Explainable Interactive Learning that allows continuous learning under human supervision. The results on the (semi-) synthetic and real-world data show that including human domain knowledge positively influences classification results. Including continuous learning allows for iterative learning of new domains and tasks. Further, it improves learning with limited data. The improvement in the model is evident in real-world data.

In hindsight to RQ III.(1), our evaluation indicates that continuous learning utilizing the replay strategy still works best. Consistent with many studies, regularization-based methods, specifically in task-free settings, fail [270]. Even the proposed label trick did not improve the result on those methods. With more research and better-performing regularization methods, the results could be improved. Overall, regularization-based methods are a promising direction for continuous learning, as they do need less resources than memory-based methods and do not change the architecture of a learner. In 4 out of the 7 cases, CAIPI in replay setting performed best. In the remaining experiments, where RRR performed better, significantly larger datasets were used (e.g., MNIST contains 70000 compared to 10000/5000 for tabular/time or the data increase in the real-world domain experiment as all views are added). We hypothesize that this is due to CAIPI data augmentation, which could be helpful in cases with low data. However, a closer inspection of the relationship between the feedback algorithm and the dataset size on performance still needs to be conducted.

Regarding RQ III.(2), we were able to show that on real data with (noisy) human annotations the proposed feedback and continuous learner were able to learn new domains, new tasks, and furthermore refine the already trained model with domain knowledge. However, a long-term study in production is still necessary to evaluate the ability of workers to interact with explanations as well as the effects of long-term continuous learning. Further, the chosen use case is image-based and therefore more intuitive to human cognition than tabular or time series data. For tabular and time series data, additional thoughts need to be put into the visualization. Further, we did not account for human fatigue which often occurs in repetitive settings. To evade human fatigue, including heuristics for item selections inspired by active learning in high-frequency settings (e.g., settings where each step cannot be supervised due to fast processing times) might be advantageous.

Part V

Conclusion

This part provides a synthesis of the thesis. We first summarize the thesis contributions (Section 9.1). Then, we provide an outlook on future work (Section 9.2).

9 Conclusion

This thesis investigated *XAI* in the context of industrial use cases, specifically focusing on *XAI* methods, *XAI* method evaluation and *XAI*-based model revision. This chapter summarizes and synthesizes our main findings (Section 9.1) and concludes with an outline of promising future research topics (Section 9.2).

9.1 Summary

Since the adoption of the AI-ACT, interest in *XAI* has been spiking. Especially high-risk systems are now subject to regulations like transparency and human oversight. Although *XAI* research has been around for more than a decade with many real-world applications, adoption of *XAI* in the industrial domain is still hesitant.

Therefore, this thesis investigated the following research question:

How can we facilitate and improve *XAI* to enable *XAI* in industrial settings?

To answer the main research question, we built a conceptual framwork for the integration of post-hoc *XAI*. Along this conceptual framework described in Figure 1.2, we identified research gaps currently hindering the usage of *XAI* on a larger scale in industrial settings. The main research question is answered based on the developed subresearch questions in 4 parts: the foundations, *XAI* methods, *XAI* evaluation, and *XAI*-based model revision.

Part I provided the foundations and related work necessary to understand this work, starting with an introduction to supervised machine learning and *EXplainable Artificial Intelligence*. Based on the foundations, we presented first (post-hoc) *XAI* methods on the time series domain. Within related work, we depicted current works regarding (i) the application of *AI* in industrial settings, (ii) the usage of *XAI* on time series, (iii) the evaluation of *XAI*, and (iv) industrial applications of human-*AI* interaction and continuous learning.

Part II was driven by the research challenge of *XAI* for time series. Thereby, our research questions aimed to increase the accessibility of *XAI* for time series (RQ I.(1)) and provide more plausible counterfactual explanations for time series (RQ I.(2)).

RQ I. How can we facilitate and enhance *XAI* on time series?

- **RQ I.1.** How can the cross-domain application of explanation methods for deep learning based time series classification be facilitated ?
- **RQ I.2.** How can we enhance counterfactual time series explainers by including time series based transformers?

We derived from the research questions Hypothesis I:

Hypothesis I (Methods)
The adoption of XAI in industrial settings can be facilitated by providing time series explainers based on time series transformation mechanisms and an easier access to such explainers.

Based on existing (post-hoc) *XAI* methods for time series and the scikit-learn principles, we formalized and developed an architecture to standardize access to time series explainers. To test Hypothesis I, we integrated existing *XAI* methods for time series into the designed framework and enabled unified access to those methods. By providing the tool (the framework including the implemented methods) open-source via PyPi and GitHub, we enabled an easy access to time series explainers driving more widespread adoption, fostering community contributions, and continuous improvements. Further, to enable more realistic counterfactuals, we included our method, *TSEvo*, which generates counterfactual explanations based on a reference dataset and four different time series transformations in the framework. We were able to show that our method, in contrast to other counterfactual explanations for uni- and multivariate time series, can generate more proximate and in-distribution counterfactuals.

Part III investigated the evaluation of *XAI* methods specifically in the time series (RQ II.(1)) and counterfactual (RQ II.(2)) domain.

RQ II. How can we quantify *XAI* performance on time series and industrial use cases?

RQ II.1. How can we quantify the quality of time series explainers for an easier benchmarking?

RQ II.2. How can we evaluate counterfactual explainers for real-world coherence?

Hypothesis II combines both research questions to:

Hypothesis II (Evaluation)
To ensure the correctness of explanations for further use, the quality of explanation algorithms needs to be quantifiable.

Based on the framework developed in **Part II**, we investigated the performance of *XAI* for time series with the help of a standardized framework proposing tweaks for the time series domain, synthetic data with known ground truths, and trained models of various types (*LSTM*, *CNN*) with the result that adopting non-time-specific *XAI* to time series yields better results on faithfulness - i.e., explainer-model consistency, however worse results on complexity. This emphasizes the need for more time-series-specific feature attribution methods to enable explainers that replicate *ML* model behavior while still being low complexity for better understandability. Further, we proposed “Semantic Meaningfulness” as a measure to capture the realisticness of counterfactual explanation according to real-world coherence instead of the data manifold with the result that only causal approaches can fully capture such relations and surrogate-based approaches can only capture some. The empirical evaluation showed that optimization-based approaches perform worse than approaches with access to additional information like the underlying data distribution or classification model internals.

Part IV built on **Part II** and **Part III** by leveraging well-performing *XAI* explanations to help *AI* models learn iteratively from expert knowledge and prevent spurious behavior. By combining various continuous learners and interactive learners, we addressed which algorithm combinations are preferable in RQ III.(1) and how this can be applied to actual use case data RQ III.(2).

RQ III. How can we enable continuous, explanatory. and interactive model improvement?

RQ III.1. Which combinations of continuous, explanatory, and interactive are the most suitable?

RQ III.2. How does continuous, explanatory, and interactive model improvement perform on realistic data?

Based on RQ III.(1) and RQ III.(2), we formulated Hypothesis III:

Hypothesis III (Revision)
By combining Explanatory Interactive Machine Learning with continuous learning, we can enable continuous human supervision and boost model performance.

Building on the paradigms of continuous learning and explanatory interactive learning, we built *CXIL*, a framework enabling continuous learning under human supervision. The results on (semi-) synthetic data showed that the memory-based continuous learner is a suitable addition to enable fine-tuning with the help of human annotations, task learning, and domain adaption. The results on the real-world data set with noisy human annotations confirmed these results.

Returning to the overarching research question and compiling Hypothesis I-III, our proposed methods improve and simplify the adoption of *XAI* in industrial settings from the accessibility of methods, ensuring the performance of *XAI* to improving predictive models based on (well-performing) *XAI* applications on-the-fly. By developing these standardized frameworks and methods, the thesis empowers integrating *XAI* into industrial *AI*, enhancing transparency. Creating more meaningful explanations, such as realistic counterfactuals, helps bridge the gap between complex *AI* models and human understanding. Rigorous evaluation of *XAI* methods ensures their reliability. Ultimately, combining continuous learning with human oversight fosters a feedback loop, enabling *AI* to learn and improve in alignment with expert knowledge, boosting performance. Figure 9.1 summarizes the investigated problem setting and shows the locations of the contributions coming into effect.

9.2 Discussion and Future Work

The thesis provides methods and tools facilitating access to *XAI*, focusing on industrial problem settings. While we provide improvements in the area of *XAI* for time series, the evaluation of *XAI*, and *XAI*-based model revision, the restrictions employed in this thesis enable future improvements and innovation. This section discusses the thesis and proposed methods and points out future work based on the scope defined in Section 1.5.[1]

First, we discuss the framework for time series explainers and the enhancement of counterfactuals for time series (Hypothesis I), which enables standardized access, reusable functions, and easy integration of newly developed time series *XAI* algorithms to enable a democratized and easily accessible framework.

[1] Note that the results of the research questions have already been discussed in the respective chapters.

An important direction for future work is the real-world applicability of the framework. The underlying assumptions are regular time series with a consistent time step size and no missing values. To resolve this assumption, data preprocessing or imputation of missing values becomes crucial; methods include imputation, data generation-based methods, and predictor-based methods (e.g., [259]). Further, forecasting is an increasingly important problem, as it covers use cases like demand forecasting or machine time till failure. As discussed in Section 1.5, currently, statistic-based models, not in need of *XAI*, still dominate. Nevertheless, deep learning models are closing up. As research into time series forecasting is steadily proceeding (e.g., transformers on time series [152]), deep learning models will soon overtake the traditional models, creating a need for *XAI* on forecasting tasks. Some *XAI* methods are already available for forecasting (e.g., [207, 271]). They are currently mostly limited to feature attribution. As research in *XAI* is spiking, the research community continuously develops more and better methods addressing some limitations of this work. We leave the integration of such methods to future work.

Dealing with Hypothesis II, the evaluation of *XAI* for time series and causal coherence, the major limitation is related to the synthetic data sets provided in both cases. The framework for benchmarking time series assumes known time series mechanisms to generate reliable metric results, as many metrics rely on perturbing the input with informative and uninformative features. A more sophisticated synthetic time series generator cannot be used by assuming known generation mechanisms and informative features. Future work should include non-perturbation metrics specifically for faithfulness and robustness to enable the use with any dataset. Such a metric could be developed by including mechanisms to extract likely non-informative features from datasets (e.g., by considering time series features and generating perturbations based on those). Additionally, a conditioned time series generator (e.g., a *VAE* that is additionally conditioned on producing non-informative time series) might be able to generate time series or fractions of time series that do not include informative features. Such an approach would also bring the advantage of reliable metric applicability on custom data. Quantifying semantic meaningfulness relies strongly on *SCM*s and therefore the synthetic data. Currently, only synthetic data is provided, so extending the metrics with time series and images is of interest. To circumvent the issue of missing *SCM* including approaches from the ever-proceeding research on learning (structural) causal models from data (e.g., [180]) would enable the usage of the metrics to custom data.

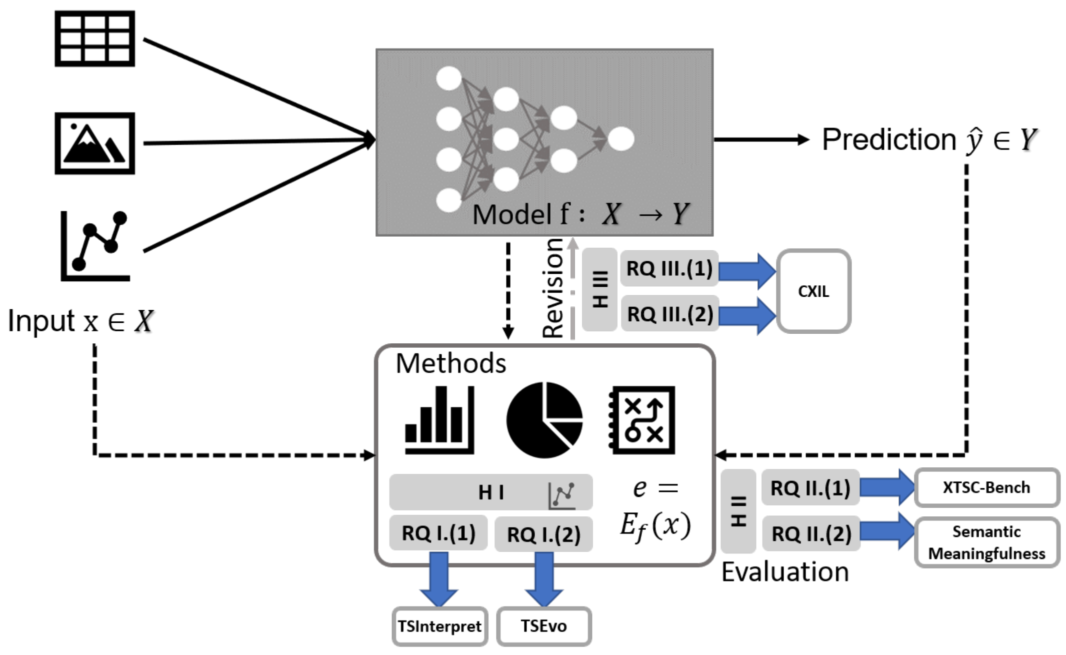


Figure 9.1: Conceptual framework for enabling (post-hoc) *XAI* in industrial settings highlighting the contributions of the thesis.

Furthermore, it would be interesting to investigate whether any of the metrics utilized correlate with what a human perceives as a good explanation. Therefore, the outcome of this thesis is not only valuable in industrial settings but also provides methods, tools, and results that have a broader applicability to other domains.

Future work concerning Hypothesis III, the model revision, focuses on enhancing the applicability and efficiency in real-world settings by specifically looking at continuous learner and human factors. So far, memory-based learners are the only adequate form of continuous learning in task-free settings. Unfortunately, those are also the continuous learners with the highest resource needs. As research in continuous learning is progressing, adding promising algorithms to the framework to improve online learning capabilities is advantageous. Further, currently, each item running through the model should be annotated. To make this less time-consuming, only selected items should be shown to the domain expert to prevent human fatigue if the use case allows it. Inspiration for such selectors can be drawn from the research field of active learning. Many industrial use cases include mixed input data types (e.g., image, tabular data); the method has only been evaluated on single data type models. While the adaption of continuous learning and interactive learning in such a setting is trivial, including an explainer able to cope with multimodel deep neural networks is challenging due to the vast input space and need for different visualization types. Taking the results into the real world, a sufficient user interface (e.g., [64]) and human-centered explanations (e.g., [165]) are essential to enable good human-computer interaction. Designing such interfaces and interactions can draw from research in the field of Human - Computer interaction.

Despite the limitations described above, the outcomes of this thesis is not only valuable in industrial settings but also provides methods, tools, and results that have a broad applicability to other domains. The proposed frameworks and methods are not tailored to a specific use case and can be applied where technically possible. An interesting application is for example the medical domain that has requirements similar to those described in the industrial setting – i.e., time series are ubiquitous (e.g., ECG), reliability and causal coherence are highly relevant as wrong decisions have an even larger impact than in the industrial domain, and extensive domain knowledge is available from doctors.

To summarize, while this thesis contributes to improve the applicability of *XAI* in industrial setting, it also sets the ground for diverse, promising research directions and the application to of the methods to other use cases.

A Appendix

A.1 TSEvo

In this section the related work for Multi-Objective Optimization and additional results for Chapter 5 are presented. Appendix A.1.2 shows the dataset-wise results for the mutation ablation study.

A.1.1 Details on the selection of the Multi-Objective Optimization Algorithm

Algorithms for multi-objective optimization aim to simultaneously optimize two or more objectives. In contrast to single-objective optimization, multi-objective optimization provide a set of points known as pareto optimal set. In a pareto optimal set, different solutions represent the trade-off solutions between conflicting objectives. A variety of mathematical techniques have been developed to obtain pareto optimal solutions [145]. However, such techniques present several limitations, e.g., the susceptible to the continuity and shape of the pareto front. Furthermore, they usually only generate one element of the pareto optimal set per execution. These limitations gave rise to more flexible and easy-to-use metaheuristics. Currently, among the most popular metaheuristics are evolutionary multi-objective optimization algorithms [42]. They fall into three main categories: pareto-based, indicator-based, and decomposition-based.

