# MetricLib: A Modular and Extensible Toolkit for Evaluation of Medical ML Datasets

Martin Seyferth
martin.seyferth@ptb.de
Physikalisch-Technische Bundesanstalt & BIFOLD
Berlin, Germany

Katinka Becker
katinka.becker@ptb.de
Physikalisch-Technische Bundesanstalt
Berlin, Germany

Tobias Schaeffter
tobias.schaeffter@ptb.de
Physikalisch-Technische Bundesanstalt & TU Berlin & Einstein Center for Digital Future
Berlin, Germany

Daniel Schwabe
daniel.schwabe@ptb.de
Physikalisch-Technische Bundesanstalt
Berlin, Germany

Matthias Boehm
matthias.boehm@tu-berlin.de
TU Berlin & BIFOLD
Berlin, Germany

## Abstract

Machine Learning (ML) applications are increasingly applied in the healthcare domain, raising the need for so-called trustworthy AI. One fundamental pillar of trustworthy AI is systematic data quality assessment. However, existing ML data quality (DQ) evaluation tools are typically limited to tabular data, lack extensibility and assess only a fraction of data quality aspects. Moreover, many existing tools are unaware of the specific requirements of the given ML task. This prevents them from offering a comprehensive DQ evaluation. To address these issues, we present `MetricLib`: an extensible toolkit for holistic data quality evaluation of medical ML datasets, based on the theoretical METRIC-framework for trustworthy AI in medicine. The toolkit is able to process a range of data modalities in a memory-efficient manner. While a core set of DQ metrics is implemented, `MetricLib` is easily extensible with custom metrics and therefore allows investigation of use-case-specific requirements. Additionally, by aggregated DQ scores, the tool enables the efficient identification of data quality gaps. For displaying all results through a graphical user interface, `MetricLib` is complemented by `MetricLibUI`. The UI enables targeted, fit-for-purpose data quality analysis based on quantitative and qualitative information.

## Keywords

Data quality, data quality evaluation, machine learning, artificial intelligence, health data

## 1 Introduction

The development of machine learning models is accelerating at an unprecedented speed. This trend poses major challenges to regulation, such as the conformity assessment of medical devices. Particularly the black-box nature of deep learning models requires thorough performance investigation beyond predictive performance. According to the European AI Act [19], data quality is a cornerstone of trustworthy AI. However, the open question is how this high-level requirement translates into technical requirements and how to operationalize it for large-scale datasets.

**Data Quality Dimensions**: As a first step towards translating abstract data quality requirements into technical ones, the METRIC-Framework for trustworthy AI in medicine [16] was introduced. It clusters DQ aspects into five categories: *measurement process*, *timeliness*, *representativeness*, *informativeness* and *consistency*. Each *cluster* is further divided into several DQ *dimensions* and *subdimensions*. This theoretical framework is equipped with metrics for data quality evaluation [4] and shall now be operationalized in `MetricLib`. However, a fully automatic, holistic data quality assessment remains intractable, as it requires knowledge of the data source, the target system, the humans involved and particularly the task at hand [10].

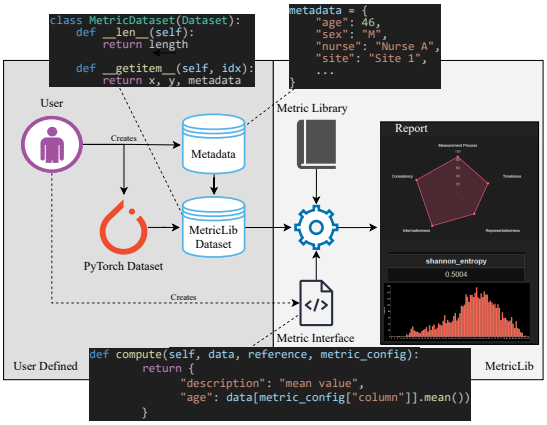


**Figure 1: Components of Data Quality Evaluation.**

**Existing Data Quality Tools**: A variety of DQ tools already exists [6, 12, 14, 17]. However, these tools are mainly designed to repair errors in tabular data. Additionally, they offer only limited capabilities for analyzing distributional characteristics or complex modalities of medical data, such as imaging or medical time series modalities.

