DQSOps: Data Quality Scoring Operations Framework for Data-Driven Applications

Firas Bayram
firas.bayram@kau.se
Dept of Mathematics and Computer Science, Karlstad University
Karlstad, Sweden

Bestoun S. Ahmed
bestoun@kau.se
Dept of Mathematics and Computer Science, Karlstad University
Karlstad, Sweden
Department of Computer Science, Czech Technical University
Prague, Czech Republic

Erik Hallin
Anton Engman
erik.hallin@uddeholm.com
anton.engman@uddeholm.com
Uddeholms AB
Hagfors, Värmlands län, Sweden

ABSTRACT

Data quality assessment has become a prominent component in the successful execution of complex data-driven artificial intelligence (AI) software systems. In practice, real-world applications generate huge volumes of data at speeds. These data streams require analysis and preprocessing before being permanently stored or used in a learning task. Therefore, significant attention has been paid to the systematic management and construction of high-quality datasets. Nevertheless, managing voluminous and high-velocity data streams is usually performed manually (i.e. offline), making it an impractical strategy in production environments. To address this challenge, DataOps has emerged to achieve life-cycle automation of data processes using DevOps principles. However, determining the data quality based on a fitness scale constitutes a complex task within the framework of DataOps. This paper presents a novel Data Quality Scoring Operations (DQSOps) framework that yields a quality score for production data in DataOps workflows. The framework incorporates two scoring approaches, an ML prediction-based approach that predicts the data quality score and a standard-based approach that periodically produces the ground-truth scores based on assessing several data quality dimensions. We deploy the DQSOps framework in a real-world industrial use case. The results show that DQSOps achieves significant computational speedup rates compared to the conventional approach of data quality scoring while maintaining high prediction performance.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Software and its engineering** → **Software development techniques**; **Software verification and validation**.

KEYWORDS

Automated data scoring, DataOps, data assessment, data quality dimensions, mutation testing

EASE '23, June 14–16, 2023, Oulu, Finland

ACM ISBN 979-8-4007-0044-6/23/06.
https://doi.org/10.1145/3593434.3593445

ACM Reference Format:
Firas Bayram, Bestoun S. Ahmed, Erik Hallin, and Anton Engman. 2023. DQSOps: Data Quality Scoring Operations Framework for Data-Driven Applications. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE '23), June 14–16, 2023, Oulu, Finland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3593434.3593445

1 INTRODUCTION

In the era of big data explosion, data has become the most critical asset of artificial intelligence (AI) software projects [41]. The primary motivation is that the data accompany machine learning (ML) software throughout their life cycle. Notably, huge volumes of data are being collected for industries and businesses at an ever-escalating rate. The volume and velocity pose a formidable challenge for ML software systems in production, which are required to make (near) real-time decisions [53]. In real-life scenarios, the challenge is twofold: assessing the quality of the data flowing in the system and processing it within defined time frames.

Data quality is not a new topic; its roots are traced back to the literature on database management systems [56]. However, the advancement of data-driven systems has recently shifted the topic towards AI research [5]. It is widely recognized that the performance of ML projects is mainly dependent on the quality of the underlying data [26]. Data quality is a multifaceted concept that defines several quality dimensions based on the relative nature of the problem [42]. Typically, data quality dimensions are identified according to the context and domain, which implies that data quality dimensions may change with application. Each dimension of the data quality measures the condition of the data concerning a specific aspect which is usually summarized in a quality index or metric that is used as an indicator of data validity [54].

Data quality scoring holds significant relevance in data quality assessment and is defined as the methodology for obtaining a score metric for each dimension of the data quality of a given set of observations in a dataset [52]. In the data quality scoring literature, researchers investigate the procedures to measure and quantify the different dimensions of data quality within the target application. The scores are interpreted as the fitness scale corresponding to certain data quality dimensions for each data record or chunk. Data quality scores are used in data storage management or leveraged in predictive ML software by introducing acceptability thresholds [38]. Acceptability thresholds filter out data instances that do not meet

quality standards. As a result, the data quality scoring methodology enables the characterization of data records according to their quality through quantitative analysis.

The conventional procedure for data quality scoring is executed through a code that manually inspects and validates the condition of the data with respect to the defined data quality dimensions [51]. However, this traditional procedure is resource intensive, especially if the number of data quality dimensions is high, and generally requires human intervention [43]. Furthermore, in deployment environments, this manual procedure is not practical for ML software that should perform tasks automatically [33]. Recently, the DevOps principles, continuous integration and continuous delivery (CI/CD), have been adopted and introduced to build productionized ML software, known as Machine Learning Operations (MLOps) [28]. MLOps systems allow the deployment of ML software in a formalized way throughout the life-cycle of AI applications [7]. An indispensable element of MLOps is DataOps, which is the approach that manages and processes the production data systematically and continuously. DataOps pipelines are designed to prepare accurate and reliable data that the ML model will use. However, scoring the streaming data windows is challenging and yet to be explored within the context of DataOps for production systems that are characterized by high sampling rates.

This paper contributes to the current advancement in the DataOps field by presenting a novel Data Quality Scoring Operations (DQSOps) framework that can be streamlined in DataOps workflows. DQSOps can be viewed as a general data scoring methodology to automate scoring the quality of streaming data. The generalizability of DQSOps is enabled by incorporating configurable components that can be easily adjusted and extended based on the problem. In practical implementations, DQSOps accelerates scoring the quality of acquired production data irrespective of the number of defined quality dimensions. The speedup is achieved by employing an innovative ML predictor that determines the quality score of the processed data window. Using an ML predictor rather than the standard data scoring methodology substantially reduces the overall processing time. However, a test oracle is implemented to continuously evaluate the ML accuracy throughout the system evolution to sustain high performance for the ML predictor. The test oracle uses ground-truth data quality scores that are periodically produced by a standard-based approach. Additionally, we introduce a new data mutation component, inspired by the mutation testing principles, to simulate data quality issues in the data window, and thus facilitate the initialization phase of our framework.

The rest of the paper is structured as follows: Section 2 reviews the related work in the field and provides an overview of the data quality scoring task. Section 3 presents the methodology used to develop our data quality scoring framework. The experimental results of our use cases are reported in Section 4. The threats to generalize our proposed framework are discussed in Section 5. Finally, Section 6 concludes the paper with a summary of the remarks and future perspective.

2 BACKGROUND AND PREVIOUS WORK

This section provides a general overview of data quality dimensions and related definitions. Then we review studies devoted to assessing data quality in the literature.

