The Journal of Systems and Software 209 (2024) 111907

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Software engineering practices for machine learning — Adoption, effects, and team assessment☆

Alex Serban [a,*], Koen van der Blom [a,b], Holger Hoos [a,c,d], Joost Visser [a]

[a] Leiden University, The Netherlands
[b] Sorbonne Université, France
[c] RWTH Aachen University, Germany
[d] University of British Columbia, Canada

ARTICLE INFO

Keywords:
Software engineering
Machine learning
Engineering practices
Maturity Models
Artificial Intelligence

ABSTRACT

Machine learning (ML) is extensively used in production-ready applications, calling for mature engineering techniques to ensure robust development, deployment and maintenance. Given the potential negative impact machine learning (ML) can have on people, society or the environment, engineering techniques that can ensure robustness against technical errors and adversarial attacks are of considerable importance. In this work, we investigate how teams of experts develop, deploy and maintain software with ML components. Moreover, we link what teams do to the effects they aim to achieve and provide means for improvement. Towards this goal, we performed a mixed-methods study with a sequential exploratory strategy. First, we performed a systematic literature review through which we mined both academic and grey literature, and compiled a catalogue of engineering practices for ML. Second, we validated this catalogue using a large-scale survey, which measured the degree of adoption of the practices and their perceived effects. Third, we ran validation interviews with practitioners to add depth to the survey results. The catalogue covers a broad range of practices for engineering software systems with ML components and for ensuring non-functional properties that fall under the umbrella of trustworthy ML, such as fairness, security or accountability. Here, we present the results of our study, which indicate, for example, that larger and more experienced teams tend to adopt more practices, but that trustworthiness practices tend to be neglected. Moreover, we show that the effects measured in our survey, such as team agility or accountability, can be predicted quite accurately from groups of practices. This allowed us to contrast the importance of the practices for these effects as well as adoption rates, revealing, for example, that widely adopted practices are, in reality, less important with respect to some effects. For instance, writing reusable scripts for data cleaning and merging is highly adopted, but has a limited impact on reproducibility. Overall, our study provides a quantitative assessment of ML engineering practices and their impact on desirable properties of software with ML components, by which we open multiple avenues for improving the adoption of useful practices.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

1. Introduction

Machine learning (ML) techniques are used across a broad range of commercial and non-commercial applications. To reap the benefits of ML components in production-ready applications, engineering methods that ensure robust development, deployment and maintenance must be applied, or new methods tailored to ML must be developed. Likewise, since ML components pose inherent threats to users or the environment in mission and safety critical applications, engineering methods that ensure robustness against a wide range of technical and societal factors must be adopted.

Because the field of ML is evolving rapidly, with new methods, tools and experience reports being published daily, practitioners face a challenging task when disseminating and selecting useful methods or practices for their projects. Moreover, broad empirical evidence about the relevance of the available methods across domains and organisations is missing. We aim to disseminate, validate, organise and systematise practical engineering guidance for the development of soft-

☆ Editor: Dr. Nicole Novielli.

* Corresponding author at: Leiden University, The Netherlands.
E-mail address: serban.alexc@gmail.com (A. Serban).

https://doi.org/10.1016/j.jss.2023.111907
Received 21 October 2022; Received in revised form 5 September 2023; Accepted 19 November 2023
Available online 28 November 2023

ware with ML components. Further, we aim to collect and disseminate empirical evidence for the importance of engineering practices across domains and organisations, and publish a comprehensive data set that can be used to develop instruments that help teams improve the quality of their engineering methods. We also discuss promising directions for developing quality models using the published data set, and introduce an exploratory use-case in which the data set is used to assess the engineering ability of teams developing software with ML components.

In this work, we present the results from a series of two studies that collect, validate and measure the adoption of engineering best practices for ML. In both studies, we used multi-vocal literature reviews to compile engineering best practices from academic and non-academic literature; we also ran interviews and surveys with practitioners to validate the relevance of the practices we have identified, to measure adoption and to assess the effects of adopting these practices. The first study focused on engineering practices for the entire ML development life-cycle (Serban et al., 2020c), while the second study complemented the first with a focus on engineering practices for trustworthy ML (Serban et al., 2021).

We now extend the two previous studies (Serban et al., 2020c, 2021) with additional interviews, as well as a joint analysis of the results, to yield novel insights. The disjoint previous studies did not allow for a comprehensive and broad treatment, as is needed for the field of software engineering and machine learning. Moreover, in this paper we introduce the complete catalogue of practices and draw conclusions from it, which was not possible in the independent evaluation of the previous two studies. Here, we conducted additional validation interviews to complement the quantitative results of the previous studies with a qualitative practitioners' perspective. The joint analysis allows us to present novel results including (i) a deep, comprehensive, catalogue of software engineering practices for ML discussed and validated as one rather than independently on parts, (ii) improved, more reliable statistical results, and (iii) a comprehensive data set that can be used to work towards assessment methods for teams developing software with ML components. We also discuss promising directions and use-cases that can help the development of such instruments.

In detail, the main contributions of our work are as follows. First, we summarise academic and non-academic literature on the topic of software engineering (SE) for ML in a comprehensive catalogue of best practices that cover the entire ML development life-cycle and includes highly relevant topics such as robustness, fairness and AutoML (the latter also being novel to this study). This catalogue can be used by practitioners to assess and improve their development process, and as a gateway to related literature on the topic. Second, we measure the adoption of the practices and their effects by surveying more than 400 practitioners. The adoption rates allow us to rank the practices and uncover relationships between them. Third, we measure the effects of adopting the practices and discover linear and non-linear relationships between groups of practices and beneficial effects. These relationships allow us to quantify the contribution of the practices to the effects, which together with the adoption rate can be interpreted as a cost–benefit analysis of adopting practices for a desired effect. Moreover, by jointly analysing the results of the previous studies, these relationships are now statistically more reliable and accurate. Last, based on the published data, we discuss the future development of instruments to assess the quality and engineering ability of teams of practitioners developing software with ML components.

The remainder of this article is organised as follows. We start by discussing background information and related work (Section 2). Next, we describe the study design (Section 3) and introduce the catalogue of engineering practices for ML (Section 4) followed by an analysis of the practice adoption (Section 5), practice ranking (Section 6) and an analysis of practices and effects (Section 7). Afterwards, we zoom in on individual sets of practices (Section 8) and discuss the future development of instruments for assessing the quality of teams developing software with ML components (Section 9). We end with a high-level discussion of our findings (Section 10) and draw a number of conclusions (Section 11). All data and code for analysis, visualisation and reproduction of the experiments presented are publicly available (Serban et al., 2020a).

2. Background and related work

The literature on SE for ML can be broadly classified into (i) *challenges* raised by the adoption of ML components (e.g., Arpteg et al., 2018; Ishikawa and Yoshioka, 2019; Lwakatare et al., 2019; Bosch et al., 2021; Humbatova et al., 2020; Lenarduzzi et al., 2021; Kumeno, 2019; Muccini and Vaidhyanathan, 2021), (ii) *solutions to challenges*, such as engineering practices, guidelines or experiences with SE for ML (e.g., Sculley et al., 2015; Breck et al., 2017; Zhang et al., 2020; Weiss and Tonella, 2021; Amershi et al., 2019) and (iii) SE *quality models* for ML (e.g., Breck et al., 2017; Akkiraju et al., 2020; Kuwajima et al., 2020).

Arpteg et al. (2018) defined a broad set of engineering challenges for development, deployment and organisational issues that arise from the adoption of ML components. In particular, they observed that versioning artefacts related to ML, monitoring deployed models and testing ML components are particularly hard challenges that remain to be solved. Ishikawa and Yoshioka (2019) and Wan et al. (2019) showed that engineers perceive testing and quality evaluation of ML components as very difficult challenges compared with traditional software components, because test oracles for ML are missing, ML components behave non-deterministically, or test coverage is hard to define. Lwakatare et al. (2019) introduced a comprehensive taxonomy to classify the challenges stemming from adoption of ML components. This taxonomy was later used as a starting point for other SE for ML taxonomies, such as defining research agendas (Bosch et al., 2021), classifying faults (Humbatova et al., 2020) or positioning quality models for ML (Lenarduzzi et al., 2021). A broad overview of the challenges in SE for ML was also introduced by Kumeno (2019), while other publications present challenges met at individual stages of the ML development life-cycle, e.g., requirements engineering (Nakamichi et al., 2020; Villamizar et al., 2021) or software architecture (Muccini and Vaidhyanathan, 2021).

Sculley et al. (2015) were among the first to present both challenges and introduce solutions in SE for ML. They used the framework of technical debt to define risk factors for ML components, and presented initial methods to manage the risks. Furthermore, they argued that ML components are subject to all maintenance issues from traditional software development, as well as new issues specific to ML. Breck et al. (2017) studied the challenges raised by testing ML components and proposed twenty-eight testing and monitoring practices suited for different stages of the development life-cycle of ML software. Additionally, they also investigated the benefits of adopting the practices, which allowed the development of a maturity model with the ultimate goal of measuring technical debt.

A broader overview of ML testing was introduced by Zhang et al. (2020). Since then, testing ML components has received increasing attention, and multiple testing frameworks have been introduced. In particular, attention has focused on acquiring more testing concepts from the general field of ML and on integrating them into SE test suits. For instance, Weiss and Tonella (2021) focused on uncertainty testing, while Chakraborty et al. (2021) focused on bias testing. Testing metrics specifically designed for neural networks have been summarised by Usman et al. (2022), while broader concerns regarding fault tolerance and misbehaviours of ML components have been introduced by Myllyaho et al. (2022).

Amershi et al. (2019) presented a study conducted at Microsoft, aimed at collecting challenges and practices in SE for ML across different teams. Their study reported on a broad set of challenges and practices corresponding to all stages of the ML development life-cycle. Using the experience of the respondents and the set of challenges,

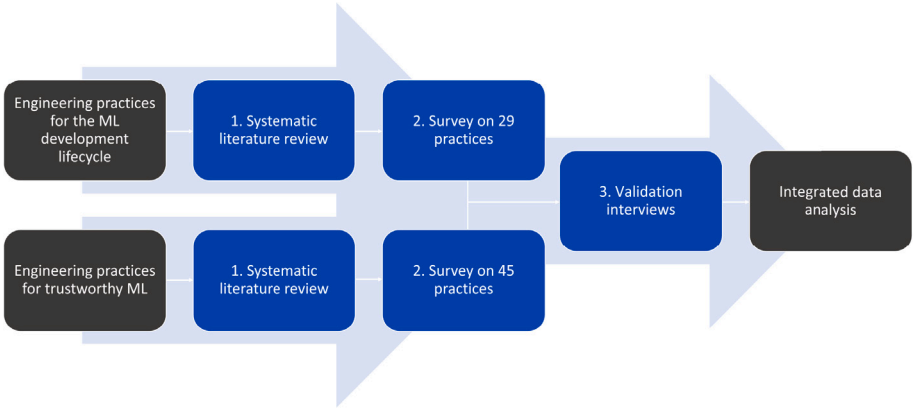


Fig. 1. Study design stages, where the numbered blue blocks correspond to the three individual stages of the study.

they built a maturity model which was used to assess each team. However, the set of challenges and reported practices are very broad and often not directly actionable. Moreover, they represent the opinions of team members from Microsoft, where typically more resources are dedicated to ensuring adoption of best practices than within smaller or less experienced companies.