**Contribution**: As a step towards semi-automatic data quality evaluation of medical ML data, we demonstrate `MetricLib`, an extensible toolkit for modular data quality evaluation. The contribution of `MetricLib` is twofold: first, `MetricLib` provides a core set of data quality metrics aligned with the METRIC-framework [16]. Second, the existing data quality metrics can easily be extended with modular, easy-to-use and use-case-specific data quality metrics (by inheriting from an abstract `Metric` class). Additionally, `MetricLib` provides data quality scores for the five *clusters* of the

*EDBT '26, Tampere (Finland)*



10.48786/edbt.2026.66

**Table 1: DQ Clusters [16] and Excerpt of Associated Metrics.**

| Cluster | Description | Metric |
|---|---|---|
| **Measurement Process** | Technical or human influences that affect the data acquisition process. | Sample Entropy [24] or signal-to-noise ratio (SNR) [13] for noise level estimation, Limit of Quantification of a measurement [2], Completeness [1] of recording metadata. |
| **Timeliness** | Changes in time and whether they are appropriately reflected. | Currentness [8] of datapoint recordings. |
| **Representativeness** | Appropriate and comprehensive representation of the targeted population in the dataset. | Wasserstein Distance [20] to a reference distribution, Hill numbers [22] of types, Generalized Imbalance Ratio of labels [9], Demographic Parity of protected groups [5]. |
| **Informativeness** | Does the data convey the information it describes. | Prevalence of duplicates [15], Pearson's Correlation between features and labels [21]. |
| **Consistency** | The consistency of presentation and composition of the dataset. | Syntactic Accuracy of text features [3], Maximum Mean Discrepancy between subgroups [7]. |

METRIC-framework. The scores enable fast navigation through the DQ *dimensions* and, thus, allow rapid identification of weak spots. Despite the toolkit being originally designed for medical data, it can be adopted for other domains as well, due to its extensibility.

## 2 System Overview

In this section, we describe `MetricLib` and `MetricLibUI`, with `MetricLib` being a Python package that enables users to comprehensively evaluate data quality. The lifecycle of a `MetricLib` report is shown in Figure 1 and comprises the following steps:

(1) create a `PyTorch`-compatible dataset with metadata,
(2) choose metrics from the metric core set,
(3) create custom metrics according to the `Metric` class,
(4) evaluate the data quality report.

`MetricLibUI` is a user interface on top of `MetricLib`. The graphical interface enables data quality analysis beyond quantitative metrics by allowing interactions with the dataset.

### 2.1 `MetricLib`[1]

Here, we describe the data, metrics and reports of `MetricLib`.

**`MetricLib` Dataset**: Our data loading interface is inspired by PyTorch [11] datasets. A `MetricLib` dataset implements two methods: a dataset length method called `__len__` and a `__getitem__` method, returning an individual datapoint at a given index. Every *datapoint* then consists of three components:

- a tensor `x`, used as input to an ML model. The tensor can, for instance, represent an image or a time series,
- a label tensor `y`, representing the labels of `x`,
- a metadata dictionary associated with this datapoint.

This data format has two major advantages: First, the data can be consumed in a streaming fashion without loading the full data into memory. Second, the same dataset can be reused for training a `PyTorch` model after the data quality evaluation succeeded. After providing a dataset in the instructed format, the setup for evaluating the core set of data quality metrics is complete.

**Core Data Quality Metrics**: `MetricLib` provides a core set of built-in data quality metrics that can be used immediately in the report. An excerpt of the implemented metrics collected by Becker et al. [4] can be found in Table 1. The full list is available in the repository's README. With the core set of data quality metrics, it is possible to get a first impression of the suitability of a dataset according to the METRIC framework.