2.1 Data Quality Dimensions

Data quality dimensions are defined to assess the quality of the data. Each dimension captures a specific characteristic of the evaluated data within a particular task. Knight [31] has presented a combined conceptual framework for the quality of information systems. ML software concerns the relation between the system's data and task attributes in information systems. Therefore, contextual quality is relevant to build an ML solution, as it is identified in the interactive space between the data of a system and the attributes of the task [19].

The contextual quality comprises five quality dimensions: *value-added*, *relevancy*, *timeliness*, *appropriate data* and *completeness* [31]. However, since these quality dimensions are context-based, they must be adapted to the task and the system's characteristics. Wand and Wang [55] described the dimensions of data quality as subjective and should be defined within the application of the system that generates the data. In a more recent study, [52], these dimensions have been revised for continuous quality management. The revised data quality dimensions are: *accuracy*, *completeness*, *consistency*, *Timeliness*. However, for evolving ML software systems, distributional changes are likely to occur during system evolution [39]. Therefore, we will extend these quality measures and include *skewness* [15] to monitor the distribution of incoming data. Descriptions of the data quality dimensions are provided with the following definitions:

1. **Accuracy:** Measures whether the observed data value represents the actual value.
2. **Completeness:** Checks whether the observed data include missing values.
3. **Consistency:** Verifies whether the observed values meet the integrity constraints of the domain.
4. **Timeliness:** Describes whether the data are up-to-date for the corresponding task.
5. **Skewness:** Computes the distribution deviation of the observed data from a reference distribution.

2.2 Related Work

Data quality assessment has been the subject of many early works in the literature [45]. Chug *et al.* [15] have proposed a method that quantifies the quality of a given dataset. The method uses nine data quality dimensions to estimate the quality of the dataset. As a result, the method provides a score, report, and label for the data quality of the dataset. Similarly, for big data systems, Taleb *et al.* [52] proposed the Big Data Quality Management Framework (BDQMF) to address data quality issues in big data systems based on several data quality dimensions. The framework included several components to manage, validate, and monitor the data quality. Furthermore, data quality issues were investigated both at the cell instance and the schema levels of the dataset. BDQMF framework also quantifies scores of data quality aspects. However, the authors did not experimentally evaluate the framework. For business processes,

Data Quality Validation Methodology (DQVM) was proposed to evaluate the effects of data quality on the process outcome [10]. The methodology allows domain experts to assign the corresponding weights for each data quality dimension based on their relative importance to the business process. The deviation of quality scores is then observed between fault-free and fault-injected datasets, and the impact on the process is eventually evaluated.

Other approaches have investigated quantifying an individual data quality dimension. Heinrich *et al.* [25] proposed a quality metric based on probability theory to assess *semantic consistency*. The metric indicates the degree to which assessed data is contradiction-free. Similarly, *minimality*, or *uniqueness*, was investigated to measure redundancies in data at the data- and schema-level [20]. The method uses hierarchical clustering and similarity calculation techniques to calculate the metric. For Internet of Things (IoT) systems, Byabazaire *et al.* [9] presented a real-time data quality assessment framework. To measure data quality, the framework uses *trust* as a metric for quantification. The correlation between trust score and root mean square error (RMSE) and mean absolute error (MAE) is calculated using Pearson's correlation coefficient to validate if trust is a good indicator of data quality. With the limited approaches to address data quality scoring in the context of production systems, we design a framework that can be utilized efficiently in real-world problem scenarios. The proposed framework can handle streaming data with high sampling rates, as opposed to the existing methods that are designed for offline static datasets. Additionally, our proposed framework can effectively accommodate a high number of quality dimensions.

3 DATA QUALITY SCORING OPERATIONS (DQSOPS) FRAMEWORK

This section introduces the methodology to deliver our proposed DQSOps framework. DQSOps can efficiently evaluate the quality of production data streams based on quantitative analysis. The output of DQSOps is a score metric that serves as an indicator of the overall validity of the collected data window, and is calculated based on assessing several data quality dimensions. The scores can be used to rank the system data according to their quality. Thus they can be selected for the various business analysis tasks that may require different levels of data quality. In practice, DQSOps significantly reduces the time to score the data window by employing an ML predictor that replaces the conventional data scoring method. The run time is irrespective of the number of quality dimensions, making it convenient for real-world applications characterized by high sampling rates. In addition, the performance of the ML predictor is periodically evaluated by executing test oracles.

The overall pipeline of the presented framework is shown in Figure 1. Upon receiving the data streams from data-generating sources, the stream is segmented into windows of data samples according to a pre-defined data window size. Inspired by the mutation testing principles [27], we introduce a *data mutant simulator* component to obtain full control over the experimental conditions. The component is applied to simulate *nonequivalent data mutants* within the data window according to pre-configured parameters that are loaded from a configuration file that stores the pre-specified mutation percentage of each data quality dimension. Nonequivalent mutants are used to induce meaningful changes in the problem [50]. Typical examples of data mutants are missing or inconsistent data values, or anomalies. Consequently, we utilize the data mutation component to simulate erroneous data that would affect the data quality according to the specified dimensions. Furthermore, the component is especially useful during the initialization phase, as will be discussed further in Section 3.3.

After preparing the data window, the method activator component is invoked to activate the appropriate workflow path for data scoring based on particular decisions. The criteria for selecting the approach will be further detailed in Section 3.4. The candidate quality scoring approaches are: the standard scoring method, a regression ML model, or a retrain signal. The standard scoring method calculates the ground-truth data quality scores of the defined dimensions and stores it in a repository. Whereas the ML model predicts the data quality score of each data window. If the retrain signal is activated, the ML model will be retrained using the ground-truth scores stored in the repository. During the retraining process, both the current prediction- and standard-based approaches will be used to score the data quality until the new model is ready to replace the current one. Furthermore, the performance of the regression model is continuously monitored using a test oracle that evaluates the predicted data quality scores. The next subsections will delve deeper into the mechanisms of the main components of the DQSOps framework.