Pareto-based algorithms use a selection mechanism based on pareto ranking. The core idea of pareto ranking is to rank the population according to pareto optimality and include a density estimator to maintain as many different solutions in the population as possible. Since its proposal by Goldberg [78], numerous algorithm implementations based their selection scheme on pareto Ranking (e.g., Multi-Objective Genetic Algorithm (MOGA) [72] or Nondominated Sorting Algorithm (NSGA) [227]). However, those first pareto-based algorithms had a high computational complexity as well as a lack of elitism. Elitism is the concept of retaining the best solutions obtained in the population to prevent the evolutionary operators from destroying well-performing solutions. NSGA-II [52], the predecessor of NSGA, introduced elitism, a more efficient ranking schema, and a crowded comparison estimator as density estimator. Despite the well-known limitations of the crowd comparison operator and the rapid increase of non-dominated solutions, when dealing with more than three objectives, NSGA-II is still used by researchers today [100].

Indicator-based algorithms were introduced to tackle the increasing number of non-dominated solutions in multi-objective problems with more than three objectives. The modified selection scheme of indicator-based algorithms incorporates a performance measure(e.g., hypervolume or ϵ indicator). Zitzler et al. [274] provided an algorithmic framework for incorporation of performance indicators into the selection scheme in Indicator Based Evolutionary Algorithm (IBEA). Interest was sparked by the introduction of the S Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMA) [66] that combines the crossover, mutation, and non-dominated sorting of NSGA-II with the hypervolume. However, as SMS-EMA and its modified versions rely on the calculation of the exact hypervolume contributions, it becomes computationally very expensive with the increasing number of objectives [28].

The key idea of decomposition-based approaches is transforming a multi-objective problem into a variety of single-objective problems that are solved to generate non-dominated solutions to the original problem. Decomposition methods allow the generation of non-convex and disconnected pareto front to overcome the limitation of linear aggregation functions. MOEA/D [184] decomposes a multi-objective problem into several scalar subproblems by scalarizing the objective functions with different weight vectors. Each subproblem is optimized simultaneously using only information from its neighboring subproblems, which leads to a lower complexity than NSGA-II at each generation and easier diversity maintenance. However, MOEA/D has other limitations: the population size is the size of the weight vectors, which leads for many-objective problems to an impractically large number of weight vectors[100]. For handling this difficulty, NSGA-III [54] adopts both: decomposition and reference points. Ishibuchi et al. [100] showed that the performance of decomposition-based MOEAs (including MOEA/D and NSGA-III) are highly sensitve to pareto front shapes and can be outperformed on those shapes even by NSGA-II. Furthermore, Ishibuchi et al. [100] stated, that for high performance of decomposition-based method: 1) The triangular shape of the pareto front is the same as or similar to the distribution of the weight vectors. 2) The pareto front has to be small in comparison with the feasible region in the objective space, and 3) The decision variables should be separable. Nevertheless, decomposition-based approaches are still an active area of research.

Taking into account that our optimization problem only has three objectives, all of the above introduced algorithm are a valid choice. However, we exclude indicator-based methods since our optimization problem does not fall into the area of many-objective problems and therefore we avoid the extra complexity of calculating an indicator. Further, we exclude decomposition-based methods since we cannot fulfill two of the three reasons Ishibuchi et al. [100] analyzed as being responsible for high performance. The exclusion of indicator- and decomposition-based approaches leaves us with the pareto-based approaches. NSGA-II compared to MOGA and NSGA implements a more efficient ranking scheme leading to increasing popularity. Despite its age, researchers still apply NSGA-II to numerous problems in the last decade (e.g., [224, 57, 146]), including a counterfactual generating problem with tabular data [48]. Furthermore, the performance of NSGA-II has been tested in many competitive studies (e.g., [100, 185, 75]). Therefore, we decided to use NSGA-II.

A.1.2 Additional Visualizations for Mutation Types

This section contains the remaining results. Figure A.1 visualizes the counterfactuals achieved with the different mutation types on the first test image.

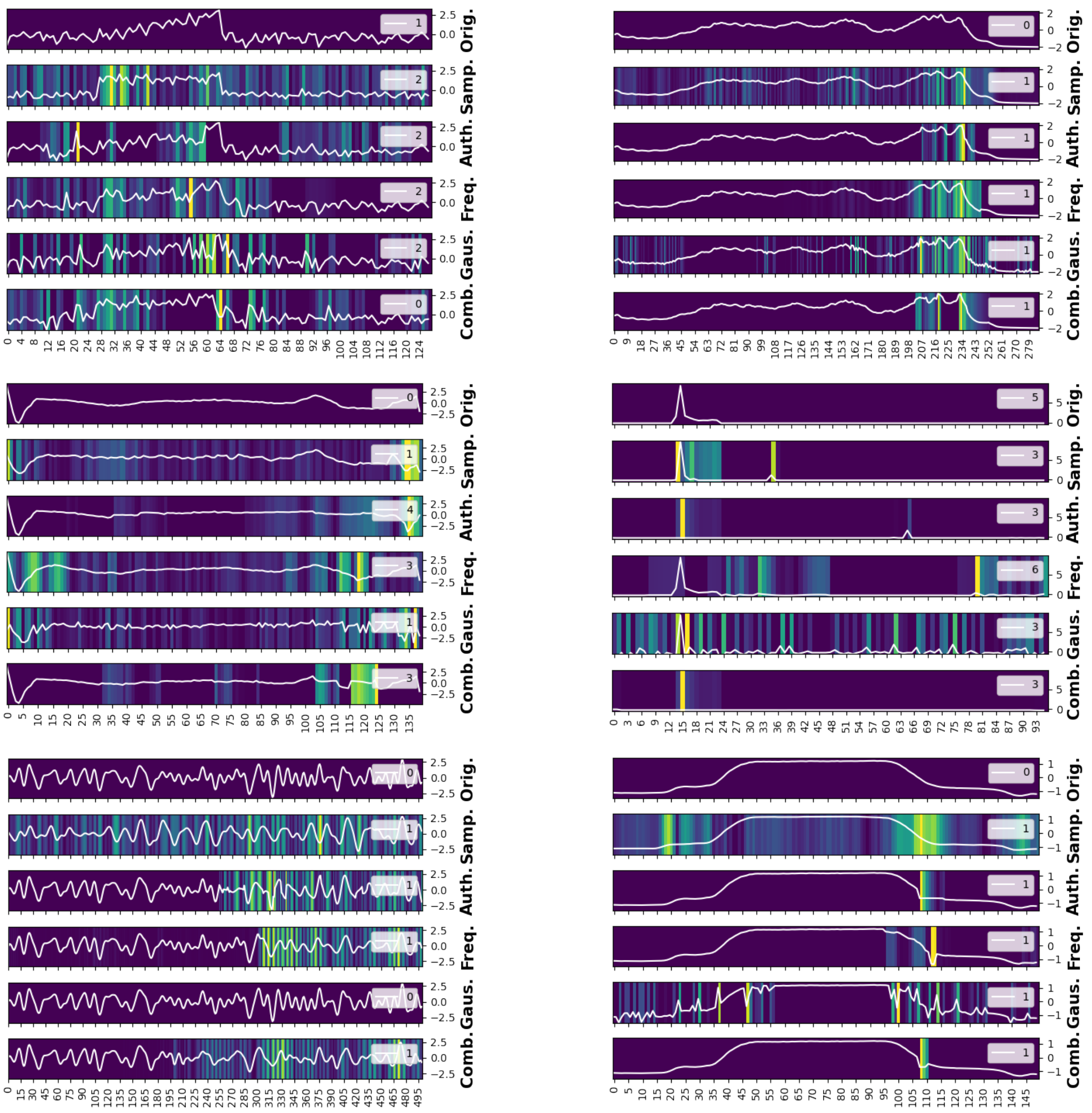


Figure A.1: Counterfactual for the first time series of the test set from CBR, Coffee, ECG5000, Electric Devices, GunPoint and FordA obtained with the different mutation types. If the labels are consistent with the original classification, the method failed to generate a true counterfactual.

A.2 XTSC-Bench

This appendix provides additional explanations of the evaluation settings and visualizations of the results from Section 6.3. Section 6.2.6 explains tweaks to the counterfactual explanations to use the benchmarking tool and Appendix A.2.1 - Appendix A.2.3 visualize additional results.

A.2.1 Results split on Informative Features Types

Figure A.3 and Figure A.4 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.The explainers perform similarly on robustness, faithfulness, and complexity on uni- and multivariate data across the different informative feature types. 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” $\rightarrow$ “SmallMiddle” $\rightarrow$ “Rare”).

A.2.2 Results split on Classifier Models

Figure A.5 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 is below 90% or the explainer is not applicable to the classifier. On average, explainers on LSTMs result in less complex explanations than CNN. Explainers on CNN and LSTM perform similarly on reliability for example and perturbation-based approaches. On faithfulness and robustness, explainers on CNNs and LSTMs show no dominant behavior. Therefore, on most metrics, the classifier type has no larger influence on the explainer’s performance. However, on multivariate data, a slightly higher reliability can be observed for gradient-based approaches without TSR for explanations based on CNN. The performance increase cannot be observed after applying TSR, indicating that “traditional” gradient-based approaches work well for multivariate data with *CNN* classifiers and that LSTM-based gradient explainers need improvement.

A.2.3 Faithfulness: Comparison of Baselines

Figure A.2 compares the faithfulness metric with the generation baseline used for the synthetic data to the “traditionally” used baselines mean and uniform. As the uniform baseline performs, on average, similar to the known generation process baseline, we advise users with non-synthetic data to use the uniform baseline.

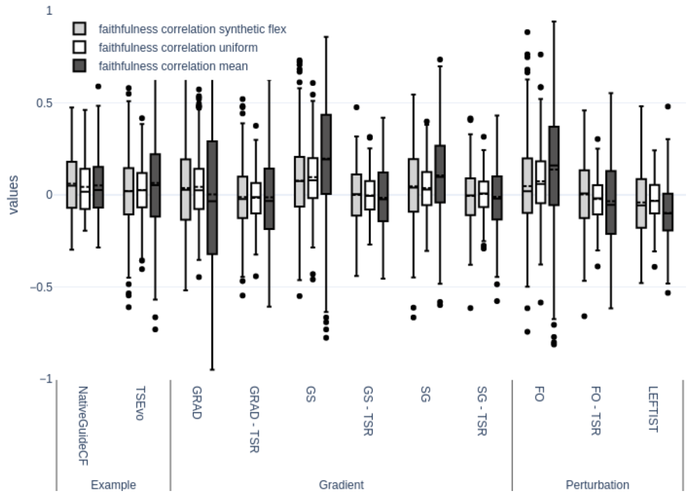
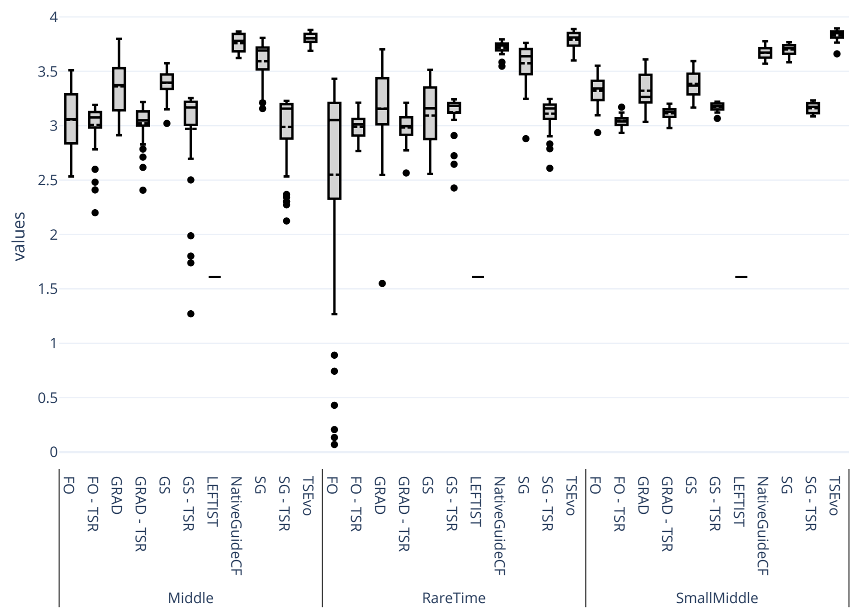
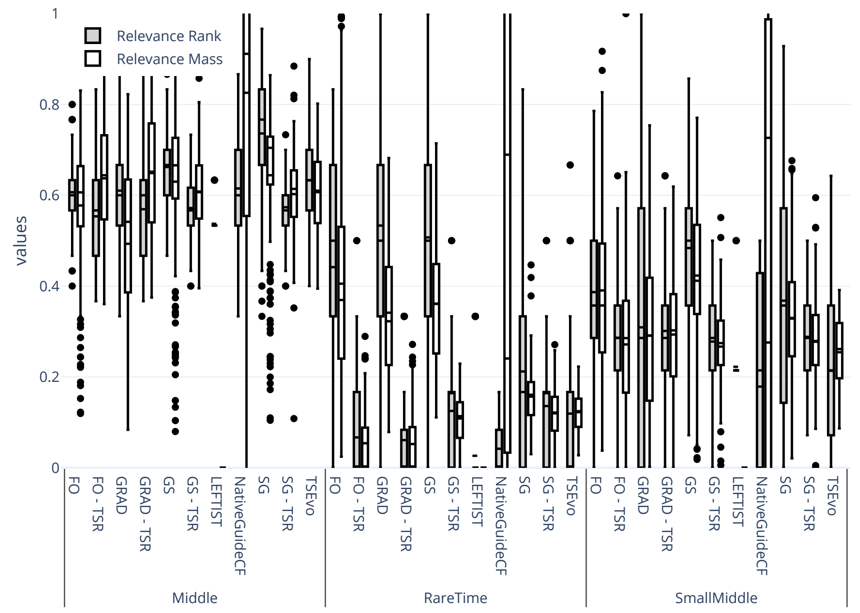


Figure A.2: Comparison of baselines used in the calculation of the faithfulness metric.

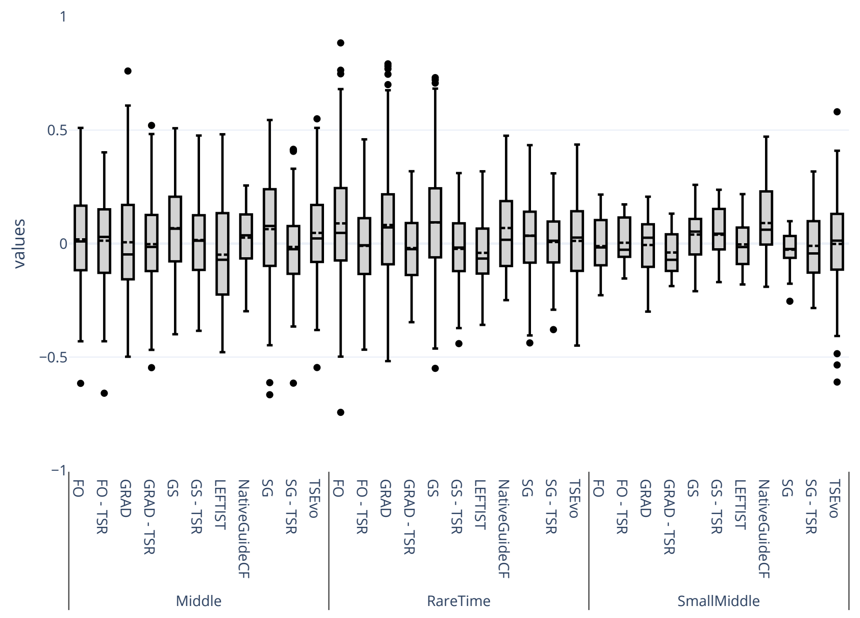


(a) Complexity Univariate

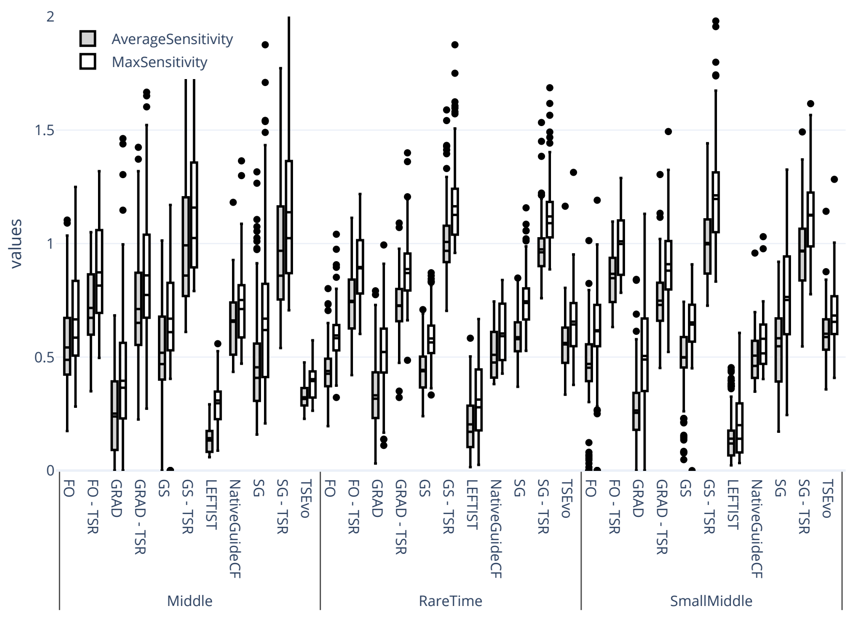


(b) Reliability Univariate

Figure A.3: Informative-feature-wise explainer performance on complexity, reliability, faithfulness, and robustness averaged over all generation processes.