**Custom Metric Creation**: While `MetricLib` offers a core set of metrics, accounting for the specific ML task requires custom supplementary measures. For leveraging the library's memory efficiency and evaluation speed, it is necessary to implement metrics as one of the two principal classes:

(1) `TabularMetric`: A metric operating on tabular data. It is required to implement the `compute` method, which accepts the dataset as a dataframe, a configuration dictionary and a reference dataset as arguments. The reference dataset is useful whenever a comparison to external data is necessary. Thereby, a `TabularMetric` follows the implementation of the data quality tool Metis [23]. An example of a `compute` method can be found in Figure 1.
(2) `StreamMetric`: A metric where datapoints are accessed incrementally, which requires an `aggregate` method that accepts one datapoint, a reference value and a configuration dictionary as input. The method summarizes a single datapoint into a scalar number. The return value is stored after each call. A `StreamMetric` is also required to have a `compute` method, where the input data contains the aggregation results for each datapoint, which had run through `aggregate`. A `StreamMetric` is for example helpful, when not all model inputs fit into RAM at once.

Metrics that inherit from `TabularMetric` or `StreamMetric` are automatically available for the final report generation. In addition to the metric value, `compute` methods may return a `cluster` name from Table 1. The *cluster* name determines which score the metric contributes to. Additionally, a `threshold` value is required for the score calculation. It represents the ideal value for the metric and is used for the score calculation in Equation 1. It is noteworthy that thresholds should only be set if they are derived from a benchmark dataset or are intuitively justified.

**Report Structure**: A data quality report is generated with the purpose of identifying data quality gaps. The report comprises all available metric values, as well as complementary visualizations. Additionally, the report includes a score for each *cluster* of the METRIC-framework. For $C$ being the set of metrics associated with one *cluster*, the score for that *cluster* is calculated as:

$$\min_{f \in C} \left(1 + \frac{\min(f(X) - t_f, 0)}{t_f}\right), \tag{1}$$

where $X$ is the set of datapoints and $t_f \in \mathbb{R}$ is the given threshold for metric $f : X \to \mathbb{R}$. A low score indicates that at least one metric value is low relative to the threshold. However, caution is advisable, as this definition is not applicable to metrics that decrease with increasing dataset quality. One example of such a metric is the Generalized Imbalance Ratio of labels [9], where a low value represents a balance between minority and majority classes. Nevertheless, it is straightforward to turn a decreasing metric into an increasing one by inverting the sign of this particular metric.

Additionally, `MetricLib` enables the comparison of different datasets, as multiple datasets can be added to the report.

[1] https://github.com/PTBresearch/MetricLib

**Table 2: List of Columns Processed by `MetricLibUI`.**

| Category | Column | Description |
|---|---|---|
| **Patient** | *Age*, *Sex*, *Ethnicity*, *Weight*, *Height* | Patient demographic information contributing to dataset variety. |
| **Recording Metadata** | *Date*, *Nurse*, *Site*, *Device* | Recording information indicating data-source heterogeneity and recency. |
| **ML Model** | *Model input*, *Label* | Paths to training samples and associated label indicators. |

## 2.2 `MetricLibUI`[2]

`MetricLibUI` is the user interface on top of `MetricLib`. It enables interaction with the underlying dataset and data quality metrics.

**Input Format**: `MetricLibUI` accepts images and time series with metadata as input. The metadata and label information for an image or time series are stored in a CSV file. Each row of the CSV file contains a *path* to the associated image/time series file, the label(s) and metadata. If the *path* is absent, only metrics for labels and metadata are applied. Due to the metadata format, metrics are directly transferable from metadata to general tabular datasets. From the *paths*, *label* column(s) and *metadata*, `MetricLibUI` builds a `MetricLib` dataset. `MetricLibUI` is currently limited to multi-label and multi-class classification tasks. The CSV file and the image/time series data needs to be in the data folder at the root level of the project.

**Metric Configuration**: Additional input is needed to evaluate the metric core set. Most importantly, `MetricLibUI` is required to be aware of the column names representing the *path*, the *label* and recording/patient *metadata* to create a column mapping. A list of supported columns is provided in Table 2. The supported columns are tailored to medical datasets. Unsupported columns can be mapped to an additional *other* category. `MetricLibUI` renames the original columns to columns from Table 2 and columns mapped to *other* will be stored under their original name. However, all columns mapped to *other* do not automatically contribute to the metric values, but user-created custom metrics have access to the *other* column. From that point onward, the newly created dataframe is referred to as the *processed dataframe*. The mapping concludes the configuration and the core set of metrics can be evaluated.