3.1 Scoring the Data Quality Dimensions

The standard-based approach is a core element of DQSOps and used to quantify the ground-truth scores of the data quality dimensions that are used to train the ML model. Each data quality dimension will be reflected in one quality score. In particular, this score quantifies the level of fitness of the collected data window with respect to the specific data quality dimension. The relevant data quality dimensions are: accuracy, completeness, consistency, timeliness, and skewness, as discussed in Section 2.1. As shown in Figure 1, meta-information is loaded that helps calculate the scores of the data quality dimensions from a configuration file. For instance, meta-information may include the values that characterize the integrity constraints of the specific problem, such as the maximum and minimum value limits. The configuration file could also include the path of auxiliary files that supports the calculation of some scores such as the anomaly detector or data distribution. After calculating the data quality scores, all score values are standardized between 0 and 1 using min-max normalization to obtain uniform scales across the data quality dimensions, as discussed in the next section. The data quality scores are retrieved as follows:

3.1.1 Accuracy Score. Accuracy score is calculated by finding the proportion of the anomalous datum in the processed data window [52]: $Accuracy = \frac{NAV}{N}$, where NAV is the total number of anomalous values detected in the data window and N is the size of the data window. Anomalies may appear in the system due to several factors, such as malicious activities, hardware failures, inaccuracies in data collection, or adversarial attacks [12]. From a data quality perspective, anomalous data records represent abnormal values of unhealthy data instances and therefore are considered an indicator

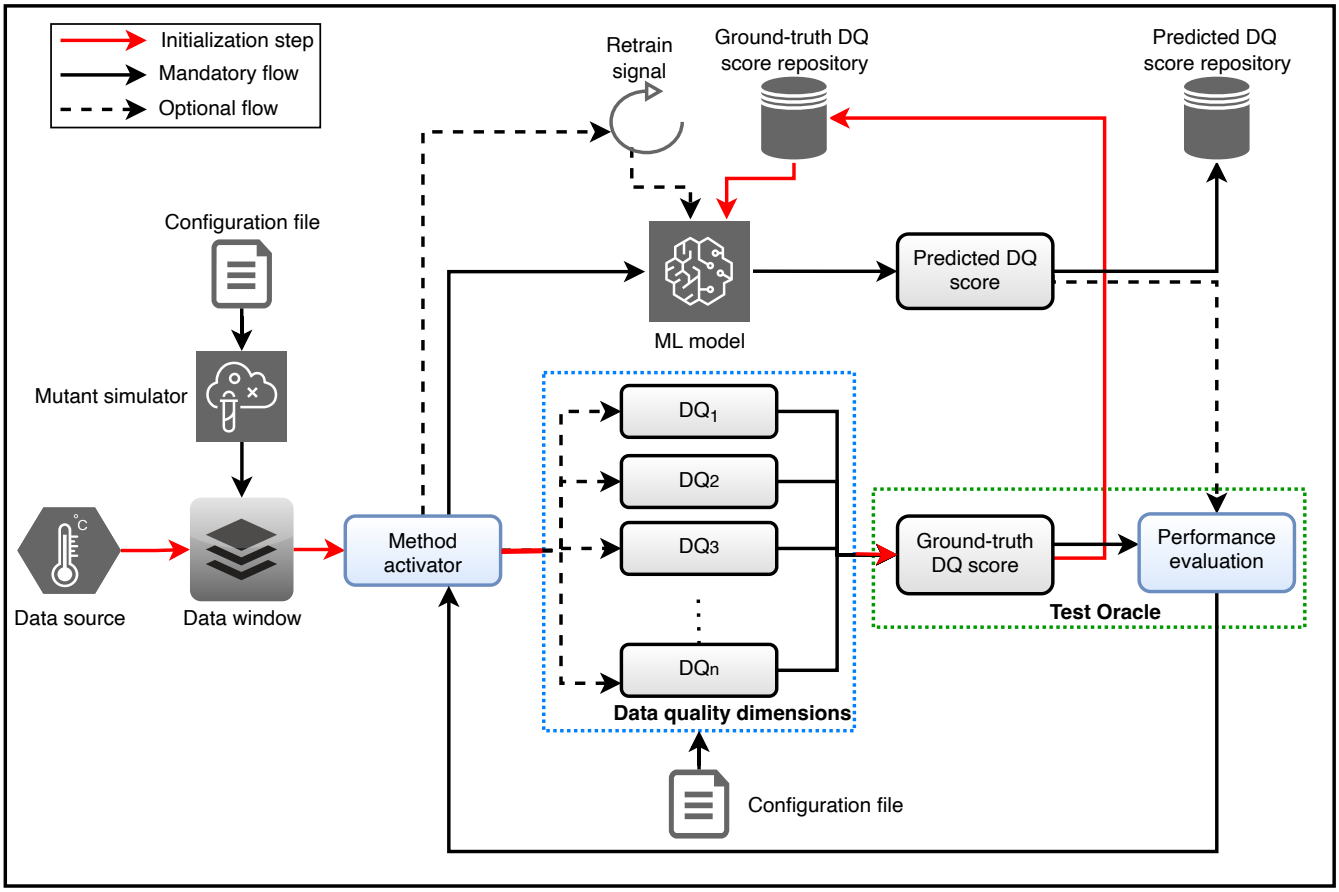


Figure 1: DQSOps framework, the red path represents the initialization phase.

of low-quality data [49]. The anomaly detector can be obtained in the initialization step; see Section 3.3.

3.1.2 Completeness Score. The fraction of missing values that are observed in the data window and can be found as follows [24]: $Completeness = \frac{NNV}{N}$, where NNV is the number of missing values such as NA (Not Available) or NULL observations and N is the size of the data window. Missing values are a popular problem of data quality that could be an indicator of disconnection or damage to the data source [46].

3.1.3 Consistency Score. The integrity constraints vary depending on the domain and the application conditions. Therefore, they are defined in problem-specific settings. For example, in some domains, data observations cannot be negative or their values should fall within a particular range. After defining the integrity rules for the data values, the consistency score can be calculated as the fraction of data values that do not meet the integrity constraints [47]: $Consistency = \frac{NCV}{N}$, where NCV is the number of consistent values and N is the size of the data window. Data consistency is a fundamental issue for data quality, as it checks data conflicts that can detect errors in the data recording process [22].

3.1.4 Timeliness Score. Timeliness, or data currency, is a semantic measure that characterizes whether the data fits the application domain [29]. To quantify the timeliness of the data, we used a goodness-of-fit test. Goodness-of-fit tests measure the likelihood that current data are sampled from a specific cumulative distribution function(cdf) or probability density function (pdf) of the underlying data-generating distribution [2]. Several goodness-of-fit test techniques can be applied according to the nature of the data [17]. The most popular tests are the Kolmogorov-Smirnov, Anderson-Darling, and Cramér-von Mises statistical tests [21]. Each test calculates test statics that is interpreted for the fitness of the data with the compared distribution.

