

SURVEY

Open Access



Quality assurance strategies for machine learning applications in big data analytics: an overview

Mihajlo Ogrizović^{1*}, Dražen Drašković¹ and Dragan Bojić¹

*Correspondence:
ogrizovic@etf.bg.ac.rs

¹ School of Electrical Engineering,
Department of Computer
Science and Information
Technology, University
of Belgrade, Belgrade, Serbia

Abstract

Machine learning (ML) models have gained significant attention in a variety of applications, from computer vision to natural language processing, and are almost always based on big data. There are a growing number of applications and products with built-in machine learning models, and this is the area where software engineering, artificial intelligence and data science meet. The requirement for a system to operate in a real-world environment poses many challenges, such as how to design for wrong predictions the model may make; How to assure safety and security despite possible mistakes; which qualities matter beyond a model's prediction accuracy; How can we identify and measure important quality requirements, including learning and inference latency, scalability, explainability, fairness, privacy, robustness, and safety. It has become crucial to test thoroughly these models to assess their capabilities and potential errors. Existing software testing methods have been adapted and refined to discover faults in machine learning and deep learning models. This paper covers a taxonomy, a methodologically uniform presentation of all presented solutions to the aforementioned issues, as well as conclusions about possible future development trends. The main contributions of this paper are a classification that closely follows the structure of the ML-pipeline, a precisely defined role of each team member within that pipeline, an overview of trends and challenges in the combination of ML and big data analytics, with uses in the domains of industry and education.

Keywords: Quality assurance, Machine learning, Big data, Model quality, Data quality, ML pipeline quality, Software in production, Integration and system testing

Introduction

Since the 1950s and the Turing Test, the field of artificial intelligence (AI) has had many ups and downs [94]. The first AI software, mostly based on games (e.g., chess), was developed precisely in those years [24, 91]. Then software systems were created that are used in medicine as decision-making systems [73, 130], and very soon after that, AI-tools for education [10, 146], and in the form of more complex games [4, 53, 126] and AI simulations [32]. Traditional algorithms have been limiting for solving some important problems, so researchers have devised new algorithms and techniques, such as genetic algorithms, gradient descent, Newton's method, simulated

annealing, and other optimization techniques in software development [2, 97, 138]. After that, the term machine learning, as well as neural networks, quickly came into use, until the emergence of deep learning as a subfield of AI. Finally, we are discussing generative AI and AI-human collaboration as a revolution in this field [36, 39, 40].

A large number of modern software systems today include at least some techniques and algorithms of artificial intelligence, and most often, it is machine learning (ML), as AI subfield. Professionals use ML techniques actively in software systems based on predictive analytics (e.g., real estate or cars price estimation [43, 86], stock market [115, 158], conclusion on soil yield based on soil quality [30] and weather forecast prediction [119, 128], or preferred medical therapy in patients [74, 96], recognition of shapes and images (e.g., in systems for recognizing different images at airports [136], parking lots [3], or in software for autonomous vehicles [81]), analysis of collected images (e.g., in medical diagnostics [5, 113, 139, 152] and in industrial facilities for anomaly detection [75, 141]), and natural language processing (e.g., in software that recognizes clinical texts [78], fake news [62], hate speech [123], suicide [59], classify user comments and reviews [117, 163], filter e-mail messages according to content and reject spam messages [93], speech recognition and processing systems [29, 102], sentiment analysis and machine translation [8, 33]). Also, in e-commerce, almost every company uses smart systems or subsystems based on recommending content [9, 28]. In education, the number of AI software is growing, primarily in the recognition of handwritten text and automatic grading of student tests [61, 104].

In order to be able to implement and test well the AI-software systems that we want to develop, it is important to do a good first step of collecting and analyzing client requirements, as well as designing the software system, but that is not enough. For a ML model to produce correct results within such software, it is important that the software has sufficient large data sets. If the data set is relatively small, no matter how well the software is designed and implemented, as well as the model within it, it may not be able to reach sufficiently high-quality conclusions. Therefore, the motivation for working on this review paper is to determine which quality assurance (QA) strategies we should use when developing ML applications based for big data analytics.

In the first step of the research, we wanted to analyze industrial trends in developing complex systems based on big data analytics and ML models. We included several review papers based on ML and DL software system testing. We analyzed the characteristics and methodologies on which the authors of those studies based their work. In the second step, we analyzed the existing solutions of individual testing techniques and mapped them into six categories as a new type of classification. Finally, with the growing interest in AI applications in big data-driven software development [99], we highlighted the advantages of the new classification and the possible importance for software engineers, software system architects, and big data analysts. In this research process, innovative methods were used: Specialization (proposal of a new specific taxonomy for QA strategies in ML software systems based on big data) as a dominant method, as well as Adaptation with Transgranularization (proposal of using QA techniques used in traditional software now on modern ML-based software) which are three of the proposed ten scientific methods in [11].

This survey paper should answer many questions, such as: Can we use different testing techniques to develop better ML models in software? Can the quality of the input data have a decisive effect on the output data of the software, and what techniques can we use to improve the data quality? What components make ML pipelines, and how can we improve the whole process, from analyzing client requirements to producing such ML systems with validation, verification, and other techniques? How many roles do integration and system testing play if we know the nature of the ML components that make up the software system? What else constitutes responsible ML engineering in developing ML applications in big data analytics?

The structure of this paper consists of six chapters. After the introductory chapter, the second chapter describes the problem to be solved. The third chapter describes other review papers covering this multidisciplinary field, which intersects software development, ML, and big data analytics. The fourth chapter describes a new taxonomy and the existing research works are mapped in the proposed taxonomy. The fifth chapter discusses challenges and opportunities in the QA process for ML big data software systems. A conclusion is given at the end of the paper.

Problem definition

For the sake of getting clear about the context of this review, there are some terms that we will now define for the remainder of our paper—namely, big data analytics, machine learning (ML) and product quality. Big data refers to data sets that are highly voluminous, large and complex. Due to the size and complexity of these data sets, they become difficult to process with traditional data processing applications. Big data typically consist of massive databases with structured and unstructured information collected from social media, sensors, internet transactions, and other digital sources. Big data is generally depicted by four main features, that is called four Vs: volume, velocity, variety, veracity. Volume is related to amount of data captured/collected. Big data refers to huge datasets from several terabytes to petabytes and more. Velocity: refers to the speed at which the data is generated, collected and processed.

In most big data environments, data is being generated in near-real time or even in real time. Data needs to be processed and analysed rapidly so that the insights that will be derived from it can be provided to the stakeholders rapidly—often in a matter of seconds. Variety describes the many different types of data that can be found in a big data set. Different big data sources can produce structured data (data in a database, in a spreadsheet, in a form, etc.), semi-structured data (an XML file, a JSON document, etc.) and unstructured data (a text document, an image, a video). Veracity describes the quality of the data. Data from big data sources tends to be of varying levels of accuracy and trustworthiness. The concept of veracity is a concern about how much we can trust the data and make decisions based on it, knowing full well that the data can be grossly inaccurate, inconsistent and biased. Big data analytics is about digging for useful information patterns in large volumes of input and applying them to management, prediction, and other analytic functions.

Machine learning (a part of intelligence) focuses on deriving functions from observations or training data sources. In this field a ML algorithm, often called a modeling method is used to establish how the function (ie. ML model) is acquired from the

data that is observed. The act of feeding data into the algorithm to acquire the model is commonly known as training the model while the models capability to produce results, for inputs in line, with the training data domain is referred to as model inference, which involves making predictions. The most frequently used ML methods in big data analytics are: Decision Trees and Random Forests; Support Vector Machines (SVM); K-means Clustering; Principal Component Analysis (PCA); Gradient Boosting Machines (GBM); and deep neural networks.

The construction of ML models typically follows a structured pipeline, shown in Fig. 1. Before the model can be learned, clarity regarding its purpose or objectives (model requirements) is essential. After that, the collection of training data (data collection), identification of expected outcomes for these data (data labeling), and data preparation are undertaken. Data preparation involves activities such as identifying and rectifying errors, addressing missing data, and transforming the data into a suitable format for ML algorithms—these activities are referred to as data cleaning and feature engineering. Following the training stage with a ML algorithm, the model undergoes evaluation. If deemed satisfactory, it can be deployed and often monitored in production environments.

Products, with machine learning (ML) elements are evaluated based on their performance, dependability and user satisfaction regarding the ML features they offer. This assessment covers aspects, such as;

- Accuracy—Refers to how well the machine learning model correctly predicts or classifies outputs based on the given input data.
- Performance—The speed and efficiency of the ML algorithms in processing data and producing results within reasonable timeframes.
- Robustness—The consistency and reliability of ML components across different conditions, datasets and environments.
- Fairness—Ensuring that ML algorithms do not show biases or discriminate against groups or individuals when making decisions or predictions.
- Interpretability—The transparency and clarity of ML models enabling users to comprehend how decisions or predictions are reached.
- Scalability—The ability of ML components to handle amounts of data and increased usage without drops in performance.

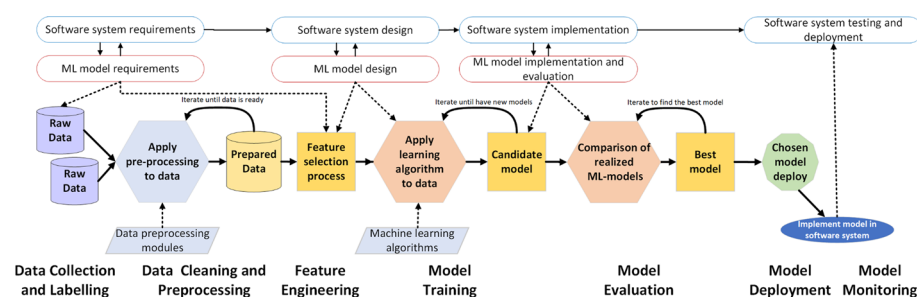


Fig. 1 ML pipeline within the life cycle of ML-software development for big data analytics

In terms of manufacturing quality for products incorporating ML elements it involves adhering to standards, specifications and best practices, throughout the development, deployment and maintenance stages of the products ML aspects. Key aspects include:

- Making sure that the data used to train machine learning models is precise, comprehensive and reflects the target population accurately.
- Carrying out validation and testing procedures to confirm the accuracy, resilience and efficiency of the models in situations, including tricky scenarios and potential failures.
- Setting up solid version control systems and maintaining detailed documentation to monitor changes made to the machine learning models, datasets and codebase. This enables reproducibility of outcomes, eases troubleshooting and auditing processes.
- Ensuring that the machine learning components adhere to regulations, standards and industry norms in tightly regulated sectors like healthcare, finance and automotive industries.
- When rolling out machine learning models in production settings, having mechanisms in place to identify and address issues like model drift, performance decline or security risks.
- Establishing procedures for monitoring, assessment and enhancement of the machine learning components over time. Utilizing feedback from users, stakeholders, as real world usage data helps refine performance continually.

Quality assurance (QA) involves measures taken to guarantee that products or services meet defined requirements and standards. A series of methodical tasks are carried out to avoid flaws, mistakes or shortcomings, in the goods or services created.

Existing surveys and classifications

There have been different approaches done through survey or overview papers in classifying testing methodologies in ML, Big Data or in general AI applications. A discussion and comparison of a couple of representative papers follows. In the remainder of the paper, other survey papers on testing ML systems will be mentioned, but are omitted from this discussion, due to the fact that they will be discussed later on in the paper.

Batarseh [7] defines a new AI assurance definition and maps different AI subarea (for example: data science, genetic algorithms, reinforcement learning, computer vision, ML, NLP, etc.), in past and future period. The authors suggest that for every application of AI in software systems, three views should be analyzed: AI domain, AI subarea and AI goal.

Braiek [13] gives an overview of papers based on testing ML software and gives a classification based on the testing method used on the ML software. Similar concepts used in regular software testing are used here as well, with the classification being provided being the following:

- Approaches that aim to detect conceptual and implementation errors in data (e.g. [80, 114]), and

- Approaches that aim to detect conceptual and implementation errors in ML models:
- Approaches that aim to detect conceptual errors in ML models and
 - Black box testing approaches for ML models (e.g. [44, 165])
 - White box testing approaches for ML models (e.g. [109, 150])
- Approaches that aim to detect errors in ML code implementations: Numerical-based testing: Finite-difference techniques (e.g. [64]), Property-based testing (e.g. [45, 64]), Metamorphic testing (e.g. [35, 98]), Mutation testing (e.g. [89]), Coverage-Guided fuzzing (e.g. [105]) & Proof-based testing (e.g. [127]).

Zhang [166] analyses and reports data on the research distribution, datasets, and trends that characterize the ML testing literature. The paper analyzes the literature based on the following classifications, which contains a few topics:

- Testing Workflow—Test Input Generation (Domain-specific Test Input Synthesis (e.g. [145]), Fuzz and Search-based Test Input Generation (e.g. [47]), Symbolic Execution Based Test Input Generation (e.g. [118]), Synthetic Data to Test Learning Program (e.g. [111]), Test Oracle (Metamorphic Relations as Test Oracles (e.g. [19]), Cross-Referencing as Test Oracles (e.g. [110]), Test Adequacy (Test Coverage (e.g. [109]), Mutation Testing (e.g. [89]), Surprise Adequacy (e.g. [76]), Rule-based Checking of Test Adequacy (e.g. [14]), Test Prioritization and Reduction (e.g. [16]), Bug Report Analysis (e.g. [149]), Debug and Repair (e.g. [34]), Testing Framework and Tools (e.g. [101]).
- Testing Properties—Correctness (e.g. [116]), Model Relevance (e.g. [157]), Robustness and Security (e.g. [17]), Efficiency (e.g. [77]), Fairness (e.g. [41]), Interpretability (e.g. [23]), Privacy (e.g. [31])
- Testing Components—Bug Detection in Data (e.g. [55]), Bug Detection in Learning Program (e.g. [92]), Bug Detection in Framework (e.g. [137])
- Application Scenario—Autonomous Driving (e.g. [156]), Machine Translation, Natural Language Inference (e.g. [100]).

Sugali [144] overviews the different issues and challenges of AI/ML application testing, but doesn't go further into researching actual solutions nor providing any sort of classification method.

In [134], the authors describe the five-step interactive rapid review study, as well as its results, which include newly created a ML testing taxonomy that aims to guide the formulation of practical ML testing challenges at an appropriate abstraction level to support identification and design of relevant research. Another result is a list of twelve open questions from practitioners guided by the taxonomy, out of which three questions are mapped to the selection of studies extracted for this paper. These three research questions include:

- How to test the dataset? The dataset is evolving, which motivates the following sub-questions: How to identify mislabeled data in the dataset? Adequacy testing, i.e., how to assess and improve data (scenario) coverage of the training and test datasets in terms of diversity? How to assess potential bias after the training/test split? How valid is the data and its use? Is the data used for testing within the operational design domain? Or did some of the data originate from another source?
- Are there complementary metrics to assess model correctness (accuracy)? (e.g., edge case measures, uncertainty scores, aggregated metrics across scenes)
- How to generate new test cases for testing the model? (e.g., synthetic data, data augmentation, guided search)

In the paper by Arshad et al. [6], the authors do a review of Big Data testing techniques based on two primary research questions: (RQ1) What are the existing techniques for testing Big Data?

(RQ2) What type of challenges or limitations exist in testing Big Data? For RQ1, the authors identified over twenty methods for testing Big Data, where some of them include: Cube testing (e.g. [46]), Combinatorial technique (e.g. [84]), A MongoDB and XML-Based Functional Testing Technique (e.g. [164]), Transformation testing (e.g. [148]).

Sherin [129] presents an overview of the area of testing ML programs by answering fourteen research questions. One of the research questions is based on the type of approach used in testing of the ML system, where the results of the search show the following approaches: Metamorphic testing, Coverage based testing, Adversarial testing, Mutation testing, Symbolic & Concolic testing, Multi-implementation testing, Evolutionary computing.

Ji [60] provides a survey of quality assurance technologies that have major roles in big data applications, and extracts quality attributes of the software used on big data applications (some of them are correctness, performance, availability, scalability and reliability) and the factors that influence these attributes. Strengths and weaknesses of each QA solution, as well as the architectural details are provided.