De Souza Nascimento et al. (2019) aimed to identify the challenges faced by small companies in developing software with ML components by running interviews with practitioners. They compiled a set of checklists to help practitioners overcome the challenges faced by small companies. In spite of light validation (with only two interview participants), the checklist provides a good indication of the challenges faced by small organisations, such as identifying relevant business metrics or establishing processes for data handling.

Washizaki et al. (2019, 2022) and Serban and Visser (2022) studied software architecture design patterns and tactics for the development of software with ML components. Both studies observe that software architecture for ML is (still) driven by more traditional concerns, such as scalability or component coupling, and less by ML-specific concerns, such as trustworthiness. Moreover, both studies present comprehensive sets of patterns and tactics which help overcome both traditional and ML-specific concerns. For a comprehensive overview of SE for ML methods, practices or checklists, we recommend the work of Martínez-Fernández et al. (2021). Operational patterns for responsible development of ML components have also been introduced by Lu et al. (2022).

The methods used to overcome the challenges posed by the adoption of ML components have fuelled broad interest to develop or assess quality and maturity models for systems with ML components. As previously mentioned, Breck et al. (2017) introduced a maturity model based on testing practices, which has been used as a proxy to measure technical debt. Akkiraju et al. (2020) proposed a re-interpretation of the Capability Maturity Model for software with ML components based on personal experiences, Mikkonen et al. (2021) analysed the maintainability of software with ML components, while Lavin et al. (2021) proposed a framework to assess the readiness of ML components.

Kuwajima et al. (2020) analysed the ISO/IEC 25000 standard series on System and Software Quality Requirements and Evaluation (SQuaRE) (International Organization for Standardization, 2016) and discovered several gaps when applied to software with ML components. In particular, they noticed the lack of requirements specification and robustness of ML components, which greatly impacts the quality elements of the standard. A similar conclusion was drawn by Serban and Visser (2022) when analysing the quality attributes of the ISO 25010 standard with respect to ML components. Efforts to enhance these models to support ML components are being carried out, as shown by the development of ISO 21448 (Schnellbach and Griessnig, 2019) in the automotive domain or by recently announced developments of the ISO/IEC JTC 1/SC 42 group. In spite of these efforts, as of this writing, a standardised quality assessment for software with ML components is not available.

As mentioned before, our work builds on two previously published studies (Serban et al., 2020c, 2021). In the first of these, we mined both academic and grey literature and identified twenty-nine engineering best practices for software with ML components (Serban et al., 2020c). We also conducted a survey with over three hundred participants to determine the degree of adoption for the practices and to validate their perceived effects. In the second study, we extended the set of practices with fourteen new practices for the development of trustworthy ML components and two practices for AutoML (Serban et al., 2021; van der Blom et al., 2021). Since additional responses were received since the last publication, in the following, we present more robust analyses of the results, as well as results from jointly analysing the studies. Moreover, we extend the validation by interviewing more practitioners.

To catalogue and describe the practices, we took inspiration from well-known cataloguing techniques in SE, such as design patterns (Gamma et al., 1994; Lämmel and Visser, 2002), refactoring (Fowler, 2018), structured formats for documenting metrics (Bouwers et al., 2014) and implementation tactics (Cruz and Abreu, 2019).

While several novel studies have used the two previous publications this paper is based on, e.g., Giray (2021), Bogner et al. (2021), Lewis et al. (2021), Mojica-Hanke et al. (2023), Li et al. (2022), Golendukhina et al. (2022) the focus has been on the more traditional engineering practices presented by Serban et al. (2020c) and less on studying trustworthiness practices presented by Serban et al. (2021). For example, when evaluating what software quality means for AI engineers, Golendukhina et al. (2022) did not find any trustworthiness related quality concerns. Given the negative impact that ML components may have on users and society, the failure to embed trustworthiness practices in the software development life-cycle or quality assessment of software systems is worrying. We consider that by promoting a comprehensive catalogue of practices that treats trustworthiness as an intrinsic component rather than a separate concern (like the one introduced in our study) practitioners will prioritise trustworthiness practices together with other concerns, rather than treat these concerns separately.

3. Study design

This section presents details about the design of the studies performed and discusses their individual stages. Our studies consisted of a mixed-methods approach with a sequential exploratory strategy (Easterbrook et al., 2008). The process was organised in three stages, executed in a series of two studies. The first stage consisted of a *SLR* designed to identify engineering best practices for the development of software with ML components. The second stage consisted of a *survey* designed to gather quantitative data on the adoption of practices and

Table 1
Systematic literature review research questions.

ID	Research Question	Motivation
RQ1	Which are the engineering best practices for developing software with ML components?	Understand technical and organisational best practices for software with ML components.
RQ2	What are the specific engineering best practices that can be used to develop trustworthy ML components?	Identify actionable recommendations for software engineers who are developing ML components, focusing on ensuring the components are reliable, secure, and free from bias following the seven requirements for trustworthy AI presented by the high-level expert group set up by the European Commission (High-Level Expert Group on AI, 2021).

on the resulting effects. The last stage consisted of *validation interviews* with practitioners designed to determine the relevance of the practices and to obtain new insights into the trends observed in the survey. By triangulating data collected in these three stages, we expect more reliable results (Easterbrook et al., 2008). An illustration of the study design is provided in Fig. 1 while the design of the three stages is described in the following sections.

3.1. Engineering practices for the entire ML development life-cycle

The first study in the series aimed to identify engineering practices for all stages of the development life-cycle and consisted of the three stages illustrated in Fig. 1 and briefly discussed below. For a more detailed overview of the study design we refer the reader to Serban et al. (2020c).

Systematic literature review. The research protocol for conducting the SLR was defined using the guidelines from Kitchenham and Charters (2007). Since a consistent body of non-academic (grey) literature proposes engineering practices for ML, the SLR was designed to include both academic (white) and non-academic (grey) literature following the process described by Garousi et al. (2019).

The research question and its underlying motivation for the SLR is described in Table 1 (RQ1), and was designed to discover broad technical and organisational practices used during the development of software with ML components. To ensure adequate representation of academic and non-academic studies, the search strategy included multiple information sources. First, we used multiple queries to retrieve studies from the Google and Google Scholar search engines. We found these search engines to subsume other information sources typically used in SE SLRs (Shahin et al., 2017). To define the search queries, we followed the guidelines by Kitchenham and Charters (2007), and defined queries that comprised of two elements; the first suggesting the field of research and the second suggesting stages of the development life-cycle. For the first element, we used the terms “machine learning” and “deep learning”; we excluded the term “artificial intelligence”, because it has a broader scope and also includes not just algorithms or engineering methods, but also philosophical and regulatory debates. For the second element of the query, we used keywords that define individual phases or the entire process. A complete list of keywords is provided in Appendix, Table 8.

To filter out documents, we used the criteria formulated by Garousi et al. (2019), i.e., using the authoritativeness of the outlets and authors as well as the objectivity of the style and content. Moreover, we filtered out duplicates based on the articles’ title and limited the search to articles published in the past five years (2015–2020), corresponding to the rising interest and adoption of ML technologies.

From the manually inspected articles we extracted information about (i) demographic factors and (ii) engineering best practices or recommendations for the development of software with ML components. To analyse the demographics data, we used descriptive statistics and to synthesise the engineering practices, we used thematic analysis (Braun and Clarke, 2006).

Survey. To validate the set of practices obtained from the SLR, we ran a survey with both researchers and practitioners, which consisted of a questionnaire asking participants if, at the moment of taking the survey, their team adopted the engineering practices in projects with ML components. The questionnaire was designed following the recommendations from Kitchenham and Pfleeger (2008) and Ciolkowski et al. (2003), and consisted of (i) 5 preliminary questions used to distinguish between the teams’ profiles, (ii) 31 questions mapped to the initial practices (from which two practices had two questions each), and (iii) 4 questions focused on the perceived effects of adopting the practices.

Before broad distribution of our survey, we invited several practitioners with varying backgrounds to an exploratory interview session that included the survey pilot. The participants’ profiles are illustrated in the first five rows of Table 2. All participants were part of teams developing software with ML components, and all agreed with the relevance of all practices. Moreover, all challenges met by the participants were well represented in the initial set of practices. The feedback from the interviews was used to refine the questionnaire, adding new answers or rewording the questions.

To distribute the survey, we chose a snowball strategy. At the beginning, we reached out to our network of contacts and to all authors of the publications used to extract the practices and asked them to distribute the survey through their networks. Moreover, we advertised the survey through channels commonly used by practitioners, such as Twitter, Medium, HackerNoon, Dev.to and the Meetup groups for ML in various regions.

3.2. Engineering practices for trustworthy ML

The second study in the series aimed to identify engineering practices for the development of trustworthy ML components (Serban et al., 2021) and was organised using the same guidelines as described in Section 3.1. Moreover, at this stage the set of practices was complemented with two more practices related to AutoML, to improve the coverage of this topic compared to the first study. In this section we only discuss specific design choices for this study and refer the reader to Section 3.1 and Serban et al. (2021) for more details.