In our experiments, a two-sample Kolmogorov-Smirnov statistical test is used to calculate the goodness-of-fit metric. Kolmogorov-Smirnov test is a non-parametric statistical tool to determine whether two samples are drawn from the same distribution [48]. The main motivation for adopting the Kolmogorov-Smirnov test is that it is a powerful method for small subsets [3], as in our use-case settings. For two empirical cumulative distribution functions $\hat{F}_1$ and $\hat{F}_2$ for two independent random samples $X = X_1, \ldots, X_n$ and $Y = Y_1, \ldots, Y_m$ respectively, the Kolmogorov–Smirnov test statistic

is computed as [32]:

$$KS = \max_{1 \leqslant i \leqslant N} \left| \hat{F}_1(Z_i) - \hat{F}_2(Z_i) \right|, \quad (1)$$

where Z is the combined sample of X and Y, $N = n + m$.

3.1.5 Skewness Score. In real-world applications, especially the Internet of Things (IoT), where data are collected from sensors, distributional drift (or shift) is one of the most frequent data issues in the system [23]. Data flows are validated against distributional deviations that induce skewness in the data distribution [11]. To calculate the distributional skewness score, the divergence magnitude can be calculated to measure the dissimilarity between the distributions of the current data window and the historical data [34]. There are numerous methods to calculate the divergence measure; Jensen-Shannon (JSD) and Kullback-Leibler (KLD) Divergences are the most popular ones [44].

For our framework, we have used the JSD metric to calculate the skewness score. JSD metric is a symmetrization of the KLD metric. The main property of JSD is that it is bounded in the interval $[0, 1]$, while the KLD value may be infinite [37]. According to JSD, the dissimilarity between two probability distributions P and Q is calculated as [36]:

$$\mathrm{JSD}(P\|Q) = \mathrm{JSD}(P\|Q) := H\left(\frac{P+Q}{2}\right) - \frac{H(P) + H(Q}{2}, \quad (2)$$

where the function H denotes Shannon's entropy and is given by:

$$H(p) = -\int p(\mathbf{Y}) \log p(\mathbf{Y}) d\mathbf{Y}. \quad (3)$$

A final remark on the data quality dimensions defined in this section is that other definitions can be introduced according to the task. For example, accuracy, completeness, and consistency scores can be defined as '1' if the data window includes the corresponding data quality issue and '0' otherwise [6], rather than using the fraction of erroneous data instances. This can be adopted in safety-critical domains with a lower tolerance level for faulty data, such as medical applications and autonomous cars [13]. In our industrial use cases, using fractions of erroneous data is more suitable since potentially dropping data instances (i.e., giving them a score of '1') is not preferred.

3.2 Finding the Consolidated Data Quality Score

After calculating the score values of the data quality dimensions, the values are aggregated in a consolidated score that represents the quality of the data window [16]. The most basic approach to aggregate the metrics is by taking the arithmetic mean, or its variants, such as the weighted average, of the data quality dimensions. However, it was argued that the arithmetic mean does not provide sound aggregation [24]. Additionally, aggregation methods are sensitive to the scales of the variables. Therefore, the calculation of the aggregated metric will be dominated by the dimensions with large scales, and the effect of the dimensions with low scales will be negligible.

Before aggregating the quality scores, the matrix DQ_{nm} that contains the quality scores of the n data instance of m dimension is standardized. The standardization step is essential before aggregation so that the different data quality metrics can be equally integrated into the overall quality score [18]. The standardization, also known as whitening, can be achieved by calculating the z-score of each element dq_{ij} of the matrix DQ_{nm}: $z_{ij} = \frac{(dq_{ij} - \mu_j)}{\sigma_j}$, where μ_j is the mean μ_j is the standard deviation of the respective column. The standardization process would give the different data quality dimensions uniform weights when calculating the aggregation metric.

In a recent survey [53], it was shown that the principal component analysis-based methods are the most widely used techniques to detect data errors. Therefore, we follow the literature and use principal component analysis (PCA) to extract the score that represents the overall quality of the data windows. PCA is a widely used dimensionality reduction technique that extracts the most crucial information from the data. PCA maps each data instance of d dimensional space to k dimensional space using an orthogonal transformation such that $k < d$. The first principal component is constructed by finding the direction of maximum variance [1]. The principal components of dataset X are found by solving the eigenvalue problem:

$$\Sigma \mathbf{M} = \lambda \mathbf{M}, \quad (4)$$

where Σ is the covariance matrix $\sum = XX^T$ that measures the correlation between the variables of the original data X, λ is the vector of eigenvalues, and M is the matrix of eigenvectors that contains the principal components ordered by the size of their eigenvalue.

3.3 Initialization Phase

Before deployment, the DQSOps framework requires preparing a warm-start of the ML predictor using training data that includes ground-truth quality scores. The initialization phase of DQSOps is indicated in Figure 1 by the red path. Through this phase, a reliable ML predictor will be delivered to be put into practice in production. To produce the ML predictor, ground-truth labels are found, as discussed in the previous sections, and stored in a repository that will be used to train the ML model. The prediction accuracy of the ML model is monitored until it reaches a predetermined threshold τ evaluated with respect to a performance metric. The ML predictor is deployed in the real-world problem upon reaching this threshold. In this phase, the mutant simulator component plays an essential role in the learning task. The mutants would help the model learn to identify data quality issues by providing the model with more extensive and diverse quality issues that could potentially arise in real-life environments. Furthermore, the component accelerates the learning process, compared to waiting for data quality issues that may not be frequent in reality, which can be time-consuming.

In addition to producing the ML predictor, other meta-information files are prepared in this phase to be used in practice. These files are the anomaly detection model used to calculate the accuracy score and a reference data distribution used to calculate the skewness score. These files should be prepared using high-quality clean data to enhance the reliability of the ground-truth quality scores. The paths of these files are then stored in the configuration file to be loaded once the DQSOps framework is eventually deployed in production.

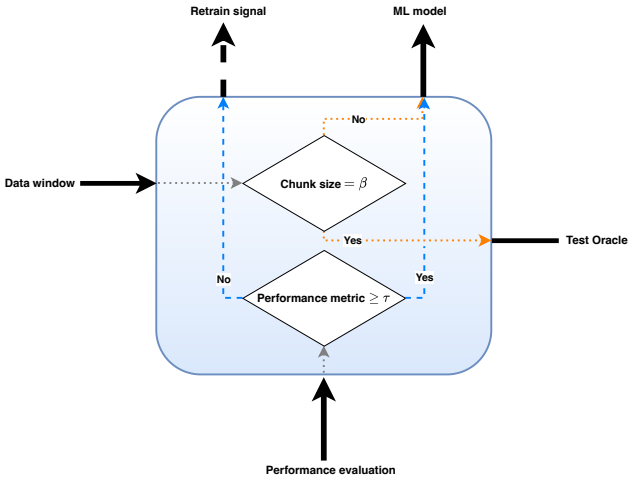


Figure 2: Flowchart diagram of the method activator component

3.4 Method Activator Component

One of the core elements of DQSOps is the method activator which orchestrates the pipeline flow. As illustrated in the decision flowchart in Figure 2, the component uses a set of predetermined criteria to execute the appropriate approach for data quality scoring. Once the data window is collected, a decision is made based on the chunk size. The activator repeatedly executes the ML model to obtain the data quality scores until the chunk size reaches a preconfigured threshold β. Figure 3 illustrates the mechanism for processing the data windows and chunks. The threshold β represents how frequently we want to test the ML model performance. Once the threshold is reached by collecting β data windows, the activator executes an evaluation methodology using the standard-based approach. Subsequently, the standard approach produces the ground-truth scores of n data windows, where $1 \leq n < \beta$. This is necessary to perform continuous monitoring of the ML model performance.