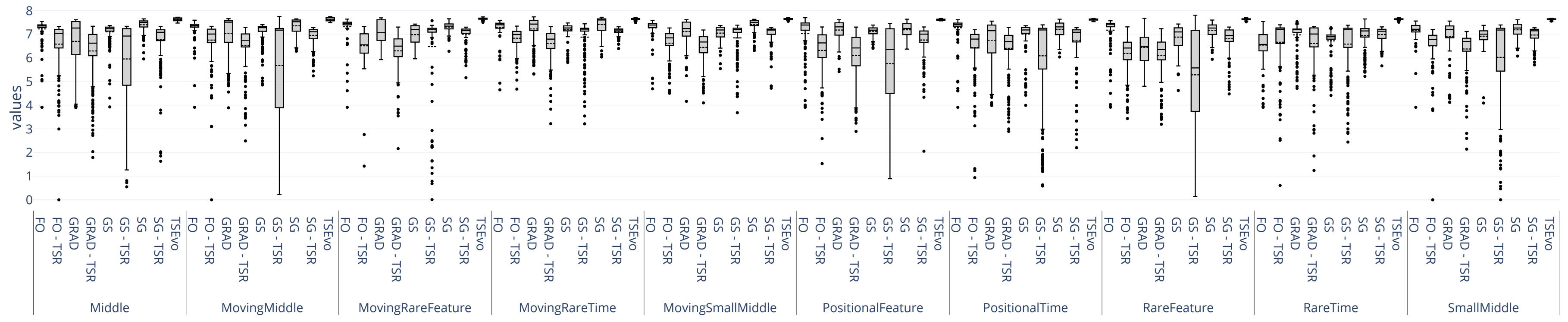


(c) Faithfulness Univariate

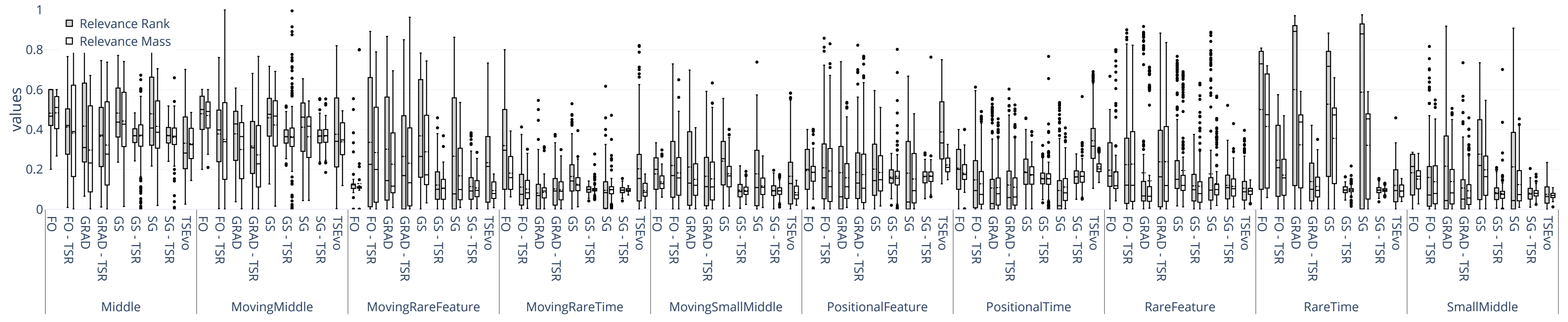


(d) Robustness Univariate

Figure A.3: Informative-feature-wise explainer performance on complexity, reliability, faithfulness, and robustness averaged over all generation processes.

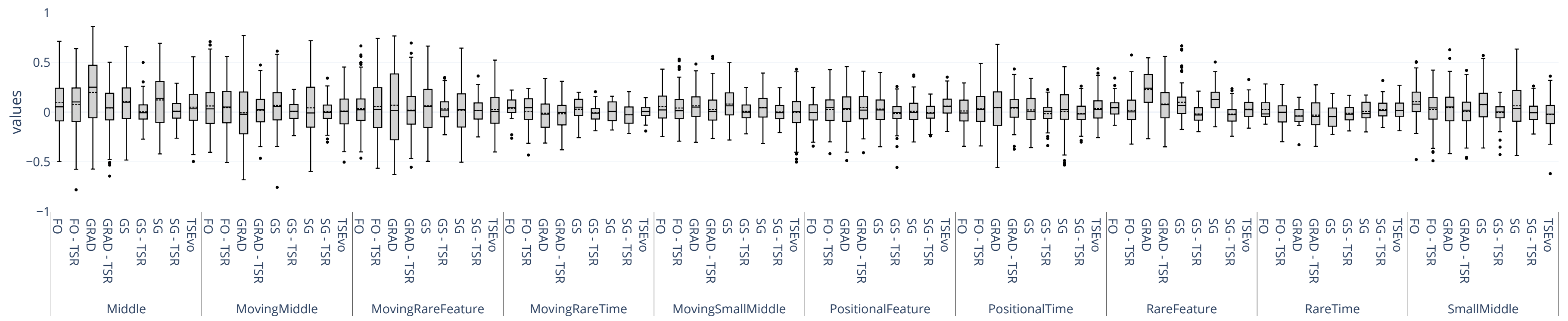


(a) Complexity Multivariate

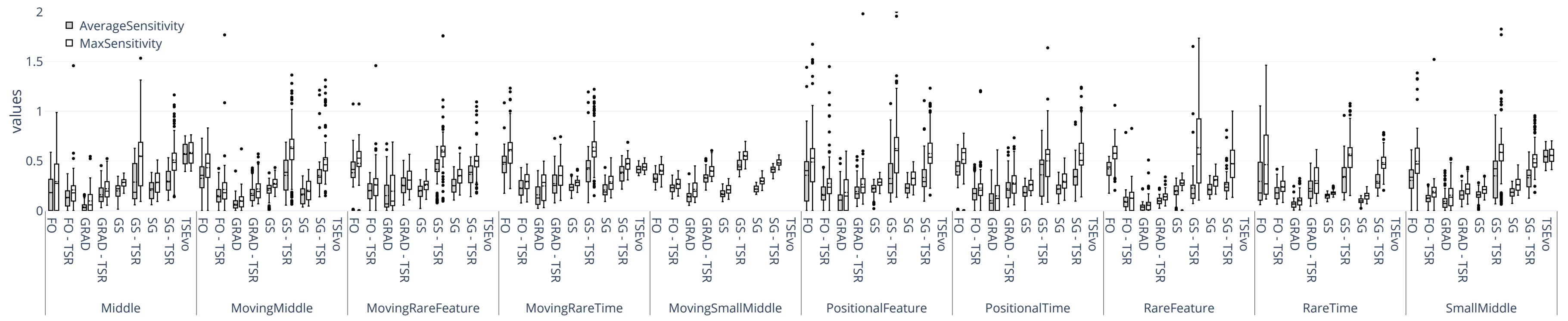


(b) Reliability Multivariate

Figure A.4: Informative-feature-wise explainer performance on complexity, reliability, faithfulness, and robustness averaged over all generation processes.

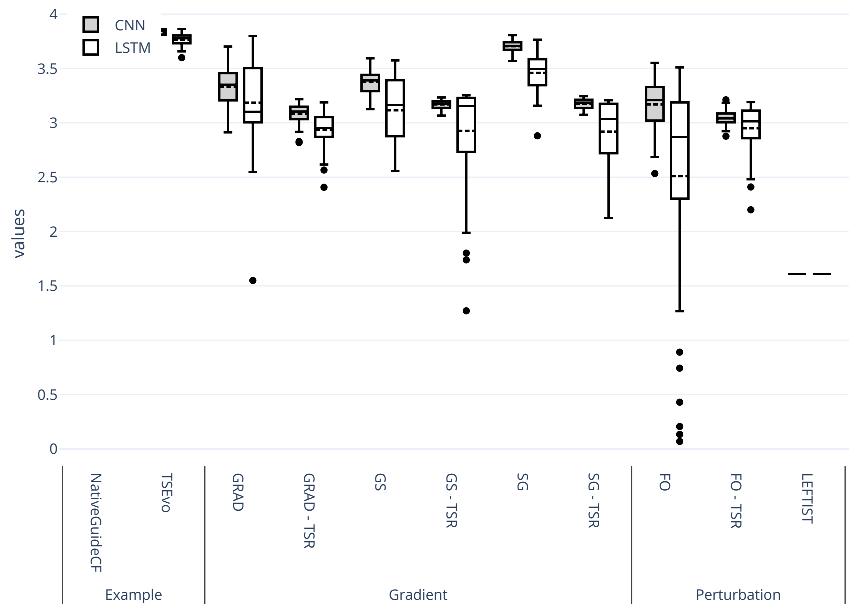


(c) Faithfulness Multivariate

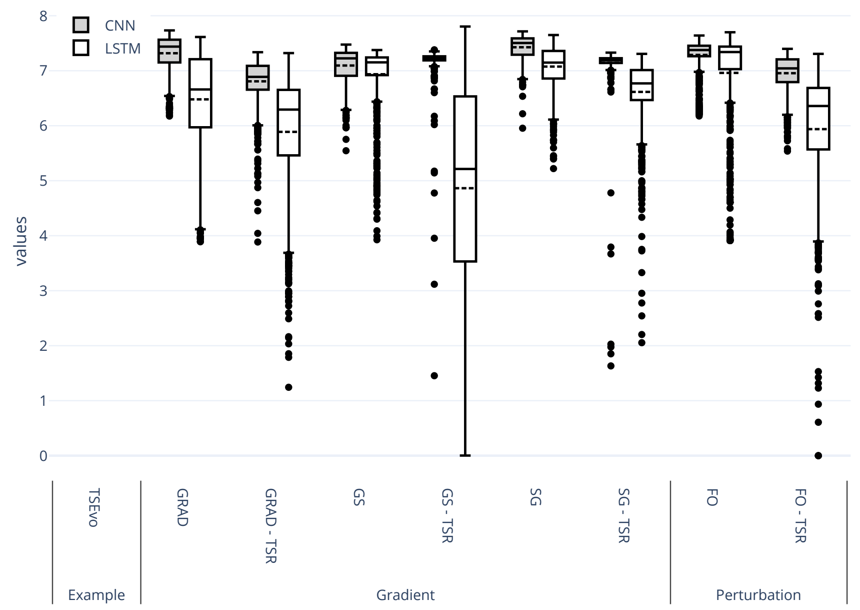


(d) Robustness Multivariate

Figure A.4: Informative-feature-wise explainer performance on complexity, reliability, faithfulness, and robustness averaged over all generation processes.

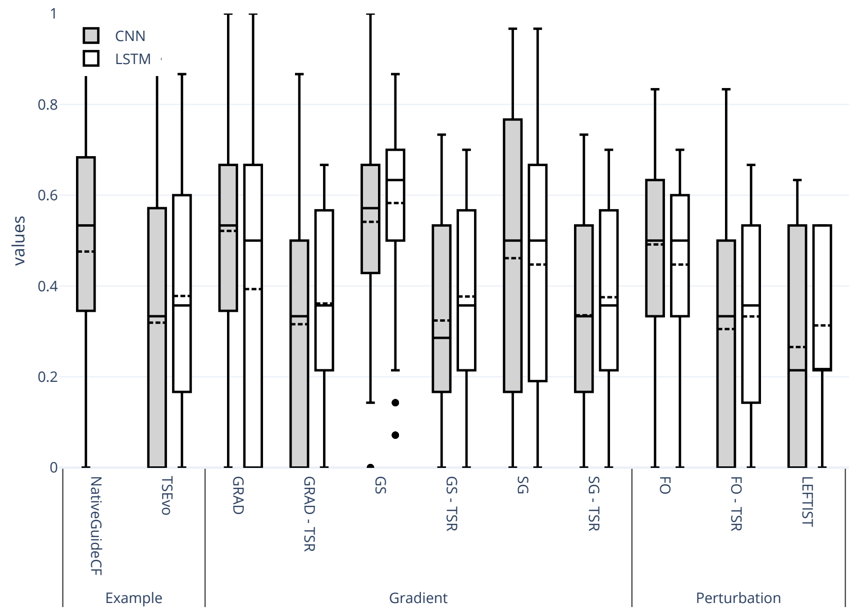


(a) Complexity Univariate

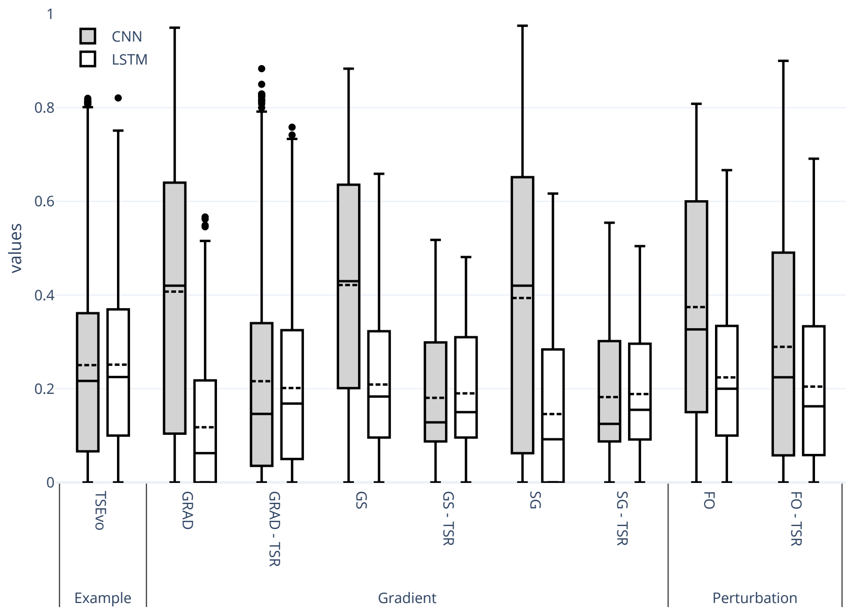


(b) Complexity Multivariate

Figure A.5: Explainer Performance on complexity, reliability, faithfulness, and robustness averaged over all datasets and split on the used classifier.