Systematic literature review. The SLR was guided by the second research question (RQ2) in Table 1 and focused specifically on *actionable* engineering practices, i.e., practices that can be directly adopted and integrated by practitioners into their development processes and avoids broader, relevant but non-actionable, debates. RQ2 is first motivated by the lack of engineering practices for trustworthy ML in the study answering RQ1. One reason for this absence is that trustworthy concerns become more important and visible once regulatory and ethical considerations around the use of ML components were developed. For example, the EU’s GDPR requires that individuals have the right to explanations of automated decisions made about them. Similarly, the EU’s guidelines for trustworthy artificial intelligence (AI) establish a more comprehensive framework targeting lawful, ethical and technical robustness concerns. Additionally, various ethical frameworks have been developed, such as the IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems, which emphasise the need for transparency, accountability, and fairness in the development of autonomous and intelligent systems. By focusing on trustworthiness in the development of ML components, RQ2 is directly addressing these important regulatory and ethical considerations. Second, RQ2 is motivated by the belief that strong considerations for trustworthiness

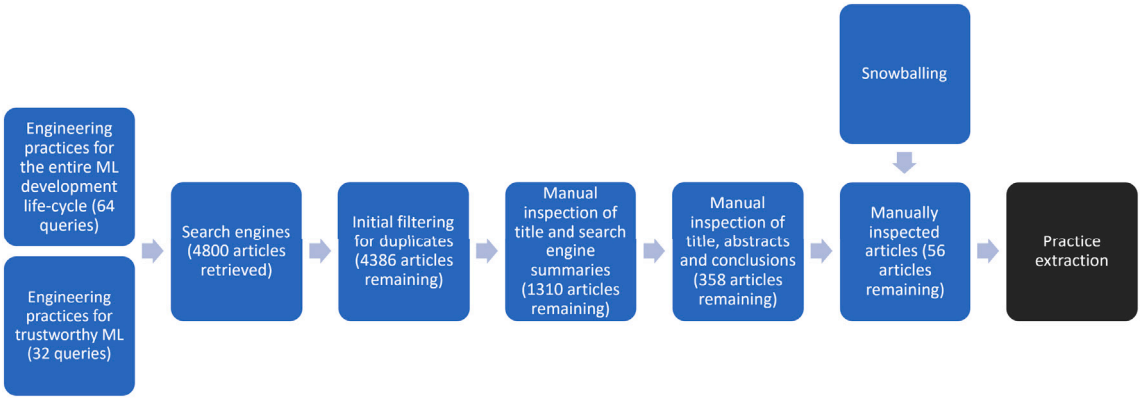


Fig. 2. Selection of articles from the systematic literature review.

are only necessary in the development of some applications where biased or incorrect decisions could have serious consequences, and not for all applications. Therefore, trustworthiness is often considered a separate concern from the development life-cycle, similar to how security was interpreted before moving to security-by-design (McGraw, 2004), and the practices for trustworthy ML are disconnected from the practices for the entire ML development life-cycle. While there is some overlap between RQ1 and RQ2, we find RQ2 necessary for addressing in detail the field of trustworthiness. This needs arises from the fact that trustworthiness concerns remain unaddressed when focusing solely on answering RQ1.

To search for documents that cover trustworthy ML, we used keywords for the second element of the query using the seven requirements for robustness from the guidelines for trustworthy AI presented by the high-level expert group set up by the European Commission (High-Level Expert Group on AI, 2021), as at the time of writing they were the most comprehensive set of guidelines that specifically target software with ML components and not broader autonomous or intelligent systems (Chatila and Havens, 2019). A complete list of keywords is provided in Appendix, Table 9.

Survey. This stage ran similarly to the first study, but was extended with 16 additional practices and two more effects, as identified using the SLR. The questionnaire was validated before distribution using a pilot interview consisting of two participants (with the profile illustrated in the last two rows of Table 2), which all agreed with the importance of all practices.

3.3. Validation interviews

To validate the final outcomes of the studies and to gain deeper insights from the results regarding the state of practice, we performed interviews with selected practitioners after we collected all the survey data.

Protocol. The interview protocol was designed following the guidelines by Hove and Anda (2005). All interviews were conducted online, through video calls. To enable participants to become familiar with the interview objectives (Hove and Anda, 2005), we asked participants to answer the survey prior to the interview.

The interviews were structured in four sections. First, during the introduction, we discussed the research goals and background. Second, we asked participants to share data about their background and about the projects (involving ML components) they are working on. Third, we asked participants general information about the survey, e.g., whether any question was unclear or if they had made any specific observations. In this section, we also asked participants about their answers, focusing on the questions where they had indicated low levels of practice adoption. This part enabled a broader discussion about the challenges faced and about the suitability of the practices to different projects. Fourth, we asked participants to provide open-ended comments and to share other experiences.

Table 2
Profiles of the pilot interview subjects.

Id	Company profile	Team size	Experience
P1	Tech Startup	5–6 ppl.	1–2 years
P2	Tech company	10–15 ppl.	>5 years
P3	Research lab	5–6 ppl.	2–5 years
P4	Tech Startup	10–15 ppl.	2–5 years
P5	Non-tech company	6–9 ppl.	1–2 years
P6	Research lab	10–15 ppl.	2–5 years
P7	Tech company	5–6 ppl.	2–5 years

Participants. The interview participants were recruited using purposeful sampling (Palinkas et al., 2015). To recruit participants, we used our network of contacts, which enabled to select participants involved in building software with ML components. The participants' profiles are presented in Table 7 and will be discussed in Section 10.

Data analysis. To process the interviews, we used thematic analysis, as recommended by Cruzes and Dyba (2011). The process consisted of (i) data extraction – the interviews were transcribed, read, and key points were extracted, (ii) data coding – the answers were categorised and linked to the practices, (iii) code to themes translation – for each transcript the codes were combined into potential themes (e.g., testing), (iv) high-order theme modelling – the themes were compared and merged, (v) synthesis assessment – arguments for the extracted data were established, e.g., in terms of agreement.

4. A catalogue of engineering practices for ML

This section presents the process of compiling a catalogue of engineering practices for ML concerning the entire ML development life-cycle and including trustworthiness. It also introduces the selection of articles from the SLRs, the methodology used to compile the catalogue and the taxonomy used to group the practices. The section ends with an example practice taken from our online catalogue. The selection of articles from our systematic literature reviews (SLRs) is illustrated in Fig. 2. In the first stage, we ran the search strings on the selected engines and retrieved 4800 results. Next, we *automatically* filtered out duplicates and *manually* inspected the resulting articles' titles and summaries provided by the search engines. In case the content was deemed relevant or the title or summary was inconclusive, the articles were selected for the next stage. A total of 358 articles were selected for further manual evaluation. These articles were downloaded and their abstracts, introduction and conclusions were read. The inclusion criteria were applied, and a snowballing strategy was

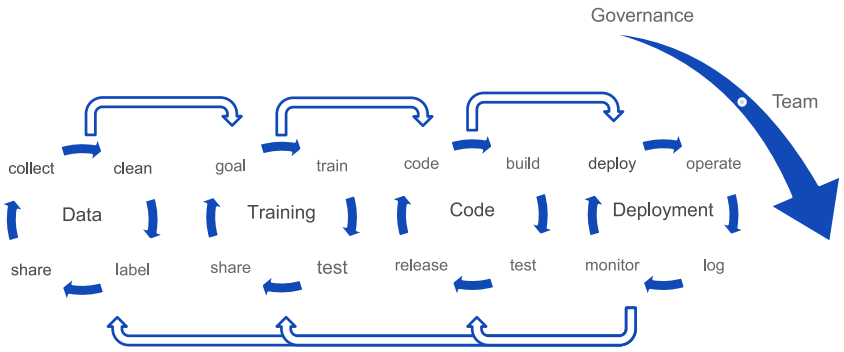


Fig. 3. Taxonomy mapping on the ML development process.

used to identify potential further articles from the references of ones already identified. This yielded 56 relevant articles, which were read completely by multiple authors and used to extract the practices.

Compiling a catalogue of engineering practices for ML. To extract the practices, two of the authors read each article in its entirety and generated draft practices (also called initial codes), i.e., preliminary lists of practices, tests or recommendations for engineering software with ML components. We observed that some articles only suggest high-level goals (e.g., ensure that trained ML models can be traced back to the training data and scripts) without detailing the steps needed to achieve the goals. Other articles provided detailed steps for achieving the goals (e.g., use versioning for data, models, configurations and training scripts (van der Weide et al., 2017; Seyffarth, 2019; Megler, 2019)). To differentiate such articles, we added the high-level goals to a special group, which we called "effects" and the rest to a group we called "practices".

Next, the codes with similar goals or recommendations were merged under the supervision of at least two authors from our team. The resulting codes were checked against each other and refined under the supervision of all authors from our team, in order to validate that they are supported by sufficient evidence. Last, clear and concise names, descriptions and statements of intent and motivation were defined for each practice.

Extracting a common taxonomy for the practices. The manually inspected documents provide a grouping or classification of practices based on the ML specific software development life-cycle. For example, Amershi et al. (2019) define a life-cycle consisting of nine stages, while Sato et al. (2019) divide similar activities into only six stages. These processes have roots in early development models for data mining, such as CRISP-DM (Wirth and Hipp, 2000).

When trying to define a common life-cycle model for the practices, we found that no single division of the software development life-cycle for ML emerges as most authoritative. Therefore, we reconstructed a broad taxonomy that is compatible with all proposals in literature and consists of the following six stages:

- **Data** - Practices used before training, when collecting or preparing the data needed to train ML models.
- **Training** - Practices used in preparing and running training experiments (such as planning, development or deployment of training processes).
- **Deployment** - Practices used to prepare models for deployment, to deploy, to monitor or to maintain models.
- **Coding** - Practices used for implementation and production of runnable code, primarily stemming from traditional SE, and for writing, testing, and deploying code.
- **Team** - Practices used by teams to organise, communicate, align and decide trade-offs among team members
- **Governance** - Practices used to ensure responsible use of ML techniques and to achieve desirable properties, such as fairness and transparency.

A mapping of this taxonomy onto the ML development process, which is compatible with previous work (Amershi et al., 2019; Sato et al., 2019; Wirth and Hipp, 2000), is illustrated in Fig. 3. For example, the "Data" class maps one to one to other data taxonomies (Amershi et al., 2019; Wirth and Hipp, 2000) while the "Training" class maps one to many to other finer-grained taxonomies, which include, for example, "Feature engineering" and "Model Training" (Amershi et al., 2019) or "Model Building" (Sato et al., 2019). Our taxonomy also features two high-level classes which gather practices concerning all steps of the development process – "Governance" and "Team" – and which are not addressed in previous taxonomies (Amershi et al., 2019; Sato et al., 2019; Wirth and Hipp, 2000).

Complementing the catalogue of practices. While the initial selection of practices reflected the ML development life-cycle, it lacked some of the relevant practices from traditional SE. Since practitioners which focus on ML may overlook practices from traditional SE, after compiling the initial catalogue, we complemented it with six additional practices from traditional SE – three of technical nature, which fall in the "Coding" class, and three related to social aspects, which fall in the "Team" class (practices 24, 25, 26 and 35, 36, 37, see Table 3). These practices were selected, because they were considered challenging, yet essential for software development (Visser et al., 2016).

Moreover, while the initial set of practices contained one practice related to AutoML – a set of technologies aimed at facilitating the effective use of ML methods by automating different stages of the life-cycle – we later realised that this left part of the spectrum of AutoML applications uncovered. Therefore, considering the potential impact of AutoML, we added two new practices (van der Blom et al., 2021).