**Data Quality Report**: The report comprises the elements:

- *processed dataframe*,
- aggregated data quality scores,
- metric evaluation results and
- data distribution visualizations.

The *processed dataframe* has filtering options. Running a filter reevaluates of the metrics and updates visualizations.

# 3 Demonstration Scenario

For this demonstration, we assume that a data scientist has the task of evaluating whether a given subsample of PTB-XL, a large open-source ECG database [18], is suitable for training a diagnosis assistance system. The dataset contains 12-lead ECGs with a sampling frequency of 500 Hz, the gold standard for ECG diagnosis [25]. It consists of a `metadata.csv` with a path to an ECG recording, patients' demographic information, recording source information, recording date and diagnostic information. The `metadata.csv` was preprocessed such that the diagnoses fall into the categories: myocardial infarction, hypertrophy, conduction disturbance, ST/T change, normal. To evaluate the quality of the dataset, 6 steps are executed (illustrated in Figure 2):

[2]https://github.com/PTBresearch/MetricLibUI

**Setup**: Full setup instructions are in the repository's README. Once initiated, `MetricLibUI` mounts the data folder and allows the data scientist to select the CSV file representing the dataset.

**Step 1. Field Mapping Creation**: It is required to map CSV columns to *processable* columns in Table 2. If multiple columns are mapped to *label column*, the problem is treated as a multi-label task. Next, the report with its metric values is loaded.

**Step 2. Inspection of General Information**: This section of the report provides an overview of the dataset, including the name of the CSV file, the number of records and features, the total number of missing values and the signal file type.

**Step 3. Qualitative Data Analysis**: The processed *metadata* is displayed together with the corresponding *label* information and enables verification of qualitative dimensions of the METRIC-framework. Selecting a row of the *processed dataframe* visualizes the underlying file from the file *path* in its respective format, in this case a 12-lead ECG recording.

**Step 4. Data Quality Scores**: The data scientist starts the quantitative evaluation by inspecting the data quality scores from Equation 1 for each *cluster* of the METRIC-framework. For the sample case at hand, participants observe that this subset of PTB-XL does not perform well in terms of *representativeness*. Given the high scores for other clusters, the extensive METRIC-framework and metric values (Table 1) confirm the dataset's high quality in terms of *Measurement Process*, *Timeliness*, *Informativeness* and *Consistency*.

**Step 5. Metric Inspection**: Once the weak *clusters* have been identified, it is essential to determine the cause of the low score. The data scientist can choose a data quality *cluster*, *dimension* or *subdimension* from the navigation. Once a *subdimension* is selected, each metric value contributing to the *subdimension* is listed. The data scientist can now examine each metric individually (5a). Furthermore, charts are provided (5b) to aid in the interpretation of these values. In this case, the data scientist explores the coverage dimension and identifies a low demographic parity metric, which is due to the low presence of one sex in the NORM cohort.

**Step 6. (Optional) Adding Metrics**: The data scientist might discover that the specific task at hand requires a metric beyond the core set. For example, analyzing ECG data requires a metric that evaluates the distribution of heartbeat frequencies.. Thus, a custom metric is created, following the `StreamMetric` interface. If the metric's return values contain the *dimension* and *subdimension* it contributes to, the metrics result is automatically part of the report.

# 4 Conclusions

`MetricLib` offers a great foundation for evaluating the suitability of a dataset for a given ML task with its core set of DQ metrics. These metrics are calculated in an efficient manner, even for complex data modalities. Due to its extensibility, task-specific requirements can be incorporated into the workflow and, thereby, facilitate holistic data quality evaluation. `MetricLibUI` offers interactions with data quality metrics, supports a better understanding of metric values and allows a more targeted evaluation. Nevertheless, more, potentially task-specific data quality metrics are needed. Due to its design, `MetricLib` provides a strong basis for this future expansion.