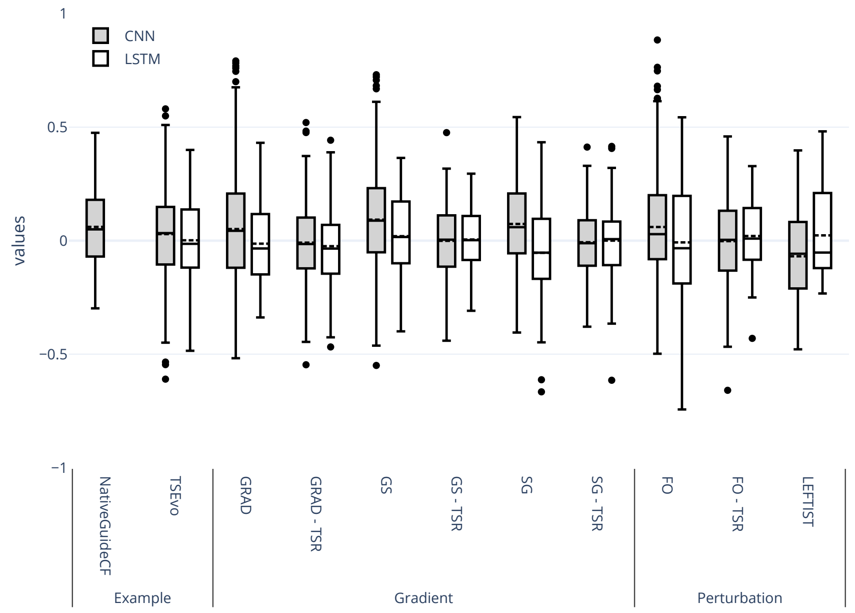


(c) Reliability Univariate

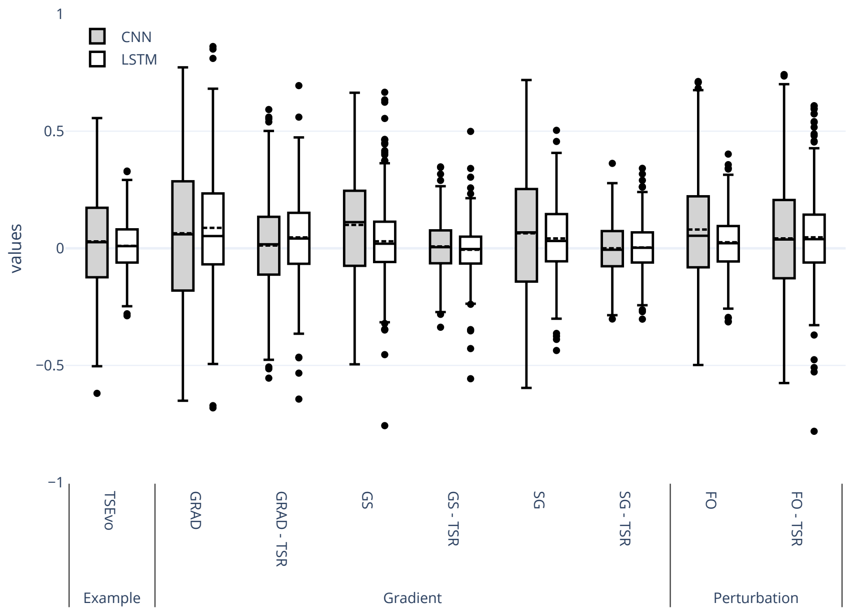


(d) Reliability Multivariate

Figure A.5: Explainer Performance on complexity, reliability, faithfulness, and robustness averaged over all datasets and split on the used classifier.

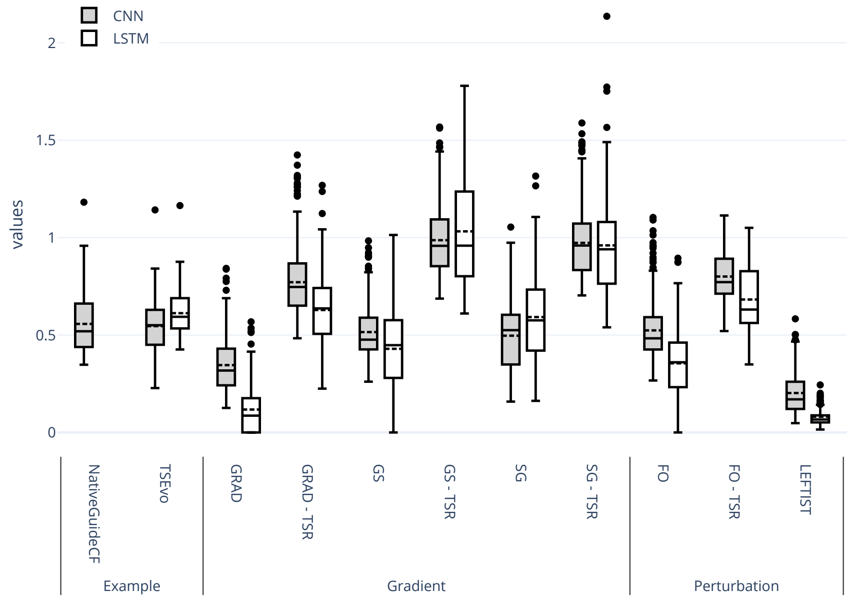


(e) Faithfulness Univariate

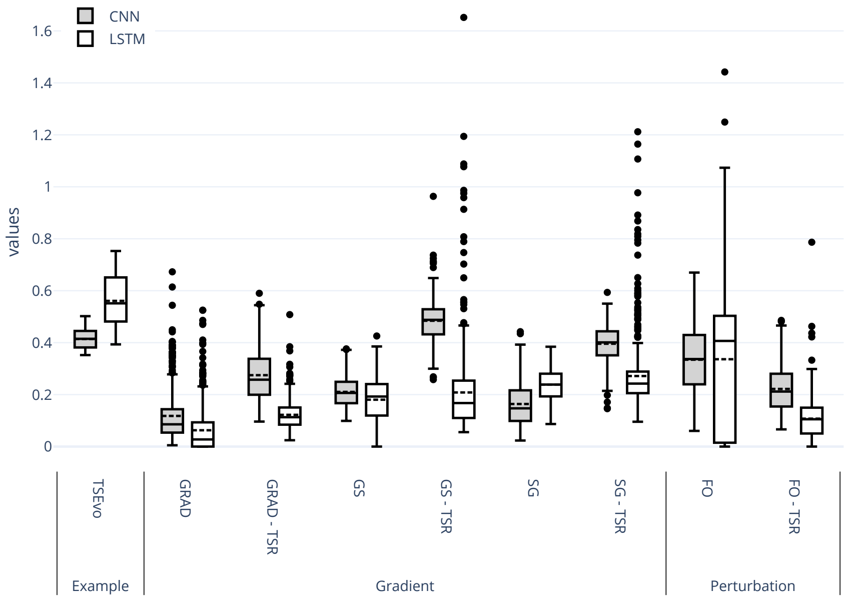


(f) Faithfulness Multivariate

Figure A.5: Explainer Performance on complexity, reliability, faithfulness, and robustness averaged over all datasets and split on the used classifier.



(g) Robustness Univariate



(h) Robustness Multivariate

Figure A.5: Explainer Performance on complexity, reliability, faithfulness, and robustness averaged over all datasets and split on the used classifier.

A.3 Semantic Meaningfulness

This section visualizes the causal graphs belonging to the synthetic dataset introduced in Section 7.2.4.

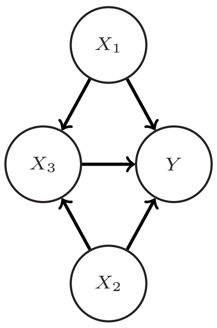


Figure A.6: Synthetic datasets.

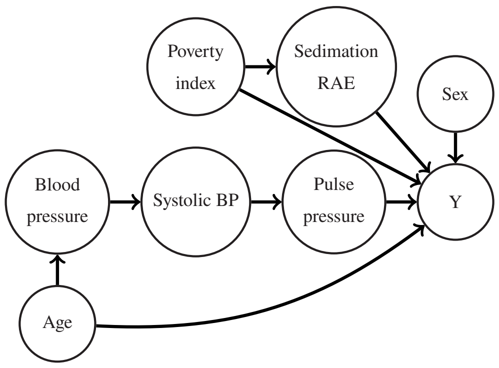


Figure A.7: Nutrition dataset.

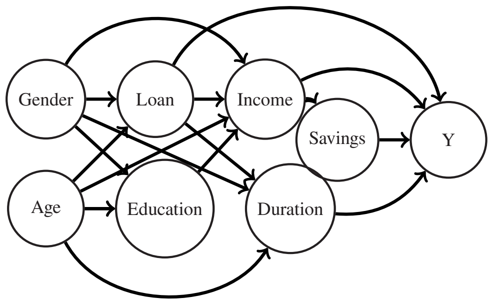


Figure A.8: Credit dataset.

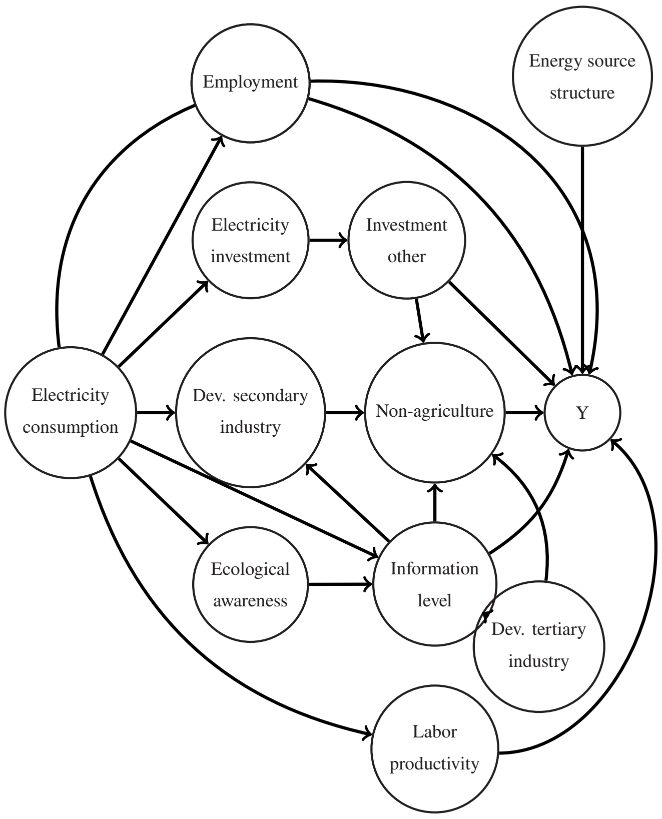


Figure A.9: Economic dataset.

List of Figures

List of Tables

List of Abbreviations

A

AI *Artificial Intelligence* i, 3, 5, 7, 11, 33, 34, 37, 42, 91, 93, 107, 109

AR *Actionable Recourse* 25, 83

AUC *Area Under the Curve* 82, 131

C

C-CHVAE *Counterfactual Conditional Heterogeneous Autoencoder* 26, 83, 84

CF *Counterfactuals* 24, 25

CL *Continuous Learning* 37, 93, 95

CLUE *Counterfactual Latent Uncertainty Explanations* 27, 83, 84

CNN *Convolutional Neural Network* 4, 13, 14, 20, 23, 108, 116, 127

COMTE *Counterfactual Explanations for Machine Learning on Multivariate Time Series Data* 30, 44, 56–59, 127

CR *Causal Recourse* 27, 82, 84

CRUDS *Counterfactual Recourse Using Disentangled Subspaces* 27, 83, 84

CXIL Continuous *EXplanatory Interactive Learning* 93, 94, 98, 103, 109

D

DL *Deep Learning* 3, 4, 7, 93

E

EWC *Elastic Weight Consolidation* 96, 100

F

FA *Feature Attribution* 19–21, 43, 44

FO *Feature Occlusion* 23, 28, 29, 68

FOCUS *Flexible Optimizable Counterfactuals for Tree Ensembles* 28, 83

FT *Feature Tweaking* 28, 83

G

GS *Growing Spheres* 25, 68, 83, 84

I

IB *Instance-Based Methods* 19, 21, 24, 36, 44, 74, 75

IG *Integrated Gradients* 24, 28

L

LEFTIST *Local Explainer For TIme Series classificaTion* 31, 44, 68–73, 127, 128

LIME *Locally Interpretable Model-Agnostic Explainer* 21, 22

LSTM *Long Short Term Memory* 4, 15, 108, 127

LT *Label Trick* 97

M

MAS *Memory Aware Synapses* 96, 100

ML *Machine Learning* 3, 9, 11, 12, 16, 19, 20, 33, 77, 78, 81, 84–89, 108, 128

MLP *Multilayer Percepton* 13, 82, 83, 87

MOC *Multi-Objective Counterfactuals* 26

N

NG *Native Guide* 29, 30, 44, 56–59, 68–72, 127, 128

NSGA-II *Non-dominated Sorting Genetic Algorithm* 26, 47, 48

R

RB *Rule-Based Methods* 21

RF *Random Forest* 82

RNN *Recurrent Neural Network* 13–15

RRR *Right for the Right Reasons* 98–100

S

SCM *Structural Causal Model* 27, 75–82, 84, 85, 87–90, 110, 128

SETS Text 44

SG *Smooth Gradients* 24, 29, 68

SGD *Stochastic Gradient Descent* 16

SHAP *SHapley Additive exPlanations* 22, 29, 43, 94, 95

SI *Synaptic Intelligence* 96, 100

SMO *Semantic Meaningful Output* 77–79, 83–90, 128

SMR *Semantic Meaningful Relations* 77–79, 83–89, 128

T

TSEvo Text 44

TSR *Temporal Saliency Rescaling* 29, 43, 44, 68, 73

V

VAE *Variational Autoencoder* 26, 27, 83, 84, 110

W

W-CF *Wachter Counterfactual* 25, 26, 28, 47, 56–58, 83, 84

X

XAI *EXplainable Artificial Intelligence* 3–10, 19–21, 24, 28, 29, 33–37, 39, 41–43, 46, 60, 61, 74, 75, 91, 93, 107–111, 127, 128, 131

XIL *EXplanatory Interactive Learning* 37, 93

Bibliography

[1] Agnar Aamodt; Enric Plaza. “Case-based reasoning: foundational issues, methodological variations, and system approaches”. In: *AI Commun.* Vol. 7. 1. IOS Press, 1994, 39–59. DOI: `10.3233/AIC-1994-7104`.

[2] Amina Adadi; Mohammed Berrada. “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* 6 (2018), pp. 52138–52160. DOI: `10.1109/ACCESS.2018.2870052`.

[3] Homayun Afrabandpey; Michael Spranger. “Feasible and Desirable Counterfactual Generation by Preserving Human Defined Constraints”. In: *arXiv:2210.05993* (2022).

[4] Namita Agarwal; Saikat Das. “Interpretable Machine Learning Tools: A Survey”. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. Canberra, ACT, Australia: IEEE, 2020, pp. 1528–1534. ISBN: 978-1-72812-547-3. DOI: `10.1109/SSCI47803.2020.9308260`.

[5] Carlos Agostinho et al. “Explainability as the key ingredient for AI adoption in Industry 5.0 settings”. In: *Frontiers in Artificial Intelligence* 6 (2023). DOI: `10.3389/frai.2023.1264372`.

[6] Imran Ahmed; Gwanggil Jeon; Francesco Piccialli. “From Artificial Intelligence to Explainable Artificial Intelligence in Industry 4.0: A Survey on What, How, and Where”. In: *IEEE Transactions on Industrial Informatics* 18.8 (2022), pp. 5031–5042. DOI: `10.1109/TII.2022.3146552`.

[7] Maximilian Alber et al. “iNNvestigate Neural Networks!” In: *Journal of Machine Learning Research* 20.93 (2019), pp. 1–8.

[8] Zoe Alexander; Duen Horng Chau; Christopher Saldaña. “An Interrogative Survey of Explainable AI in Manufacturing”. In: *IEEE Transactions on Industrial Informatics* 20.5 (2024), pp. 7069–7081. DOI: `10.1109/TII.2024.3361489`.

[9] Sajid Ali et al. “Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence”. In: *Information Fusion* 99 (2023), p. 101805. DOI: `10.1016/j.inffus.2023.101805`.

[10] Rahaf Aljundi et al. “Memory Aware Synapses: Learning What (not) to Forget”. In: *Computer Vision – ECCV 2018*. Munich, Germany: Springer, 2018, 144–161. ISBN: 978-3-030-01218-2. DOI: `10.1007/978-3-030-01219-9_9`.

[11] David Alvarez-Melis; Tommi S. Jaakkola. “Towards robust interpretability with self-explaining neural networks”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Vol. 31. NIPS’18. Montréal, Canada: Curran Associates Inc., 2018, 7786–7795.