The resulting 45 practices are listed in Table 3 and the six effects in Table 4. The practices are also available to practitioners in a more elaborate format in an online catalogue.[1] A curated reading list with references, further relevant literature as well as a selection of supporting tools is also maintained online.[2]

Example practice. Fig. 4 illustrates how practice 4 from Table 3 appears in the online catalogue. The difficulty estimation found below the title and the category to the right will be explained in Section 6. Besides this, the practice catalogue takes inspiration from established SE cataloguing techniques, particularly from design patterns (Gamma et al., 1994; Lämmel and Visser, 2002), to detail intent, motivation and applicability, and to provide broader descriptions of practices. Moreover, adoption plots grouped by various demographic factors, such as team size or experience are displayed in the online catalogue. These plots help practitioners to compare their adoption rates with other teams covered by our data set, and to reason whether improving particular demographic factors, such as the team size, can lead to better

[1] https://se-ml.github.io/practices/
[2] https://github.com/SE-ML/awesome-seml

Table 3
Engineering practices for ML, where DT – Data, TR – Training, CD – Code, DP – Deployment, TM – Team, GV – Governance and N – new, T – traditional, M – modified and TR – trustworthiness practices.

Nr.	Title	Class	Type	References	Rank
1	Use Sanity Checks for All External Data Sources	DT	N	Alkis Polyzotis Martin A Zinkevich Steven Whang Sudip Roy (2017), Breuel (2019)	23
2	Check that Input Data is Complete, Balanced and Well Distributed	DT	N	Alkis Polyzotis Martin A Zinkevich Steven Whang Sudip Roy (2017), Sculley et al. (2015), Megler (2019), Breck et al. (2017)	14
3	Test for Social Bias in Training Data	DT	TR	Liu et al. (2019), Kleinberg et al. (2016), Hardt (2020)	30
4	Write Reusable Scripts for Data Cleaning and Merging	DT	N	Algorithmia (2019), Alkis Polyzotis Martin A Zinkevich Steven Whang Sudip Roy (2017), Breuel (2019)	2
5	Ensure Data Labelling is Performed in a Strictly Controlled Process	DT	N	Roh et al. (2019), Prendki (2018), Cloudfactory (2019), Altexsoft (2018)	8
6	Prevent Discriminatory Data Attributes from Being Used as Model Features	DT	TR	Hardt (2020)	33
7	Use Privacy-Preserving Machine Learning Techniques	DT	TR	Brundage et al. (2020)	42
8	Make Data Sets Available on Shared Infrastructure (private or public)	DT	N	Megler (2019), Khomh et al. (2018), Herron (2019), John (2019)	10
9	Share a Clearly Defined Training Objective within the Team	TR	N	Microsoft Blog (2019), Megler (2019), Zinkevich (2019)	6
10	Capture the Training Objective in a Metric that is Easy to Measure and Understand	TR	N	Microsoft Blog (2019), Talagala (2018), Zinkevich (2019)	1
11	Test all Feature Extraction Code	TR	M	Sato et al. (2019), Breck et al. (2017)	24
12	Assign an Owner to Each Feature and Document its Rationale	TR	M	Zinkevich (2019)	28
13	Actively Remove or Archive Features That are Not Used	TR	N	Sculley et al. (2015), Zinkevich (2019)	27
14	Employ Interpretable Models When Possible	TR	TR	Brundage et al. (2020), High-Level Expert Group on AI (2021), Molnar (2020)	34
15	Peer Review Training Scripts	TR	M	Breck et al. (2016)	16
16	Enable Parallel Training Experiments	TR	N	Sato et al. (2019), Seyffarth (2019)	3
17	Automate Feature Generation and Selection	TR	N	Dean (2019), Tunguz (2020), ZelrosAI (2019)	39
18	Automate Hyper-Parameter Optimisation and Model Selection	TR	N	Hutter et al. (2019), Mayo (2017)	21
19	Automate Configuration of Algorithms or Model Structure	TR	N	Dean (2019), Tunguz (2020), ZelrosAI (2019)	37
20	Continuously Measure Model Quality and Performance	TR	N	Zinkevich (2019), Google Devs (2019)	4
21	Assess and Manage Subgroup Bias	TR	TR	Hébert-Johnson et al. (2018), Kearns et al. (2018)	45
22	Use Versioning for Data, Model, Configurations and Training Scripts	TR	M	Le (2019), Hummer et al. (2019), Seyffarth (2019), Megler (2019)	5
23	Share Status and Outcomes of Experiments Within the Team	TR	N	Le (2019), Herron (2019)	7
24	Run Automated Regression Tests	CD	T	Seyffarth (2019), Breck et al. (2017)	25
25	Use Continuous Integration	CD	T	Sato et al. (2019), Breck et al. (2017)	31
26	Use Static Analysis to Check Code Quality	CD	T	Visser et al. (2016)	20
27	Assure Application Security	CD	TR	Brundage et al. (2020), National Science and Technology Council (US). Select Committee on Artificial Intelligence (2021)	41
28	Automate Model Deployment	DP	M	Sapp (2017), Dunning and Friedman (2019), Talagala (2018)	17
29	Enable Shadow Deployment	DP	M	Dunning and Friedman (2019), Baylor et al. (2017), van der Weide et al. (2017)	22
30	Continuously Monitor the Behaviour of Deployed Models	DP	N	Sato et al. (2019), Dunning and Friedman (2019), Seyffarth (2019), Google Devs (2019)	13
31	Perform Checks to Detect Skews between Models	DP	N	Sato et al. (2019), Zinkevich (2019), Google Devs (2019), Baylor et al. (2017)	18
32	Enable Automatic Roll Backs for Production Models	DP	M	Sato et al. (2019), Dunning and Friedman (2019)	12
33	Log Production Predictions with the Model's Version and Input Data	DP	M	Hummer et al. (2019), Sridhar et al. (2018), Megler (2019)	19
34	Provide Audit Trails	DP	TR	High-Level Expert Group on AI (2021), Brundage et al. (2020), Raji et al. (2020)	40
35	Use A Collaborative Development Platform	TM	T	Booch and Brown (2003), Storey et al. (2016)	9
36	Work Against a Shared Backlog	TM	T	Sedano et al. (2019), Sutherland and Schwaber (2013)	15
37	Communicate, Align, and Collaborate With Multidisciplinary Team Members	TM	T	Faraj and Sproull (2000)	11
38	Decide Trade-Offs through Defined Team Process	TM	TR	Fandel (1990), Branke et al. (2008), Ruhe (2002)	29
39	Establish Responsible AI Values	GV	TR	High-Level Expert Group on AI (2021), Brundage et al. (2020)	36
40	Perform Risk Assessments	GV	TR	High-Level Expert Group on AI (2021), Brundage et al. (2020), Raji et al. (2020)	43
41	Enforce Fairness and Privacy	GV	TR	Google AI Blog (2019a,b), Breck et al. (2017)	26
42	Inform Users on Machine Learning Usage	GV	TR	High-Level Expert Group on AI (2021), Mitchell et al. (2019)	35
43	Explain Results and Decisions to Users	GV	TR	High-Level Expert Group on AI (2021)	38
44	Provide Safe Channels to Raise Concerns	GV	TR	High-Level Expert Group on AI (2021), Brundage et al. (2020)	32
45	Have Your Application Audited	GV	TR	High-Level Expert Group on AI (2021), Brundage et al. (2020)	44

Table 4
Description of the effects studied.

Nr.	Effects	Description
1	Agility	The team can quickly experiment with new data and algorithms, and quickly assess and deploy new models
2	Software Quality	The software produced is of high quality (technical and functional)
3	Team Effectiveness	Experts with different skill sets (e.g., data science, software development, operations) collaborate efficiently
4	Traceability	Outcomes of production models can easily be traced back to model configuration and input data
5	Reproducibility	Past behaviour of the models or applications can easily and precisely be reproduced
6	Accountability	The team can demonstrate the ML application being developed adheres to the principles of Responsible AI (e.g., fairness, transparency, etc.).

Write Reusable Scripts for Data Cleaning and Merging **Category:** *Data*
Difficulty: *Basic*

Intent
Avoid untidy data wrangling scripts, reuse code and increase reproducibility.
Motivation
Data cleaning and merging are exploratory processes and tend to lack structure. These processes involve manual steps or poorly structured code which can not be reused later. Needless to mention, such code cannot be integrated into processing pipelines.
Applicability
Reusable data cleaning scripts should be written for any ML application that does not use raw or standard data sets.
Description
Often, training ML models is preceded by an exploratory phase, in which non-structured code is written or manual steps are performed, in order to get the data into the right format, merge several data sources, etc. Especially when using notebooks, there is a tendency to write ad-hoc data processing scripts, which depend on variables already stored in memory when running previous cells.
Before moving to the training phase, it is important to convert this code into reusable scripts and move it into methods that can be called and tested individually. This will enable code reuse and ease integration into processing pipelines.
Adoption
Adoption plots, grouped by various demographic factors.
Related
Practices 11 and 24 from Table 3.
References
[2, 21, 93]

Fig. 4. Example of practice in online catalogue.

practice adoption. Related practices and references for in-depth details complete the catalogue.

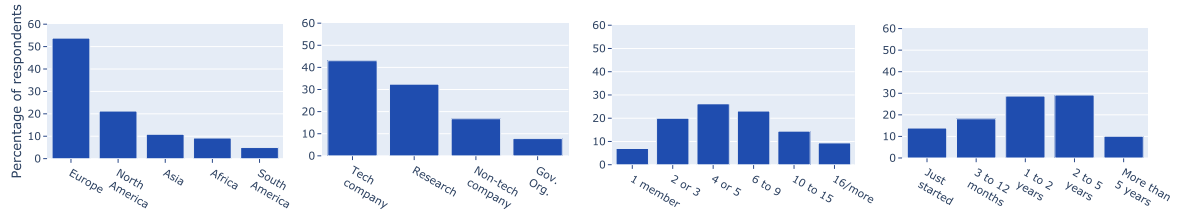
As can be seen in Fig. 4, the *intent* for practice 4 is to avoid cluttered data processing scripts, in order to increase reusability and reproducibility. This practice is *motivated* by the fact that data cleaning and merging are exploratory processes by nature, and thus typically lack structure and clearly defined goals. Therefore, the code related to these processes also tends to lack structure, which makes it hard to maintain, reuse or integrate into more complex pipelines. The *applicability* of this practice is broad, as any ML application where data processing precedes training faces these challenges. The practice *description* emphasises use-cases in which these challenges are exacerbated – e.g., when using coding notebooks – and proposes strategies for applying the practice, such as defining units of code that can be called and tested individually.

Figs. 5(b), 5(c), 5(d) show the percentage of respondents grouped by organisation type, team size and team experience. The results in all of these figures are as expected, since e.g., the majority of answers come from tech companies or the majority of teams have 4–5 or 6–9 members (as in Agile).