Nti [103] outlines approaches in using ML in Big Data Analytics and presents an all-inclusive and detailed evaluation of previous studies on Big Data Analytics using ML. The paper also provides valuable features of compared techniques in Big Data Analytics using ML.

Another classification is based on whether the paper proposes black or white box testing (most propose black box testing), with subdivisions related to types of generated test artifacts and kinds of testing techniques.

We propose a new taxonomy elaborated in the next chapter based on categories we identified in the existing surveys and classifications. However, the focus of the new taxonomy will be to address contemporary problems and challenges in industry, as well as academia. The main goals of our newly proposed taxonomic classes and the guidelines that accompany that classification in the survey paper are:

The taxonomy in our paper will have some similarities to the already mentioned ones in this section. However, the focus of the taxonomy that will be used in this paper will rely more on industrial problems and challenges. Some of the papers discussed in this

section, such as [60] outline the challenges that are discovered during the research phase. The takeaway in the discussion presented with these challenges is more on general problems that are present in Big Data Application development, while the taxonomy in our paper is focused on the ML development pipeline. Another mentioned paper that also discusses and outlines the challenges of QA in Big Data is [103], which covers challenges that are discussed in our paper more in detail in Sect. "Responsible engineering".

The advantages of our newly proposed classification and the guidelines that accompany that classification in this paper are:

- The classification should follow closely the structure of the ML-pipeline defined in chapter 2. In this way, each team member that is responsible for a particular step in the ML-pipeline (e.g. Model data scientist, ML developer, ML tester, Big data analyst, etc.) is able to easily identify relevant QA techniques for that step.
- We focused on the combination of ML and big data analytics. We included all state-of-the-art and the latest research works on the topic being addressed.
- Apart from uses in the industry, taxonomy should serve to identify topics and areas for new university courses in the intersection of classical software engineering and data science in the area of interest.

A new taxonomy

A survey of publications for the topic of quality assurance of ML approaches for big data analytics, grouped by the taxonomy of QA strategies introduced in chapter 3, is presented in this chapter. Each presented research is mapped to seven Ws:

- Who? Authors of research.
- When? Publishing year.
- Why? The aim of the research.
- Whom? The target group for which the research is intended.
- Where? The institution and research group that conducted the research / developed the methodology or tool/framework.
- What/Which? What is the solution? / To which category does the solution belong?
- How? How the research went and what are the main phases of the applied methodology?

Figure 2 shows the new taxonomy with all possible QA aspects in machine learning systems for big data analytics, covered in this survey paper.

Model quality

Quality assessment of ML models is not the same as for traditional software. Given that ML models, unlike traditional software, do not have clear correctness specifications, we cannot perform simple comparison of the obtained output with the expected one. Validation boils down to how well the model fits the data. We accept that some predictions will be wrong and try to estimate how often the model makes mistakes for our data. This means that model evaluation focuses on checking whether the model fits the problem and helps us solve it. This is quite different from the way software is usually tested on

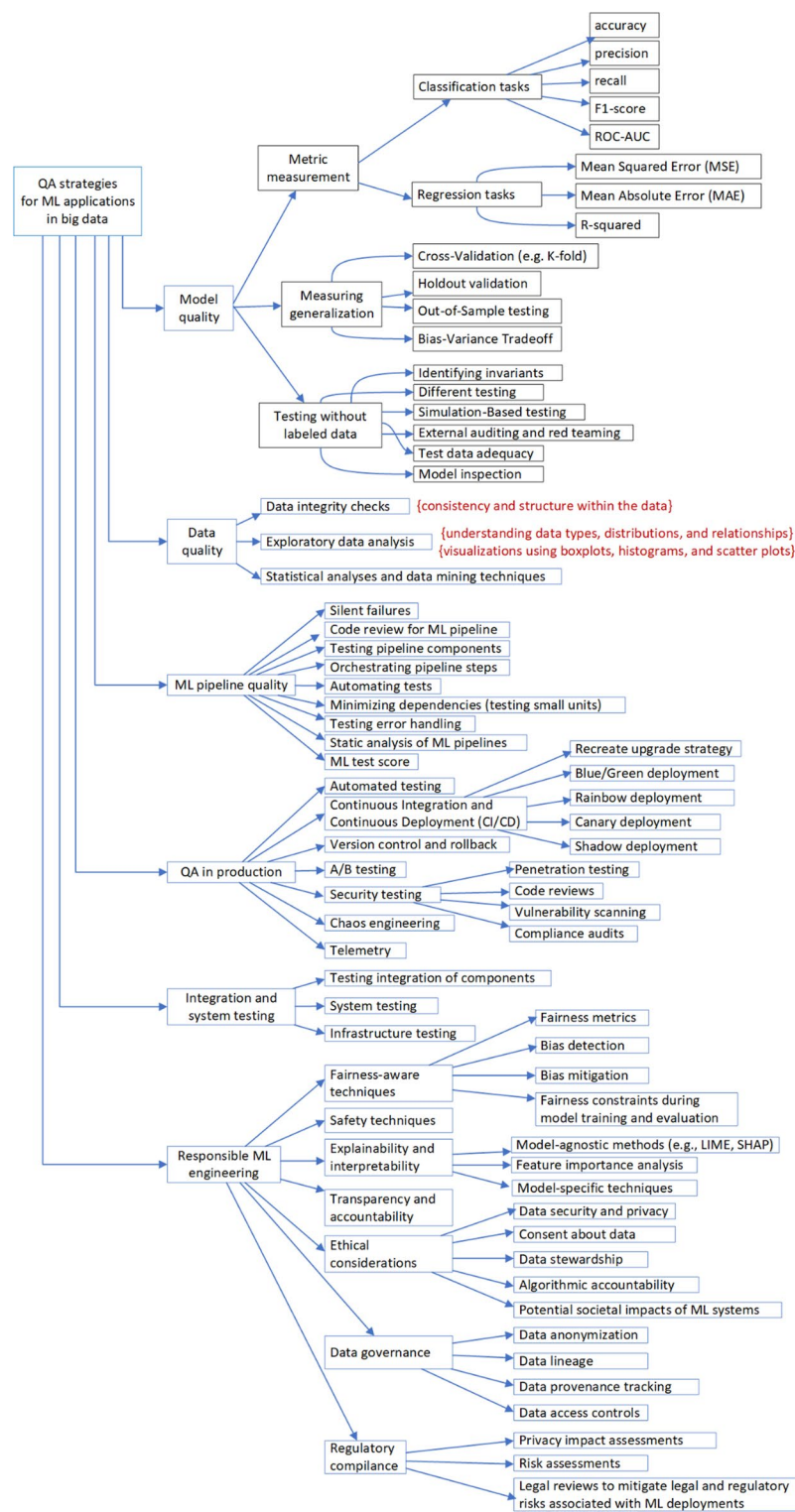


Fig. 2 Category tree of QA strategies in ML-systems

correctness, and is more like software validation activities such as acceptance and end user testing.

The accuracy of prediction is used to assess the quality of ML models. Model accuracy is measured differently based on the type of task: classification, regression, or ranking. A classification task's accuracy, for example spam detection or image segmentation, is measured by the percentage of correct predictions, taking into account false positives and false negatives via recall and precision metrics. A range of accuracy measures, such as mean squared error or mean average precision, are used for regression models that predict numerical values or ranking models that predict search results. The quality evaluation of generative models is more challenging. These models are evaluated for output quality using similarity evaluation techniques, human judgment, and assessment of certain features.

It is useful to compare model performance with some defined baseline when assessing models. It is possible to define a baseline by following some simple rules or by taking existing state-of-the-art models, so the model output results are thus interpreted meaningfully based on factors like the cost of errors and the degree of improvement over existing solutions.

The generalization measure refers to how well the model performs on new data from the target distribution, which it has not encountered during training. The main problem is overfitting—if the model overfits the training data, it leads to poor performance on new data. Balanced model fitting during training is achieved by finding the right values of hyperparameters that control model complexity, such as regularization.

Separation of data for training, validation, and model testing is necessary to avoid overfitting. Major errors in measuring model accuracy include the use of unrepresentative test data, misleading accuracy measures, data leakage, and label leakage. Data leakage occurs in a situation where the ML model already has information about the test data in the training data, but this information would not be available at prediction time, for example, if the entire data set is used for feature engineering before dividing it into training sets and testing. The consequence of overfitting is overestimated performance and unreliable results. Another common mistake is label leakage, where the model learns irrelevant correlations in the data. Label leakage detection can be done using explanatory techniques to analyze model decisions.

Model testing strategies

Model accuracy is used as a basic quality indicator, but reliance on accuracy as a performance metric can result in inaccurate estimates of the true quality of a model. For instance, one model might perform well on some portion of the input distribution and badly on others, or might fail outright on some unique inputs, perhaps relatively rare ones, in which case an inaccuracy won't be well-represented in overall accuracy scores. Slicing and behavioral testing are two strategies that can be used to provide more nuanced information on how or where a given model underperforms.

Slicing consists of testing model accuracy on certain subsets of data (so-called 'slices'), which can have certain attributes or characteristics relating to the problem. For instance, a slice could be a subset of the training data with a high degree of correlation (eg, data for stocks traded at the same time). You can look at accuracy for each of these slices in

isolation, and thus see where the model is performing poorly and try to come up with a fix or mitigation.

Behavioural testing consists of curating (finding, organizing, annotating and maintaining) test data to test certain ‘behaviours’ of a model in isolation. This allows us to ensure that models are learning the appropriate concepts so that they will generalize.

There are various approaches to testing ML models without the need for labeled data, such as identifying invariants, conducting differential testing, simulation-based testing, external auditing, test data adequacy and model inspection. The application of model quality methods within the ML-pipeline is shown in Fig. 1.

Identifying Invariants: Invariants, which express expected behaviors of a model (assertions that always hold true) without needing specific labels, are crucial for testing robustness and fairness. These invariants can be tested using unlabeled or random data and metamorphic relations, ensuring stability under different conditions. An example of metamorphic relation is: if a search query returns a set of results, then if we add a new search condition to a query, the search should return a subset of the previous result.

Differential Testing: This strategy compares a model’s predictions with those of a reference implementation or multiple models to identify discrepancies. This technique is particularly useful when replacing an existing model with a new one, or when a reliable reference exists.

Simulation-Based Testing: Simulators allow us to assess how well a model performs under various conditions, including scenarios that may be challenging to replicate in real-world testing (e.g. driving autonomous vehicles).

External Auditing and Red Teaming: Red teaming involves deliberate attempts to circumvent model safety mechanisms, while crowd-sourced audits involve end users from diverse backgrounds testing for discriminatory behavior. Both methods uncover issues not caught by traditional testing but lack structured approaches.

Test Data Adequacy: Judging when testing is sufficient involves considering factors like data representativeness and coverage of relevant behaviors or subgroups. There are no established criteria for model testing adequacy, leading to debates on the responsibility of companies in model deployment.

Model Inspection: Developers inspect models to ensure learned rules are plausible and not reliant on spurious correlations. Explainability tools help interpret complex models like neural networks.

Selected papers on model quality assessment

The papers overviewed in this category were extracted from a systematic search extended with snowballing, done through Google Scholar and arXiv, with a search criteria that fit the following boolean: (“ML” OR “machine learning” OR “big data” OR “AI”) AND “model quality”. Table 1 shows the selected research papers and articles analyzed in this survey, which are related to the quality of the model.

The book by Ackerman et al. [1] presents a software testing framework for ML, which covers the aspects of model quality such as correctness, fit, and generalization. The authors propose a test-driven development approach that uses test cases to specify and evaluate model behaviors and requirements. The book also discusses the challenges and opportunities of applying software testing techniques to ML systems.

Table 1 An overview of model quality references

Features/Ref. ID	Pub. date	Aim/goal	Target group			Research organization	Solution	Methodology
			ML dev	ML tst	Rs			
[1]	2022	To provide a systematic and rigorous way of testing ML models	+	+		IBM Research Lab	Experim. analysis	By using test cases to specify and evaluate model behaviors and requirements
[72]	2024	To educate and inform readers about model quality and its testing strategies	+	+	+	Carnegie Mellon University	Review article/survey	By explaining the concept of model quality and its dimensions, and introducing various testing strategies with examples and illustrations
[132]	2022	To explore step-by-step quality model construction process for industrial purpose	+		+	Fraunhofer Institute for Experimental Soft. Eng	Research article	How to develop ML systems based on an industrial use case
[12]	2023	To improve model quality using a proposed framework for automatically generating classification models	+	+	+	Wageningen University & Research	Framework	By using feature engineering methods, improve the quality of a given classification model and propose the highly accurate model in a short time
[63]	2020	To realize the model assertion interface, that can be integrated into the ML development, and an API that can generate weak labels for data where the assertion fails	+			Stanford University	Prototype library for model assertions	Develop model assertions mechanism in a novel library, that can apply to real-world ML systems, and algorithm BAL for selecting data for active learning
[135]	2015	To provide a technique that automates the search for a high-quality predictive model with comparable quality to those found using exhaustive strategies but an order of magnitude more efficiently than the standard baseline approach	+		+	UC Berkley, Brown University & UCLA	System	By introducing TuPAQ, a component of the MLbase system, which decides on an efficient parallel execution strategy during model training, and uses sophisticated techniques both to identify new hyperparameter configurations to try and to proactively eliminate models which are unlikely to provide good results

Legend: ML dev - ML developers, ML tst - ML testers, Rs - researchers

The article by Kästner [72] explains the concept of model quality and its dimensions, such as correctness, fit, and generalization. It also introduces various testing strategies to measure and improve model quality, such as slicing, behavioral testing, and identifying invariants. The article provides examples and illustrations to help readers understand the challenges and best practices of model QA.

The paper by Siebert et al. [132] presents a systematic approach on how to implement quality models for ML systems, demonstrated on an industrial use case. The quality model consists of six steps: Define quality meta-model, Define use case and application context, Identify relevant ML quality requirements, Identify relevant entities of an ML system, Identify reference elements of an ML quality model and Instantiate quality model for use case. The described process improves on work that has already been done in the field, described in [155], where a meta quality model that defines the structure of operationalised quality models has been described.

The paper by Boeschoten et al. [12] focuses on the current state of the art of feature engineering techniques, which are investigated, compared, and implemented to reach an optimal classifier model. Examples include missing data imputation, variable transformation, encoding, scaling, aggregating, dimension reduction, and feature creation/extraction/selection. These techniques are combined and tested with several ML models for robustness, efficacy, and ease of use purposes. To simplify the development of all these prediction models, a new framework has been created using the CARET package. The framework automatizes the process of creating different classification models, and through this process, gives the optimal classification model.

In the paper by Kang et al. [63] the authors introduce the abstraction of model assertions for monitoring and continuously improving ML models. Model assertions are implemented in a Python library, OMG, that can be used with existing ML frameworks. An evaluation on assertions is demonstrated on four ML applications: understanding TV news, AVs, video analytics, and classifying medical readings. Assertions are implemented for systematic errors reported by ML users in these domains, including checking for consistency between sensors, domain knowledge about object locations in videos, and medical knowledge about heart patterns. The demonstrated new algorithms for active learning and weak supervision via assertions are shown to improve model quality over existing methods. An API is proposed for consistency assertions that can automatically generate weak labels for data where the assertion fails, and show that weak supervision via these labels can improve relative model quality by up to 46%.

In the paper by Sparks et al. [135] the authors describe and implement the TuPAQ algorithm and system, implemented in Apache Spark, that automatically finds and trains models for a user's predictive application with comparable quality to those found using exhaustive strategies, however with better performances. The authors evaluate several points in the design space with respect to each of our logical and physical optimizations, and demonstrate that proper selection of each can dramatically improve both accuracy and efficiency. Experimental results are presented on large, distributed datasets up to Terabytes in size, demonstrating that TUPAQ's search techniques converge to high quality models an order of magnitude faster than a standard model search strategy.

Data quality

ML systems are only as good as their data: bad data quality negatively affects the production of biased, unreliable or out-of-date models, affecting decision-making processes and lowering system performance over time.

Forms of poor quality include: accuracy—was it correctly recorded?; completeness—was all relevant recorded?; uniqueness—were all occasions recorded only once?; consistency—does it agree with each other?; and currentness—was it kept up to date?. The difference between inaccurate data and imprecise data is huge; an increase in quantity can easily be undone through poor quality; there is no such thing as ‘raw data’—regardless of the source, every reading from a sensor is subjective and is dictated by design decisions about what to record, how to code and what noise to tolerate.