[12] Julia Amann et al. “Explainability for artificial intelligence in healthcare: a multidisciplinary perspective”. In: *BMC Medical Informatics and Decision Making* 20.1 (2020), p. 310. DOI: `10.1186/s12911-020-01332-6`.

[13] Marco Ancona; Enea Ceolini; Cengiz Öztireli; Markus Gross. “Towards better understanding of gradient-based attribution methods for Deep Neural Networks”. In: *6th International Conference on Learning Representations*. OpenReview.net, 2018.

[14] Christopher J. Anders et al. “Finding and removing Clever Hans: Using explanation methods to debug and improve deep models”. In: *Information Fusion* 77 (2022), pp. 261–295. DOI: `10.1016/j.inffus.2021.07.015`.

[15] Mathias Anneken et al. “Explainable AI for sensor-based sorting systems”. In: *tm - Technisches Messen* 90.3 (2023), pp. 154–166. DOI: `10.1515/teme-2022-0097`.

[16] Javier Antorán et al. “Getting a CLUE: A Method for Explaining Uncertainty Estimates”. In: *9th International Conference on Learning Representations*. OpenReview.net, 2021.

[17] Leila Arras; Ahmed Osman; Wojciech Samek. “CLEVR-XAI: A benchmark dataset for the ground truth evaluation of neural network explanations”. In: *Information Fusion* 81 (2022), pp. 14–40. DOI: `10.1016/j.inffus.2021.11.008`.

[18] Vijay Arya et al. “AI Explainability 360 Toolkit”. In: *8th ACM IKDD CODS and 26th COMAD*. Bangalore India: ACM, 2021, pp. 376–379. ISBN: 978-1-4503-8817-7. DOI: `10.1145/3430984.3430987`.

[19] Emre Ates; Burak Aksar; Vitus J. Leung; Ayse K. Coskun. “Counterfactual Explanations for Multivariate Time Series”. In: *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*. IEEE. IEEE, 2021, pp. 1–8. DOI: `10.1109/ICAPAI49758.2021.9462056`.

[20] Anthony Bagnall et al. “The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances”. In: *Data Mining and Knowledge Discovery* 31.3 (2017), pp. 606–660. DOI: `10.1007/s10618-016-0483-9`.

[21] Anthony Bagnall et al. “The UEA multivariate time series classification archive, 2018”. In: *arXiv:1811.00075*. arXiv: 1811.00075. arXiv, 2018.

[22] Omar Bahri; Soukaina Filali Boubrahimi; Shah Muhammad Hamdi. “Shapelet-Based Counterfactual Explanations for Multivariate Time Series”. In: *arXiv:2208.10462* (2022).

[23] Hubert Baniecki et al. “dalex: Responsible Machine Learning with Interactive Explainability and Fairness in Python”. In: *Journal of Machine Learning Research* 22.214 (2021), pp. 1–7.

[24] Alejandro Barredo Arrieta et al. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: *Information Fusion* 58 (2020), pp. 82–115. DOI: `https://doi.org/10.1016/j.inffus.2019.12.012`.

[25] Mariam Barry; Albert Bifet; Jean-Luc Billy. “StreamAI: Dealing with Challenges of Continual Learning Systems for Serving AI in Production”. In: *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. Melbourne, Australia: IEEE, 2023, pp. 134–137. ISBN: 979-8-3503-0037-6. DOI: `10.1109/ICSE-SEIP58684.2023.00017`.

[26] João Bento et al. “TimeSHAP: Explaining Recurrent Models through Sequence Perturbations”. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. ACM, 2021, pp. 2565–2573. DOI: `10.1145/3447548.3467166`.

[27] Massimo Bertolini; Davide Mezzogori; Mattia Neroni; Francesco Zammori. “Machine Learning for industrial applications: A comprehensive literature review”. In: *Expert Systems with Applications* 175 (2021), p. 114820. DOI: `10.1016/j.eswa.2021.114820`.

[28] N. Beume et al. “On the Complexity of Computing the Hypervolume Indicator”. In: *IEEE Transactions on Evolutionary Computation* 13.5 (2009), pp. 1075–1082. DOI: `10.1109/TEVC.2009.2015575`.

[29] Umang Bhatt; Adrian Weller; José M. F. Moura. “Evaluating and aggregating feature-based model explanations”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. IJCAI’20. Yokohama, Yokohama, Japan: ijcai.org, 2021. ISBN: 9780999241165. DOI: `10.24963/IJCAI.2020/417`.

[30] Alexander Binder et al. “Morphological and molecular breast cancer profiling through explainable machine learning”. In: *Nature Machine Intelligence* 3.4 (2021), pp. 355–366. DOI: `10.1038/s42256-021-00303-4`.

[31] Szymon Bobek et al. “Why Industry 5.0 Needs XAI 2.0?” In: *xAI (Late-breaking Work, Demos, Doctoral Consortium)*. Vol. 3554. Lisbon, Portugal: CEUR Workshop Proceedings, 2023, pp. 1–6.

[32] Eugenio Brusa; Luca Cibrario; Cristiana Delprete; Luigi Gianpio Di Maggio. “Explainable AI for Machine Fault Diagnosis: Understanding Features’ Contribution in Machine Learning Models for Industrial Condition Monitoring”. In: *Applied Sciences* 13.4 (2023), p. 2038. DOI: 10.3390/app13042038.

[33] Lars Buitinck et al. “API design for machine learning software: experiences from the scikit-learn project”. In: *arXiv:1309.0238* (2013).

[34] Nadia Burkart; Marco F. Huber. “A Survey on the Explainability of Supervised Machine Learning”. In: *Journal of Artificial Intelligence Research* 70 (2021), pp. 245–317. DOI: 10.1613/jair.1.12228.

[35] Ruth M. J. Byrne. “Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. Macao, China: ijcai.org, 2019, pp. 6276–6282. ISBN: 978-0-9992411-4-1. DOI: 10.24963/ijcai.2019/876.

[36] Rich Caruana et al. “Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Sydney NSW Australia: ACM, 2015, pp. 1721–1730. ISBN: 978-1-4503-3664-2. DOI: 10.1145/2783258.2788613.

[37] J. A. Carvajal Soto; F. Tavakolizadeh; D. Gyulai. “An online machine learning framework for early detection of product failures in an Industry 4.0 context”. In: *International Journal of Computer Integrated Manufacturing* 32.4-5 (2019), pp. 452–465. DOI: 10.1080/0951192X.2019.1571238.

[38] J. A. Carvajal Soto; F. Tavakolizadeh; D. Gyulai. “An online machine learning framework for early detection of product failures in an Industry 4.0 context”. In: *International Journal of Computer Integrated Manufacturing* 32.4-5 (2019), pp. 452–465. DOI: 10.1080/0951192X.2019.1571238.

[39] Diogo V. Carvalho; Eduardo M. Pereira; Jaime S. Cardoso. “Machine Learning Interpretability: A Survey on Methods and Metrics”. In: *Electronics* 8.8 (2019), p. 832. DOI: 10.3390/electronics8080832.

[40] Augustin-Louis Cauchy. *Oeuvres complètes*. 1st ed. Cambridge University Press, 2009. ISBN: 978-1-108-00277-6. DOI: 10.1017/CBO9780511702396.

[41] Michael Chromik; Andreas Butz. “Human-XAI Interaction: A Review and Design Principles for Explanation User Interfaces”. In: *Human-Computer Interaction – INTERACT 2021*. Ed. by Carmelo Ardito et al. Vol. 12933. Series Title: Lecture Notes in Computer Science. Cham: Springer, 2021, pp. 619–640. ISBN: 978-3-030-85616-8. DOI: 10.1007/978-3-030-85616-8_36.

[42] Carlos A. Coello Coello; Margarita Reyes-Sierra. “Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art”. In: *International Journal of Computational Intelligence Research* 2.3 (2006). DOI: 10.5019/j.ijcir.2006.68.

[43] Bianca Maria Colosimo; Fabio Centofanti. “Model Interpretability, Explainability and Trust for Manufacturing 4.0”. In: *Interpretability for Industry 4.0 : Statistical and Machine Learning Approaches*. Ed. by Antonio Lepore; Biagio Palumbo; Jean-Michel Poggi. Cham: Springer, 2022, pp. 21–36. ISBN: 978-3-031-12402-0. DOI: 10.1007/978-3-031-12402-0_2.

[44] European Commission. “Proposal for a regulation of the European Parliament and the Council laying down harmonised rules on Artificial Intelligence (Artificial Intelligence Act) and amending certain Union legislative acts”. In: *EUR-Lex-52021PC0206* (2023).

[45] C. S. Cox et al. “Plan and operation of the NHANES I Epidemiologic Followup Study, 1992”. In: *Vital and Health Statistics. Ser. 1, Programs and Collection Procedures* 35 (1997), pp. 1–231.

[46] Jonathan Crabbé; Mihaela Van Der Schaar. “Explaining time series predictions with dynamic masks”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. PMLR. 2021, pp. 2166–2177.

[47] Logan Cummins et al. “Explainable predictive maintenance: a survey of current methods, challenges and opportunities”. In: *IEEE Access* 12 (2024), pp. 57574–57602.

[48] Susanne Dandl; Christoph Molnar; Martin Binder; Bernd Bischl. “Multi-Objective Counterfactual Explanations”. In: *Parallel Problem Solving from Nature*. Springer, 2020, 448–469. ISBN: 978-3-030-58111-4. DOI: `10.1007/978-3-030-58112-1_31`.

[49] Arun Das; Paul Rad. “Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey”. In: *arXiv:2006.11371* (2020).

[50] Amit Datta; Michael Carl Tschantz; Anupam Datta. “Automated Experiments on Ad Privacy Settings: A Tale of Opacity, Choice, and Discrimination”. In: *arXiv:1408.6491* (2015).

[51] Hoang Anh Dau et al. “The UCR Time Series Archive”. In: *arXiv:1810.07758*. arXiv: 1810.07758. arXiv, 2019.

[52] K. Deb; A. Pratap; S. Agarwal; T. Meyarivan. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: `10.1109/4235.996017`.

[53] Kalyanmoy Deb. “An introduction to genetic algorithms”. In: *Sadhana* 24.4-5 (1999), pp. 293–315. DOI: `10.1007/BF02823145`.

[54] Kalyanmoy Deb; Himanshu Jain. “An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints”. In: *IEEE Transactions on Evolutionary Computation* 18.4 (2014), pp. 577–601. DOI: `10.1109/TEVC.2013.2281535`.

[55] Eoin Delaney; Derek Greene; Mark T. Keane. “Instance-Based Counterfactual Explanations for Time Series Classification”. In: *Case-Based Reasoning Research and Development*. Vol. 12877. Series Title: Lecture Notes in Computer Science. Cham: Springer, 2021, pp. 32–47. ISBN: 978-3-030-86957-1. DOI: `10.1007/978-3-030-86957-1_3`.

[56] Li Deng; Dong Yu. “Deep Learning: Methods and Applications”. In: *Foundations and Trends® in Signal Processing* 7.3–4 (2014), pp. 197–387. DOI: `10.1561/2000000039`.

[57] S. Dhanalakshmi; S. Kannan; K. Mahadevan; S. Baskar. “Application of modified NSGA-II algorithm to Combined Economic and Emission Dispatch problem”. In: *International Journal of Electrical Power & Energy Systems* 33.4 (2011), pp. 992–1002. DOI: `10.1016/j.ijepes.2011.01.014`.

[58] Amit Dhurandhar et al. “Explanations based on the missing: Towards contrastive explanations with pertinent negatives”. In: *Advances in neural information processing systems*. Vol. 31. Curran Associates, Inc., 2018, pp. 590–601.

[59] R. Doddaiah; P. Parvatharaju; E. Rundensteiner; T. Hartvigsen. “Class-Specific Explainability for Deep Time Series Classifiers”. In: *2022 IEEE International Conference on Data Mining (ICDM)*. Los Alamitos, CA, USA: IEEE, 2022, pp. 101–110. DOI: `10.1109/ICDM54844.2022.00020`.

[60] Filip Karlo Dosilovic; Mario Brcic; Nikica Hlupic. “Explainable artificial intelligence: A survey”. In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018, pp. 0210–0215. ISBN: 978-953-233-095-3. DOI: `10.23919/MIPRO.2018.8400040`.

[61] Michael Downs et al. “Cruds: Counterfactual recourse using disentangled subspace”. In: *ICML WHI* 2020 (2020), pp. 1–23.

[62] Julia Dressel; Hany Farid. “The accuracy, fairness, and limits of predicting recidivism”. In: *Science Advances* 4.1 (2018). DOI: 10.1126/sciadv.aao5580.

[63] John Duchi; Elad Hazan; Yoram Singer. “Adaptive subgradient methods for online leaning and stochastic optimization”. In: *Journal of machine learning research* 12.7 (2011), pp. 2121–2159. ISSN: 1532-4435.

[64] John J. Dudley; Per Ola Kristensson. “A Review of User Interface Design for Interactive Machine Learning”. In: *ACM Transactions on Interactive Intelligent Systems* 8.2 (2018), pp. 1–37. DOI: 10.1145/3185517.

[65] C. Emmanouilidis; S. Waschull; J. A. C. Bokhorst; J. C. Wortmann. “Human in the AI Loop in Production Environments”. In: *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems*. Ed. by Alexandre Dolgui et al. Vol. 633. Series Title: IFIP Advances in Information and Communication Technology. Cham: Springer, 2021, pp. 331–342. ISBN: 978-3-030-85910-7. DOI: 10.1007/978-3-030-85910-7_35.

[66] Michael Emmerich; Nicola Beume; Boris Naujoks. “An EMO Algorithm Using the Hypervolume Measure as Selection Criterion”. In: *Evolutionary Multi-Criterion Optimization*. Vol. 3410. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 62–76. ISBN: 978-3-540-31880-4. DOI: 10.1007/978-3-540-31880-4_5.

[67] Carlos A. Escobar; Megan E. McGovern; Ruben Morales-Menendez. “Quality 4.0: a review of big data challenges in manufacturing”. In: *Journal of Intelligent Manufacturing* 32.8 (2021), pp. 2319–2334. DOI: 10.1007/s10845-021-01765-4.

[68] Philippe Esling; Carlos Agon. “Time-series data mining”. In: *ACM Computing Surveys* 45.1 (2012), pp. 1–34. DOI: 10.1145/2379776.2379788.

[69] Mojtaba A. Farahani; M.R. McCormick; Ramy Harik; Thorsten Wuest. “Time-series classification in smart manufacturing systems: An experimental evaluation of state-of-the-art machine learning algorithms”. In: *Robotics and Computer-Integrated Manufacturing* 91 (2025), p. 102839. DOI: https://doi.org/10.1016/j.rcim.2024.102839.

[70] Hassan Ismail Fawaz et al. “Deep learning for time series classification: a review”. In: *Data Mining and Knowledge Discovery* 33.4 (2019). arXiv: 1809.04356, pp. 917–963. DOI: 10.1007/s10618-019-00619-1.

[71] Ruth Fong; Mandela Patrick; Andrea Vedaldi. “Understanding Deep Networks via Extremal Perturbations and Smooth Masks”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 2950–2958. DOI: 10.1109/ICCV.2019.00304.

[72] Carlos M Fonseca; Peter J Fleming. “Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization.” In: *Proceedings of the 5th International Conference on Genetic Algorithms*. Vol. 93. Morgan Kaufmann Publishers Inc., 1993, pp. 416–423. ISBN: 1-55860-299-2. DOI: 10.5555/645513.657757.

[73] Felix Friedrich; Wolfgang Stammer; Patrick Schramowski; Kristian Kersting. “A typology for exploring the mitigation of shortcut behaviour”. In: *Nature Machine Intelligence* 5.3 (2023), pp. 319–330. DOI: 10.1038/s42256-023-00612-w.

[74] Krishna Gade et al. “Explainable AI in Industry”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA: ACM, 2019, pp. 3203–3204. ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3332281.