5. Practice adoption

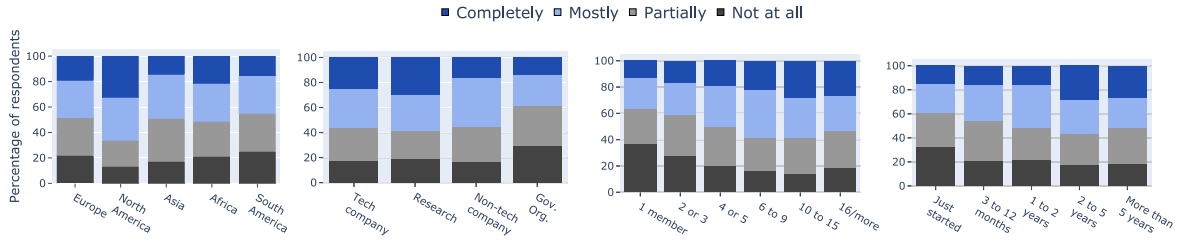
This section presents the adoption of practices as revealed by the two surveys, discussing different demographic factors and their impact on the results. In total, we received 504 valid responses — 378 from our first, and 126 from our second study. To discard responses outside our target audience, we used the prerequisites to filter out all answers that came from respondents that were not part of a team working on software with ML components. Moreover, we applied a finer-grained filtering using the percentage of questions that were answered in the prerequisites (at least 50%) and in the practice adoption sections of our survey (at least 50%). This process resulted in 313 answers from our first study and 95 from the second, for a total of 408 valid answers.

A demographic characterisation of the respondents is illustrated in Fig. 5. Fig. 5(a) shows the percentage of respondents grouped by region (using the location attribute of the surveys). We observe that respondents from Europe have a higher contribution, although other regions are also well represented. The possible bias arising from the over-representation of Europe will be discussed in Section 10. We note that in our first study, we had no respondents from Africa (Serban et al., 2020c). After advertising the second study in that region, we obtained



(a) Respondents grouped by regions. (b) Respondents grouped by organisation type. (c) Respondents grouped by team size. (d) Respondents grouped by team ML experience

Fig. 5. Demographic information describing the survey participants. All plots show the percentage of respondents, grouped by various demographic factors.



(a) Adoption of practices grouped by regions. (b) Adoption of practices grouped by organisation type. (c) Adoption of practices grouped by team size. (d) Adoption of practices grouped by team ML experience.

Fig. 6. Adoption of practices grouped by various demographic factors. All plots show the percentage of answers, grouped by the answer types illustrated in the plot legend..

a higher number of respondents from Africa than from South America. Moreover, the Oceania region was grouped with Asia.

Fig. 5(b) shows the percentage of respondents grouped by organisation type. The highest percentages of answers are from respondents working for tech companies (e.g., social media platforms, semiconductors) and for research labs. These results are expected, because both types of organisations play leading roles in the ML research landscape. Nonetheless, we also observe a fair representation of non-tech companies (e.g., enterprises, banks, retailers) and governmental organisations.

Figs. 5(c) and 5(d) further illustrate the percentage of respondents grouped by team size and experience in developing software with ML components. We notice that the majority of teams have 4–5 or 6–9 members, which corresponds to the typical size of software development teams (e.g., as recommended in Agile). Furthermore, the majority of teams have between 1–2 or 2–5 years of experience with ML, which is consistent with the more recent growth in popularity of ML technologies.

Using the demographic factors presented previously, we can evaluate the practice adoption across different groups. These results are illustrated in Fig. 6, where the answers are grouped and normalised to percentages following the Likert scale from our survey. Fig. 6(a) shows the adoption of practices grouped by regions. While Europe is over-represented, the practice adoption for this region does not present striking differences when compared to other regions, e.g., Asia, Africa or South America. On the other hand, the practice adoption for North America is significantly higher, as reflected by large numbers of "Completely" and "Mostly" answers. Since North America is well represented in our data set, it is likely that practitioners from this region indeed exhibit higher adoption of practices. Similarly, since Europe does not present striking differences when compared to other regions, it is likely that the bias introduced by over-representation is not significant.

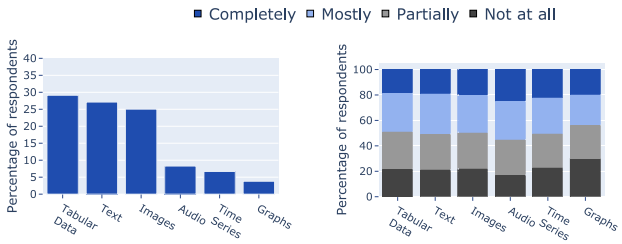
Fig. 6(b) illustrates the adoption of practices grouped by organisation type. We observe that tech companies and research labs lead in practice adoption, while non-tech companies and governmental organisation have overall lower adoption rates. In particular, non-tech companies have lower practice adoption in the "Completely" category, but competitive adoption in the other categories. Governmental organisations have a large percentage of responses in the adoption categories "Not at all" and "Partially". Similar trends were observed in concurrent work, where organisations outside "Big Tech" report lower adoption of practices (Hopkins and Booth, 2021).

Figs. 6(c) and 6(d) illustrate the adoption trends by team size and team experience. For both classes, we observe a trend towards higher adoption as we move from small/inexperienced teams to larger or more experienced teams. A contrasting trend can be observed only for the last category of the plots, i.e., for larger team size or more experience the practice adoption is slowly decreasing. This result may indicate that practitioners who started early may be unaware of practices developed more recently, or teams with too many members may fail to distribute tasks efficiently (as is the case with traditional software development teams (Begel and Nagappan, 2007)).

Figs. 6(b), 6(c), 6(d) present the adoption rates for organisation type, team size and team experience. The results are also as anticipated, since tech companies and research labs lead in practice adoption or practice adoption increases with team experience.

These demographic results confirm that the survey questions were clear, and the answer scale introduced little bias, as most results were consistent with expectations. We also analysed the adoption rates based on the types of data used by respondents and based on the practice type. Since the practices are general and apply to any context (e.g., to any project constraint, such as data type), we do not expect the adoption rates to change significantly between data types. However, the type of data directly influences the choice of algorithms and can indirectly influence practice adoption; for example, when using deep neural networks to process images or text, training is not preceded by a feature extraction step.

The distribution of data types and the adoption of practices grouped by data type is illustrated in Fig. 7. We note that tabular data, text and images have similar representation in our set of survey responses, while audio, time series and graphs have a lower representation (which implicitly makes their adoption rates less reliable). For the first three classes, we observe similar practice adoption rates, while for the last three classes, the adoption rates exhibit small variances (Fig. 7(b)). Given that the majority of answers are for the first three classes of data,



(a) Distribution of data types.

(b) Adoption of practices grouped by data type.

Fig. 7. Distribution and adoption of practices grouped by data type.

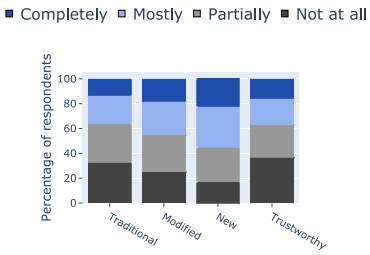


Fig. 8. Adoption of practices grouped by practice type.

for which the adoption rates do not vary significantly, we conclude that the set of practices is largely independent of the data type used.

Fig. 8 illustrates the adoption rate for the practices grouped by the practice type defined as follows: (i) *traditional* practices are used both in traditional software development projects and for ML, (ii) *modified* practices are used in traditional software development, but require changes to accommodate specific needs in the context of ML (e.g., versioning), (iii) *new* practices are only used for ML, and (iv) trustworthiness practices are engineering practices dedicated to the development of trustworthy ML components, such as alleviating bias in data sets. As seen in Fig. 8, practitioners tend to adopt new practices – specifically developed for ML – more than modified or traditional practices. This trend was also observed previously (Serban et al., 2020c) and may reveal that many ML practitioners are unaware of traditional software engineering practices. The least adopted practices are those for trustworthy development of ML components – which is a cause for concern, considering the potential negative impact ML systems can have on their users and society.

6. Practice ranking

This section presents the algorithm used to rank the practices and evaluates how robust this algorithm is against other possible choices. Using the adoption rates for each practice, we can rank the practices from the most adopted to the least adopted.

The ranking can be used as an indicator of the popularity of each practice. Considering the Likert scale from our survey, common metrics for ranking the practices are the mean, the median or the mean normalised by the maximum level of adoption for each practice (Nashimoto and Wright, 2007). However, given the relatively small number of levels on the Likert scale, the median will not vary much. Moreover, the mean is known to be sensitive to noise and sample size, and the maximum adoption level observed for each practice is "Completely".

To overcome these weaknesses, we defined a ranking algorithm that uses *levels of adoption*, meant to cancel out some noise stemming from fuzzy boundaries between neighbouring points on the Likert scale. The algorithm computes for each practice:

- The percentage of respondents with *at least high* adoption by counting the "Completely" answers; the percentage of respondents with *at least medium* adoption by counting the "Completely" and "Mostly" answers; and the percentage of respondents with *at least low* adoption by counting the "Completely", "Mostly" and "Partially" answers.
- A rank for each practice using each separate level of adoption.
- An average of individual adoption level ranks, which is used to order the practices and obtain the final rank.

Therefore, the final rank for each practice is the average rank on the three levels of adoption, which are meant to make the ranking algorithm robust to noise and sample size. To validate this approach, we compared the ranking algorithm with ranking based on the mean adoption level, by repeatedly subsampling our data using sample sizes between 50 and 90% of the data. For each sample size, we sampled and ranked the practices 1000 times and evaluated the variance in rank. We averaged this rank variance and took its square root (equivalent to average standard deviation). The results, displayed in Fig. 10, show that our choice for ranking leads to consistently smaller variability in the final ranking, for all sample sizes.

The final ranks for all practices are displayed in Table 3 and illustrated in Fig. 9, where the highest rank – 1 – corresponds to the most adopted practice and the lowest rank – 45 – corresponds to the least adopted practice. We note that the most adopted practice (practice 10) relates to establishing and communicating clearly defined training objectives within the team. Writing reusable scripts for processing data (practice 4), enabling parallel experiments (practice 16), continuous monitoring of the model during experimentation (practice 20) and versioning (practice 5) also appear in the top five positions. We observe that the top most adopted practices are new or modified practices and not traditional engineering practices. This result entails that practitioners focus more on ML-related engineering practices and tend to neglect more traditional practices, as also observed in Fig. 8.

The least adopted practices are all related to trustworthy development of ML. Besides these, practices related to AutoML (such as automation of feature generation and selection or automation for algorithm configuration and model structure, practices 17, 19), continuous integration (practice 25), and feature management (practices 12, 13) are among the least adopted. Some of these practices require more resources or a higher level of technical sophistication, which makes their adoption more difficult. For example, AutoML and continuous integration require more advanced infrastructure, and AutoML also requires additional expertise.