The evaluation is conducted using a specified test oracle [35], as shown in Figure 1. The specified test oracle requires oracle information and oracle procedure [4]. The oracle information characterizes the expected output, which is obtained by the ground-truth scores by the standard-based approach. For the oracle procedure, a relevant performance metric, such as the prediction error, is used to evaluate the predicted quality score of the ML model using the oracle information. Afterward, the result of the test oracle is forwarded to the activator, which compares it with a tolerance level τ. The level represents the minimum accuracy required to achieve and can be set based on quality requirements or inferred by observing the ML performance in the initialization phase; see Section 3.3. If the performance of the ML model falls below the desired level, a retrain signal is sent to update the ML model using the newly obtained data quality scores. This strategy ensures the efficiency and reliability of the ML model over time.

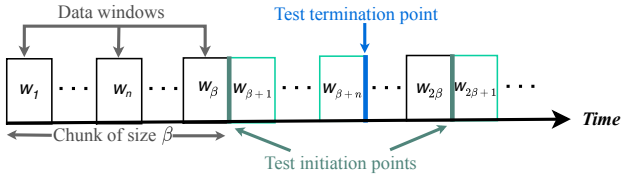


Figure 3: Mechanism of processing the data streams

4 IMPLEMENTATION AND EVALUATION

The framework is applied and evaluated in a real-world industrial use case. The production results are analyzed in terms of predictive performance and time-efficiency criteria. An analysis of the effects of the mutation percentage on the initialization phase is also presented.

4.1 Industrial Use Case Description

The industrial use case in which we implement our framework is the industrial process of Electroslag Remelting (ESR) vacuum pumping at Uddeholm steel company in Sweden[1]. The vacuum pump is used to ensure the production of high-quality steel. This is achieved by extracting the air oxygen from the furnace. In production, each vacuum pump event lasts up to 20 minutes. The observation of interest is the pressure value generated inside the vacuum chamber. To record the pressure values, a sensor is connected to the furnace and registers the value every millisecond. The pressure data windows are continuously transmitted every second through Apache Kafka streaming platform[2] to enable real-time analysis. The pressure value should gradually decrease in a proper pump event until it reaches the desired minimum value within 20 minutes. However, in some cases, an improper pump event occurs, and the minimum pressure value is not met so the furnace will be re-initiated. Therefore, it is essential to distinguish the status of the pump events. The task is to quantify the quality of the pressure data collected from each pump event using the proposed DQSOps framework. For a production environment, DQSOps is developed using Python programming language and deployed in the system through a Docker container[3]. The Docker ecosystem allows the deployment flexibly and efficiently regardless of the underlying system. For ML models, we use two popular decision tree-based algorithms, random forests (RF) [8] and extreme gradient boosting (XGBoost) [14] models. RF and XGBoost are two efficient ensemble algorithms that have been widely used in various applications due to their simplicity, high performance, and interpretability property, making them suitable for industrial applications [30].

4.2 Predictive Performance Evaluation

The performance results of the prediction-based methods of our framework are evaluated in the production environment of our use case. These methods are based on the regression algorithm that predicts the data quality score rather than calculating it using the

[1]https://www.uddeholm.com/
[2]https://kafka.apache.org/
[3]https://www.docker.com/

Table 1: Average performance of the RF prediction-based method

DQD	MAE			R^2		
	mean	std	CV	mean	std	CV
1	0.0148	0.0052	0.3529	0.7133	0.2111	0.2959
2	0.4142	0.0561	0.1353	0.7325	0.2289	0.3125
3	0.5604	0.0928	0.1656	0.7978	0.1353	0.1696
4	0.6411	0.1032	0.1610	0.8435	0.0362	0.0430
5	0.6132	0.0000	0.0000	0.8936	0.0000	0.0000

standard approach. The performance is evaluated using two performance metrics, the mean absolute error (MAE) and the coefficient of determination R^2 given by the formulae:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}, \tag{5}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}, \tag{6}$$

where y_i is the ground-truth score calculated using the oracle-based approach, $\hat{y}_i$ is the predicted value using the regression algorithm, and $\bar{y}$ represents the mean of all scores.

Table 1 shows the performance of the RF prediction-based method, and Table 2 shows the performance of the XGBoost prediction-based method. The arithmetic mean of the error rates of all possible combinations of data quality dimensions of the same size is recorded. We have also calculated the Coefficient of Variation (CV) to measure the relative dispersion of the error rates of each experiment. CV is calculated as the ratio between the standard deviation (std) σ and the mean μ of the population: $CV = \frac{\sigma}{\mu}$. Moreover, to provide insights into the distribution of the DQ scores and to assess the significance of the error rates compared to the statistics of the DQ scores, we computed the summary statistics for the DQ score values presented in Table 3 for the different experimental settings. These statistics offer an overview of the DQ score ranges and their standard deviation, which allows us to contextualize the performance of the ML models.

As for the predictive performance, for both ML algorithms, the MAE value increases with the number of quality dimensions, as the true scoring function becomes more difficult to capture by the algorithms. The MAE metric's average value is low in experiments that are performed using a single data quality dimension. This is because most of the data windows are of high quality, so their quality score is close to 0. However, the R^2 metric increases with the size of the data quality dimensions for both RF and XGBosst as the algorithms fit the data better. From the tables, we can also see that the difference in performance is not significant between RF and XGBoost, with the latter showing slightly lower MAE and higher R^2 values. However, the significant run-time efficiency achieved by the XGBoost model, as will be presented in the subsequent analysis, makes it the most superior ML model to deploy in the DQSOps framework.