[75] Bhargav Gadhvi; Vimal Savsani; Vivek Patel. “Multi-Objective Optimization of Vehicle Passive Suspension System Using NSGA-II, SPEA2 and PESA-II”. In: *Procedia Technology* 23 (2016). 3rd International Conference on Innovations in Automation and Mechatronics Engineering 2016, ICIAME 2016 05-06 February, 2016, pp. 361–368. ISSN: 2212-0173. DOI: `https://doi.org/10.1016/j.protcy.2016.03.038`.

[76] Moncef Garouani et al. “Towards big industrial data mining through explainable automated machine learning”. In: *The International Journal of Advanced Manufacturing Technology* 120.1-2 (2022), pp. 1169–1188. DOI: `10.1007/s00170-022-08761-9`.

[77] Ross Girshick; Jeff Donahue; Trevor Darrell; Jitendra Malik. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, 2014, pp. 580–587. ISBN: 978-1-4799-5118-5. DOI: `10.1109/CVPR.2014.81`.

[78] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass: Addison-Wesley Pub. Co, 1989. ISBN: 978-0-201-15767-3.

[79] Ian Goodfellow; Yoshua Bengio; Aaron Courville. *Deep learning*. Adaptive computation and machine learning. Cambridge, Mass: The MIT press, 2016. ISBN: 978-0-262-03561-3.

[80] Yash Goyal et al. “Counterfactual visual explanations”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR, 2019, pp. 2376–2384.

[81] Riccardo Guidotti et al. “A Survey of Methods for Explaining Black Box Models”. In: *ACM Computing Surveys* 51.5 (2019), pp. 1–42. DOI: `10.1145/3236009`.

[82] Riccardo Guidotti et al. “Explaining Any Time Series Classifier”. In: *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*. Atlanta, GA, USA: IEEE, 2020, pp. 167–176. ISBN: 978-1-72814-144-2. DOI: `10.1109/CogMI50398.2020.00029`.

[83] Riccardo Guidotti et al. “Factual and Counterfactual Explanations for Black Box Decision Making”. In: *IEEE Intelligent Systems* 34.6 (2019), pp. 14–23. DOI: `10.1109/MIS.2019.2957223`.

[84] Mael Guilleme; Veronique Masson; Laurence Roze; Alexandre Termier. “Agnostic Local Explanation for Time Series Classification”. In: *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. Portland, OR, USA: IEEE, 2019, pp. 432–439. ISBN: 978-1-72813-798-8. DOI: `10.1109/ICTAI.2019.00067`.

[85] Isabelle Guyon; Steve Gunn; Asa Ben Hur; Gideon Dror. “Design and Analysis of the NIPS2003 Challenge”. In: *Feature Extraction*. Ed. by Isabelle Guyon; Masoud Nikravesh; Steve Gunn; Lotfi A. Zadeh. Vol. 207. Series Title: Studies in Fuzziness and Soft Computing. Berlin, Heidelberg: Springer, 2006, pp. 237–263. ISBN: 978-3-540-35488-8. DOI: `10.1007/978-3-540-35488-8_10`.

[86] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. First edition, fifth release. Beijing Boston Farnham Sebastopol Tokyo: O’Reilly, 2018. ISBN: 978-1-4919-6229-9.

[87] Patrick Hall et al. *Machine Learning Interpretability with H2O Driverless AI*. Available online at: `http://artifacts.h2o.ai.s3.amazonaws.com/releases/ai/h2o/dai/rel-1.6.1-12/docs/booklets/MLIBooklet.pdf`, last accessed on 31.05.2022. 2017.

[88] Kaiming He; Xiangyu Zhang; Shaoqing Ren; Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[89] Anna Hedström et al. “Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations and beyond”. In: *J. Mach. Learn. Res.* Vol. 24. 1. JMLR.org, 2024.

[90] Sepp Hochreiter; Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: `10.1162/neco.1997.9.8.1735`.

[91] Andreas Holzinger et al. “Explainable AI Methods - A Brief Overview”. In: *xxAI - Beyond Explainable AI*. Ed. by Andreas Holzinger et al. Vol. 13200. Series Title: Lecture Notes in Computer Science. Cham: Springer, 2022, pp. 13–38. ISBN: 978-3-031-04083-2. DOI: `10.1007/978-3-031-04083-2_2`.

[92] Brigt Håvardstun; Cèsar Ferri; Kristian Flikka; Jan Arne Telle. “XAI for Time Series Classification: Evaluating the Benefits of Model Inspection for End-Users”. In: *Explainable Artificial Intelligence*. Ed. by Luca Longo; Sebastian Lapuschkin; Christin Seifert. Vol. 2155. Series Title: Communications in Computer and Information Science. Cham: Springer Nature, 2024, pp. 439–453. ISBN: 978-3-031-63800-8. DOI: `10.1007/978-3-031-63800-8_22`.

[93] Jacqueline Höllig; Cedric Kulbach; Steffen Thoma. “TSEvo: Evolutionary Counterfactual Explanations for Time Series Classification”. In: *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*. Nassau, Bahamas: IEEE, 2022, pp. 29–36. ISBN: 978-1-66546-283-9. DOI: `10.1109/ICMLA55696.2022.00013`.

[94] Jacqueline Höllig; Cedric Kulbach; Steffen Thoma. “TSInterpret: A Python Package for the Interpretability of Time Series Classification”. In: *Journal of Open Source Software* 8.85 (2023), p. 5220. ISSN: 2475-9066. DOI: `10.21105/joss.05220`.

[95] Jacqueline Höllig; Cedric Kulbach; Steffen Thoma. *TSInterpret: A unified framework for time series interpretability*. arXiv:2208.05280. 2022.

[96] Jacqueline Höllig; Aniek F. Markus; Jef De Slegte; Prachi Bagave. “Semantic Meaningfulness: Evaluating Counterfactual Approaches for Real-World Plausibility and Feasibility”. In: *Explainable Artificial Intelligence*. Ed. by Luca Longo. Vol. 1902. Series Title: Communications in Computer and Information Science. Cham: Springer Nature Switzerland, 2023, pp. 636–659. ISBN: 978-3-031-44066-3 978-3-031-44067-0. DOI: `10.1007/978-3-031-44067-0_32`.

[97] Jacqueline Höllig; Steffen Thoma; Florian Grimm. “XTSC-Bench: Quantitative Benchmarking for Explainers on Time Series Classification”. In: *2023 22st IEEE International Conference on Machine Learning and Applications (ICMLA)*. Jacksonville, Florida, USA: IEEE, 2023, pp. 1126–1131. DOI: `10.1109/ICMLA58977.2023.00168`.

[98] Jacqueline Höllig; Steffen Thoma; Cedric Kulbach. “Evolutionary Counterfactual Visual Explanation”. In: *KI (Workshops)*. Vol. 3457. Trier, Germany: CEUR Workshop Proceedings, 2022.

[99] Hisao Ishibuchi; Yuji Sakane; Noritaka Tsukamoto; Yusuke Nojima. “Evolutionary many-objective optimization by NSGA-II and MOEA/D with large populations”. In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. San Antonio, TX, USA: IEEE, 2009, pp. 1758–1763. ISBN: 978-1-4244-2793-2. DOI: `10.1109/ICSMC.2009.5346628`.

[100] Hisao Ishibuchi; Yu Setoguchi; Hiroyuki Masuda; Yusuke Nojima. “Performance of Decomposition-Based Many-Objective Algorithms Strongly Depends on Pareto Front Shapes”. In: *IEEE Transactions on Evolutionary Computation* 21.2 (2017), pp. 169–190. DOI: `10.1109/TEVC.2016.2587749`.

[101] Aya Abdelsalam Ismail; Mohamed Gunady; Héctor Corrada Bravo; Soheil Feizi. “Benchmarking deep learning interpretability in time series predictions”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Vol. 33. NIPS ’20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[102] Senthil Kumar Jagatheesaperumal et al. “The Duo of Artificial Intelligence and Big Data for Industry 4.0: Applications, Techniques, Challenges, and Future Research Directions”. In: *IEEE Internet of Things Journal* 9.15 (2022), pp. 12861–12885. DOI: `10.1109/JIOT.2021.3139827`.

[103] Gargi Joshi; Rahee Walambe; Ketan Kotecha. “A Review on Explainability in Multimodal Deep Neural Nets”. In: *IEEE Access* 9 (2021), pp. 59800–59821. DOI: `10.1109/ACCESS.2021.3070212`.

[104] Amir-Hossein Karimi; Gilles Barthe; Borja Balle; Isabel Valera. “Model-agnostic counterfactual explanations for consequential decisions”. In: *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS*. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 895–905.

[105] Amir-Hossein Karimi; Gilles Barthe; Bernhard Schölkopf; Isabel Valera. “A Survey of Algorithmic Recourse: Contrastive Explanations and Consequential Recommendations”. In: *ACM Computing Surveys* 55.5 (2023), pp. 1–29. DOI: `10.1145/3527848`.

[106] Amir-Hossein Karimi; Julius von Kügelgen; Bernhard Schölkopf; Isabel Valera. “Algorithmic recourse under imperfect causal knowledge: a probabilistic approach”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS ’20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[107] Amir-Hossein Karimi; Bernhard Schölkopf; Isabel Valera. “Algorithmic Recourse: from Counterfactual Explanations to Interventions”. In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. Virtual Event Canada: ACM, 2021, pp. 353–362. ISBN: 978-1-4503-8309-7. DOI: `10.1145/3442188.3445899`.

[108] Mark T. Keane; Eoin M. Kenny; Eoin Delaney; Barry Smyth. “If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021*. ijcai.org, 2021, pp. 4466–4474.

[109] Christopher J. Kelly et al. “Key challenges for delivering clinical impact with artificial intelligence”. In: *BMC Medicine* 17.1 (2019), p. 195. DOI: `10.1186/s12916-019-1426-2`.

[110] Amir E. Khandani; Adlar J. Kim; Andrew W. Lo. “Consumer credit-risk models via machine-learning algorithms”. In: *Journal of Banking & Finance* 34.11 (2010), pp. 2767–2787. DOI: `10.1016/j.jbankfin.2010.06.001`.

[111] Been Kim. “Interactive and Interpretable Machine Learning Models for Human Machine Collaboration”. Phd. Massachusetts Institute of Technology, 2015.

[112] Been Kim; Rajiv Khanna; Oluwasanmi Koyejo. “Examples are not enough, learn to criticize! criticism for interpretability”. In: *Advances in Neural Information Processing Systems*. Vol. 29. NIPS’16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 2288–2296. ISBN: 978-1-5108-3881-9. DOI: `10.5555/3157096.3157352`.

[113] Sung Wook Kim; Jun Ho Kong; Sang Won Lee; Seungchul Lee. “Recent Advances of Artificial Intelligence in Manufacturing Industrial Sectors: A Review”. In: *International Journal of Precision Engineering and Manufacturing* 23.1 (2022), pp. 111–129. DOI: `10.1007/s12541-021-00600-3`.

[114] Diederik P. Kingma; Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations*. 2017.

[115] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the National Academy of Sciences* 114.13 (2017), pp. 3521–3526. DOI: `10.1073/pnas.1611835114`.

[116] Dani Kiyasseh; Tingting Zhu; David Clifton. “A clinical deep learning framework for continually learning from cardiac signals across diseases, time, modalities, and institutions”. In: *Nature Communications* 12.1 (2021), p. 4221. DOI: `10.1038/s41467-021-24483-0`.

[117] Janis Klaise; Arnaud Van Looveren; Giovanni Vacanti; Alexandru Coca. "Alibi Explain: Algorithms for Explaining Machine Learning Models". In: *Journal of Machine Learning Research* 22.181 (2021), pp. 1–7.

[118] Narine Kokhlikyan et al. "Captum: A unified and generic model interpretability library for PyTorch". In: *arXiv:2009.07896* (2020).

[119] Arzam Kotriwala et al. "XAI for Operations in the Process Industry – Applications, Theses, and Research Directions". In: *AAAI spring symposium: combining machine learning with knowledge engineering*. Vol. 2846. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 1–12.

[120] Alex Krizhevsky; Ilya Sutskever; Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems*. Vol. 25. 6. Curran Associates, Inc., 2017, pp. 84–90. DOI: `10.1145/3065386`.

[121] Fenareti Lampathaki; Carlos Agostinho; Yuri Glikman; Michele Sesana. "Moving from 'black box' to 'glass box' Artificial Intelligence in Manufacturing with XMANAI". In: *2021 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. Cardiff, United Kingdom: IEEE, 2021, pp. 1–6. ISBN: 978-1-66544-963-2. DOI: `10.1109/ICE/ITMC52061.2021.9570236`.

[122] Thibault Laugel; Marie-Jeanne Lesot; Christophe Marsala; Marcin Detyniecki. "Issues with post-hoc counterfactual explanations: a discussion". In: *arXiv:1906.04774* (2019).

[123] Thibault Laugel et al. "Comparison-Based Inverse Classification for Interpretability in Machine Learning". In: *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations*. Vol. 853. Cham: Springer, 2018, pp. 100–111. ISBN: 978-3-319-91473-2. DOI: `10.1007/978-3-319-91473-2\_9`.

[124] Thibault Laugel et al. "The dangers of post-hoc interpretability: unjustified counterfactual explanations". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. IJCAI'19. Macao, China: AAAI Press, 2019, 2801–2807. ISBN: 9780999241141.

[125] Walter Laurito et al. "AIDA-Vis – Automatic Data Visualization with Human Preferences". In: (2022). ISBN: 9783885797203 Publisher: Gesellschaft für Informatik, Bonn. ISSN: 1617-5468. DOI: `10.18420/INF2022_27`.

[126] Andrew R. Lawrence; Marcus Kaiser; Rui Sampaio; Maksim Sipos. "Data Generating Process to Evaluate Causal Discovery Techniques for Time Series Data". In: *arXiv:2104.08043* (2021).

[127] Thach Le Nguyen et al. "Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations". In: *Data mining and knowledge discovery* 33.4 (2019), pp. 1183–1222.

[128] Benjamin Leblanc; Pascal Germain. "On the Relationship Between Interpretability and Explainability in Machine Learning". In: *arXiv:2311.11491* (2024).

[129] Michael van Lent; William Fisher; Michael Mancuso. "An explainable artificial intelligence system for small-unit tactical behavior". In: *Proceedings of the national conference on artificial intelligence*. MIT Press, 2004, pp. 900–907.

[130] Simon Letzgus et al. "Toward Explainable Artificial Intelligence for Regression Models: A methodological perspective". In: *IEEE Signal Processing Magazine* 39.4 (2022), pp. 40–58. DOI: `10.1109/MSP.2022.3153277`.

[131] Peiyu Li; Omar Bahri; Soukaïna Filali Boubrahimi; Shah Muhammad Hamdi. "Attention-Based Counterfactual Explanation for Multivariate Time Series". In: *Big Data Analytics and Knowledge Discovery*. Ed. by Robert Wrembel et al. Vol. 14148. Series Title: Lecture Notes in Computer Science. Cham: Springer Nature, 2023, pp. 287–293. ISBN: 978-3-031-39831-5. DOI: `10.1007/978-3-031-39831-5_26`.

[132] Zhizhong Li; Derek Hoiem. “Learning Without Forgetting”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe; Jiri Matas; Nicu Sebe; Max Welling. Cham: Springer International Publishing, 2016, pp. 614–629. ISBN: 978-3-319-46493-0.

[133] Q. Vera Liao; Kush R. Varshney. “Human-Centered Explainable AI (XAI): From Algorithms to User Experiences”. In: *arXiv:2110.10790* (2022).

[134] Zachary C. Lipton. “The Mythos of Model Interpretability”. In: *Commun. ACM* 61.10 (2017). arXiv: 1606.03490, pp. 36–43.

[135] Shusen Liu; Bhavya Kailkhura; Donald Loveland; Yong Han. “Generative Counterfactual Introspection for Explainable Deep Learning”. In: *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Ottawa, ON, Canada: IEEE, 2019, pp. 1–5. ISBN: 978-1-72812-723-1. DOI: 10.1109/GlobalSIP45357.2019.8969491.

[136] David Lopez-Paz; Marc’Aurelio Ranzato. “Gradient episodic memory for continual learning”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, 6470–6479. ISBN: 9781510860964.

[137] Yang Lu. “Industry 4.0: A survey on technologies, applications and open research issues”. In: *Journal of Industrial Information Integration* 6 (2017), pp. 1–10. DOI: 10.1016/j.jii.2017.04.005.