7. Analysis of practices and effects

This section presents the relationship between practices and effects as revealed by the survey answers. Moreover, it presents multiple models to predict the effects from the practices and evaluate the importance of each practice for the effects. We analysed the relationship between groups of practices and the effects of practice adoption, which allows us to define basic groups of practices that can be adopted in order to achieve desired effects. For this analysis, we employ linear and non-linear statistical models to predict the effects perceived by the survey participants from groups of practices. The groups of practices we considered and the practice adoption effects are presented in Table 5 and illustrated in Fig. 11.

The experiment consisted of training distinct models to predict the effects from the group of practices. For training, we converted the answers to a numerical scale ranging from 1 for the first answer, "Not at all" and 4 for the last answer, "Completely" and attempted to predict the corresponding numerical effects from sets of practices. The answer scale for the effects was identical to the answer scale for the practices. The evaluation was performed using a test set consisting of 25% of the data, for each effect. Following practice 18, we used

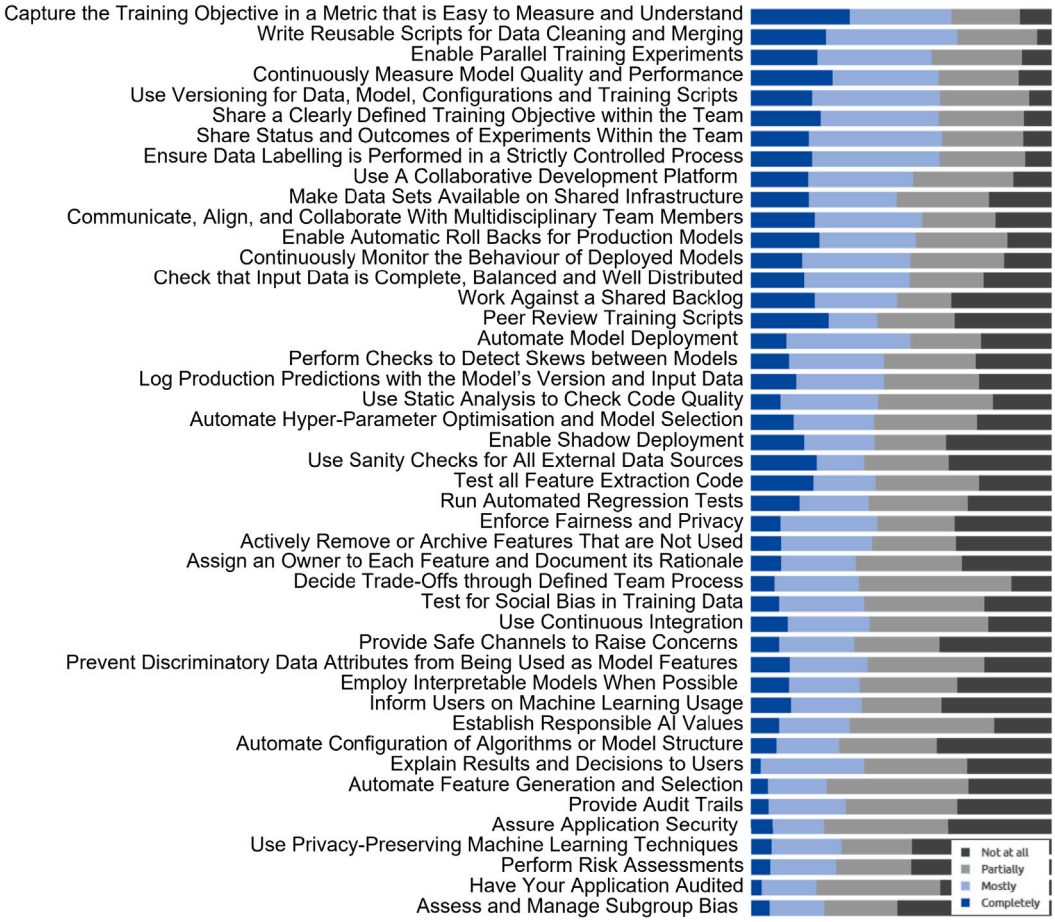


Fig. 9. Practices ordered by their rank and illustrated together with their adoption rates.

four types of models with increasing sophistication: (i) a simple linear regression model, meant to validate that a linear relationship exists between the group of practices and the effects; (ii) a random forest (RF) regression model with manually determined hyper-parameter settings and feature selection, meant to validate that a non-linear relationship exists between the practices and the effects; (iii) a RF regression model with hyper-parameters optimised through grid search, meant to boost the performance of the manual RF model; and (iv) an AutoML system that performed automatic model selection and hyper-parameter optimisation. We convert the categorical effects to numbers and perform regression instead of classification because the outcomes have a natural order. In the attached replication package we also experiment with classification, which entails similar results (Serban et al., 2020a).

Training was performed using 5-fold cross-validation. For all experiments, we used under-sampling on the training data to remove class imbalance. We also experimented with over-sampling algorithms (in particular with SMOTE (Chawla et al., 2002; Torgo et al., 2013)), but the increase in performance was not significant. The grid search RF used 384 candidate configurations for each of the five folds. For AutoML, we employed auto-sklearn (Feurer et al., 2015), where we tested both a modest time budget of 1 h and a larger budget of 5 h. This budget is significantly higher than the automatic hyper-parameter search through grid search, which required less than 10 min to run. For the evaluation of candidate models from AutoML we employed 5-fold cross validation and report the results from the 1 h budget models, as increasing the budget did not yield significant improvements.

Model performance was tested using three standard evaluation metrics: (i) mean squared error (MSE), (ii) the R^2 coefficient of determination and (iii) the Spearman correlation coefficient (ρ) for predicted and true outcomes. The results for all models are shown in Table 5.

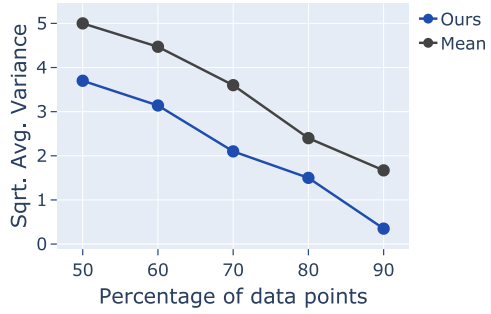


Fig. 10. Ranking variance at different sample sizes. The average variance is computed using 1000 different data samples for each sample size.

We observe that, in all cases, the effects can be predicted with low error and high coefficient of determination, both for the linear and non-linear models. However, the RF models outperform linear models in all cases, which indicates that some practices have a non-linear impact on the effects.

We also note that grid search yields performance improvements in all cases, which further motivates practice 18. AutoML models outperform the grid search RF in one case and yield competitive results in all others. For reproducibility, grid search has the highest correlation, but AutoML has a lower error with similar correlation.

Overall, the correlation between the predicted and true outcomes is high for all models, which indicates that the effects can be accurately

Table 5
Mean squared error (MSE), R^2 and Spearman correlation coefficient (ρ) between the predicted and the true outcomes for distinct models trained to predict the effects from the practices in the second column, where RF is Random Forest Regression. The results are extracted from a test data set consisting of 25% of the data. For each effect, the best performing model is shown in bold.

Effects	Practices	MSE/R^2/ρ - Linear Regression	MSE/R^2/ρ - RF	MSE/R^2/ρ - RF Grid Search	MSE/R^2/ρ - AutoML
Agility	16, 25, 28, 29, 30, 37	0.46/0.54/0.75	0.40/0.67/0.74	**0.36/0.79/0.86**	0.38/0.77/0.85
Software Quality	12, 13, 15, 24, 25, 26	0.32/0.45/0.72	0.39/0.51/0.85	**0.44/0.50/0.83**	0.44/0.50/0.82
Team Effectiveness	9, 35, 36, 37, 38	0.55/0.59/0.81	0.25/0.60/0.84	0.19/0.61/0.87	**0.18/0.62/0.88**
Traceability	4, 8, 22, 30, 33, 36	0.46/0.48/0.72	0.45/0.56/0.75	**0.42/0.57/0.80**	0.40/0.55/0.78
Reproducibility	4, 8, 13, 22, 23, 35	0.58/0.32/0.59	0.76/0.47/0.86	**0.67/0.55/0.87**	0.65/0.55/0.85
Accountability	39, 40, 41, 42, 43, 44, 45	0.34/0.39/0.76	0.65/0.49/0.80	**0.31/0.24/0.81**	0.33/0.28/0.80

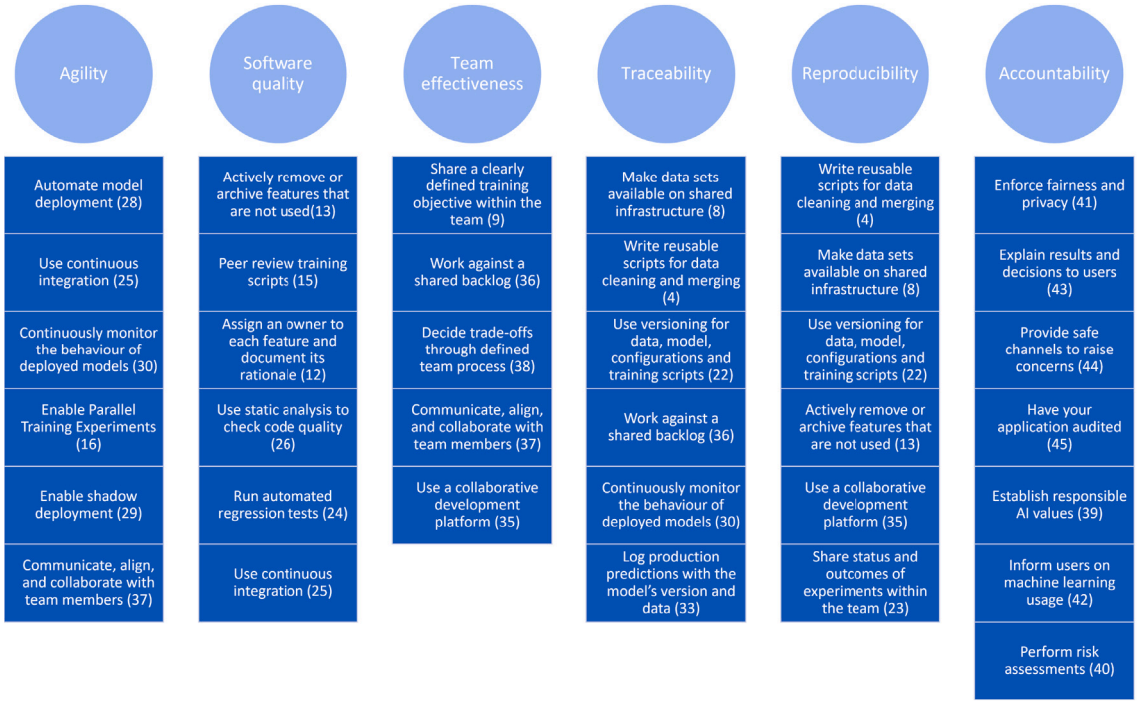


Fig. 11. Effects and the practices used to predict them, ordered by their importance.