Data integrity checks—often database schema-based—rigidly enforce data structure and integrity to a data model—a logic of consistency where data is constrained to mathematical relationships centering around referential integrity such as ‘one-to-one’, ‘one-to-many’ or ‘many-to-many’ for a specific record. Schema enforcement in relational schema is rather strict with some relational database management systems refusing data that violate schema constraints. While enforcing data schema in schema-less data formats that are popular in the ML context is more challenging, analysis of structures, use of schema languages, and constraints and validation tools make it possible to enforce data schema and minimise schema drift that is highly important to communicate data among groups with less effort. But more advanced schema lingo and tooling such as automatic schema inference tools force or allow users to discern likely schema from sample data in a sample set, which is an important step in schema adoption.

Exploratory data analysis includes interpreting data types, distributions and relations (often this is done by visualisations, such as boxplots, histograms and scatter plots) to see what trends, oddities or errors exist; to build a better understanding of the data. By applying statistical modelling and data mining techniques, developers can see other relationships that exist in the data.

Anomaly detection usually identifies some pattern in the data that is atypical, redundancy can be a clue to data consistency, and domain knowledge can add to that mix. We can have techniques to repair those data anomalies, or at least flag them. Automated methods come in handy for some of these steps, but if it is serious that could lead to an intervention from one of the data science workers.

This is akin to the use of data linting tools to flag unusual features in datasets (‘data antipatterns’ or ‘data smells’). Data linting tools apply heuristics to attempt to discover implicit issues in datasets thrown at ML models, for example the existence of outliers or the presence of improper way of encoding data. There is a difference between data linting and the application of formalised data schemas that encode assumptions about data. The latter codifies a world of certainty, establishes how decisions about a dataset should be made, yet there are situations when it’s inadequate as a tool. This may be the case in situations where no schema exists, or where the existing schema is wrong or incomplete when put to use for a specific purpose.

The most common type of ML system decay is a phenomenon known as data drift, where an ML system’s data distribution, problem context or data format shifts over time and begins to degrade model performance. Data drift often manifests as a shift in the

likelihood distribution of input data; in these cases, you might be able to detect it by comparing probability distributions of the current input data with past inference output data or model training data. Concept drift arises as a result of changes in the problem context so that you should update model labels or features in order to improve performance. Lastly, schema drift is a result of shifts in the data format or assumption, where you need to version and enforce a schema for applications and presentation.

Monitoring for drift entails periodically checking current data distributions for outliers, examining the relative importance of features to make sure that they are still important in current data, checking for the presence of poorly formatted inputs, and checking prediction accuracy over time. Responses to drift models include re-training the model periodically on new data, and if concept drift is indeed detected, discarding or re-labelling old training data. Within the ML pipeline, proactive approaches include collecting additional data for anticipated changes or simulating an anticipatory data scenario to augment training data and add context; they also include encoding more context as features.

Far more effective (if at all possible) is improving processes related to data generation, collection and labelling, as opposed to cleaning them afterwards. Coordination between producers and consumers of data needs to be built so that all parties develop a richer understanding of quality issues and ideally some quantitative quality goals are set. The documentation of data—from specifications, to provenance and metadata—in general will help facilitate such coordination, and help to ensure data integrity.

Further quality of data requirements includes: secure and archival long-term storage of the data, infrastructure for scalable storage and processing, and data privacy requirements.

The application of data quality methods within the ML-pipeline, the part which includes data collection and labelling, data preprocessing with cleaning, and feature engineering, is shown in Fig. 1.

Selected papers on data quality

The papers overviewed in this category were extracted from a systematic search extended with snowballing, done through Google Scholar and arXiv, with a search criteria that fit the following boolean: (“ML” OR “machine learning” or “AI”) AND “data quality”) OR “big data quality”. Table 2 shows the selected research papers and articles analyzed in this survey, which are related to the quality of the data.

The paper [147] proposes a big data quality framework that aims to enhance the pre-processing activities and strengthen data control. The framework uses a new concept called big data quality profile, which captures quality outline, requirements, attributes, dimensions, scores, and rules. The framework also uses exploratory profiling and sampling components to estimate data quality before and after an intermediate pre-processing phase. The paper discusses the implementation and evaluation of the framework and its benefits for big data analytics.

Fadlallah [38] provides a scoping review to study the existing context-aware data quality assessment solutions. The strength and weaknesses of each outlined solution is discussed. The paper demonstrates that context awareness with the ability to handle big data could not be guaranteed through the existing data quality assessment solutions. A

Table 2 An overview of data quality references

Features/Ref. ID	Pub. date	Aim/goal	Target group			Research organization	Solution	Methodology
			ML dev	RS	BDA			
[147]	2021	To develop a big data quality framework that uses big data quality profile, exploratory profiling, and sampling components. To address the challenges of ensuring data quality and its impact on analytics decisions			+	Zayed University, UAE Uni., Concordia University	Framework	By applying various quality metrics, rules, and functions to estimate and improve big data quality before and after pre-processing
[38]	2023	A comparison of the quality models/solutions to reach a comprehensive view, that cover aspects of context awareness when assessing data quality, and recommendation for design of any big data quality service	+		+	Saint-Joseph University	Review article/survey	Cover assessing the big data quality and achieving context awareness during data quality assessing
[15]	2022	To observe ML/model behavior in terms of data quality: analyze the performance of 5 algorithms for the three tasks of classification, regression, and clustering		+	+	Hasso Plattner Institute, University of Potsdam	Experim. analysis	By using six selected data quality dimensions (consistent representation, completeness, feature accuracy, target accuracy, uniqueness, target class balance) find the relationship with the performance of ML algorithms
[20]	2021	To show a strong correlation between ML-system performance and the quality of datasets. To propose data quality evaluation framework with the data quality criteria	+		+	University of North Texas	Framework	By exposing the quality problem in the big data medical sets, demonstrating the evaluation of data quality during validation process, and their impacts to overall performance of the ML systems
[26]	2018	To develop big data quality framework based on DL techniques and statistical model algorithm	+		+	University of Arkansas at Little Rock	Framework	By reviewing related works, propose a new unique data profiling architecture with three parts: data preparation, ML module and data visualization
[66]	2021	To provide an overview of solutions and problems about data quality management	+		+	Carnegie Mellon University	Review article/survey	By analyzing data and possible problems, try to find relationship between data quality and their impacts on ML-system

Legend: ML dev - ML developers, RS - researchers, BDA - Big data analysts, data scientists, and managers

comparison of existing quality models and solutions led the authors to a set of recommendations framed in a methodological framework shaping the design and implementation of any context-aware data quality service for big data.

The paper [15] explores empirically the relationship between six data quality dimensions (Consistent representation, Completeness, Feature accuracy, Target accuracy, Uniqueness and Target class balance) and the performance of fifteen widely used ML algorithms that cover classification, regression, and clustering. This process is done with the goal of explaining the performance of these algorithms in terms of data quality. Three scenarios are outlined through the experiments based on the AI pipeline steps that were fed with polluted data: polluted training data, test data, or both. A comprehensive experimental study is presented to understand the relation between data quality and ML-model performance under the umbrella of data-centric AI, providing: A systematic empirical study that investigates the relation between six data quality dimensions and the performance of fifteen ML algorithms, A simulation of real life scenarios concerning data in ML-pipelines.

The research in [20] answers a few important questions regarding the data quality to the performance of ML. Cross-validation might not be sufficient to validate the performance of a ML system since the test and the training datasets may share a significant amount of data if the original dataset contains many duplicated data items. The authors discuss whether there could be designed an approach for better evaluating the system. An experimental study was conducted to answer these questions and proposed the solutions based on the experiment results. If the quality of the dataset is not high enough for building a ML system with desirable performance, what are the best strategies that can be applied for improving the system performance? Questions related to best practices for using transfer learning to improve the performance of the ML system, and the best strategies that can be applied for improving the system performance if the quality of the dataset is not high enough for building a ML system with desirable performance, are answered through experiments of a group of transfer-learning-based performance improvement strategies on the medical concept normalization system.

In a paper by Dai et al. [26], a review of relevant works is presented as well as a discussion of ML techniques, tools, and statistical quality models. A creative data profiling framework is demonstrated based on deep learning and statistical model algorithms for improving data quality based on KNIME and WEKA. The architecture of the framework contains three parts: data preparation, ML, and data visualization. Data preparation focuses on identifying basic problems and cleaning data. ML contains neural network and statistical quality control models for discovering complex outlier data. Data visualization displays risk data, error data, and correct data. The authors use public Arkansas officials' salaries, one of the open datasets available from the state of Arkansas' official website, to demonstrate how to identify outlier data for improving data quality via ML.

The article by Kästner [66] gives an overview of solutions and problems about data quality management. An overview of data quality criteria is given, which contains concepts such as: Accuracy, Completeness, Uniqueness, Consistency, and Currentness. A discussion of different approaches on applying data consistency, and detection and repairing of potential data quality problem is provided. The author outlines that before any automated checks, quality assessment usually requires a good understanding of the

data, which almost always starts with exploratory data analysis. A common problem in products with ML components of data and assumptions about data changing over time is also discussed.

Pipeline quality

ML pipelines can experience silent failures, such as model training failure due to a small volume of data, model training stalling on stale data due to a failure of the database connector library, using default values instead of real data from a sensor, or the dropping of telemetry data. Such problems might not crash the system and can therefore go undetected, but they can have serious consequences. Their silent nature exists because ML algorithms are well-suited for corrupted, noisy data, the absence of all input tests leads to a lack of detection for how different inputs behave with other system components, and error detection and recovery mechanisms can fall short.

Code review, testing pipeline components, orchestrating pipeline steps, automating ad-hoc tests, error handling or migrating tests to a new language are just a sample of techniques to ensure quality assurance in ML pipelines.

Code review. While ML model testing is challenging, pipeline code is like regular code, and thus amenable to traditional software QA approaches. To address the issues enumerated above, code review processes for ML pipelines can be established that human analysts would ideally undergo. A review of pipeline code to look for (undesired) errors or inconsistencies in code, incorrect implementations of ML model code, as well as finding violations of best practices (can all) help reduce the occurrence of silent failures and will increase the reliability, as well as the predictive improvement baneful for the cause of ML in business and government alike.

When we think about how to test an ML pipeline, what we're really thinking about is testing the components of the pipeline. When we test code, what we're typically doing is executing the code with a set of inputs, and checking that the output behaves according to our expectations. It's usually safer to test simple things. Things that are easy to write down and validate based on what we already know about the world. Modularity helps with testability. The easier it is to describe the boundary between a component and its context, the easier it is to test that component. You can think of example input cases and come up with expected output cases, and reverse the process, and apply the tests to the component itself. Modularity makes it all possible. Some practical implications of modularity for ML pipelines include: how to coordinate the steps of a pipeline; how to write tests for these steps; and which dependencies we need to break for tests to be useful.

Another practical concept is how testing of error cases works for ML pipelines; what constitute good error boundaries and how testing should focus on these points; and what's the role of static analysis in ML pipelines. Existing static analysis tools focus primarily on style issues and simple bug patterns but may expand in the future. Examples include tools like DataLinter for detecting inefficient data encoding patterns, Pythia for identifying shape mismatches in TensorFlow programs, Leakage Analysis for detecting data leakage, PySmell for identifying code smells in data science code, mlint for analyzing pipeline infrastructure, etc.

One approach to quantify QA activities is an ML test score. The test score covers a range of issues including feature tests, model tests, ML infrastructure tests, and production monitoring. Teams can score points for each issue tested and for each issue where tests are automated.

Selected papers on ML pipeline quality

The papers overviewed in this category were extracted from a systematic search extended with snowballing, done through Google Scholar and arXiv, with a search criteria that fit the following boolean: (“ML” OR “machine learning” OR “big data” OR “AI”) AND (“pipeline” OR “pipeline quality”). Table 3 shows the selected research papers and articles analyzed in this survey, which are related to the ML-pipeline quality.

The article by Kästner [69] defines the concept of pipeline quality and its challenges, such as data drift, model decay, and pipeline failures. It also presents various techniques and tools to monitor and improve pipeline quality, such as data validation, model retraining, pipeline testing, and pipeline orchestration. The article provides examples and code snippets to demonstrate how to implement pipeline QA using TensorFlow Extended and Apache Airflow.

In [27], the authors propose a fresh method to address these issues by introducing an innovative model-driven strategy. This approach empowers ML designers to create models of ML pipelines while considering quality aspects. Specifically, they enhance the metamodel of Feature Models to accommodate the definition of both Quality and Feature Models. These models encompass features and quality characteristics, which are either connected to the features or can be realized through them. Furthermore, their proposed meta-model enables the definition of functional and quality requirements by handpicking necessary quality attributes from the Quality and Feature model and by specifying attributes associated with features.

The manuscript by [140], aimed to establish a robust foundation of research on AI development pipelines by conducting a thorough review of existing literature and conducting semi-structured interviews. Through a Multivocal Literature Review (MLR), 151 relevant sources were identified, leading to the recognition of five primary terms used in the field. The paper also explored key triggers initiating pipeline execution and outlined the four stages of pipeline development: Data Handling, Model Learning, Software Development, and System Operations, each accompanied by specific tasks and challenges.

The paper [42] takes a broader approach for automated ML by first understanding the complexity of the problem before proposing a solution, emphasizing its importance. The main focus is on analyzing the characteristics of the fitness landscape, utilizing a new tool for pipeline generation to explore how automatic methods perform within this landscape. Findings reveal the high complexity of pipeline optimization, with multiple scattered optimal solutions and a notable lack of generality. However, depending on search dimensions, model quality objectives, and data characteristics, basic search techniques can yield results aligned with user expectations.

The outlined chapter in [50] explores the issues and challenges of ML development and deployment. Key requirements and design considerations behind ML deployment solutions are covered. This is done by first defining the lifecycle of the ML software, and

Table 3 An overview of pipeline quality references

Features/Ref. ID	Pub. date	Aim/goal	Target group				Research organization	Solution	Methodology
			ML	Ex/Des	ML dev	Rs DS			
[69]	2022	To explain the concept of pipeline quality and its challenges, and how to monitor and improve it	+		+	+	Carnegie Mellon University	Review article/survey	By defining the concept of pipeline quality and its challenges, and presenting various techniques and tools with examples and code snippets
[27]	2022	To model ML pipelines, their requirements and quality characteristics of algorithms in each pipeline phase	+		+	+	University L'Aquila	New approach for ML pipelines	Realization of a graphical editor which permits to create Quality and Feature models, representing the Product-Line Architecture of the ML pipelines, and to specify functional and quality requirements in final production version
[140]	2023	To create a strategy for continuous AI development based on 4 stages pipeline: Data Handling, Model Learning, SW Development and System Operations	+		+	+	University of Innsbruck	Review article/survey	Three methods are realized: Multivocal Literature Review process (from Formal Sources—Google Scholar, and Informal Sources—Google/YouTube), Taxonomy Creation Strategy and Summarize Qualitative Content Analysis
[42]	2018	To assess the complexity of the optimization problem and uncover its underlying characteristics. To develop algorithms capable of navigating the problem space more intelligently			+	+	University of the Basque Country	Research article	Through crafting a series of experiments aimed at deepening comprehension of the automatic pipeline generation issue, followed by identifying strengths in search methodologies
[50]	2021	To explore the issues and challenges of ML development and deployment	+		+	+	Miracle Finland Oy, Habana Labs & Dell Technical Leadership Community	Review article / survey	By defining the differences between traditional and ML software, the challenges of ML deployment and lifecycle. Solutions and practices are also given for mentioned problems

Table 3 (continued)

Features/Ref. ID	Pub. date	Aim/goal	Target group				Research organization	Solution	Methodology
			ML	Ex/Des	ML dev	Rs			
[167]	2020	To verify the feasibility of building an effective ML pipeline with CI/CD abilities on machines with specific hardware configuration	+		+		College of Computer, National University of Defence Technology	Framework	By deploying & configuring Kubeflow, Gitea and Drone to connect each other, as well as dividing the pipeline into two functional parts: the completion of ML tasks and the compilation of ML task definitions

Legend: ML Ex/Des - ML experts/designers, ML dev - ML developers, Rs - researchers, DS - Data scientists

each of its steps. Specific solution components, best practices, and tooling are outlined. Examples include, recommendations on how to Automate Data Science Steps and how to use MLOps for an automated ML Pipeline.