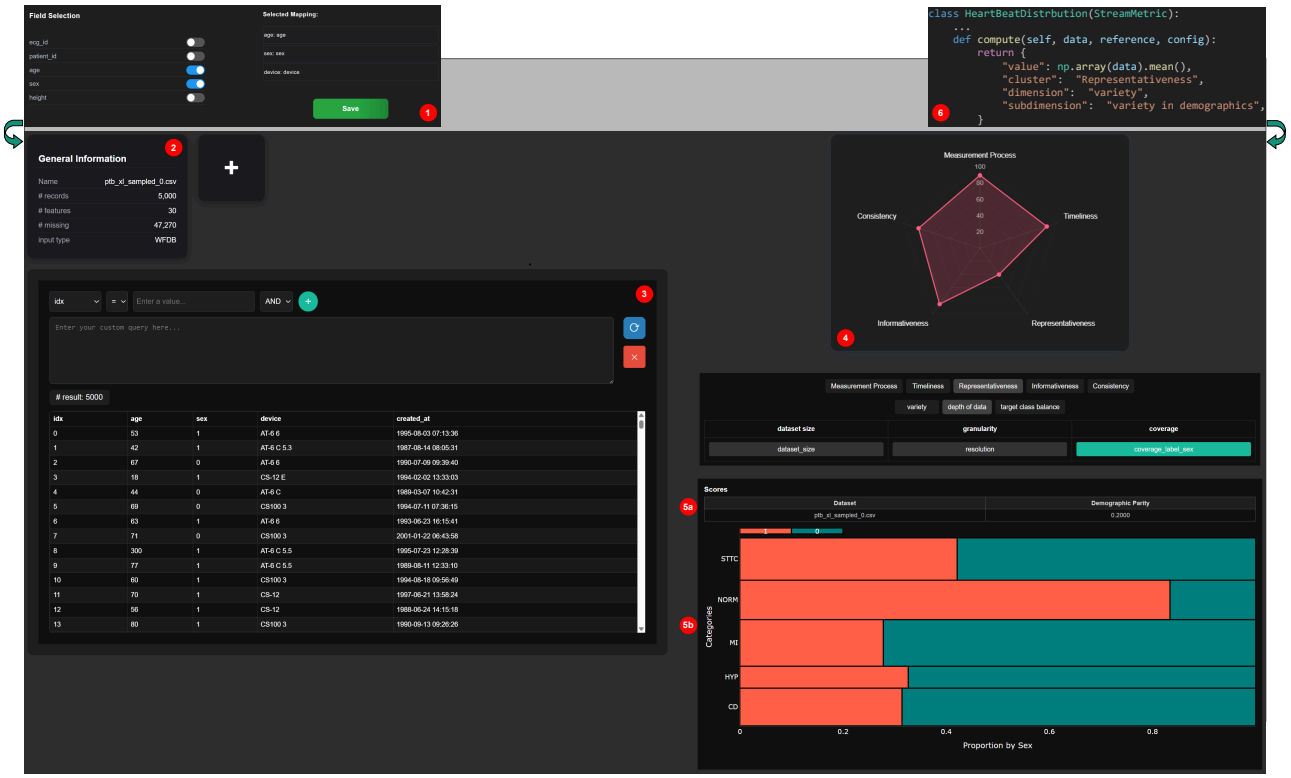


**Figure 2: `MetricLibUI` Data Quality Evaluation Demonstration.**

## Acknowledgments

This work has received funding from the European Union's Digital Europe Programme under grant agreement No. 101100700 (TEF-Health) and the German Federal Ministry of Education and Research under research grant BIFOLD25B.

## References

[1] Roger Blake and Paul Mangiameli. 2011. The Effects and Interactions of Data Quality and Problem Complexity on Classification. *Journal of Data and Information Quality* 2, 2 (2011). https://doi.org/10.1145/1891879.1891881

[2] Armbruster et al. 1994. Limit of detection (LQD)/limit of quantitation (LOQ): comparison of the empirical and the statistical methods exemplified with GC-MS assays of abused drugs. *Clinical chemistry* 40 (1994). https://pubmed.ncbi.nlm.nih.gov/8013092

[3] Batini et al. 2016. Data and information quality. *Cham, Switzerland: Springer International Publishing* 63 (2016).