Fig. 12. Practice importance to each effect studied as revealed by the RF Grid Search algorithm from Table 5.

predicted from the groups of practices. The effects and the practices that contribute to the effects are illustrated in Fig. 11.

The positive predictions allow us to suggest that adoption of groups of practices is efficient to achieve the effects studied. Moreover, the models allow us to study the contribution of each practice to the final effect.

Importance of practices. Using the models developed above, we further investigated the contribution of each practice to the effects we

Table 6
Practices for developing trustworthy ML applications and the requirements for trustworthy AI (High-Level Expert Group on AI, 2021) addressed by them.

Nr.	Title	Class	Requirement
14	Employ interpretable models whenever possible	Training	Human agency and oversight
38	Decide trade-offs through an established team process	Team	Human agency and oversight
27	Assure application security	Code	Technical robustness and safety
40	Perform risk assessments	Governance	Technical robustness and safety
7	Use privacy preserving ML techniques	Data	Privacy and data governance
39	Establish responsible AI values	Governance	Transparency
42	Inform users on ML usage	Governance	Transparency
43	Explain results and decisions to users	Governance	Transparency
44	Provide safe channels to raise concerns	Governance	Transparency
3	Test for social bias in training data	Data	Diversity, non-discrimination, fairness
6	Prevent discriminatory data attributes from being used as model features	Data	Diversity, non-discrimination, fairness
21	Assess and manage subgroup bias	Training	Diversity, non-discrimination, fairness
34	Provide audit trails	Deployment	Accountability
45	Have your application audited	Governance	Accountability

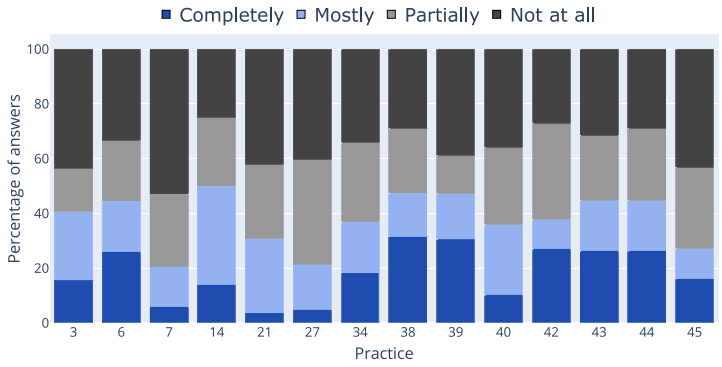


Fig. 13. Adoption of practices for developing trustworthy ML applications.

considered. This analysis allows us to determine which practices are most important for each effect. Moreover, we contrast the importance of the practices with their adoption rates – used as a proxy for their difficulty – to determine the return of investment when adopting a practice.

To evaluate the importance of the practices, we use the *Shapley* value – a well-known concept from cooperative game theory that is frequently used to explain ML models. Although in Table 5, the AutoML models perform similarly, and in one case better, we decided to use the grid search random forest (RF) models in our analysis, because of their simplicity and to maintain consistency across the analysis.

For all effects, we computed the Shapley values for both the training and test data sets and obtained consistent results for all models. The Shapley value for each practice reflects the increase in predictive accuracy caused by the practice averaged over all possible subsets of practices used in a model. To showcase the importance of the practices, we illustrate in Fig. 12 the Shapley values obtained from the test data together with the normalised practice ranks. Moreover, the practices in Fig. 11 are ordered based on the Shapley values.

We note that important practices are shown to have low adoption rates, but also that less important practices have high adoption rates. For example, practices (40) and (45) which are important for "Accountability" have very low adoption. Similarly, practices (4) and (8) are less important for "Reproducibility", but have high adoption rates. These results are not entirely unexpected, as other factors such as the criticality of the project, which could not be considered in our analysis, can influence the practice adoption rates.

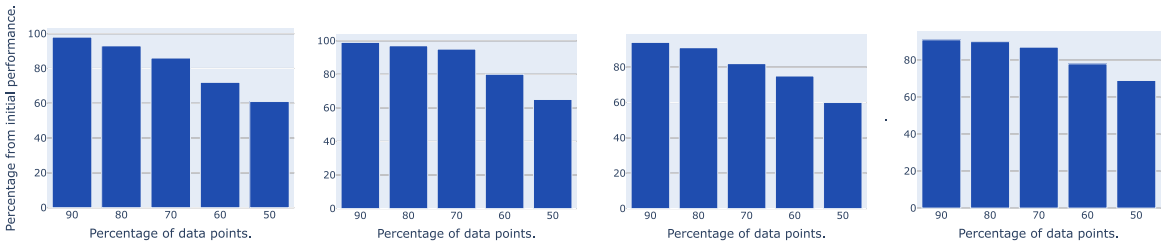
Nonetheless, since the effects can be well predicted from the practices, the analyses in this section can also be interpreted as a data-driven recommendation model for practice adoption conditioned by the desire to achieve certain effects. In light of this view, practitioners can choose to either adopt the practices ordered by their contribution to the effects, as revealed in Fig. 12, or take into consideration individual project constraints for their adoption. For example, practitioners who do not use feature-based ML can ignore practice 13 for software quality. As both a generalisable and project specific model is difficult to envision, the adoption of practices can always be determined in light of specific project constraints, and can be decided within teams of practitioners (i.e., following practice 38).

8. Analyses of individual sets of practices: Engineering practices for trustworthy machine learning applications

This section presents a different way of analysing the practices, considering the grouping by practice type. In addition to analyses by grouping practices based on the effect they contribute to, one can also analyse the practices grouped by practice type. As an example, in this section, we discuss the analysis of practices for engineering trustworthy ML applications from our second study. Similar analyses can be performed for other groups of practices, using the practice class grouping from Table 3 or any other custom grouping. For example, van der Blom et al. analysed the group of practices specific to AutoML to uncover the reasons which facilitate or inhibit their adoption (van der Blom et al., 2021). Their results are of interest not only to ML practitioners, but also to developers of AutoML tools and frameworks.

As mentioned in Section 3.2, the practices for engineering trustworthy ML were mined from the literature, guided by the requirements for trustworthy AI developed by the European Commission's high-level expert group on AI (High-Level Expert Group on AI, 2021). Our second survey yielded 126 responses, of which 95 were complete and suitable for this analysis (see Section 5 for details).

Table 6 presents a mapping between the engineering practices and the requirements for trustworthy AI (High-Level Expert Group on AI, 2021) addressed by them. We observe that all requirements are addressed by at least one practice, except "Societal and environmental well-being". Some practices, notably (3), (6) and (7), do address societal aspects, but in the context of diversity, non-discrimination



(a) Agility. (b) Software quality. (c) Team effectiveness. (d) Traceability.

Fig. 14. Impact of data set size on the effect prediction from sets of practices.

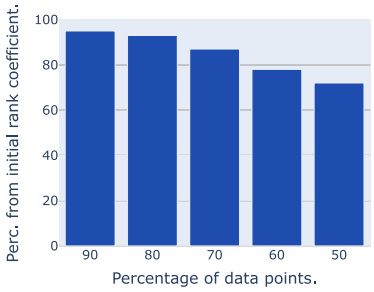


Fig. 15. Impact of data set size on practice correlations.

and fairness. Environmental well-being is only indirectly addressed by practice (39).

Fig. 13 presents the adoption of practices for trustworthy ML, omitting the answers specifically constructed to avoid noisy results. One such answer is for practice (14), where "using interpretable models is not possible for our use case" was used to account for applications where black box models are used.

Overall, the adoption of practices for trustworthy ML is rather low, as can be observed from the large percentage of "Not at all" and "Partially" answers (see Fig. 13). The least adopted practice – practice (27) on assuring application security – may raise concerns, considering the substantial interest in software and ML security from both academia and industry. Our conjecture is that the large body of work on security of ML (e.g., the work done on adversarial examples) is still limited in practical applicability.

Conversely, the practices related to establishing team processes for trade-off analysis (38), to establishing responsible AI values (39) and to explaining results to users (43) have higher adoption. This result could be due to more mature legislation in the area (such as the right to an explanation in the EU). Nonetheless, compared with other practice types, the adoption of practices for trustworthy ML is low (see Fig. 8).

As seen in this example, the analysis of focused groups of practices can help researchers and practitioners with diverse backgrounds to approach, investigate and understand concerns in isolation.

9. The potential of the collected data to support quality assessment models

This section analyses the robustness and representativeness of the collected data set, and explores a use-case in which the data can be used to support quality assessment models for teams developing software with ML components.

9.1. Robustness and representativeness analysis

As mentioned in Section 2, assessing the quality of systems with ML components or of the teams developing them is still an open and important problem. Given the broad range of practices from Table 3 and the relatively high number of responses from our survey, we consider that the data collected in this series of studies can be used to support such models. Particularly since completely measurable benchmarks – as in more traditional software engineering (Baggen et al., 2012) – are difficult to define because ML components add new dimensions to the software systems they are part of, such as data related concerns, model uncertainty and others (Serban et al., 2020b).

Measuring and comparing comprehensive practice adoption rates within teams can potentially alleviate the impact of hidden factors that affect the quality of ML systems, especially when the practices target fundamental concerns, intrinsic to quality. For example, the practices in the "Data" category target broad and fundamental concerns across the data life cycle – from data collection and labelling to bias testing – without which it is difficult to envision high quality ML software systems. Nevertheless, practice adoption rates measured through online questionnaires are subject to specific biases. Concretely, to support a general quality model the data volume must be sufficiently high to be robust against individual response biases such as teams not being able to adopt practices due to specific constraints (i.e., following the law of large numbers).

To test whether the data set collected in our study is robust, we performed a series of data tests. First, since the effects could be predicted from the practices with high accuracy, we evaluated the impact of the data set size on the results in Section 7. This analysis provides quantitative results about the impact of the data set size on underlying assumptions – such as whether the effects can be determined from sets of practices. For this analysis, we ran the experiments in Table 5 and randomly removed between 10 and 50 percent of the data. Afterwards, we evaluated the impact on the prediction performance (in this case the increase in MSE). The experiments were conducted using the same methodology as in Section 7, i.e., using 5-fold cross-validation and under-sampling to remove class imbalance. For each percentage of data removed, we sampled the data dropped randomly 5 times and report the average performance relative to using the full data set. The analysis ran for the first 4 effects, common to both studies. We plot the results for the grid search RF models in Fig. 14. We observe that removing up to 30% of the data maintains approximately 80% of the initial performance, for all effects studied. Therefore, we can consider the data set sufficiently large to draw robust conclusions.