The paper [167] provides a valuable reference for ML pipeline platform construction in practice, by building a functional ML platform with DevOps capability from existing continuous integration (CI) and continuous delivery (CD) tools and Kubeflow. The authors also constructed and ran ML pipelines to train models with different layers and hyperparameters while time and computing resources consumed were recorded. An analysis of the time and resource consumption of each step in the ML pipeline is presented, as well as an exploration of the consumption concerning the ML platform and computational models. A proposal of potential performance bottlenecks such as GPU utilization is also provided.

Integration and system quality

ML models, like other system modules, could be individually inspected, tested, even stressed—but problems can surface only when these are chained together. The integrated system needs testing too, and then there are specifics of integration where a subsystem might be a single model or a combination of system modules. Additionally, unlike for other software, the quality of ML predictions is entirely fallible. Figure 1 illustrates integration and system testing methods in an ML pipeline. Compare notes on the final extra step of integration of components (supervised by the system module), where, at least, some of the components are ML Models.

Feature interactions create situations in which the behaviour of the composed system differs from the behaviour of the components taken separately—an important source of poor system quality, and a particularly significant research area in systems of systems and architectures. Here's an everyday real-world example of feature interactions [106]. Imagine a smart home system, where two vendors have each supplied a component for voice control and facial recognition functionality. Each component was independently developed and separately tested, and it works as expected. When you say: 'Hey, Alexa, turn on the lights,' the voice-control component recognises the command; when the lights are on, the facial-recognition component will recognise you as the appropriate user and the lights stay on. However, when the two components are combined into one home, the following can happen: 'Hey, Alexa, turn on the lights,' the voice-control component will recognise the command, but the facial-recognition component also recognises you as a face, and will set off an alarm saying that you were recognised by the camera as another person. The voice-control component tries to talk to the alarm, but the alarm says: 'Please do not talk to me; you're not allowed here, this is my house,' and continues blaring. Here, both components are reacting to the same environmental factor, light, and in opposite ways that each separately failed to reveal; one tries to turn on the light and the other tries to prevent a person from entering. This is a conflict that the components can't detect in isolation and which can't be detected individually, but which can be identified once the components are embedded in a specific context.

Testing these interactions involves integration testing – the process of verifying the functionality of the combination of ML and non-ML components between the extremes of unit testing (testing individual components) and system testing (testing the entire

product). An integration test of an image-captioning system might examine interactions between an object detector, a language model and a caption reranker, ingredients of the system.

With system testing, you test and assess whether the system works properly from the beginning to the end—for example, that it is useful, usable, and safe. Acceptance testing, a type of system test, addresses whether the system works for you/the user or meets all the requirements requested. With testing in production, you're able to see how the system responds and works in the real world—issues that might not be seen in 'offline' tests are reflected here.

Alongside testing for core functionality, testing the infrastructure for deployment, operations and monitoring is equally vital: the ability to reliably and continuously deploy products, operate them smoothly and alert upon errors and incidents all play significant roles in ensuring that a system functions as expected in production. Chaos engineering—where failure is injected into production systems—pushes the system to its limits by emulating real-world conditions for fault scenarios.

While tricky to test—but critical nonetheless—is the monitoring and alerting infrastructure that tracks issues in real time and calls attention to them. Running fire drills or other smoke tests—that is, injecting issues into production systems—makes it possible to see whether monitoring and alerting systems work as designed.

Parallels can be drawn in the realm of system and software quality to include the shift from ISO/IEC 9126 to ISO/IEC 25000 series, simply titled SQuaRE [56, 57]. ISO/IEC 9126 was established in 1991 and outlined six software quality characteristics. In 2014, SQuaRE, yet another standard explicitly tailored to policing quality measures, elements, models, characteristics and sub-characteristics, was adopted by the same committee. By employing a tree structure, the established quality characteristics of the ISO/IEC 9126 standard on software quality were simplified and amplified. SQuaRE's top-level quality models are product quality, data quality, quality in use and IT service quality, and represent a system quality framework most people either have experience with or have heard of at least.

Selected papers on integration and system quality

The papers overviewed in this category were extracted from a systematic search extended with snowballing, done through Google Scholar and arXiv, with a search criteria that fit the following boolean: ("ML" OR "machine learning" OR "big data" OR "AI") AND ("integration testing" OR "system testing" OR "integration quality" OR "system quality"). Table 4 shows the selected research papers and articles analyzed in this survey, which are related to the integration and system quality.

The article by [70] defines the concept of integration and system testing for ML systems, and how it differs from traditional software testing. It also discusses the challenges and best practices of integration and system testing, such as defining test scenarios, generating test data, measuring test coverage, and automating testing processes. The article provides examples and illustrations to help readers understand the importance and methods of integration and system testing for ML systems.

Siebert [131] outlines the development of a concrete quality model for a ML system, focusing on an industrial application. Firstly, it reviews existing literature and identifies

Table 4 An overview of integration and system quality references

Features/Ref. ID	Pub. date	Aim/goal	Target group			Research organization	Solution	Methodology
			ML dev	ML tst	RS			
[70]	2022	To define and explain the concept of integration and system testing for ML systems, and its challenges and best practices	+	+	+	Carnegie Mellon University	Review article/survey	Define and explain the concept of integration and system testing and its challenges and best practices and provide examples and illustrations
[131]	2020	To develop a quality model for an industrial-based ML system	+			Fraunhofer IESE, Fujitsu Laboratories Ltd	Research article with guidelines	The approach used to derive a quality model: 1) Understand the use case/problem. 2) Identify views on the SW system and measurement objects 3) Define quality attributes and metrics
[48]	2021	To design a methodology based on established security principles for the development of complex ML system	+		+	Naval Postgrad. School Monterey	New methodology	Introduction of a security methodology based on Zero-Trust and defense-in-depth principles, with examples
[160]	2019	To develop an adversarial test generation framework for closed-loop control systems with RNN		+		Arizona State University	Framework	Experimental analyze realized with 2 system containing NNS (Non-linear system with FNN controller and Steam Condenser with RNN Controller) using different search methods
[106]	2019	To understand feature interaction in datasets and to propose a new function for feature interaction measurement based on prediction performance	+		+	Dankook University	Experim. analysis	Demonstrate feature interaction function with many permutation on two sample datasets (PimaIndians-Diabetes dataset and BostonHousing dataset)
[82]	2020	Analysis of existing research dealing with the design of ML systems and providing guidelines on how to integrate and design safety-critical ML systems	+		+	Denso Corporation and Tokyo Institute of Technology	Review article / survey	Overview of related works and research directions for requirements of ML models and for verification of ML models

Table 4 (continued)

Features/Ref. ID	Pub. date	Aim/goal	Target group			Research organization	Solution	Methodology
			ML dev	ML tst	RS			
[112]	2020	To present a new methodological QA approach, evAla, for new developed ML-based systems (products/services), and their systematic quality risk evaluation	+	+		Volkswagen Group	New methodology (evAla approach)	Four steps: 1) the product/service risks evaluation against potential quality issues; 2) check weaknesses of the ML-system by team; 3) decisions about mitigation actions; 4) define actions

Legend: ML dev - ML developers, ML tst - ML testers, RS - researchers

gaps that the study aims to address. Secondly, it defines various perspectives on ML systems and the pertinent measurement criteria, crucial for evaluation within specific contexts. Thirdly, the methodological approach for quality modeling, tailored to industrial applications, is detailed, encompassing quality aspects and metrics for each measurement criterion. This model facilitates practitioners in specifying and evaluating quality requirements for ML systems. Lastly, the article discusses the relevance of identified quality aspects through expert evaluation and highlights key insights gained, along with proposing future research directions, particularly examining quality variations across different types of ML systems.

The aim of the research by [48] is to aid in designing systems with robust security measures, capable of preventing and mitigating threats from ML components, even when the specific attack methods are unknown during system development. All system elements, including humans, engineered systems, ML components, and external environment, are treated as untrusted, emphasizing the need for stringent security measures, particularly for ML components with decision-making authority or behavior that can't be verified. Proposed security measures in described methodology can involve employing generic solutions from knowledge databases, such as developing separate ML components by isolated teams with voter block logic.

Yaghoubi [160] presents advancements in automatically generating adversarial test cases for nonlinear control systems incorporating neural network (NN) components. The approach utilizes Signal Temporal Logic (STL) to specify system properties and employs a framework employing functional gradient descent to search for adversarial tests. The method integrates local optimal control-based search with a global optimizer to address the non-convex nature of the optimization problem. Notably, the proposed approach doesn't necessitate analytical information about the system model or NN architecture, relying instead on readily available linearizations of the closed-loop system at specified operating points. The assumption is made that the NNs in the system employ differentiable activation functions, which aligns with common training approaches based on gradients.

The study by [82] explores the challenges associated with safety-critical ML systems, reviewing related works and suggesting future research directions. The proposed ideal training process integrates deductive requirements and data-driven training, considering test data as requirements specification and training data as design specification. Results highlight issues such as the absence of requirements and design specifications, interpretability, and robustness in ML models, emphasizing the importance of requirements specification and verification for open environments. Quality models for ML systems are also examined, indicating that the absence of requirements and robustness significantly impacts conventional system quality models.

[Poth2020] introduces a systematic approach, *evAIa* (*evaluate AI approaches*), designed to address questions regarding the quality assurance (QA) and testing methods of ML systems. As ML-based components become increasingly prevalent in industrial products and services, there is a growing need to ensure the quality and reliability of these data-driven functionalities. The proposed approach emphasizes the importance of safeguarding ML-based components, whether they serve as additional features or form the core of a product/service offering. It aims to bridge the gap between established

systematic testing standards, such as ISO/IEC/IEEE 29119 series, and the specific requirements of AI and ML QA, while being adaptable to various development and operational models.

Testing and experimenting in production

Many developers today also run tests in production rather than in a controlled environment, because of the difficulty of addressing model quality issues like overfitting, data drift, or training on non-representative data. Model quality in production is often also largely evaluated with proxy measures—for example, if the output of a prediction model can be revised by the user, we might measure label accuracy only through the rate of user corrections. Proxy measures remain useful because they make it possible to compare two or more models in the same environment, and to monitor the trend in a given measure over time. Telemetry (the set of information collected from a system by its monitoring agents) and the use of automated statistical methods to organise it into meaningful reports are essential for monitoring several measures of quality and gaining an understanding of model performance despite lots of noise and uncertainty in each of those measures. Testing in production can be problematic due to potential harms to users caused by quality issues, as well as the capital investment required for the infrastructure and the design of telemetry.

Telemetry refers to the collection of data from a system at runtime, and the transmission of that data to where it can be analysed. Different methods of telemetry reveal different things about the system and provide varying degrees of confidence in how accurate the model of that system is, while requiring varying degrees of accuracy, reliability, cost and latency. Generally, the design of telemetry follows a three-step process: defining the quality metric, gathering the data relevant to that metric, and operationalising how you compute the metric you defined from the data you gathered. Setting up telemetry involves choosing logging and observability libraries for the system to emit data against. Scaling telemetry requires being thoughtful about the volume of data to be emitted, adding sampling or feature extraction so the data can be stored more efficiently.

In cases where the privacy concerns of telemetry data collection are more acute, you might turn to federated learning with ML models kept and run on the user's device. This is the stance taken by some organisations such as Mozilla and the OpenAI initiative. However, such approaches still limit the type and volume of telemetry that is available for collection. The fundamental architectural and UI designs chosen limit this availability once more: a ML server-based system tends to be able to access more data than a client-side one.

Beyond just observing a system, experimentation in production allows developers to assess changes and detect issues before widespread deployment, with many companies running multiple experiments concurrently to optimize user experiences. Various aspects of experimentation in production include A/B experiments [37], canary releases [151], and other forms of testing like shadow releases [125], blue/green deployments [161, 162], and chaos engineering [58]. There is a need for sophisticated experimental designs when multiple experiments are conducted simultaneously taking care to minimize user exposure to potential risks associated with experimental changes.

A/B experiments involve randomly dividing users into groups to test different variants of a product [79]. It outlines the components needed for A/B experiments. The infrastructure support needed for experimentation, include feature flags (Boolean options that are used to decide between two control flow paths in the implementation), telemetry, and statistical analysis tools to determine the significance of observed differences between groups. Dashboards can be used to visualize A/B experiment results, including measures of effect size and confidence.

The canary release idea is to treat *every* release like an experiment that is initially only rolled out to a small number of users, to limit exposure in case the release performs poorly [18]. If the new release performs worse than the previous release according to telemetry (with all the statistical tests of A/B experiments), it is rolled back so that all users receive the previous release [142].

Chaos engineering, pioneered by Netflix, is a method of testing system robustness in production environments [58]. Deliberately inducing faults, such as server failures, allows teams to observe how the system responds and encourages the development of more resilient software.

Other forms of experiments that mandate running both the old release and the new one in parallel, include shadow releases (only show old system results to the user, internally compare new results with old one) [125] and blue/green deployments (switch all users at once to new version) [161, 162].

Responsible experimentation is necessary, considering the potential impact on users and the ethical considerations involved, such as minimizing harm and ensuring user safety and well-being. There is a need for automation, quick rollback mechanisms, and ethical reviews to conduct experiments responsibly and ethically. The application of methods for QA in production within the ML-pipeline is shown in Fig. 1.

Selected papers on testing and experimenting in production

The papers overviewed in this category were extracted from a systematic search extended with snowballing, done through Google Scholar and arXiv, with a search criteria that fit the following boolean: (“ML” OR “machine learning” OR “big data” OR “AI”) AND (“canary testing” OR “production testing” OR “production experimenting” OR “A/B testing” OR “telemetry” OR “chaos engineering”). Tables 5 and 6 show the selected research papers and articles analyzed in this survey, which are related to the testing and experimenting in production for ML-system development.

Studer [143] paper proposes a process model for the development of ML applications, covering six phases from defining the scope to maintaining the deployed ML application: business and data understanding, data preparation, modeling, evaluation, and deployment. With each task of the process, this work proposes QA methodology that is suitable to address challenges in ML development that are identified in the form of risks, drawn from practical experience and scientific literature. The process model expands on the CRISP-DM data mining process model. The presented work proposes an industry- and application neutral process model tailored for ML applications with a focus on technical tasks for QA.

Kästner [67] defines the concept of testing and experimenting in production for ML systems, and why it is necessary and beneficial. It also introduces various techniques and

Table 5 An overview of testing and experimenting in production references

Features/Ref. ID	Pub. date	Aim/goal	Target group			Research organization	Solution	Methodology
			ML dev	ML	tst RS			
[143]	2021	To provide industry process model for the development of ML apps with QA methodology	+	+		Mercedes-Benz AG, AI Research group	Research article with industry application	Use QA methodology into data mining process model CRISP-DM to cover the specifics of a ML application development
[67]	2021	To define and explain the concept of testing and experimenting in production for ML systems, and its techniques and tools	+	+	+	Carnegie Mellon University	Review article/survey	By defining and explaining the concept of testing and experimenting in production and its techniques and tools, and providing examples and illustrations
[51]	2020	To propose directions for testing ML systems using the Chaos Engineering (CE) approach		+		Universidad de los Andes, Universidad Nacional de Colombia and ADL	Review article/survey	The analysis of existing literature, own experience of the authors with distributed software systems, and a survey with 25 participants
[21]	2022	To propose a fault-tolerant, resilient and reliable testing method based on the Chaos Engineering (CE) in big data systems		+	+	China Mobile Information Tech. Co.Ltd. Beijing	Experm. analysis	The analysis on their own system and other systems
[49]	2020	Review of existing satellite health monitoring systems and using telemetry data mining techniques in real-time monitoring			+	Cairo University, Helwan University and AI Azhar University	Research article	Use modern ML techniques in telemetry mining, discuss about some representative methods to solve satellite health problems, followed by illustrative examples
[142]	2023	To show advantages of "canary deployment"		+	+	University of Applied Sciences FH Campus Wien	Experm. Analysis	Implementation of canary deployment by a simple static file service. Realization of service that stores state. Third use case includes multiple parallel canary deployments

Legend: ML dev - ML developers, ML tst - ML testers, RS - researchers

Table 6 An overview of testing and experimenting in production references (continuation of Table 5)

Features/Ref. ID	Pub. date	Aim/goal	Target group			Research organization	Solution	Methodology
			ML dev	ML tst	RS			
[65]	2020	To design a new course based on AI-Enabled systems			+	Carnegie Mellon University	Research article	Course design, define scope, develop lectures and assignments, and evaluation
[122]	2020	To provide a comprehensive guide to testing and experimenting ML systems in production	+	+		/	Book	The book covers the broader topic of designing systems in production and their testing, including examples from industry
[120]	2020	To analyze the existing solutions for functional testing ML-based systems		+		Universita della Svizzera Italiana, Lugano	Review article / survey	A systematic mapping study was conducted to explore testing techniques for MLs using 33 research questions
[85]	2017	To find out a way to group customers in the data sample in order to achieve an optimal difference between the buckets	+		+	Nordstrom/Hautelook, Algoma University, National Geological Library of China & Tianjin Polytechnic University	Experim. analysis	With the analysis result of real data collected during joining an industry project

Legend: ML dev - ML developers, ML tst - ML testers, RS - researchers

tools to conduct testing and experimenting in production, such as A/B testing, shadow deployment, canary deployment, and online monitoring. The article provides examples and illustrations to help readers understand the advantages and challenges of testing and experimenting in production for ML systems.