[4] Becker et al. 2026. Metric Hub: A metric library and practical selection workflow for use-case-driven data quality assessment in medical AI. https://doi.org/10.48550/ARXIV.2601.22702

[5] Dwork et al. 2012. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. 214–226. https://doi.org/10.1145/2090236.2090255

[6] Dallachiesa et al. 2013. NADEEF: a commodity data cleaning system. *SIGMOD* (2013). https://doi.org/10.1145/2463676.2465327

[7] Gretton et al. 2012. A Kernel Two-Sample Test. *Journal of Machine Learning Research* 13 (2012), 723–773. https://www.jmlr.org/papers/v13/gretton12a.html

[8] Heinrich et al. 2007. How to measure data quality? A metric-based approach. (2007). https://api.semanticscholar.org/CorpusID:15446057

[9] Khan et al. 2024. A Review of Ensemble Learning and Data Augmentation Models for Class Imbalanced Problems: Combination, Implementation and Evaluation. *Expert Systems with Applications* 244 (2024). https://doi.org/10.1016/j.eswa.2023.122778

[10] Mohammed et al. 2025. The Five Facets of Data Quality Assessment. *SIGMOD* (2025). https://doi.org/10.1145/3749116.3749120

[11] Paszke et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS* 32 (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[12] Rekatsina et al. 2017. HoloClean: holistic data repairs with probabilistic inference. *VLDB* (2017). https://doi.org/10.14778/3137628.3137631

[13] Rawash et al. 2024. Advanced Low-Pass Filters for Signal Processing: A Comparative Study on Gaussian, Mittag-Leffler, and Savitzky-Golay Filters. *Mathematical Modelling of Engineering Problems* 11 (2024). https://doi.org/10.18280/mmep.110713

[14] Schelter et al. 2019. Unit Testing Data with Deequ. *SIGMOD* (2019). https://doi.org/10.1145/3299869.3320210

[15] Steinkamp et al. 2022. Prevalence and sources of duplicate information in the electronic medical record. *JAMA Network Open* 5 (2022). https://doi.org/10.1001/jamanetworkopen.2022.33348

[16] Schwabe et al. 2024. The METRIC-framework for assessing data quality for trustworthy AI in medicine: a systematic review. *NPJ digital medicine* (2024). https://doi.org/10.1038/s41746-024-01196-4

[17] Siddiqi et al. 2024. SAGA: A Scalable Framework for Optimizing Data Cleaning Pipelines for Machine Learning Applications. *SIGMOD* (2024). https://doi.org/10.1145/3617338

[18] Wagner et al. 2020. PTB-XL, a Large Publicly Available Electrocardiography Dataset. *Scientific Data* 7 (2020). https://doi.org/10.1038/s41597-020-0495-6

[19] European Union. 2024. Artificial Intelligence Act (Regulation EU 2024/1689). https://eur-lex.europa.eu/eli/reg/2024/1689/oj

[20] Peyré Gabriel and Cuturi Marco. 2019. Computational Optimal Transport with Applications to Data Sciences. *Foundations and Trends® in Machine Learning* 11 (2019). https://doi.org/10.1561/2200000073

[21] Mark A Hall. 1999. *Correlation-based feature selection for machine learning*. Ph. D. Dissertation. The University of Waikato.

[22] Mark O. Hill. 1973. Diversity and evenness: A unifying notation and its consequences. *Ecology* 54 (1973). https://doi.org/10.2307/1934352

[23] HPI-Information-Systems. 2025. Metis: A framework to automatically assess the quality of tabular data. https://github.com/HPI-Information-Systems/Metis GitHub repository. Accessed 4 December 2025.

[24] Joshua S. Richman and J. Randall Moorman. 2000. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology* 278 (2000). https://doi.org/10.1152/ajpheart.2000.278.6.h2039

[25] Hans-Peter Schuster and Hans-Jürgen Trappe. 2017. *EKG-Kurs für Isabel* (7th ed.). Georg Thieme Verlag.