Table 2: Average performance of the XGBoost prediction-based method

DQD	MAE			R^2		
	mean	std	CV	mean	std	CV
1	0.0138	0.0036	0.2617	0.7820	0.1690	0.2161
2	0.4242	0.0619	0.1459	0.7231	0.2341	0.3237
3	0.5483	0.1044	0.1904	0.7846	0.1919	0.2445
4	0.5934	0.1075	0.1812	0.8582	0.0641	0.0746
5	0.6320	0.0000	0.0000	0.8916	0.0000	0.0000

Table 3: Statistics of data quality scores

DQD	DQ score range	DQ scores std
1	[0, 0.9345]	0.0896
2	[-15.173, 4.820]	2.27
3	[-13.819, 5.472]	2.355
4	[-12.204, 8.152]	2.93
5	[-9.899, 9.855]	2.846

4.3 Time Speedup Evaluation

To evaluate the run-time efficiency, we compare the time required to execute the standard and prediction-based data quality scoring approaches. Experiments were conducted with different numbers of data quality dimensions to analyze the effect of the size of the data quality dimensions on the execution time. For example, we have five possible combinations in total for a single data quality dimension. Each combination evaluates an individual data quality dimension separately. While for two data quality dimensions, ten pairs of data quality dimensions can be constructed from the five overall quality dimensions defined in our framework, and so on. Table 4 summarizes the average computation time of each method per data window in seconds.

Analogously to the analysis presented in the previous section, we analyzed the computational run-time of different sizes of data quality dimensions. We calculated the summary statistics and Coefficient of Variation (CV) for the computational run-time of each experiment to explore the dispersion of the results. We can observe that the CV is higher for the standard-based scoring method. This is because calculating completeness and consistency scores requires less computational time than calculating the goodness-of-fit or skewness. A simple fraction is required to calculate completeness and consistency scores, while goodness-of-fit and skewness involve estimating probability distributions. Hence, variability is higher when conducting experiments with a single data quality dimension. The CV values decrease as the number of data quality dimensions increases, and the run-time variation of the different experiments decreases. However, the standard deviation values are lower, and the CV values are closer for prediction-based scoring methods than for the standard-based approach. This signifies that the level of dispersion around the mean is lower, and the execution times are relatively closer for prediction-based methods.

For the standard-based method, the average computational time is proportional to the number of quality dimensions. In contrast,

Table 4: Time required to score a data window using the different approaches in seconds

DQD	Standard-based			RF prediction-based				XGBoost prediction-based			
	mean	std	CV	mean	std	CV	Speedup	mean	std	CV	Speedup
1	0.19057	0.32218	1.69066	0.01953	0.00142	0.07260	**9.76x**	0.00050	0.00004	0.08774	**381.90x**
2	0.37671	0.39409	1.04614	0.01268	0.00035	0.02740	**29.71x**	0.00103	0.00007	0.06453	**366.45x**
3	0.57191	0.40000	0.69941	0.01259	0.00042	0.03334	**45.44x**	0.00109	0.00004	0.03334	**523.72x**
4	0.75229	0.32335	0.42983	0.01245	0.00030	0.02405	**60.42x**	0.00118	0.00004	0.03472	**638.07x**
5	0.94101	0.00000	0.00000	0.01265	0.00000	0.00000	**74.38x**	0.00115	0.00000	0.00000	**818.98x**

the average computational time does not depend on the number of quality dimensions for prediction-based approaches. We can see that the summary statistics are similar for each ML model used to predict the data quality score in all experiments. This finding has substantial practical implications on the sensitivity to the number of quality dimensions for production environments. The run-time for the prediction-based scoring process is agnostic to the number of quality dimensions. In contrast, the run-time would increase with the number of quality dimensions for the standard scoring process. Regarding speedup rates, the results in Table 4 show significant levels of run-time efficiency with the prediction-based quality scoring approach. Specifically, the random forest (RF) prediction-based method registers a 10x speedup increase over the standard-based method for a single quality dimension. It reaches approximately 75x when using all the data quality dimensions. Similarly, for the XGBoost prediction-based method, the speedup is approximately 382x when using one quality dimension to roughly reach 819x when using all quality dimensions. However, comparing the different ML models, we see that the XGBoost algorithm requires significantly less run-time than RF. This is because RF relied on more complex ensembles of decision trees to make predictions than XGBoost during the hyperparameter tuning phase.

4.4 Mutation Percentage Impact

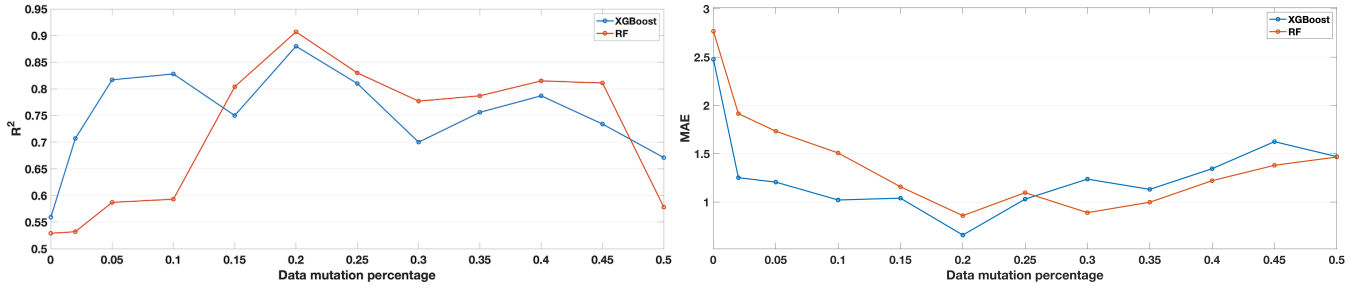
As discussed in Section 3, data mutation methodology was carried out to simulate issues that affect the quality of the processed data window according to a specified percentage of mutations. Specifically, we utilized the whole set of data quality dimensions, which are five in total, in this experiment. To analyze the impact of the mutation percentage in the initialization phase, we have observed the result of the test oracle by adjusting the percentage value using a validation set. The test oracle was evaluated in terms of R^2 and MAE performance metrics for each ML algorithm. The results of the RF and XGBoost algorithms are summarized in Figure 4.

For both algorithms, we can see that the evaluation metrics demonstrate a U-shaped trend. Specifically, for R^2 as displayed in Figure 4a, the value begins at a rate slightly above 0.5 for both algorithms when no data mutants are simulated, then improving to reach approximately 0.9 using a data mutation percentage of 20%. After reaching the peak, the performance metric starts to drop as we increase the percentage of data mutants to reach approximately 0.68 and 0.58 for XGBoost and RF, respectively, at a mutation percentage of 50%. Identically as shown in Figure 4b, the curve of the MAE metric follows the same trend. Beginning with the highest error levels with noise mutants introduced in the training data to achieve optimal performance with MAE of 0.65 for XGBoost and 0.86 for RF at 20% of the percentage of mutants. After that, inducing more data mutants does not improve the training process and the error steadily increases with higher mutation percentages.