[138] Ana Lucic; Harrie Oosterhuis; Hinda Haned; Maarten De Rijke. “FOCUS: Flexible Optimizable Counterfactual Explanations for Tree Ensembles”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.5 (2022), pp. 5313–5322. DOI: 10.1609/aaai.v36i5.20468.

[139] Scott M. Lundberg; Su-In Lee. “A unified approach to interpreting model predictions”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Vol. 30. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, 4768–4777. ISBN: 9781510860964.

[140] Divyat Mahajan; Chenhao Tan; Amit Sharma. “Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers”. In: *arXiv:1912.03277* (2020).

[141] Spyros Makridakis; Evangelos Spiliotis; Vassilios Assimakopoulos. “M5 accuracy competition: Results, findings, and conclusions”. In: *International Journal of Forecasting* 38.4 (2022), pp. 1346–1364. DOI: 10.1016/j.ijforecast.2021.11.013.

[142] Spyros Makridakis; Evangelos Spiliotis; Vassilios Assimakopoulos. “The M5 competition: Background, organization, and implementation”. In: *International Journal of Forecasting* 38.4 (2022), pp. 1325–1336. DOI: 10.1016/j.ijforecast.2021.07.007.

[143] Ričards Marcinkevičs; Julia E. Vogt. “Interpretability and Explainability: A Machine Learning Zoo Mini-tour”. In: *arXiv:2012.01805* (2023).

[144] Aniek F. Markus; Jan A. Kors; Peter R. Rijnbeek. “The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies”. In: *Journal of Biomedical Informatics* 113 (2021), p. 103655. DOI: 10.1016/j.jbi.2020.103655.

[145] R.T. Marler; J.S. Arora. “Survey of multi-objective optimization methods for engineering”. In: *Structural and Multidisciplinary Optimization* 26.6 (2004), pp. 369–395. DOI: 10.1007/s00158-003-0368-6.

[146] Anabel Martínez-Vargas et al. “Application of NSGA-II algorithm to the spectrum assignment problem in spectrum sharing networks”. In: *Applied Soft Computing* 39 (2016), pp. 188–198. DOI: 10.1016/j.asoc.2015.11.010.

[147] Benjamin Maschler; Hannes Vietz; Nasser Jazdi; Michael Weyrich. “Continual Learning of Fault Prediction for Turbofan Engines using Deep Learning with Elastic Weight Consolidation”. In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vienna, Austria: IEEE, 2020, pp. 959–966. ISBN: 978-1-72818-956-7. DOI: `10.1109/ETFA46521.2020.9211903`.

[148] Benjamin Maschler et al. “Insights and Example Use Cases on Industrial Transfer Learning”. In: *Procedia CIRP* 107 (2022), pp. 511–516. DOI: `10.1016/j.procir.2022.05.017`.

[149] Warren S. McCulloch; Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133. DOI: `10.1007/BF02478259`.

[150] Han Meng; Christian Wagner; Isaac Triguero. “Explaining time series classifiers through meaningful perturbation and optimisation”. In: *Information Sciences* 645 (2023), p. 119334. DOI: `10.1016/j.ins.2023.119334`.

[151] Raphael Meudec. *tf-explain*. 2021. DOI: `10.5281/ZENODO.5711704`.

[152] John A. Miller et al. “A Survey of Deep Learning and Foundation Models for Time Series Forecasting”. In: *arXiv:2401.13912* (2024).

[153] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial Intelligence* 267 (2019), pp. 1–38. DOI: `https://doi.org/10.1016/j.artint.2018.07.007`.

[154] Tom M. Mitchell. *Machine Learning*. McGraw-Hill series in computer science. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[155] Navid Mohammadi Foumani et al. “Deep Learning for Time Series Classification and Extrinsic Regression: A Current Survey”. In: *ACM Comput. Surv.* 56.9 (2024). DOI: `10.1145/3649448`.

[156] Christoph Molnar. *Interpretable machine learning: a guide for making black box models explainable*. Second edition. Munich, Germany: Christoph Molnar, 2022. ISBN: 979-8411463330.

[157] Ramaravind Kommiya Mothilal; Amit Sharma; Chenhao Tan. “Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations”. In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. arXiv: 1905.07697. ACM, 2020, pp. 607–617. DOI: `10.1145/3351095.3372850`.

[158] Felix Mujkanovic et al. “timeXplain – A Framework for Explaining the Predictions of Time Series Classifiers”. In: *arXiv:2007.07606* (2020).

[159] Mohsin Munir et al. “TSXplain: Demystification of DNN Decisions for Time-Series Using Natural Language and Statistical Features”. In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings*. Munich, Germany: Springer, 2019, 426–439. ISBN: 978-3-030-30492-8. DOI: `10.1007/978-3-030-30493-5_43`.

[160] W. James Murdoch et al. “Definitions, methods, and applications in interpretable machine learning”. In: *Proceedings of the National Academy of Sciences* 116.44 (2019), pp. 22071–22080. DOI: `10.1073/pnas.1900654116`.

[161] Alexander Naumann et al. “Literature Review: Computer Vision Applications in Transportation Logistics and Warehousing”. In: *arXiv:2304.06009* (2023).

[162] Inês Neves et al. “Interpretable heartbeat classification using local model-agnostic explanations on ECGs”. In: *Computers in Biology and Medicine* 133 (2021), p. 104393. DOI: `10.1016/j.compbiomed.2021.104393`.

[163] An-phi Nguyen; María Rodríguez Martínez. “On quantitative aspects of model interpretability”. In: *arXiv:2007.07584* (2020).

[164] Huu Du Nguyen; Kim Phuc Tran. "Artificial Intelligence for Smart Manufacturing in Industry 5.0: Methods, Applications, and Challenges". In: *Artificial Intelligence for Smart Manufacturing*. Ed. by Kim Phuc Tran. Series Title: Springer Series in Reliability Engineering. Cham: Springer, 2023, pp. 5–33. ISBN: 978-3-031-30510-8. DOI: `10.1007/978-3-031-30510-8_2`.

[165] Thu Nguyen; Alessandro Canossa; Jichen Zhu. "How Human-Centered Explainable AI Interface Are Designed and Evaluated: A Systematic Survey". In: *arXiv:2403.14496* (2024).

[166] Alexander Nikitin; Letizia Iannucci; Samuel Kaski. "TSGM: A Flexible Framework for Generative Modeling of Synthetic Time Series". In: *arXiv:2305.11567* (2024).

[167] Harsha Nori; Samuel Jenkins; Paul Koch; Rich Caruana. "InterpretML: A Unified Framework for Machine Learning Interpretability". In: *arXiv:1909.09223* (2019).

[168] Oracle. *Skater*. Available online at: `https://github.com/oracle/Skater`, last accessed on 31.05.2022. 2017.

[169] eli5 org. *ELI5*. Available online at: `https://github.com/eli5-org/eli5`, last accessed on 31.05.2022. 2017.

[170] Utku Ozbulak. *PyTorch CNN Visualizations*. Available Online at `https://github.com/utkuozbulak/pytorch-cnn-visualizations`, last accessed 31.08.2024. 2019.

[171] Cecilia Panigutti et al. "The role of explainable AI in the context of the AI Act". In: *2023 ACM Conference on Fairness, Accountability, and Transparency*. Chicago IL USA: ACM, 2023, pp. 1139–1150. ISBN: 9798400701924. DOI: `10.1145/3593013.3594069`.

[172] German I. Parisi et al. "Continual lifelong learning with neural networks: A review". In: *Neural Networks* 113 (2019), pp. 54–71. DOI: `10.1016/j.neunet.2019.01.012`.

[173] Martin Pawelczyk; Klaus Broelemann; Gjergji Kasneci. "Learning Model-Agnostic Counterfactual Explanations for Tabular Data". In: *Proceedings of The Web Conference 2020*. Taipei Taiwan: ACM, 2020, pp. 3126–3132. ISBN: 978-1-4503-7023-3. DOI: `10.1145/3366423.3380087`.

[174] Martin Pawelczyk et al. "CARLA: A Python Library to Benchmark Algorithmic Recourse and Counterfactual Explanation Algorithms". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Vol. 1. 2021.

[175] Judea Pearl. *Causality: models, reasoning, and inference*. Cambridge, U.K. ; New York: Cambridge University Press, 2000. ISBN: 978-0-521-77362-1.

[176] Ricardo Silva Peres et al. "Industrial Artificial Intelligence in Industry 4.0 - Systematic Review, Challenges and Outlook". In: *IEEE Access* 8 (2020), pp. 220121–220139. DOI: `10.1109/ACCESS.2020.3042874`.

[177] Maya Perlmutter; Ryan Gifford; Samantha Krening. "Impact of example-based XAI for neural networks on trust, understanding, and performance". In: *International Journal of Human-Computer Studies* 188 (2024), p. 103277. DOI: `10.1016/j.ijhcs.2024.103277`.

[178] Jonas Peters; Dominik Janzing; Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. Adaptive computation and machine learning. Cambridge, Mass: The MIT press, 2017. ISBN: 978-0-262-03731-0.

[179] Oleg S. Pianykh et al. "Continuous Learning AI in Radiology: Implementation Principles and Early Applications". In: *Radiology* 297.1 (2020), pp. 6–14. DOI: `10.1148/radiol.2020200038`.

[180] Audrey Poinsot et al. "Learning Structural Causal Models through Deep Generative Models: Methods, Guarantees, and Challenges". In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*. ijcai.org, 2024, pp. 8207–8215.

[181] Teodora Popordanoska; Mohit Kumar; Stefano Teso. “Toward Machine-Guided, Human-Initiated Explanatory Interactive Learning”. In: *arXiv:2007.10018* (2020).

[182] Rafael Poyiadzi et al. “FACE: Feasible and Actionable Counterfactual Explanations”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. ACM, 2020, 344–350. DOI: `10.1145/3375627.3375850`.

[183] Rafael Poyiadzi et al. “FACE: Feasible and Actionable Counterfactual Explanations”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. New York NY USA: ACM, 2020, pp. 344–350. ISBN: 978-1-4503-7110-0. DOI: `10.1145/3375627.3375850`.

[184] Qingfu Zhang; Hui Li. “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition”. In: *IEEE Transactions on Evolutionary Computation* 11.6 (2007), pp. 712–731. DOI: `10.1109/TEVC.2007.892759`.

[185] M. Rabiee; M. Zandieh; P. Ramezani. “Bi-objective partial flexible job shop scheduling problem: NSGA-II, NRGA, MOGA and PAES approaches”. In: *International Journal of Production Research* 50.24 (2012), pp. 7327–7342. DOI: `10.1080/00207543.2011.648280`.

[186] Maithra Raghu et al. “The Algorithmic Automation Problem: Prediction, Triage, and Human Effort”. In: *arXiv:1903.12220* (2019).

[187] Alvin Rajkomar et al. “Scalable and accurate deep learning with electronic health records”. In: *npj Digital Medicine* 1.1 (2018), p. 18. DOI: `10.1038/s41746-018-0029-1`.

[188] Abbavaram Gowtham Reddy; Vineeth N Balasubramanian. “On Causally Disentangled Representation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 7. AAAI Press, 2022, pp. 8089–8097. DOI: `10.1609/aaai.v36i7.20781`.

[189] Carl O. Retzlaff et al. “Post-hoc vs ante-hoc explanations: xAI design guidelines for data scientists”. In: *Cognitive Systems Research* 86 (2024), p. 101243. DOI: `10.1016/j.cogsys.2024.101243`.

[190] Marco Tulio Ribeiro; Sameer Singh; Carlos Guestrin. “Anchors: High-Precision Model-Agnostic Explanations”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 32. 1. AAAI Press, 2018, pp. 1527–1535.

[191] Marco Tulio Ribeiro; Sameer Singh; 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*. KDD ’16. San Francisco, California, USA: ACM, 2016, 1135–1144. ISBN: 9781450342322. DOI: `10.1145/2939672.2939778`.

[192] Laura Rieger; Chandan Singh; W. James Murdoch; Bin Yu. “Interpretations are useful: penalizing explanations to align neural networks with prior knowledge”. In: *arXiv:1909.13584* (2019). DOI: `10.48550/ARXIV.1909.13584`.

[193] Raúl Rojas. *Neural Networks*. Berlin, Heidelberg: Springer, 1996. ISBN: 978-3-642-61068-4. DOI: `10.1007/978-3-642-61068-4`.

[194] Thomas Rojat et al. “Explainable Artificial Intelligence (XAI) on TimeSeries Data: A Survey”. In: *arXiv:2104.00950* (2021).

[195] Andrew Slavin Ross; Michael C. Hughes; Finale Doshi-Velez. “Right for the right reasons: training differentiable models by constraining their explanations”. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI’17. Melbourne, Australia: AAAI Press, 2017, 2662–2670. ISBN: 9780999241103.

[196] Jože M. Rožanec et al. “Human in the AI Loop via xAI and Active Learning for Visual Inspection”. In: *Artificial Intelligence in Manufacturing*. Ed. by John Soldatos. Cham: Springer Nature, 2024, pp. 381–406. ISBN: 978-3-031-46452-2. DOI: `10.1007/978-3-031-46452-2_22`.

[197] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv:1609.04747* (2017).

[198] Alejandro Pasos Ruiz et al. “The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances”. In: *Data Mining and Knowledge Discovery* 35.2 (2021), pp. 401–449. DOI: `10.1007/s10618-020-00727-3`.

[199] David E. Rumelhart; Geoffrey E. Hinton; Ronald J. Williams. “Learning internal representations by error propagation”. In: *Neurocomputing, Volume 1*. Ed. by James A. Anderson; Edward Rosenfeld. The MIT Press, 1988, pp. 675–695. ISBN: 978-0-262-26713-7. DOI: `10.7551/mitpress/4943.003.0128`.

[200] Stuart J. Russell et al. *Artificial intelligence: a modern approach*. Fourth edition, global edition. Pearson series in artificial intelligence. Harlow: Pearson, 2022. ISBN: 978-1-292-40113-3.

[201] Andrei A. Rusu et al. “Progressive Neural Networks”. In: *arXiv:1606.04671* (2022).

[202] Wojciech Samek; Alexander Binder; Sebastian Lapuschkin; Klaus-Robert Muller. “Understanding and Comparing Deep Neural Networks for Age and Gender Classification”. In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. Venice, Italy: IEEE, 2017, pp. 1629–1638. ISBN: 978-1-5386-1034-3. DOI: `10.1109/ICCVW.2017.191`.

[203] Wojciech Samek et al. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Ed. by Wojciech Samek et al. Vol. 11700. Lecture Notes in Computer Science. Cham: Springer, 2019. ISBN: 978-3-030-28954-6. DOI: `10.1007/978-3-030-28954-6`.

[204] Isak Samsten. *isaksamsten/wildboar: wildboar*. 2020. DOI: `10.5281/ZENODO.4264063`.

[205] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: `10.1147/rd.33.0210`.

[206] Anirban Sarkar; Deepak Vijaykeerthy; Anindya Sarkar; Vineeth N Balasubramanian. “A Framework for Learning Ante-hoc Explainable Models via Concepts”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 10276–10285. DOI: `10.1109/CVPR52688.2022.01004`.

[207] Udo Schlegel; Duy Lam Vo; Daniel A. Keim; Daniel Seebacher. “TS-MULE: Local Interpretable Model-Agnostic Explanations for Time Series Forecast Models”. In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Vol. 1524. Series Title: Communications in Computer and Information Science. Cham: Springer, 2021, pp. 5–14. ISBN: 978-3-030-93736-2. DOI: `10.1007/978-3-030-93736-2_1`.

[208] Udo Schlegel et al. “Towards a Rigorous Evaluation of XAI Methods on Time Series”. In: *2019 IEEE/CVF International Conference on Computer Vision Workshops*. IEEE, 2019, pp. 4197–4201.

[209] Philipp Schmidt; Felix Biessmann. “Quantifying Interpretability and Trust in Machine Learning Systems”. In: *arXiv:1901.08558* (2019).

[210] Patrick Schramowski et al. “Making deep neural networks right for the right scientific reasons by interacting with their explanations”. In: *Nature Machine Intelligence* 2.8 (2020), pp. 476–486.