Hernandez-Serrato et al. [51] realize a literature review about Chaos Engineering and ML-systems with conditions of uncertainty, and an online survey with 25 engineers with industry experience. They write a report of relevant directions and challenges for the CE application and Site Reliability Engineering, consisting of gathering data, observability and data analysis, adoption, platform-specific tools, automated management of failure scenarios, costs and coordination.

Chen [21] propose a fault-tolerant, resilient and reliable testing method based on the Chaos Engineering (CE) in big data systems.

The research by [49] offers a comprehensive perspective on space systems and ground control operations, particularly focusing on the utilization of telemetry data mining with ML technology. An essential aspect discussed is the detection of anomalies in satellite operations, which involves identifying patterns in data that deviate from expected behavior. Various methods, including classification-based techniques, statistics-based approaches, aggregation, and swarm intelligence, are explored for identifying such patterns. ML and data mining techniques are widely applied across scientific and engineering domains for tasks like information processing, decision-making, and optimization.

In software engineering, canary deployment refers to the method of introducing an updated server into production gradually, by directing a small portion of user requests to it, followed by monitoring and analyzing predefined metrics, known as canary analysis, to assess the performance of the new software version. These metrics typically include measurements like request success rate or latency. The duration of the analysis phase varies depending on the deployment scale, ranging from seconds to days to gather sufficient information for decision-making. Research by [142] examines “Canary deployment” tools and assesses them based on their resource usage during deployment and their alignment with industry and academia requirements.

The development of large AI-systems based on big data is a very common subject of new programs in university courses. At Carnegie Mellon University, a course “Software Engineering for AI-Enabled Systems” has been developed that includes data science topics with a software-engineering perspective [65]. Lessons include analysis of requirements and system goals, with emphasis on the specification of AI-components, system architecture, including model performance (learning time, model size, etc.), implementation and operations, QA and production process.

After graduation, software engineers increasingly specialize in the positions of ML developers or data scientists. However, their lack of experience in working with large systems is often a big obstacle in finding their first job. Book by [122] provides a comprehensive guide to testing and experimenting ML systems in production. The book covers topics such as testing strategies, experimentation methods, monitoring techniques, debugging tools, and deployment practices. The book also includes case studies and examples from industry leaders.

Review paper by [120] presents findings from a systematic mapping study on functional testing techniques for ML systems, aimed at organizing and synthesizing literature

in this area, with the objective is to offer insights into the potential and accessibility of these techniques. Formulating 33 research questions, the study systematically collected and analyzed 70 relevant papers, examining the addressed problems, proposed approaches, and empirical evaluations. To ensure validity, repeatability, and reliability, a systematic protocol was designed for paper identification and classification, with detailed documentation of the research process and measures taken to address potential validity threats.

In [85] the authors try to find out a way to group customers in the data sample in order to achieve an optimal difference between the buckets, due to the fact that more on-line experiments have been done in E-Commerce in order to understand the behavior of users or customers and then apply the data analysis technique to provide business guidance with the use of A/B testing. Based on the analysis result of real data collected during joining an industry project, the authors conclude that the problem is complex and the meaningful conclusions have to be drawn with caution from business experiments such as A/B testing, due to the vast variation in the data.

Responsible engineering

Responsible engineering demands that ML systems be developed, tested and maintained in a manner that meets ethical and social standards. Responsible ML also promotes the development of ML systems that are not just highly accurate, but are robust, safe and secure; fair, inclusive, transparent; understandable and explainable; accountable and governable; and support sustainable, flourishing societies and ecosystems. [133] is a good reference on this topic—it describes two new software libraries designed to further increase the configurability, reproducibility and robustness of AI.

The nonprofit AlgorithmWatch counted nearly 200 ethics guidelines and reported that every principle (for transparency, equality/non-discrimination, accountability, and safety) is repeated more or less verbatim across them, with some of them additionally emphasising societal benefits and the protection of human rights [25].

Versioning, provenance, and reproducibility (V2R3) is concerned with tracking and maintaining control over the variants of ML models/code/data, their dependencies and lineage, and the ability to repeat or check their results. V2R3 properties are key to maintaining the integrity of ML systems in terms of debugging, auditing and (re)validation. Kästner [68] surveys the issues and techniques relevant to achieving each of these properties in practice for ML pipelines.

Interpretability and explain ability is the extent to which an ML model can provide humans with interpretable reasons for certain predictions and decisions. It helps build trust, transparency and accountability into an ML system, and also facilitates human oversight and intervention, as discussed in [90], a good overview of these techniques and examples.

Fairness reflects the degree to which ML systems can avoid or minimise biased and discriminatory judgments, particularly with respect to protected or sensitive attributes of individuals or groups, including race, gender, age, disability, etc. There are a number of different definitions or characterisations, and measures/metrics of this concept, as well as methods and tools designed to mitigate unfairness and promote equitable and just outcomes resulting from ML. For an introduction and overview of some of the concepts,

methods, and challenges of fairness in ML and its connections to other research areas and domains, see [154].

The issue of safety is associated with the guarantee that the ML system will not cause physical, mental or economic damage (intentionally or unintentionally) to humans or our environment, and its construct and testing will prevent or reduce failures, errors and risks, or reacts to uncertainty and adversaries [153]. Discussion of the issue of safety in ML, the definition and measure of how safety is determined in ML, and the problem and proposed solutions for ML to be safe and reliable.

Security and privacy is the guarantee that ML systems and data are protected from unauthorised access or alteration, or malicious misuse or disclosure; a form of confidentiality assurance whereby the correctness and completeness (or integrity) of the information and resources is maintained, and the information or resource is available or accessible when required (or availability of an ML system). Techniques include detecting and protecting ML systems from various types of attacks such as data poisoning, model stealing, membership inference and adversarial examples. Xu [159] provides an overview on ML privacy-preserving techniques and frameworks, as well as the open challenges and research directions related to this direction.

Transparency refers to the extent to how far users are informed that they are subject to an automated decision-making process, and to what extent they can evaluate how and why they have been treated the way they have. The dimension of accountability refers to the extent to which humans can oversee and repair an algorithm, and which persons and what institutions are held responsible in turn of errors. Kästner [71] provides an overview of the key principles and risks of transparency and accountability in ML, and the technological possibilities and means for achieving them.

Selected papers on responsible engineering

The papers overviewed in this category were extracted from a systematic search extended with snowballing, done through Google Scholar and arXiv, with a search criteria that fit the following boolean: ("ML" OR "machine learning" OR "big data" OR "AI") AND ("responsible engineering" OR "fairness" OR "ethics" or "safety"). Tables 7, 8 and 9 show the selected research papers and articles analyzed in this survey, which are related to the responsible engineering.

Ref. [133] introduces two new software libraries for enhancing the configurability, reproducibility, and robustness of AI systems: hydra-zen (it extends the Hydra framework to standardize the process of extending the configurability and reproducibility of complex AI applications) and the rAI-toolbox (designed to enable methods for evaluating and enhancing the robustness of AI models in a way that is scalable and that composes naturally with other popular ML frameworks). These two libraries address critical needs for responsible AI engineering.

Kästner [68] by Kästner provides a comprehensive overview of the issues and techniques for topic refers to the challenges and solutions for tracking and managing the changes, dependencies, and origins of ML models, code, and data, as well as ensuring the consistency and reliability of their results in ML pipelines. This is done through versioning, provenance, and reproducibility of the ML system, processes that enable

Table 7 An overview of responsible engineering references

Features/Ref. ID	Pub. date	Aim/goal	Target group				Research organization	Solution	Methodology
			ML dev	ML tst	RS	AISS			
[133]	2022	To provide industry process model for the development of ML apps with QA methodology	+	+	+		MIT Lincoln Laboratory	Software libraries	By introducing two software libraries that address critical needs for responsible AI engineering hydra-zen and the rAI-toolbox
[68]	2021	To provide a comprehensive overview of the issues and techniques for achieving Versioning, provenance, and reproducibility in ML pipelines	+	+	+		Carnegie Mellon University	Review article/survey	By defining and explaining the challenges and solutions for tracking and managing the changes, dependencies, origins of ML models, code, data and ensuring the consistency and reliability of results
[90]	2020	To review various approaches and examples of interpretable and explainable ML models			+		ETH Zurich	Review article/survey	An overview of evaluation methods of interpretability and explainability, which are then outlined through a partial taxonomy of techniques for interpretable and explainable ML
[153]	2016	To discuss the definition and measurement of safety in ML, as well as the challenges and solutions for ensuring safe and reliable ML systems	+		+		IBM Thomas J. Watson Research Center	Research article	By critically examining the statistical ML principles of empirical and structural risk minimization from the perspective of safety
[159]	2021	To systematically review and summarize existing privacy-preserving approaches and propose a triad based model to understand and guide the evaluation of various privacy-preserving ML solutions	+				IBM Research	Review article/survey	Evaluation of various PPML solution space by decomposing features of privacy-preserving approaches and discussing the privacy guarantees, and technical utilities
[107]	2020	To introduce the problem of fairness in ML and describe some techniques for addressing it	+		+		University of Pisa & DeepMind	Framework & Approach proposal	Presentation of a unified framework that encompasses methods that can deal with different settings and fairness criteria, and introducing an approach to learn fair representations that can generalize to unseen tasks

Legend: ML dev - ML developers, ML tst - ML testers, RS - researchers, AISS - AI system stakeholders

debugging, reliable experiments, safe deployments, safeguards against security attacks and forensics when attacks occur, auditing, and accountability.

In the paper by Marcinkevičs et al. [90], the authors examine the problem of designing interpretable and explainable ML models, which lie at the core of many ML and statistical applications in medicine, economics, law, and natural sciences. The authors discuss a need for interpretable and explainable ML techniques, giving examples from several application areas and give an overview of evaluation methods of interpretability and explainability, which are then outlined through a partial taxonomy of techniques for interpretable and explainable ML with examples of several recent advancements.

The paper [153] discusses the definition and measurement of safety in ML, as well as the challenges and solutions for ensuring safe and reliable ML systems. This is done by critically examining the foundational statistical ML principles of empirical risk minimization and structural risk minimization from the perspective of safety and examining safety in formulating ML problems. The author also provides a discussion of strategies to increase the safety of sociotechnical systems with ML components.

The authors of [159] provide a systematic review of existing privacy-preserving approaches and propose a Phase, Guarantee, and Utility (PGU) triad-based model to understand and guide the evaluation of various privacy-preserving ML (PPML) solutions by decomposing their privacy-preserving functionalities. The PPML pipeline is discussed from various phases of privacy-preserving functionalities that occur in the process-chain in these systems; these include privacy-preserving data preparation, privacy-preserving model training and evaluation, privacy-preserving model deployment, and privacy-preserving model inference. The technical utility of various PPML systems is also explored, with the authors providing a classification of existing PPML solutions: data publishing approaches, data processing approaches, architecture-based approaches and hybrid approaches.

In [107], the authors introduce the the problem of fairness in ML and describe some techniques for addressing it. The authors focus the presentation on the issues outlined above, and describe some approaches to address them, with them specifying the crucial role that Causal Bayesian Network (CBNs) when discussing fairness. The authors introduce a simple post-processing method that uses optimal transport theory to impose constraints on the full shapes of distributions corresponding to different sensitive attribute. They also describe a unified fairness framework that enjoys strong theoretical guarantees, as well as introduce a method to learn fair representations that can generalize to unseen tasks. A discussion of legal restrictions with the use of sensitive attributes is provided, as well as an introduction of an in-processing method that does not require them when using the model.

In the article by Kästner [71], a description of the concepts of Transparency of the ML system and Accountability is given, as well as an overview of the key concerns related to these topics. The author provides different examples where the users are not aware of the software systems' use of ML components to make decisions, and provides different ways of achieving greater transparency about how decisions are made. A discussion of Accountability and Culpability in case things go wrong is also provided.

The study presented in [83] advances the understanding of AI governance in the context of ML model development and thus contributes to the emerging body of IS

Table 8 An overview of responsible engineering references (continuation of Table 7)

Features/Ref. ID	Pub. date	Aim/goal	Target group				Research organization	Solution	Methodology
			ML dev	ML	tst	RS			
[71]	2022	To describe the concepts of transparency and accountability in ML systems, and provide an overview of key concerns	+	+	+	+	Carnegie Mellon University	Review article/survey	By describing different ways of how can the transparency of the model be presented to the user, with demonstrated examples, as well as overviewing different aspects of accountability and culpability
[83]	2022	To answer the question of what key issues and decisions ML development projects face during their life cycle with regards to AI governance	+			+	University of Turku	Research paper	By conducting a series of semi-structured interviews among high-profile experts involved in AI and software development, and elucidating key concepts related to AI governance and how they map to different stages of ML model development life cycle
[52]	2022	To propose a solution to the dispute of responsibility in case an AI system causes a harmful outcome	+				Syracuse University & Frankfurt School of Finance & Mng	Proposal of a new methodology	Illustrating what approaching the responsibility gap problem as a conceptual engineering problem looks like
[87]	2023	To operationalize responsible AI from a system perspective through presenting a Responsible AI Pattern Catalogue				+	CSIRO	Review article/survey	By performing a systematic Multivocal Literature Review to collect patterns and by following the high-level research question of what responsible AI solutions can be identified
[121]	2023	To examine the understandings and ethical risks of ML systems, by analyzing 2 safety engineering frameworks used in socio technical domains: Failure Mode and Effect Analysis and System Theoretic Process Analysis	+			+	McGill University, Google Research and Naval Postgraduate School	Research paper	By conducting 30 semi-structured in-depth interviews with industry practitioners who shared their current practices used to assess and mitigate social and ethical risks

Legend: ML dev - ML developers, ML tst - ML testers, RS - researchers, AI SS - AI system stakeholders

Table 9 An overview of responsible engineering references (continuation of Tables 7 and 8)

Features/Ref. ID	Pub. date	Aim/goal	Target group				Research organization	Solution	Methodology
			ML dev	ML tst	RS	AISS			
[124]	2024	To facilitate the implementation of well-rounded AI/ML systems that are appropriate for potential regulatory requirements	+	+			CSIRO	Framework	By proposing a framework which consists of: proactive identification of tensions, prioritisation and weighting of ethics aspects, justification and documentation of trade-off decisions
[95]	2024	To propose a unified Non-Idealized Responsible ML strategy	+		+		University of Toronto	Metho-dological framework	By interviewing 22 technically oriented ML practitioners across seven domains to understand the characteristics of their methodological approaches to Responsible ML through the lens of ideal and non-ideal theorizing of fairness and proposing a new methodological approach, inspired by the elements of non-ideal theory
[88]	2023	To adopt a pattern-oriented responsible artificial intelligence (RAI) engineering approach and build an RAI pattern catalog to operationalize RAI from a system perspective	+				CSIRO, Westpac Banking Group & Atlassian	Proposal of a new methodology	By summarizing the major challenges in operationalizing responsible artificial intelligence at scale and introduce how we use the responsible artificial intelligence pattern catalog to address those challenges
	2023	To offer a comprehensive survey of existing studies in the field of conducting fairness testing of ML software	+		+		University College London, King's College London & Simula Research Laboratory	Review article/survey	By collecting 100 papers and organize them based on the testing workflow (i.e., how to test) and testing components (i.e., what to test)

Legend: ML dev - ML developers, ML tst - ML testers, RS - researchers, AI SS - AI system stakeholders

literature on AI system design and development. This is done through the primary focus of the paper being to answer the research question of what key issues and decisions ML development projects face during their life cycle with regards to AI governance. The answer is provided by a series of semi-structured interviews among high-profile experts involved in AI and software development. The authors manage to elucidate key concepts with this approach related to AI governance and how they map to different stages of ML model development life cycle. This study presented in the paper set out to investigate the key technical steps and decisions in the system development life cycle of ML models that impact the AI governance of the resulting system and how technical AI governance aspects map to the life cycle of ML systems.