The results of this experiment showed that the percentage of induced mutants should be carefully chosen to obtain the value that produces the optimal performance. In particular, a low mutation percentage may not introduce enough data quality issues to sufficiently train the ML model to correctly score the data quality. However, setting the mutation percentage too high could introduce many quality issues into the data, disrupting the learning process and leading to poor performance. Therefore, choosing the mutation percentage carefully to strike a balance between introducing sufficient data quality issues and avoiding introducing too many errors that could negatively impact the ML performance is critical in the initialization phase.

5 THREATS TO EXTERNAL VALIDITY

External validity concerns generalizing the findings of the proposed framework. Generalizability is the main threat in constructing frameworks for real-world applications since every use case may require a different approach to solve the problem. However, the DQSOps framework shown in Figure 1 presents an abstract pipeline workflow and was designed to be flexible without rigid specifications on its components to promote reproducibility. Therefore, it can be employed for scoring data of different natures in diverse settings with simple adjustments to the methods used in this research. For example, the scores of data quality dimensions presented in Section 3.1 were designed to handle univariate time-series data as our use case requires. However, other methods can be followed to score the quality of multivariate data [40, 52] and can be integrated into the DQSOps framework. Similarly, the rest of the components can be determined to fit the system requirements of the handled use case. Additionally, the utilization of the configuration files enhanced delivering a flexible framework. The configuration files store auxiliary meta-information such as the parameters of data quality dimensions, and mutation percentage. This meta-information can be configured based on the use case needs without modifying the underlying structure of DQSOps. Based on our analysis, we believe that using different techniques than those employed in this research would still yield similar results and findings to our study.



(a) R2 **(b) MAE**

Figure 4: Mutants percentage impact on performance

6 CONCLUSION

In this paper, we have presented a Data Quality Scoring Operations (DQSOps) framework. The framework can quantify the quality of data records and distinguish high- and low-quality data. The framework is integrated with two scoring approaches: a prediction-based approach and a standard-based approach. The prediction-based approach is used to predict the quality score of the collected data windows using an ML model. In contrast, the standard-based approach is periodically invoked to produce the ground-truth quality score. The score is found by combining several score metrics from the defined data quality dimensions and is used to design a test oracle. The test oracle continuously evaluates the ML model to activate a retrain signal to update the ML model. Furthermore, a data mutants simulator is integrated into DQSOps to induce quality issues in the data and facilitate the learning process. The framework is deployed and evaluated in a real-world industrial use case. The results showed significant speedup rates achieved by DQSOps compared to the standard scoring method while maintaining high predictive performance. An analysis of the optimal mutation percentage was also presented to gain insight into its impact on the learning process.

ACKNOWLEDGMENTS

This work has been funded by the Knowledge Foundation of Sweden (KKS) through the Synergy Project AIDA - A Holistic AI-driven Networking and Processing Framework for Industrial IoT (Rek:20200067).

REFERENCES

[1] Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics* 2, 4 (2010), 433–459.
[2] Muhammad Aslam. 2021. A new goodness of fit test in the presence of uncertain parameters. *Complex & Intelligent Systems* 7, 1 (2021), 359–365.
[3] Claudia Augste and Martin Lames. 2011. The relative age effect and success in German elite U-17 soccer teams. *Journal of sports sciences* 29, 9 (2011), 983–987.
[4] Earl T Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2014. The oracle problem in software testing: A survey. *IEEE transactions on software engineering* 41, 5 (2014), 507–525.
[5] Carlo Batini, Anisa Rula, Monica Scannapieco, and Gianluigi Viscusi. 2015. From data quality to big data quality. *Journal of Database Management (JDM)* 26, 1 (2015), 60–82.
[6] Roger Blake and Paul Mangiameli. 2011. The effects and interactions of data quality and problem complexity on classification. *Journal of Data and Information Quality (JDIQ)* 2, 2 (2011), 1–28.
[7] Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. 2021. Engineering ai systems: A research agenda. *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems* (2021), 1–19.
[8] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[9] John Byabazaire, Gregory O'Hare, and Declan T Delaney. 2022. End-to-End Data Quality Assessment Using Trust for Data Shared IoT Deployments. *IEEE Sensors Journal* (2022).
[10] Cinzia Cappiello, C Cerletti, C Fratto, and Barbara Pernici. 2018. Validating data quality actions in scoring processes. *Journal of Data and Information Quality (JDIQ)* 9, 2 (2018), 1–27.
[11] Emily Caveness, Paul Suganthan GC, Zhuo Peng, Neoklis Polyzotis, Sudip Roy, and Martin Zinkevich. 2020. Tensorflow data validation: Data analysis and validation in continuous ml pipelines. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2793–2796.
[12] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).
[13] Abraham Chan, Arpan Gujarati, Karthik Pattabiraman, and Sathish Gopalakrishnan. 2022. The Fault in Our Data Stars: Studying Mitigation Techniques against Faulty Training Data in Machine Learning Applications. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 163–171.
[14] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.
[15] Sezal Chug, Priya Kaushal, Ponnurangam Kumaraguru, and Tavpritesh Sethi. 2021. Statistical Learning to Operationalize a Domain Agnostic Data Quality Scoring. *arXiv preprint arXiv:2108.08905* (2021).
[16] Corinna Cichy and Stefan Rass. 2019. An overview of data quality frameworks. *IEEE Access* 7 (2019), 24634–24648.
[17] RalphB D'Agostino. 2017. *Goodness-of-fit-techniques*. Routledge.
[18] Koen Decancq and Maria Ana Lugo. 2012. Inequality of wellbeing: A multidimensional approach. *Economica* 79, 316 (2012), 721–746.
[19] Adenekan Dedeke. 2000. A Conceptual Framework for Developing Quality Measures for Information Systems.. In *IQ*. 126–128.
[20] Lisa Ehrlinger and Wolfram Wöß. 2018. A novel data quality metric for minimality. In *International Workshop on Data Quality and Trust in Big Data*. Springer, 1–15.
[21] Diane L Evans, John H Drew, and Lawrence M Leemis. 2008. The distribution of the Kolmogorov–Smirnov, Cramer–von Mises, and Anderson–Darling test statistics for exponential populations with estimated parameters. *Communications in Statistics—Simulation and Computation®* 37, 7 (2008), 1396–1421.
[22] Wenfei Fan. 2015. Data quality: From theory to practice. *Acm Sigmod Record* 44, 3 (2015), 7–18.
[23] Harald Foidl and Michael Felderer. 2019. Risk-based data validation in machine learning-based software systems. In *proceedings of the 3rd ACM SIGSOFT international workshop on machine learning techniques for software quality evaluation*. 13–18.
[24] Bernd Heinrich, Diana Hristova, Mathias Klier, Alexander Schiller, and Michael Szubartowicz. 2018. Requirements for data quality metrics. *Journal of Data and Information Quality (JDIQ)* 9, 2 (2018), 1–32.
[25] Bernd Heinrich, Mathias Klier, Alexander Schiller, and Gerit Wagner. 2018. Assessing data quality–A probability-based metric for semantic consistency. *Decision Support Systems* 110 (2018), 95–106.
[26] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal, and Vitobha Munigala. 2020. Overview and importance of data quality for machine learning tasks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3561–3562.
[27] Yue Jia and Mark Harman. 2010. An analysis and survey of the development of mutation testing. *IEEE transactions on software engineering* 37, 5 (2010), 649–678.
[28] Meenu Mary John, Helena Holmström Olsson, and Jan Bosch. 2021. Towards mlops: A framework and maturity model. In *2021 47th Euromicro Conference on*