[211] Gesina Schwalbe; Bettina Finzel. “A Comprehensive Taxonomy for Explainable Artificial Intelligence: A Systematic Survey of Surveys on Methods and Concepts”. In: *Data Mining and Knowledge Discovery* 38.5 (2023), pp. 3043–3101. DOI: `10.1007/s10618-022-00867-8`.

[212] Ramprasaath R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Vol. 128. 2. IEEE, 2020, pp. 336–359. DOI: `10.1007/s11263-019-01228-7`.

[213] Ramprasaath Ramasamy Selvaraju et al. “Taking a HINT: Leveraging Explanations to Make Vision and Language Models More Grounded”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2019, pp. 2591–2600. DOI: `10.1109/ICCV.2019.00268`.

[214] Fabio Sgarbossa et al. “Human factors in production and logistics systems of the future”. In: *Annual Reviews in Control* 49 (2020), pp. 295–305. DOI: `10.1016/j.arcontrol.2020.04.007`.

[215] Shubham Sharma; Jette Henderson; Joydeep Ghosh. “CERTIFAI: A Common Framework to Provide Explanations and Analyse the Fairness and Robustness of Black-box Models”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. New York NY USA: ACM, 2020, pp. 166–172. ISBN: 978-1-4503-7110-0. DOI: `10.1145/3375627.3375812`.

[216] Avanti Shrikumar; Peyton Greenside; Anshul Kundaje. “Learning important features through propagating activation differences”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. ICML’17. Sydney, NSW, Australia: JMLR, 2017, 3145–3153.

[217] Shoaib Ahmed Siddiqui; Dominique Mercier; Andreas Dengel; Sheraz Ahmed. “TSInsight: A Local-Global Attribution Framework for Interpretability in Time Series Data”. In: *Sensors* 21.21 (2021), p. 7373. DOI: `10.3390/s21217373`.

[218] Shoaib Ahmed Siddiqui et al. “TSViz: Demystification of Deep Learning Models for Time-Series Analysis”. In: *IEEE Access* 7 (2019). arXiv: 1802.02952, pp. 67027–67040. DOI: `10.1109/ACCESS.2019.2912823`.

[219] Karen Simonyan; Andrea Vedaldi; Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *arXiv:1312.6034* (2014).

[220] Torty Sivill; Peter Flach. “LIMESegment: Meaningful, Realistic Time Series Explanations”. In: *International Conference on Artificial Intelligence and Statistics*. Vol. 151. PMLR, 2022, p. 16.

[221] Daniel Smilkov et al. “SmoothGrad: removing noise by adding noise”. In: *arXiv:1706.03825* (2017).

[222] Barry Smyth; Mark T. Keane. “A Few Good Counterfactuals: Generating Interpretable, Plausible and Diverse Counterfactual Explanations”. In: *Case-Based Reasoning Research and Development*. Ed. by Mark T. Keane; Nirmalie Wiratunga. Vol. 13405. Series Title: Lecture Notes in Computer Science. Cham: Springer, 2022, pp. 18–32. ISBN: 978-3-031-14923-8. DOI: `10.1007/978-3-031-14923-8_2`.

[223] Georgios Sofianidis; Jože M. Rožanec; Dunja Mladenić; Dimosthenis Kyriazis. “A Review of Explainable Artificial Intelligence in Manufacturing”. In: *arXiv:2107.02295* (2021). DOI: `10.48550/ARXIV.2107.02295`.

[224] Hamit Soyel; Umut Tekguc; Hasan Demirel. “Application of NSGA-II to feature selection for facial expression recognition”. In: *Computers & Electrical Engineering* 37.6 (2011), pp. 1232–1240. DOI: `10.1016/j.compeleceng.2011.01.010`.

[225] Francesco Spinnato et al. “Understanding Any Time Series Classifier with a Subsequence-based Explainer”. In: *ACM Transactions on Knowledge Discovery from Data* 18.2 (2024), pp. 1–34. DOI: `10.1145/3624480`.

[226] Thomas Spooner et al. “Counterfactual Explanations for Arbitrary Regression Models”. In: *arXiv:2106.15212* (2021).

[227] M. Srinivas; L.M. Patnaik. “Adaptive probabilities of crossover and mutation in genetic algorithms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 24.4 (1994), pp. 656–667. DOI: `10.1109/21.286385`.

[228] D. Sundararajan. *The discrete fourier transform: theory, algorithms and applications*. Singapore ; River Edge, NJ: World Scientific, 2001. ISBN: 978-981-02-4521-4.

[229] Mukund Sundararajan; Ankur Taly; Qiqi Yan. “Axiomatic attribution for deep networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. ICML’17. Sydney, NSW, Australia: PMLR, 2017, 3319–3328.

[230] Gian Antonio Susto; Angelo Cenedese; Matteo Terzi. “Time-Series Classification Methods: Review and Applications to Power Systems Data”. In: *Big Data Application in Power Systems*. Ed. by Reza Arghandeh; Yuxun Zhou. Elsevier, 2018, pp. 179–220. ISBN: 978-0-12-811968-6. DOI: `10.1016/B978-0-12-811968-6.00009-7`.

[231] Christian Szegedy et al. “Going deeper with convolutions”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2015, pp. 1–9. DOI: `10.1109/CVPR.2015.7298594`.

[232] Sarah Tan; Julius Adebayo; Kori Inkpen; Ece Kamar. “Investigating Human + Machine Complementarity for Recidivism Predictions”. In: *arXiv:1808.09123* (2018).

[233] tensorflow. *Lucid*. Available online at: `https://github.com/tensorflow/lucid`, last accessed on 31.05.2022. 2018.

[234] Hasan Tercan; Philipp Deibert; Tobias Meisen. “Continual learning of neural networks for quality prediction in production using memory aware synapses and weight transfer”. In: *Journal of Intelligent Manufacturing* 33.1 (2022), pp. 283–292. DOI: `10.1007/s10845-021-01793-0`.

[235] Hasan Tercan; Alexandro Guajardo; Tobias Meisen. “Industrial Transfer Learning: Boosting Machine Learning in Production”. In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Helsinki, Finland: IEEE, 2019, pp. 274–279. ISBN: 978-1-72812-927-3. DOI: `10.1109/INDIN41052.2019.8972099`.

[236] Stefano Teso; Öznur Alkan; Wolfgang Stammer; Elizabeth Daly. “Leveraging explanations in interactive machine learning: An overview”. In: *Frontiers in Artificial Intelligence* 6 (2023), p. 1066049. DOI: `10.3389/frai.2023.1066049`.

[237] Stefano Teso; Kristian Kersting. “Explanatory Interactive Machine Learning”. In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. ACM, 2019, p. 7.

[238] Andreas Theissler; Francesco Spinnato; Udo Schlegel; Riccardo Guidotti. “Explainable AI for Time Series Classification: A Review, Taxonomy and Research Directions”. In: *IEEE Access* 10 (2022), pp. 100700–100724. DOI: `10.1109/ACCESS.2022.3207765`.

[239] Erico Tjoa; Cuntai Guan. “A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.11 (2021), pp. 4793–4813. DOI: `10.1109/TNNLS.2020.3027314`.

[240] Mohamed-Ali Tnani; Michael Feil; Klaus Diepold. “Smart Data Collection System for Brownfield CNC Milling Machines: A New Benchmark Dataset for Data-Driven Machine Monitoring”. In: *Procedia CIRP* 107 (2022), pp. 131–136. DOI: `10.1016/j.procir.2022.04.022`.

[241] Gabriele Tolomei; Fabrizio Silvestri; Andrew Haines; Mounia Lalmas. “Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Halifax NS Canada: ACM, 2017, pp. 465–474. ISBN: 978-1-4503-4887-4. DOI: `10.1145/3097983.3098039`.

[242] Sana Tonekaboni et al. “What went wrong and when? instance-wise feature importance for time-series black-box models”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Vol. 33. NIPS ’20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[243] Berk Ustun; Alexander Spangher; Yang Liu. “Actionable Recourse in Linear Classification”. In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. Atlanta GA USA: ACM, 2019, pp. 10–19. ISBN: 978-1-4503-6125-5. DOI: 10.1145/3287560.3287566.

[244] Gido M. Van De Ven; Tinne Tuytelaars; Andreas S. Tolias. “Three types of incremental learning”. In: *Nature Machine Intelligence* 4.12 (2022), pp. 1185–1197. DOI: 10.1038/s42256-022-00568-3.

[245] Arnaud Van Looveren; Janis Klaise. “Interpretable Counterfactual Explanations Guided by Prototypes”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Vol. 12976. Cham: Springer, 2021, pp. 650–665. ISBN: 978-3-030-86520-7. DOI: 10.1007/978-3-030-86520-7_40.

[246] Sahil Verma et al. “Counterfactual Explanations and Algorithmic Recourses for Machine Learning: A Review”. In: *ACM Comput. Surv.* (2024). Just Accepted. DOI: 10.1145/3677119.

[247] Vivek Vijayakumar; Fabio Sgarbossa; W. Patrick Neumann; Ahmad Sobhani. “Framework for incorporating human factors into production and logistics systems”. In: *International Journal of Production Research* 60.2 (2022), pp. 402–419. DOI: 10.1080/00207543.2021.1983225.

[248] Giulia Vilone; Luca Longo. “Notions of explainability and evaluation approaches for explainable artificial intelligence”. In: *Information Fusion* 76 (2021), pp. 89–106. DOI: 10.1016/j.inffus.2021.05.009.

[249] Paul Voigt; Axel Von Dem Bussche. “Enforcement and Fines Under the GDPR”. In: *The EU General Data Protection Regulation (GDPR)*. Cham: Springer, 2017, pp. 201–217. ISBN: 978-3-319-57959-7. DOI: 10.1007/978-3-319-57959-7_7.

[250] Paul Voigt; Axel Von Dem Bussche. *The EU General Data Protection Regulation (GDPR)*. Cham: Springer, 2017. ISBN: 978-3-319-86291-0. DOI: 10.1007/978-3-319-57959-7.

[251] Simon Vollert; Martin Atzmueller; Andreas Theissler. “Interpretable Machine Learning: A brief survey from the predictive maintenance perspective”. In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vasteras, Sweden: IEEE, 2021, pp. 01–08. ISBN: 978-1-72812-989-1. DOI: 10.1109/ETFA45728.2021.9613467.

[252] Sandra Wachter; Brent Mittelstadt; Chris Russell. “Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR”. In: *arXiv:1711.00399*. arXiv, 2018.

[253] Emily Wall; Soroush Ghorashi; Gonzalo Ramos. “Using Expert Patterns in Assisted Interactive Machine Learning: A Study in Machine Teaching”. In: *Human-Computer Interaction – INTERACT 2019*. Vol. 11748. Series Title: Lecture Notes in Computer Science. Cham: Springer, 2019, pp. 578–599. ISBN: 978-3-030-29387-1. DOI: 10.1007/978-3-030-29387-1_34.

[254] Jiafu Wan et al. “Artificial-Intelligence-Driven Customized Manufacturing Factory: Key Technologies, Applications, and Challenges”. In: *Proceedings of the IEEE* 109.4 (2021), pp. 377–398. DOI: 10.1109/JPROC.2020.3034808.

[255] Huan Wang; Zhiliang Liu; Dandan Peng; Yong Qin. “Understanding and Learning Discriminant Features based on Multiattention 1DCNN for Wheelset Bearing Fault Diagnosis”. In: *IEEE Transactions on Industrial Informatics* 16.9 (2020), pp. 5735–5745. DOI: 10.1109/TII.2019.2955540.

[256] Jiaxuan Wang; Jenna Wiens; Scott Lundberg. “Shapley flow: A graph-based approach to interpreting model predictions”. In: *International Conference on Artificial Intelligence and Statistics*. Vol. 130. PMLR, 2021, pp. 721–729.

[257] Zhendong Wang et al. “Glacier: guided locally constrained counterfactual explanations for time series classification”. In: *Machine Learning* (2024). DOI: 10.1007/s10994-023-06502-x.

[258] Sabine Waschull; Christos Emmanouilidis. “Development and application of a human-centric co-creation design method for AI-enabled systems in manufacturing”. In: *IFAC-PapersOnLine* 55.2 (2022). 14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022, pp. 516–521. DOI: 10.1016/j.ifacol.2022.04.246.

[259] Philip B. Weerakody; Kok Wai Wong; Guanjin Wang; Wendell Ela. “A review of irregular time series data handling with gated recurrent neural networks”. In: *Neurocomputing* 441 (2021), pp. 161–178. DOI: 10.1016/j.neucom.2021.02.046.

[260] James Wexler et al. “The What-If Tool: Interactive Probing of Machine Learning Models”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2019), pp. 56–65. DOI: 10.1109/TVCG.2019.2934619.

[261] Xingjiao Wu et al. “A Survey of Human-in-the-loop for Machine Learning”. In: *Future Generation Computer Systems* 135 (2022). arXiv:2108.00941, pp. 364–381. DOI: 10.1016/j.future.2022.05.014.

[262] Yue Wu et al. “Incremental Classifier Learning with Generative Adversarial Networks”. In: *arXiv:1802.00853* (2018).

[263] Guangyue Xu; Hualiu Yang; Peter Schwarz. “A strengthened relationship between electricity and economic growth in China: An empirical study with a structural equation model”. In: *Energy* 241 (2022), p. 122905. DOI: 10.1016/j.energy.2021.122905.

[264] Qiang Yang; Xindong Wu. “10 CHALLENGING PROBLEMS IN DATA MINING RESEARCH”. In: *International Journal of Information Technology & Decision Making* 5.4 (2006), pp. 597–604. DOI: 10.1142/S0219622006002258.

[265] Wenzhuo Yang; Hung Le; Silvio Savarese; Steven C. H. Hoi. “OmniXAI: A Library for Explainable AI”. In: *arXiv:2206.01612* (2022).

[266] Xianzhe Yang; Changsheng Zhu. “Industrial Expert Systems Review: A Comprehensive Analysis of Typical Applications”. In: *IEEE Access* 12 (2024), pp. 88558–88584. DOI: 10.1109/ACCESS.2024.3419047.

[267] Lexiang Ye; Eamonn Keogh. “Time series shapelets: a novel technique that allows accurate, interpretable and fast classification”. In: vol. 34. Springer Nature, 2011, pp. 149–182.

[268] Matthew D. Zeiler; Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet; Tomas Pajdla; Bernt Schiele; Tinne Tuytelaars. Vol. 8689. Cham: Springer, 2014, pp. 818–833. ISBN: 978-3-319-10590-1. DOI: 10.1007/978-3-319-10590-1_53.

[269] Friedemann Zenke; Ben Poole; Surya Ganguli. “Continual learning through synaptic intelligence”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. ICML’17. Sydney, NSW, Australia: JMLR, 2017, 3987–3995.

[270] Chen Zeno; Itay Golan; Elad Hoffer; Daniel Soudry. *Task Agnostic Continual Learning Using Online Variational Bayes*. 2019.

[271] Nan Zhang; Rami Bahsoon; Nikos Tziritas; Georgios Theodoropoulos. “Explainable Human-in-the-Loop Dynamic Data-Driven Digital Twins”. In: *Dynamic Data Driven Applications Systems*. Cham: Springer Nature, 2024, pp. 233–243. ISBN: 978-3-031-52670-1.

[272] Yuyi Zhang et al. “ShapTime: A General XAI Approach for Explainable Time Series Forecasting”. In: *Intelligent Systems and Applications*. Ed. by Kohei Arai. Vol. 822. Series Title: Lecture Notes in Networks and Systems. Cham: Springer Nature, 2024, pp. 659–673. ISBN: 978-3-031-47721-8. DOI: 10.1007/978-3-031-47721-8_45.

[273] Zhenge Zhao; Panpan Xu; Carlos Scheidegger; Liu Ren. “Human-in-the-loop Extraction of Interpretable Concepts in Deep Learning Models”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2022), 780–790. DOI: `10.1109/TVCG.2021.3114837`.

[274] Eckart Zitzler; Simon Künzli. “Indicator-Based Selection in Multiobjective Search”. In: *Parallel Problem Solving from Nature - PPSN VIII*. Vol. 3242. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 832–842. ISBN: 978-3-540-30217-9. DOI: `10.1007/978-3-540-30217-9_84`.