Himmelreich [52] proposes the idea that the dispute of responsibility in case an AI system causes a harmful outcome should be approached as a conceptual engineering problem, in order to have a more of a systematic understanding of why the concept is important in the first place.

The Multivocal Literature review presented [87] focuses on patterns that AI system stakeholders can implement to ensure that the developed AI systems are responsible throughout the entire governance and engineering lifecycle. The paper outlines a Responsible AI Pattern Catalogue with the following classification of patterns: multi-level governance patterns, trustworthy process patterns, and responsible-AI-by-design product patterns.

Rismani [121] recognizes the potential advantages of safety engineering frameworks and examines the understandings of social and ethical risks of ML systems. The authors report on ethical and social risk management practices currently used in the industry and take a developmental approach to examine how safety engineering frameworks can improve existing practices, with them focusing on two of the most successful safety engineering frameworks used in other socio technical domain Failure Mode and Effect Analysis (FMEA) and System Theoretic Process Analysis (STPA). The paper showcases the results of a conducted study with 30 semi-structured in-depth interviews with industry practitioners who shared their current practices used to assess and mitigate social and ethical risks. The authors introduced the two safety engineering frameworks, inviting them to envision how they might employ them to assess the ethical and social risks of ML systems.

The paper [124] proposes a framework with the aim of facilitating the implementation of ML/AI systems. It consists of: proactive identification of tensions, prioritization and weighting of ethics aspects, and justification and documentation of trade-off decisions. This is done due to the fact that the covered approaches shown in the paper are not likely to be appropriate for all organisations, systems, or applications, with them differing in scope, methods for measuring contexts, the types of considered context and degree of justification.

Mothilal [95] discusses the unclearness of various Responsible ML (RML) strategies. Through a conducted interview of 22 technically oriented ML practitioners across seven domains to understand the characteristics of their methodological approaches to RML through the lens of ideal and non-ideal theorizing of fairness. The authors find that practitioners' methodological approaches fall along a spectrum of idealization. Therefore, the authors discuss a new methodological approach, inspired by elements of non-ideal

theory, to structure technical practitioners' RML process and facilitate collaboration with other stakeholders.

Lu [88] asks the question of how developers can ensure that AI systems, such as Chat-GPT, are developed and adopted in a responsible way. The authors adopt a pattern-oriented responsible artificial intelligence (RAI) engineering approach and build an RAI pattern catalog to operationalize RAI from a system perspective. The authors summarize the major challenges in operationalizing RAI at scale and introduce how can the RAI pattern be used catalog to address those challenges. The risks are examined at each stage of the chatbot development process and recommend pattern-driven mitigations to evaluate the usefulness of the RAI pattern catalog in a real-world setting.

Chen et al. in [22] offer a comprehensive survey of existing studies in the field of conducting fairness testing of ML software, due to the rising attention and concern of software engineers on unfair behaviors of ML software. This is done by collecting 100 papers on this topic, and organizing them based on the testing workflow (i.e., how to test) and testing components (i.e., what to test). Research focus, trends and promising directions in the realm of fairness testing is also outlined and discussed. The paper also outlines widely adopted datasets and open-source tools for fairness testing.

Trends, challenges and opportunities in QA for ML big data systems

The trend of developing software systems based on machine learning models and software that includes big data analysis is visible in the last decade. Figure 3 shows the trend of the increase in the number of research papers on the topic of "ML system", "DL system" and "Big data analytics" (search through Google Scholar and arXiv, in the title and abstract of the paper). In Fig. 4, the trend of using key research terms in the titles of papers also shows an increase in papers based on ML- and DL-systems, as well as a significant increase in the publication of research papers in the field of big data. This diagram also shows the increasingly frequent use of the term "Quality Assurance". By searching for scientific papers that have both the terms "Machine learning" (or "Machine learning system") and "Big data analytics" in the paper title, there were 356 (search performed on March 20th, 2024), of which as many as 221 papers were published after 2020. On the other hand, 109 papers have been published that apply "Quality assurance" in their title in systems based on machine or deep learning, of which as many as 79 were scientific papers in the last 5 years. These data indicate the trend of developing software systems based on machine learning and big data is rising.

As seen in the previous section, the outlined papers were published in the previous few years (all papers shown in the taxonomy were published after 2015). This was done due to the fact that one of the aims of the paper was to show more modern approaches to ML testing. However, another conclusion that can be derived from this is the fact that this topic is very modern and relevant. A statistical chart can be seen in Fig. 5, which shows the number of papers published each year in the previous section. As seen in Fig. 5, more papers have been published in recent years, which verifies the rise of the importance of this topic. It can also be seen that the category of papers with the biggest rise in recent years is responsible engineering. This is because

LLMs, such as ChatGPT, have had a rise in popularity in more recent years, which made the question of maintaining responsibility in ML systems more important.

Out of the 50 papers outlined for the taxonomy, most have been surveys (17 papers), research papers (9 papers), or proposals of frameworks/methodologies/libraries (17 papers). Only 6 of the outlined papers have been done with some sort of experimental analysis, which shows that these analyses are the most lacking as QA strategies in ML system development with big data analytics.

We will cover challenging issues in QA for these kinds of systems from two perspectives: the big data perspective and the nature of ML algorithms perspective. A good overview of the challenges of testing and verifying Big Data systems, characterized by large volume, high velocity, and diverse data, is given in [54]. The main challenges are:

- Visualizing and explaining the results of complex and high-dimensional data processing requires finding the optimal dimension and resolution for the user to gain insights and validate the results.
- Handling the non-intuitive notion of consistency may vary depending on the trade-off between availability and performance in Big Data systems. The user may experience confusion or frustration when the data is not updated or synchronized across different requests.
- Determining the correctness of data processing, which involves many interdependent steps and computations. It may be hard to find precise test data or criteria to evaluate the correctness of an operation, especially when its impact on the overall result is small. Testing for plausibility may be the only feasible option.
- Meeting the high hardware requirements for testing, which implies replicating the workload and the environment of the real system. This may pose practical challenges such as providing enough storage space, parallelizing the processing, ensuring performance and scalability, etc.

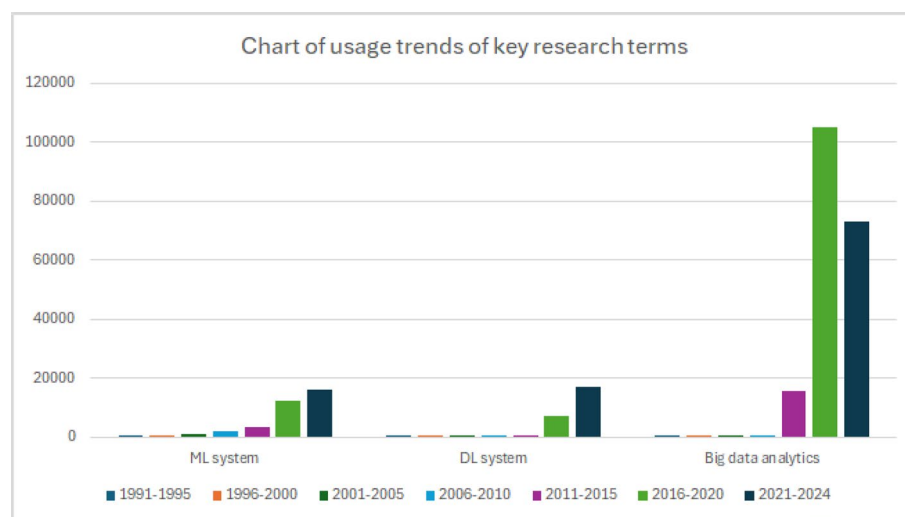


Fig. 3 Trend of used professional terms in scientific papers by years

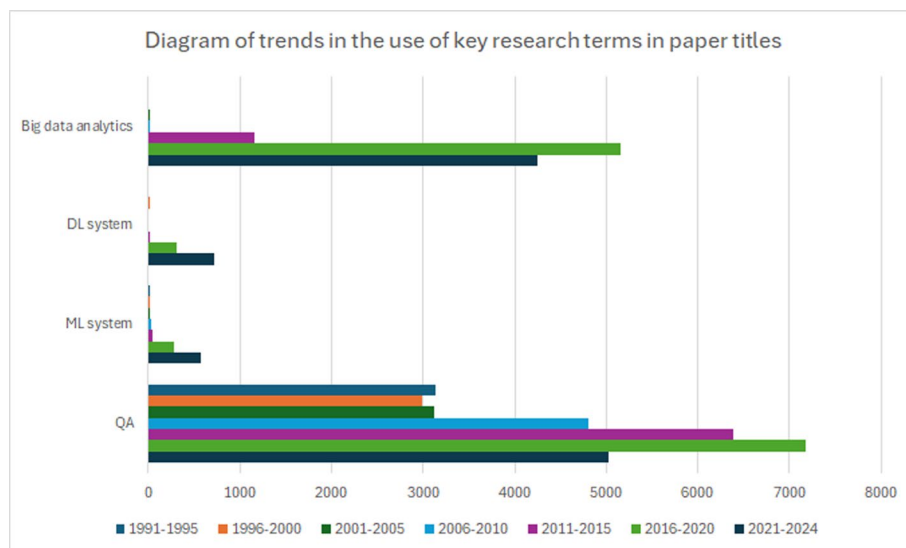


Fig. 4 Trend of used professional terms in the titles of scientific papers by year

- Generating adequate and high-quality test data, which reflects the realistic and application-specific characteristics of the data. This may be difficult to achieve for large and irregular data sets and may require a lot of storage and handling resources.
- Debugging, logging, and error-tracing the distributed data processing, which the limited capabilities of the current development tools and methods may hinder.
- Verifying the data processing, which may face a state explosion problem due to the use of growing clusters and the complexity of the computations.
- Ensuring the data quality is essential for the meaning and value of the data. This may be challenging to assess in the context of Big Data, especially when the data is aggregated or merged from different sources and types.

From the perspective of QA of general-purpose ML systems, a good overview of challenging issues is given in [166]:

- **Test Input Generation Challenges.** This task is still difficult due to the large and complex behavior of ML models. Current techniques mainly generate adversarial inputs to check the robustness of an ML system. However, these inputs are often unrealistic and do not reflect the actual input data. Therefore, a promising research direction is to explore how to generate natural test inputs and how to automatically evaluate their naturalness.
- **Test Assessment Criteria Challenges.** There is no systematic analysis of how different assessment metrics relate to each other, or how they relate to the tests' ability to reveal faults, which is a well-studied topic in conventional software testing.
- **Oracle Problem Challenges.** The oracle problem is still a hurdle in ML testing. Metamorphic relations (the expected change in the output based on the defined change in the input) are useful pseudo-oracles but they usually require manual definition. A

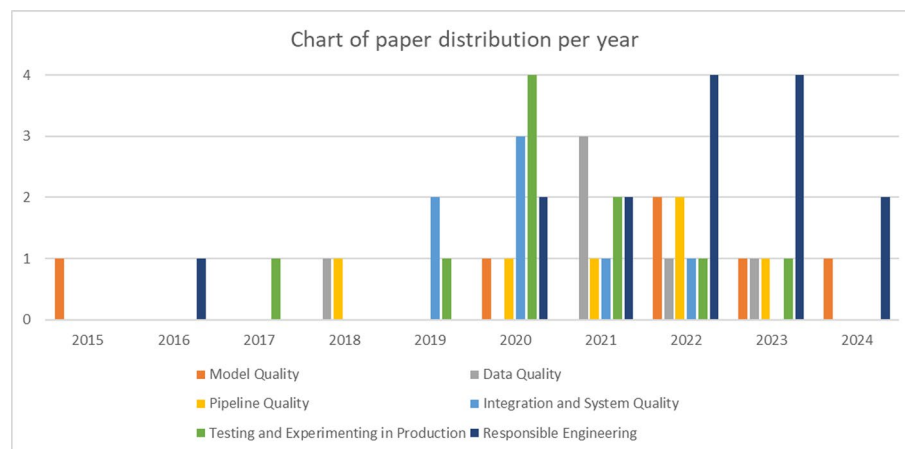


Fig. 5 Chart of paper distribution per year

remaining challenge is to automatically find and create trustworthy test oracles for ML testing.

- **Testing Other Properties Challenges.** Most research focuses on testing robustness and correctness, while few papers investigate efficiency, model relevance, or interpretability.
- **Testing Benchmarks Challenges.** Testing datasets are typically used for developing ML systems. There are hardly any benchmarks like CleverHans17 that are specifically designed for ML testing research goals, such as adversarial example generation.

Conclusion

In contrast to many research works that apply ML in the process of software testing and QA this review paper deals with the topic of applying QA techniques in the development of ML-applications, based on big data. The research is based on the analysis of the entire ML-pipeline for software development in big data analytics, and the analysis of QA techniques and methods that can be applied at each step of the development process, from requirements analysis to production, including the aspects of responsible engineering. Also, in each step it is explained how such a method can be applied and which team member should be in charge of that step.

In the context of discussing key research works and providing our insights and conclusions into those specific topics, this survey focused on three important approaches: (1) analysis of approaches in other survey papers that use QA techniques for the development of ML-applications; (2) analysis of techniques and methods that should be an integral part of the pipeline for the development of ML-applications in big data analytics; (3) proposal of a new taxonomy that analyzes each pipeline step, and based on significant research works in the existing literature, we identify recommendations and modern practices in ML-model analysis, data quality and pipeline analysis, integration process, production and responsible engineering. From (1), we concluded that there are several high-quality review papers, but that in that period the emphasis of research was not on

big data, so that domain was not included. Approach (2) gave us a very complex and meaningful pipeline for the development of ML-applications in big data analytics, composed of many steps, which led us to realize a new taxonomy (3), identify the key challenges in each step and present the possibilities of use QA for big data ML systems.

Trends show that an increasing number of software applications are being developed that have at least some domain of big data analysis or ML [54]. The guidelines and conclusions of our study can be useful to ML-developers, ML-testers, big data analytics, people dealing with data regulation, data privacy and compliance with ethical principles, researchers and others. The significance of this survey paper is multiple. Besides the usefulness for the software industry and all stakeholders involved in the development of modern ML-systems that use and process big data, the conclusions of the research can be a good basis for creating the curriculum of new university courses that combine software design and testing with ML, data science, data engineering and big data analytics. The direction of our further research is to do a broader analysis of the software tools used in the entire analyzed pipeline and thus give even greater benefit to the proposed taxonomy.

Acknowledgements

This research was supported by the Science Fund of the Republic of Serbia, Grant no. 11113, Software for Text Offences Prevention in Serbian: AI-driven Hate Speech Detection—STOP. This work was financially supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia under contract number: 451-03-65/2024-03/200103.

Author contributions

M.O. collected the data and processed it. D.D. collected the data and mapped it into a new categorization. D.B. defined the scope and the overall structure of the survey. All authors worked on problem analysis, designing a new classification and writing the first version of the manuscript.

Funding

This research was supported by the Science Fund of the Republic of Serbia, Grant no. 11113, Software for Text Offences Prevention in Serbian: AI-driven Hate Speech Detection—STOP. This work was financially supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia under contract number: 451-03-65/2024-03/200103.