Software Engineering and Advanced Applications (SEAA). IEEE, 1–8.
[29] Steven G Johnson, Stuart Speedie, Gyorgy Simon, Vipin Kumar, and Bonnie L Westra. 2016. Application of an ontology for characterizing data quality for a secondary use of EHR data. *Applied clinical informatics* 7, 01 (2016), 69–88.
[30] Sonia Kahiomba Kiangala and Zenghui Wang. 2021. An effective adaptive customization framework for small manufacturing plants using extreme gradient boosting-XGBoost and random forest ensemble learning algorithms in an Industry 4.0 environment. *Machine Learning with Applications* 4 (2021), 100024.
[31] Shirlee-ann Knight. 2011. The combined conceptual life-cycle model of information quality: part 1, an investigative framework. *International journal of information quality* 2, 3 (2011), 205–230.
[32] Paul Kvam, Brani Vidakovic, and Seong-joon Kim. 2022. *Nonparametric Statistics with Applications to Science and Engineering with R.* John Wiley & Sons.
[33] Erin LeDell and Sebastien Poirier. 2020. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, Vol. 2020.
[34] Lillian Lee. 2000. Measures of distributional similarity. *arXiv preprint cs/0001012* (2000).
[35] Nan Li and Jeff Offutt. 2016. Test oracle strategies for model-based testing. *IEEE Transactions on Software Engineering* 43, 4 (2016), 372–395.
[36] Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory* 37, 1 (1991), 145–151.
[37] Antonios Lionis, Konstantinos P Peppas, Hector E Nistazakis, and Andreas Tsigopoulos. 2021. RSSI Probability Density Functions Comparison Using Jensen-Shannon Divergence and Pearson Distribution. *Technologies* 9, 2 (2021), 26.
[38] David Loshin. 2010. *The practitioner's guide to data quality improvement.* Elsevier.
[39] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering* 31, 12 (2018), 2346–2363.
[40] Harald Martens and Magni Martens. 2001. *Multivariate analysis of quality: an introduction.* John Wiley & Sons.
[41] Xiaofeng Meng and Xiang Ci. 2013. Big data management: concepts, techniques and challenges. *Journal of computer research and development* 50, 1 (2013), 146–169.
[42] Helen-Tadesse Moges, Karel Dejaeger, Wilfried Lemahieu, and Bart Baesens. 2013. A multidimensional analysis of data quality for credit risk management: New insights and challenges. *Information & Management* 50, 1 (2013), 43–58.
[43] Michael Muller, Ingrid Lange, Dakuo Wang, David Piorkowski, Jason Tsay, Q Vera Liao, Casey Dugan, and Thomas Erickson. 2019. How data science workers work with data: Discovery, capture, curation, design, creation. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–15.
[44] Hoang-Vu Nguyen and Jilles Vreeken. 2015. Non-parametric jensen-shannon divergence. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 173–189.
[45] Jack E Olson. 2003. *Data quality: the accuracy dimension.* Elsevier.
[46] Liu Peng and Lei Lei. 2005. A review of missing data treatment methods. *Intell. Inf. Manag. Syst. Technol* 1 (2005), 412–419.
[47] Leo L Pipino, Yang W Lee, and Richard Y Wang. 2002. Data quality assessment. *Commun. ACM* 45, 4 (2002), 211–218.
[48] John Winsor Pratt and Jean Dickinson Gibbons. 2012. *Concepts of nonparametric theory.* Springer Science & Business Media.
[49] Laura Rettig, Mourad Khayati, Philippe Cudré-Mauroux, and Michał Piórkowski. 2019. Online anomaly detection over big data streams. In *Applied data science*. Springer, 289–312.
[50] David Schuler and Andreas Zeller. 2013. Covering and uncovering equivalent mutants. *Software Testing, Verification and Reliability* 23, 5 (2013), 353–374.
[51] Kelly M Sunderland, Derek Beaton, Julia Fraser, Donna Kwan, Paula M McLaughlin, Manuel Montero-Odasso, Alicia J Peltsch, Frederico Pieruccini-Faria, Demetrios J Sahlas, Richard H Swartz, et al. 2019. The utility of multivariate outlier detection techniques for data quality evaluation in large studies: an application within the ONDRI project. *BMC medical research methodology* 19, 1 (2019), 1–16.
[52] Ikbal Taleb, Mohamed Adel Serhani, Chafik Bouhaddioui, and Rachida Dssouli. 2021. Big data quality framework: a holistic approach to continuous quality management. *Journal of Big Data* 8, 1 (2021), 1–41.
[53] Hui Yie Teh, Andreas W Kempa-Liehr, and Kevin I-Kai Wang. 2020. Sensor data quality: A systematic review. *Journal of Big Data* 7, 1 (2020), 1–49.
[54] Reza Vaziri, Mehran Mohsenzadeh, and Jafar Habibi. 2019. Measuring data quality with weighted metrics. *Total Quality Management & Business Excellence* 30, 5-6 (2019), 708–720.
[55] Yair Wand and Richard Y Wang. 1996. Anchoring data quality dimensions in ontological foundations. *Commun. ACM* 39, 11 (1996), 86–95.
[56] Richard Y Wang, Veda C Storey, and Christopher P Firth. 1995. A framework for analysis of data quality research. *IEEE transactions on knowledge and data engineering* 7, 4 (1995), 623–640.