Data availability

No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate

Not applicable.

Competing interests

The authors declare no competing interests.

Received: 2 May 2024 Accepted: 13 October 2024

Published online: 30 October 2024

References

1. Ackerman S, Barash G, Farchi E, Raz O, Shehory O. Theory and practice of quality assurance for machine learning systems. New York: Springer; 2024.
2. Alhijawi B, Awajan A. Genetic algorithms: theory, genetic operators, solutions, and applications. *Evol Intel*. 2023;3:1–2.
3. Amato G, Carrara F, Falchi F, Gennaro C, Meghini C, Vairo C. Deep learning for decentralized parking lot occupancy detection. *Expert Syst Appl*. 2017;15(72):327–34.
4. Antonioni E, Suriani V, Riccio F, Nardi D. Game strategies for physical robot soccer players: a survey. *IEEE Trans Games*. 2021;13(4):342–57.
5. Anwar SM, Majid M, Qayyum A, Awais M, Alnowami M, Khan MK. Medical image analysis using convolutional neural networks: a review. *J Med Syst*. 2018;42:1–3.

6. Arshad I, Alsamhi SH, Afzal W. Big data testing techniques: taxonomy, challenges and future trends. *Comput, Mater Continua*. 2023;74(2):2739–70.
7. Batarseh FA, Freeman L, Huang CH. A survey on artificial intelligence assurance. *J Big Data*. 2021;8(1):60.
8. Batanović V, Cvetanović M, Nikolić B. A versatile framework for resource-limited sentiment articulation, annotation, and analysis of short texts. *PLoS ONE*. 2020;15(11):e0242050.
9. Batmaz Z, Yurekli A, Bilge A, Kaleli C. A review on deep learning for recommender systems: challenges and remedies. *Artif Intell Rev*. 2019;1(52):1–37.
10. Beck J, Stern M, Haugsjaa E. Applications of AI in education. *XRDS: Crossroads, ACM Mag Stud*. 1996;3(1):11–5.
11. Blagojević V, Bojić D, Bojović M, Cvetanović M, Đorđević J, Đurđević Đ, Furlan B, Gajin S, Jovanović Z, Milićev D, Milutinović V. A systematic approach to generation of new ideas for PhD research in computing. Amsterdam: Elsevier; 2017. p. 1–31.
12. Boeschoten S, Catal C, Tekinerdogan B, Lommen A, Blokland M. The automation of the development of classification models and improvement of model quality using feature engineering techniques. *Expert Syst Appl*. 2023;1(213):118912.
13. Braiek HB, Khomh F. On testing machine learning programs. *J Syst Softw*. 2020;1(164):110542.
14. Breck E, Cai S, Nielsen E, Salib M, Sculley D. The ML test score: A rubric for ML production readiness and technical debt reduction. Piscataway: IEEE; 2017. p. 1123–32.
15. Budach L, Feuerpfeil M, Ihde N, Nathansen A, Noack N, Patzlaff H, Naumann F, Harmouch H. The effects of data quality on machine learning performance. *arXiv preprint*. 2022. <https://doi.org/10.4855/arXiv.2207.14529>.
16. Byun T, Sharma V, Vijayakumar A, Rayadurgam S, Cofer D. Input prioritization for testing neural networks. Piscataway: IEEE; 2019. p. 63–70.
17. Carlini N, Wagner D. Towards evaluating the robustness of neural networks. Piscataway: IEEE; 2017. p. 39–57.
18. Carroll JJ, Anand P, Guo D. Preproduction deploys: cloud-native integration testing. Piscataway: IEEE; 2021. p. 41–8.
19. Chen TY, Cheung SC, Yiu SM. Metamorphic testing: a new approach for generating next test cases. *arXiv preprint*. 2020. <https://doi.org/10.4855/arXiv.2002.12543>.
20. Chen H, Chen J, Ding J. Data evaluation and enhancement for quality improvement of machine learning. *IEEE Trans Reliab*. 2021;70(2):831–47.
21. Chen G, Bai G, Zhang C, Wang J, Ni K, Chen Z. Big data system testing method based on chaos engineering. Piscataway: IEEE; 2022. p. 210–5.
22. Chen Z, Zhang JM, Hort M, Harman M, Sarro F. Fairness testing: a comprehensive survey and analysis of trends. *ACM Trans Softw Eng Methodol*. 2024;33(5):1–59.
23. Cheng CH, Huang CH, Ruess H, Yasuoka H. Towards dependability metrics for neural networks. Piscataway: IEEE; 2018. p. 1–4.
24. Copeland BJ. Early AI in Britain: Turing et al. *IEEE Ann Hist Comput*. 2023. <https://doi.org/10.1109/MAHC.2023.3300660>.
25. Corrêa NK, Galvão C, Santos JW, Del Pino C, Pinto EP, Barbosa C, Massmann D, Mambrini R, Galvão L, Terem E, de Oliveira N. Worldwide AI ethics: a review of 200 guidelines and recommendations for AI governance. *Patterns*. 2023. <https://doi.org/10.1016/j.patter.2023.100857>.
26. Dai W, Yoshigoe K, Parsley W. Improving data quality through deep learning and statistical models. In: *Information Technology-New Generations: 14th International Conference on Information Technology 2018*. Springer International Publishing. pp. 515–522.
27. d'Aloisio G, Di Marco A, Stilo G. Modeling quality and machine learning pipelines through extended feature models. *arXiv preprint*. 2022. <https://doi.org/10.4855/arXiv.2207.07528>.
28. Da' u A, Salim N. Recommendation system based on deep learning methods: a systematic review and new directions. *Artif Intell Rev*. 2020;53(4):2709–48.
29. Delić V, Perić Z, Sečurjski M, Jakovljević N, Nikolić J, Mišković D, Simić N, Suzić S, Delić T. Speech technology progress based on new machine learning paradigm. *Comput Intell Neurosci*. 2019. <https://doi.org/10.1155/2019/4368036>.
30. Diaz-Gonzalez FA, Vuelas J, Correa CA, Vallejo VE, Patino D. Machine learning and remote sensing techniques applied to estimate soil indicators—review. *Ecol Ind*. 2022;1(135):108517.
31. Ding Z, Wang Y, Wang G, Zhang D, Kifer D. Detecting violations of differential privacy. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security 2018 Oct 15*. pp. 475–489.
32. Draskovic D, Cvetanovic M, Nikolic B. SAIL—Software system for learning AI algorithms. *Comput Appl Eng Educ*. 2018;26(5):1195–216.
33. Draskovic D, Zecevic D, Nikolic B. Development of a multilingual model for machine sentiment analysis in the Serbian language. *Mathematics*. 2022;10(18):3236.
34. Dutta S, Zhang W, Huang Z, Misailovic S. Storm: program reduction for testing and debugging probabilistic programming systems. In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering 2019 Aug 12*. pp. 729–739.
35. Dwarakanath A, Ahuja M, Sikand S, Rao RM, Bose RJ, Dubash N, Podder S. Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. In: *Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis 2018 Jul 12*. pp. 118–128.
36. Epstein Z, Hertzmann A, Akten M, Farid H, Fjeld J, Frank MR, Groh M, Herman L, Leach N, Mahari R. Art and the science of generative AI. *Science*. 2023;380(6650):1110–1.
37. Fabijan A, Dmitriev P, McFarland C, Vermeer L, Holmström Olsson H, Bosch J. Experimentation growth: evolving trustworthy A/B testing capabilities in online software companies. *J Softw: Evol Process*. 2018;30(12):e2113.
38. Fadlallah H, Kilany R, Dhayne H, El Haddad R, Haque R, Taher Y, Jaber A. Context-aware big data quality assessment: a scoping review. *ACM J Data Inform Quality*. 2023;15(3):1–33.
39. Feuerriegel S, Hartmann J, Janiesch C, Zschech P. Generative ai. *Bus Inf Syst Eng*. 2024;66(1):111–26.
40. Fui-Hoon Nah F, Zheng R, Cai J, Siau K, Chen L. Generative AI and ChatGPT: applications, challenges, and AI-human collaboration. *J Inform Technol Case Appl Res*. 2023;25(3):277–304.

41. Galhotra S, Brun Y, Meliou A. Fairness testing: testing software for discrimination. In: Proceedings of the 2017 11th Joint meeting on foundations of software engineering 2017 Aug 21. pp. 498–510.
42. Garcíarena U, Santana R, Mendiburu A. Analysis of the complexity of the automatic pipeline generation problem. *Piscataway: IEEE*; 2018. p. 1–8.
43. Gegic E, Isakovic B, Keco D, Masetic Z, Kevric J. Car price prediction using machine learning techniques. *TEM J*. 2019;8(1):113.
44. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. *Adv Neural Infor Process Syst*. 2014;27.
45. Grosse RB, Duvenaud DK. Testing mcmc code. *arXiv preprint*. 2014. <https://doi.org/10.4855/arXiv.1412.5218>.
46. Gudipati M, Rao S, Mohan ND, Gajja NK. Big data: testing approach to overcome quality challenges. *Big Data: Chall Oppor*. 2013;11(1):65–72.
47. Guo J, Jiang Y, Zhao Y, Chen Q, Sun J. Dfuzz: Differential fuzzing testing of deep learning systems. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering 2018 Oct 26. pp. 739–743.
48. Hale B, Van Bossuyt DL, Papakonstantinou N, O'Halloran B. A zero-trust methodology for security of complex systems with machine learning components. New York City: American Society of Mechanical Engineers; 2017.
49. Hassanien AE, Darwish A, Abdelghafar S. Machine learning in telemetry data mining of space mission: basics, challenging and future directions. *Artif Intell Rev*. 2020;53(5):3201–30.
50. Helskyaho H, Yu J, Yu K, Helskyaho H, Yu J, Yu K. Delivery and automation pipeline in machine learning. *Mach Learn Oracle Database Prof: Deploy Model-Driven Appl Autom Pipelines*. 2021. https://doi.org/10.1007/978-1-4842-7032-5_8.
51. Hernández-Serrato J, Velasco A, Nifio Y, Linares-Vásquez M. Applying machine learning with chaos engineering. *Piscataway: IEEE*; 2020. p. 151–2.
52. Himmelreich J, Köhler S. Responsible AI through conceptual engineering. *Philos Technol*. 2022;35(3):60.
53. Hsu FH. Behind deep blue: building the computer that defeated the world chess champion. Princeton: Princeton University Press; 2002.
54. Hummel O, Eichelberger H, Giloj A, Werle D, Schmid K. A collection of software engineering challenges for big data system development. *Piscataway: IEEE*; 2018. p. 362–9.
55. Hynes N, Sculley D, Terry M. The data linter: Lightweight, automated sanity checking for ml data sets. In: *NIPS ML Sys Workshop 2017 (Vol. 1, No. 5)*.
56. ISO IEC 9126:2001. Software engineering—product quality. International organization for standardization, international electrotechnical commission: Tech. rep. 2001.
57. ISO IEC 25000:2014. Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE. International organization for standardization, international electrotechnical commission: Standard. 2014.
58. Jernberg H, Runeson P, Engström E. Getting Started with Chaos Engineering—design of an implementation framework in practice. In: Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) 2020 Oct 5. pp. 1–10.
59. Ji S, Pan S, Li X, Cambria E, Long G, Huang Z. Suicidal ideation detection: a review of machine learning methods and applications. *IEEE Trans Comput Soc Syst*. 2020;8(1):214–26.
60. Ji S, Li Q, Cao W, Zhang P, Muccini H. Quality assurance technologies of big data applications: a systematic literature review. *Appl Sci*. 2020;10(22):8052.
61. Jovicic V, Marinkovic M, Stojanovic S, Nikolic B. Automated assessment of pen and paper tests using computer vision. *Multimed Tools Appl*. 2024;83(1):2031–52.
62. Kaliyar RK, Goswami A, Narang P. FakeBERT: fake news detection in social media with a BERT-based deep learning approach. *Multimed Tools Appl*. 2021;80(8):11765–88.
63. Kang D, Raghavan D, Bailis P, Zaharia M. Model assertions for monitoring and improving ML models. *Proc Mach Learn Syst*. 2020;15(2):481–96.
64. Karpathy, A. CS231n Convolutional Neural Networks for Visual Recognition. <http://cs231n.github.io/neural-networks-3/>. 2018.
65. Kästner C, Kang E. Teaching software engineering for AI-enabled systems. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training 2020 Jun 27. pp. 45–48.
66. Kästner, C. Data quality for building production ML systems. Medium. <https://ckaestne.medium.com/data-quality-for-building-production-ml-systems-2e0cc7e6113f>. 2021.
67. Kästner, C. Machine learning testing and experimenting in production. Medium. <https://ckaestne.medium.com/quality-assurance-in-production-for-ml-enabled-systems-4d1b3442316f>. 2021.
68. Kästner, C. Versioning, provenance, and reproducibility in production machine learning. Medium. <https://ckaestne.medium.com/versioning-provenance-and-reproducibility-in-production-machine-learning-355c48665005>. 2021.
69. Kästner, C. Quality assurance for machine-learning pipelines. Medium. <https://ckaestne.medium.com/quality-assurance-for-machine-learning-pipelines-d495b8e5ad6a>. 2022.
70. Kästner, C. System quality. Medium. <https://ckaestne.medium.com/integration-and-system-testing-bc4db6650d1>. 2022.
71. Kästner, C. Transparency and accountability in ML-enabled systems. Medium. <https://ckaestne.medium.com/transparency-and-accountability-in-ml-enabled-systems-f8ed0b6fd183>. 2022.
72. Kästner, C. Model quality. Medium. <https://ckaestne.medium.com/model-quality-d654cea6944b>. 2024.
73. Kaul V, Enslin S, Gross SA. History of artificial intelligence in medicine. *Gastrointest Endosc*. 2020;92(4):807–12.
74. Kavakiotis I, Tsav E, Salifoglou A, Maglaveras N, Vlahavas I, Chouvarda I. Machine learning and data mining methods in diabetes research. *Comput Struct Biotechnol J*. 2017;1(15):104–16.
75. Kharitonov A, Nahhas A, Pohl M, Turowski K. Comparative analysis of machine learning models for anomaly detection in manufacturing. *Proc Comput Sci*. 2022;1(200):1288–97.

76. Kim J, Feldt R, Yoo S. Guiding deep learning system testing using surprise adequacy. Piscataway: IEEE; 2019. p. 1039–49.
77. Kirk M. Thoughtful machine learning: a test-driven approach. Sebastopol: O'Reilly Media, Inc.; 2014.
78. Knez T, Žitnik S. Multimodal learning for temporal relation extraction in clinical texts. *J Am Med Inform Assoc*. 2024. <https://doi.org/10.1093/jamia/ocae059>.
79. Kohavi R, Longbotham R, Sommerfield D, Henne RM. Controlled experiments on the web: survey and practical guide. *Data Min Knowl Disc*. 2009;18:140–81.
80. Krishnan S, Wang J, Franklin MJ, Goldberg K, Kraska T, Milo T, Wu E. SampleClean: fast and reliable analytics on dirty data. *IEEE Data Eng Bull*. 2015;38(3):59–75.
81. Kuutti S, Bowden R, Jin Y, Barber P, Fallah S. A survey of deep learning applications to autonomous vehicle control. *IEEE Trans Intell Transp Syst*. 2020;22(2):712–33.
82. Kuwajima H, Yasuoka H, Nakae T. Engineering problems in machine learning systems. *Mach Learn*. 2020;109(5):1 103–26.
83. Laato S, Birkstedt T, Mäntymäki M, Minkinen M, Mikkonen T. AI governance in the system development life cycle: Insights on responsible machine learning engineering. In: *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI 2022 May 16*. pp. 113–123.
84. Li N, Lei Y, Khan HR, Liu J, Guo Y. Applying combinatorial test data generation to big data applications. In: *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering 2016 Aug 25*. pp. 637–647.
85. Liu D, Xu S, Zhang B, Wang C, Li C, Zhou F. Issues with conducting controlled on-line experiments for e-commerce. Piscataway: IEEE; 2017. p. 187–92.
86. Lorenz F, Willwersch J, Cajias M, Fuerst F. Interpretable machine learning for real estate market analysis. *Real Estate Econ*. 2023;51(5):1178–208.
87. Lu Q, Zhu L, Xu X, Whittle J, Zowghi D, Jacquet A. Responsible ai pattern catalogue: A collection of best practices for ai governance and engineering. *ACM Comput Surveys*. 2023.
88. Lu Q, Luo Y, Zhu L, Tang M, Xu X, Whittle J. Developing responsible chatbots for financial services: a pattern-oriented responsible AI engineering approach. *IEEE Intell Syst*. 2023. <https://doi.org/10.4855/arXiv.2301.05517>.
89. Ma L, Zhang F, Sun J, Xue M, Li B, Juefei-Xu F, Xie C, Li L, Liu Y, Zhao J, Wang Y. Deepmutation: mutation testing of deep learning systems. Piscataway: IEEE; 2018. p. 100–11.
90. Marcinkevičs R, Vogt JE. Interpretability and explainability: a machine learning zoo mini-tour. *arXiv preprint*. 2020. <https://doi.org/10.4855/arXiv.2012.01805>.
91. Marsland TA. A short history of computer chess. In: Anthony Marsland T, Schaeffer J, editors. *Computers, chess, and cognition*. New York: Springer; 1990. p. 3–7.
92. McClure N. TensorFlow machine learning cookbook. Birmingham: PACKT publishing Ltd.; 2017.
93. Mehrotra T, Rajput GK, Verma M, Lakhani B, Singh N. Email spam filtering technique from various perspectives using machine learning algorithms. In: Singh TP, Tomar R, Choudhury T, Perumal T, Mahdi HF, editors. *Data driven approach towards disruptive technologies: proceedings of MIDAS 2020*. Singapore: Springer; 2021.
94. Moor J, editor. *The turing test: the elusive standard of artificial intelligence*. Dordrecht: Springer Science & Business Media; 2003.
95. Mothilal RK, Guha S, Ahmed SI. Towards a non-ideal methodological framework for responsible ML. *arXiv preprint*. 2024. <https://doi.org/10.4855/arXiv.2401.11131>.
96. Mroz T, Griffin M, Cartabuke R, Laffin L, Russo-Alvarez G, Thomas G, Smedira N, Meese T, Shost M, Habboub G. Predicting hypertension control using machine learning. *PLoS ONE*. 2024;19(3):e0299932.
97. Murillo-Morera J, Quesada-López C, Castro-Herrera C, Jenkins M. A genetic algorithm based framework for software effort prediction. *J Softw Eng Res Dev*. 2017;5:1–33.
98. Murphy C, Kaiser GE, Hu L. Properties of machine learning applications for use in metamorphic testing.
99. Najafabadi MM, Villanustre F, Khoshgoftaar TM, Seliya N, Wald R, Muharemagic E. Deep learning applications and challenges in big data analytics. *J Big Data*. 2015;2:1–21.
100. Nie Y, Wang Y, Bansal M. Analyzing compositionality-sensitivity of NLI models. In *Proceedings of the AAAI conference on artificial intelligence 2019 Jul 17, Vol. 33, No. 01*. pp. 6867–6874.
101. Nishi Y, Masuda S, Ogawa H, Uetsuki K. A test architecture for machine learning product. Piscataway: IEEE; 2018. p. 273–8.
102. Nosek T, Suzić S, Vujović M, Pekar D, Sečurjski M, Delić V. Explicit control of the level of expressiveness in DNN-based speech synthesis by embedding interpolation. St. Petersburg: Springer International Publishing; 2021. p. 472–82.
103. Nti IK, Quarcoo JA, Aning J, Fosu GK. A mini-review of machine learning in big data analytics: applications, challenges, and prospects. *Big Data Min Anal*. 2022;5(2):81–97.
104. Nurseitov D, Bostanbekov K, Kanatov M, Alimova A, Abdallah A, Abdimanap G. Classification of handwritten names of cities and handwritten text recognition using various deep learning models. *ArXiv preprint*. 2021. <https://doi.org/10.2504/aj0505114>.
105. Odena A, Olsson C, Andersen D, Goodfellow I. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning 2019 May 24*. PMLR. pp. 4901–4911.
106. Oh S. Feature interaction in terms of prediction performance. *Appl Sci*. 2019;9(23):5191.
107. Oneto L, Chiappa S. Fairness in machine learning. In: Oneto L, Navarin N, Sperduti A, Anguita D, editors. *Recent trends in learning from data: tutorials from the inns big data and deep learning conference (innsbddl2019)*. Cham: Springer International Publishing; 2020. p. 155–96.
108. Papineni K, Roukos S, Ward T, Zhu WJ. Bleu: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics 2002 Jul*. pp. 311–318.
109. Pei K, Cao Y, Yang J, Jana S. Deepxplore: Automated whitebox testing of deep learning systems. In: *proceedings of the 26th Symposium on Operating Systems Principles 2017 Oct 14*. pp. 1–18.
110. Pham HV, Lutellier T, Qi W, Tan L. CRADLE: cross-backend validation to detect and localize bugs in deep learning libraries. Piscataway: IEEE; 2019. p. 1027–38.

111. Polyzotis N, Zinkevich M, Roy S, Breck E, Whang S. Data validation for machine learning. *Proc Mach Learn Syst*. 2019;15(1):334–47.
112. Poth A, Meyer B, Schlicht P, Riel A. Quality assurance for machine learning—an approach to function and system safeguarding. *Piscataway: IEEE*; 2020. p. 22–9.
113. Puttagunta M, Ravi S. Medical image analysis based on deep learning approach. *Multimed Tools Appl*. 2021;80(16):24365–98.
114. Qi Z, Wang H, Li J, Gao H. Impacts of dirty data: and experimental evaluation. *arXiv preprint*. 2018. <https://doi.org/10.4855/arXiv.1803.06071>.
115. Qian B, Rasheed K. Stock market prediction with multiple classifiers. *Appl Intell*. 2007;26:25–33.
116. Qin Y, Wang H, Xu C, Ma X, Lu J. Syneva: evaluating ml programs by mirror program synthesis. *Piscataway: IEEE*; 2018. p. 171–82.
117. Qorich M, El Ouazzani R. Text sentiment classification of Amazon reviews using word embeddings and convolutional neural networks. *J Supercomput*. 2023;79(10):11029–54.
118. Ramanathan A, Pullum LL, Hussain F, Chakrabarty D, Jha SK. Integrating symbolic and statistical methods for testing intelligent systems: applications to machine learning and computer vision. *Piscataway: IEEE*; 2016. p. 786–91.
119. Ren X, Li X, Ren K, Song J, Xu Z, Deng K, Wang X. Deep learning-based weather prediction: a survey. *Big Data Res*. 2021;15(23):100178.
120. Riccio V, Jahangirova G, Stocco A, Humbatova N, Weiss M, Tonella P. Testing machine learning based systems: a systematic mapping. *Empir Softw Eng*. 2020;25:5193–254.
121. Rismani S, Shelby R, Smart A, Jatho E, Kroll J, Moon A, Rostamzadeh N. From plane crashes to algorithmic harm: applicability of safety engineering frameworks for responsible ML. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems 2023 Apr 19*. pp. 1–18.
122. Robinson E, Nolis J. Build a career in data science. Shelter Island: Manning Publications; 2020.
123. Saleh H, Alhothali A, Moria K. Detection of hate speech using bert and hate speech word embedding with deep model. *Appl Artif Intell*. 2023;37(1):2166719.
124. Sanderson C, Schleiger E, Douglas D, Kuhnert P, Lu Q. Resolving ethics trade-offs in implementing responsible AI. *arXiv preprint*. 2024. <https://doi.org/10.1109/CAI59869.2024.00215>.
125. Satyal S, Weber I, Paik HY, Di Ciccio C, Mendling J. Shadow testing for business process improvement. In: Panetto H, Debruyne C, Proper HA, Ardagna CA, Roman D, Meersman R, editors. *On the move to meaningful internet systems. OTM 2018 conferences: confederated international conferences: CoopIS, C&TC, and ODBASE 2018*, Valletta, Malta, October 22–26, 2018, *Proceedings, Part I*. Cham: Springer International Publishing; 2018.
126. Schaeffer J, Van den Herik HJ. Games, computers, and artificial intelligence. *Artif Intell*. 2002;134(1–2):1–7.
127. Selsam D, Liang P, Dill DL. Developing bug-free machine learning systems with formal mathematics. In: *International Conference on Machine Learning 2017 Jul 17*. PMLR. pp. 3047–3056.
128. Sharma N, Sharma P, Irwin D, Shenoy P. Predicting solar generation from weather forecasts using machine learning. *Piscataway: IEEE*; 2011. p. 528–33.
129. Sherin S, Iqbal MZ. A systematic mapping study on testing of machine learning programs. *arXiv preprint*. 2019. <https://doi.org/10.4855/arXiv.1907.09427>.
130. Shu LQ, Sun YK, Tan LH, Shu Q, Chang AC. Application of artificial intelligence in pediatrics: past, present and future. *World J Pediatrics*. 2019;1(15):105–8.
131. Siebert J, Joeckel L, Heidrich J, Nakamichi K, Ohashi K, Namba I, Yamamoto R, Aoyama M. Towards guidelines for assessing qualities of machine learning systems. Cham: Springer International Publishing; 2020. p. 17–31.
132. Siebert J, Joeckel L, Heidrich J, Trendowicz A, Nakamichi K, Ohashi K, Namba I, Yamamoto R, Aoyama M. Construction of a quality model for machine learning systems. *Softw Qual J*. 2022;30(2):307–35.
133. Soklaski R, Goodwin J, Brown O, Yee M, Matterer J. Tools and practices for responsible AI engineering. *arXiv preprint*. 2022. <https://doi.org/10.4855/arXiv.2201.05647>.
134. Song Q, Borg M, Engström E, Ardö H, Rico S. Exploring ML testing in practice: Lessons learned from an interactive rapid review with axis communications. In: *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI 2022 May 16*. pp. 10–21.
135. Sparks ER, Talwalkar A, Haas D, Franklin MJ, Jordan MI, Kraska T. Automating model search for large scale machine learning. In: *Proceedings of the Sixth ACM Symposium on Cloud Computing 2015 Aug 27*. pp. 368–380.
136. Spreeuwers LJ, Hendrikse AJ, Gerritsen KJ. Evaluation of automatic face recognition for automatic border control on actual data recorded of travellers at Schiphol Airport. *Piscataway: IEEE*; 2012. p. 1–6.
137. Srisakaokul S, Wu Z, Astorga A, Alebiosu O, Xie T. Multiple-implementation testing of supervised learning software. In: *Workshops at the thirty-second AAAI conference on artificial intelligence 2018 Jun 20*.
138. Srivastava PR, Kim TH. Application of genetic algorithm in software testing. *Int J Softw Eng Appl*. 2009;3(4):87–96.
139. Stanojević M, Drašković D, Nikolić B. Retinal disease classification based on optical coherence tomography images using convolutional neural networks. *J Electron Imaging*. 2023;32(3):032004.
140. Steidl M, Felderer M, Ramler R. The pipeline for the continuous development of artificial intelligence models—current state of research and practice. *J Syst Softw*. 2023;1(199):111615.
141. Stojanovic L, Dinic M, Stojanovic N, Stojadinovic A. Big-data-driven anomaly detection in industry (4.0): an approach and a case study. *Piscataway: IEEE*; 2016. p. 1647–52.
142. Strasser N. An evaluation of canary deployment tools. Doctoral dissertation, University of Applied Sciences; 2023.
143. Studer S, Bui TB, Drescher C, Hanuschkin A, Winkler L, Peters S, Müller KR. Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology. *Mach Learn Knowl Extr*. 2021;3(2):392–413.
144. Sugali K. Software testing: Issues and challenges of artificial intelligence & machine learning.
145. Sun Z, Zhang JM, Harman M, Papadakis M, Zhang L. Automatic testing and improvement of machine translation. In: *Proceedings of the ACM/IEEE 42nd international conference on software engineering 2020 Jun 27*. pp. 974–985.
146. Tahiru F. AI in education: A systematic literature review. *J Cases Inform Technol (JCIT)*. 2021;23(1):1–20.

147. Taleb I, Serhani MA, Bouhaddioui C, Dssouli R. Big data quality framework: a holistic approach to continuous quality management. *J Big Data*. 2021;8(1):76.
148. Tesfagiorgish DG, JunYi L. Big data transformation testing based on data reverse engineering. Piscataway: IEEE; 2015. p. 649–52.
149. Thung F, Wang S, Lo D, Jiang L. An empirical study of bugs in machine learning systems. Piscataway: IEEE; 2012. p. 271–80.
150. Tian Y, Pei K, Jana S, Ray B. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: Proceedings of the 40th international conference on software engineering 2018 May 27. pp. 303–314.
151. [Toslati2021] Toslati M, Parthasarathy S, Oliveira F, Huang H, Coskun AK. Iter8: Online experimentation in the cloud. In: Proceedings of the ACM Symposium on Cloud Computing 2021 Nov 1. pp. 289–304.
152. Vajs IA, Kvaščev GS, Papić TM, Janković MM. Eye-tracking image encoding: Autoencoders for the crossing of language boundaries in developmental dyslexia detection. *IEEE Access*. 2023;5(11):3024–33.
153. Varshney KR. Engineering safety in machine learning. Piscataway: IEEE; 2016. p. 1–5.
154. Verma S, Rubin J. Fairness definitions explained. In: Proceedings of the international workshop on software fairness 2018 May 29. pp. 1–7.
155. Wagner S, Goeb A, Heinemann L, Kläs M, Lampasona C, Lochmann K, Mayr A, Plösch R, Seidl A, Streit J, Trendowicz A. Operationalised product quality models and assessment: the Quamoco approach. *Inf Softw Technol*. 2015;1(62):101–23.
156. Wegener J, Bühler O. Evaluation of different fitness functions for the evolutionary testing of an autonomous parking system. In: Deb K, editor. Genetic and evolutionary computation—GECCO 2004: genetic and evolutionary computation conference, Seattle, WA, USA, June 26–30, 2004. Proceedings, part II. Berlin: Springer; 2004. p. 1400–12.
157. Werpachowski R, György A, Szepesvári C. Detecting overfitting via adversarial examples. *Adv Neural Infor Process Syst*. 2019;32.
158. Wu JL, Tang XR, Hsu CH. A prediction model of stock market trading actions using generative adversarial network and piecewise linear representation approaches. *Soft Comput*. 2023;27(12):8209–22.
159. Xu R, Baracaldo N, Joshi J. Privacy-preserving machine learning: methods, challenges and directions. arXiv preprint. 2021. <https://doi.org/10.4855/arXiv.2108.04417>.
160. Yaghoubi S, Fainekos G. Gray-box adversarial testing for control systems with machine learning components. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control 2019 Apr 16. pp. 179–184.
161. Yang B, Sailer A, Jain S, Tomala-Reyes AE, Singh M, Ramnath A. Service discovery based blue-green deployment technique in cloud native environments. Piscataway: IEEE; 2018. p. 185–92.
162. Yang B, Sailer A, Mohindra A. Survey and evaluation of blue-green deployment techniques in cloud native environments. In: Yangui S, Bouguettaya A, Xue X, Faci N, Gaaloul W, Qi Y, Zhou Z, Hernandez N, Nakagawa EY, editors. Service-oriented computing—ICSOC 2019 workshops: WESOACS, ASOCA, ISYCC, TBCE, and STRAPS, Toulouse, France, October 28–31, 2019, Revised Selected Papers. Cham: Springer International Publishing; 2019. p. 69–81.
163. Yessenov K, Misailovic S. Sentiment analysis of movie review comments. *Methodology*. 2009;17(17):1–7.
164. Yesudas M, S GM, Nair SK. High-volume performance test framework using big data. In: Proceedings of the 4th International Workshop on Large-Scale Testing 2015 Feb 1. pp. 13–16.
165. Zhang M, Zhang Y, Zhang L, Liu C, Khurshid S. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering 2018 Sep 3. pp. 132–142.
166. Zhang JM, Harman M, Ma L, Liu Y. Machine learning testing: survey, landscapes and horizons. *IEEE Trans Software Eng*. 2020;48(1):1–36.
167. Zhou Y, Yu Y, Ding B. Towards mlops: a case study of ml pipeline platform. Piscataway: IEEE; 2020. p. 494–500.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.