Nevertheless, the effects studied only use individual sets of practices, and not the entire data set. Second, to collect deeper insights, we studied the correlation between practice adoption levels and the impact of data set size on these correlations. The correlations were evaluated using the Spearman rank coefficient (ρ). To determine the statistical significance of these correlations, we performed t-tests with a stringent significance level of 0.01. In total, we found over 300 statistically significant medium to strong correlations ($\rho > 0.4$), spread across all practices.

Examples of strong correlations include, e.g., practice 24 on running automated regression tests, which correlates with practice 26 on using static analysis for code quality ($\rho = 0.77$). This result indicates that traditional engineering practices aimed at improving the software quality are jointly adopted. Moreover, practice 30 on monitoring deployed

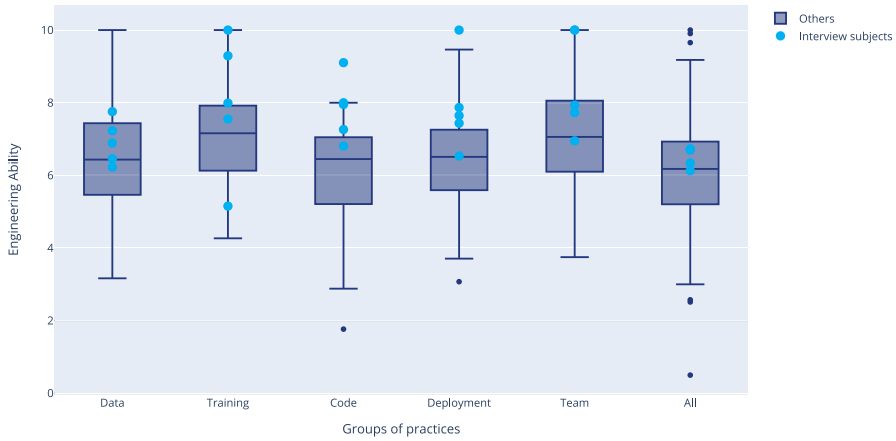


Fig. 16. Engineering ability for survey participants and for validation interview subjects.

models correlates strongly with practice 31 on performing checks to detect skews between models ($\rho = 0.72$) and with practice 33 on logging production predictions ($\rho = 0.71$). These results suggest that teams with advanced deployment and monitoring infrastructures tend to adopt (or neglect) more practices.

To evaluate the impact of data set size on the correlations, we randomly removed parts of the data set and recomputed the correlations. Similar to the previous experiment, we recompute the correlations 5 times, for each percentage of the data removed, and report the average decrease in the rank coefficient for strongly correlated practices, across all correlations. The results are illustrated in Fig. 15. Once again, the data set is robust enough to maintain approximately 80% of performance even when 40% of the data is removed. The results suggest robustness against outliers and representative results across multiple answers.

9.2. Exploratory use-case on assessing the engineering ability of ML teams

The analysis in Section 7 already revealed a potential path towards evaluating whether teams of practitioners adopting sets of practices achieve desired effects related to quality attributes. Conversely, the analysis can be used to assess and recommend improvements in case some teams want to improve certain quality aspects. In previous work, questionnaire data was also used to develop maturity models, e.g., Wendler, 2012; McGraw and Chess, 2009, to measure software evolution (Dekleva and Drehmer, 1997), usability (Tezza et al., 2011) or to perform other evaluations (Berges and Hubwieser, 2015). Since the data set appears to be robust against outliers and contains representative results (based on the previous subsection), in this section we also explore a use-case in which the data collected serves as a building block for models to evaluate the ability of teams to develop software with ML components. We comment on the potential benefits and downsides, which open promising avenues for future research.

In the proposed use-case, we consider comparing the practice adoption rates of a team against other teams in the collected data set. Engineering ability can then be assessed as, e.g., the percentile into which a given participant falls. Moreover, instead of considering the difficulty of moving from "Partially" to "Mostly", or from "Mostly" to "Completely" as equal for each practice, we would like to estimate the difficulty individually for each practice-and-answer combination using the data. To define such a benchmark, we borrow concepts from the theory of measurement – a field of study concerned with the objective measurement of knowledge and abilities from test data. In particular, we use item response theory (IRT), which is a method for scoring tests and measuring an individuals' ability on the test based on the ability of other test respondents (Embretson and Reise, 2013).

IRT can be used to compute ability from a team's responses to our practice-related survey questions, taking into account the relative difficulty of each practice, which in turn is determined from the distribution of answers to that question from the entire pool of respondents. We applied this model to assess the engineering ability of teams developing software with ML components, assuming the respondents are given the survey as an assessment. The model can be fitted for all practices as well as for single practices that may represent a topic of interest. For example, the overall engineering ability of a team can be assessed using the answers to all questions (or to all questions answered by a given respondent), while the engineering ability on groups of practices, e.g., practices in the "Data" category, can be assessed by considering only the practices from those groups. Moreover, the model can be personalised to remove practices considered irrelevant for specific teams – such as custom scenarios in which teams cannot adopt some practices due to external constraints.

Fig. 16 illustrates the results of running the assessment using the validation interview subjects, where the practices from the "Trustworthiness" class were omitted from the assessment because the interview subjects did not report major trustworthiness constraints (more details follow in Section 10.1). The ability scores were normalised between 0 and 10, similar to an exam grading score in the Netherlands, to make them easier to interpret. Here, "others" represents the set of all survey participants excluding the interview subjects and the distributions of engineering ability scores over this group are represented by boxplots. The interview subjects scored average or higher than average on most categories of practices, likely due to their previous experience with ML (Section 10.1). The plot also shows that the median engineering ability assessed on all practices evaluated is slightly above six, which indicates there is plenty of room for improvement for the teams taking the assessment.

Using such a model has several advantages. First, IRT evaluates the difficulty between different levels of practice adoption, and can be used to suggest improvements, which in turn can be weighted against the efforts needed to be spent for improvements. This results in novel, actionable, feedback for the assessed teams. Second, it provides a focused comparison to other teams in the data set, instead of comparing adoption rates for all or specific practices (as illustrated in the online catalogue). This comparison is potentially easier to understand, operate with and communicate. Third, the model is easy to extend to independent constraints, such as removing some practices that do not fit a project, or slicing the data using demographic factors – e.g., team size or experience.

Nevertheless, using the model also has disadvantages. First, the practice adoption is an imperfect proxy for ability. While some of this bias is alleviated by the robustness and representativeness of the data

Table 7
Profiles of the validation interview subjects.

Id	Company profile	Team size	Experience	Predominant data types	Predominant ML techniques
V1	Tech company	10–15 ppl.	>5 years	Time series, Tabular data	Supervised & Unsupervised
V2	Tech company	1–5 ppl.	2–5 years	Tabular data, Text	Supervised
V3	Tech company	1–5 ppl.	2–5 years	Tabular data, Text, Time Series	Supervised & Unsupervised
V4	Tech Startup	5–10 ppl.	2–5 years	Images, Videos	Supervised & Unsupervised

set, the assessment must be calibrated individually for each team to account for specific project constraints. Without individual calibration, not adopting a certain practice may in fact not reflect their inability to adopt it, but rather a conscious decision to put their effort elsewhere because the practice is not relevant or not (sufficiently) beneficial in the context of their project. Second, since the assessment does not take into account the time dimension, it needs to be evaluated multiple times, to track progress, or improvements. Last, the engineering ability is not comprehensive enough to cover all concerns needed for a quality model of software with ML components, and should be used as one building block out of potentially many others. For example, analysing stages such as requirements analysis or architectural design can reveal valuable information compared to implementation based practices.

To promote self-assessments, we publish together with the replication study a web app in which teams of practitioners can take the assessment and receive instant feedback (Serban et al., 2020a).

10. Discussion

We now present some additional discussion regarding the validation interviews and possible threats to validity.

10.1. Validation interviews

As mentioned in Section 3, to validate the outcomes of the first two stages of our studies (systematic literature review and survey) and to obtain deeper insights into the state of practice, we ran validation interviews. The interviews took a practical direction, focusing on whether the practices are actionable and the reasons why practitioners do not adopt some practices, or find them irrelevant, as practitioners' beliefs about software systems are likely to be unreliable (Shrikanth and Menzies, 2020; Shrikanth et al., 2021).

We invited interview participants from diverse teams that are known to develop and deploy software with ML components. To ensure the participants' prior experience with ML, we invited participants with at least two years of experience in developing software with ML components. The profile of the interview subjects is illustrated in Table 7, and the interview protocol was described in Section 3. We note that not all participants were originally trained for computer science, ML or SE, but entered the fields later. For example, one participant was trained in physics, while another entered into ML from another software related position. The experience column in Table 7 shows how long the participants have been active in one or more ML and SE related field. All participants confirmed that the teams they are part of are multidisciplinary and consist of members with diverse backgrounds.

Moreover, the participants worked on different application and model types, with distinct requirements. For example, (V1) worked with time series and strictly without deep learning in both unsupervised and supervised scenarios. In contrast, (V4) worked only with deep learning and image streams in supervised scenarios. This diversity allowed us to identify possible gaps or scenarios where the practices do not apply.

Before their interviews, the participants were asked to fill in the survey from our second study. During the interviews, we asked general questions regarding the survey and practices, followed by specific questions based on their survey answers. The latter allowed us to zoom in and collect data regarding issues with the adoption of the practices.

The interviews were processed using the methodology described in Section 3, and during the high-order theme modelling step several themes emerged regarding (i) the challenges faced by the participants, (ii) the ability of the survey to convey the intended information and cover the challenges, (iii) the actionability of the practices and the reasons for their adoption, (iv) the most important practices, (v) specific comments on classes of practices based on the classification from Fig. 3.

From the challenges reported, integrating multidisciplinary team members with distinct backgrounds, defining good tests for ML models as well as handling model monitoring at scale, given the large amounts of logs generated emerged as most challenging. For example, (V2) mentioned that compatibility between different versions of ML models is difficult to test and define. (V3) also mentioned that designing and implementing test scenarios for ML is a daunting but very critical task. (V2) acknowledged that code review can mediate some of the problems arising in the definition and selection of test cases. However, because not all team members have training in software engineering, code review was not straightforward to adopt. The difficulty of adopting test practices is also reflected in our survey results; for example, relatively low adoption rates were observed for practice (11) on testing the feature extraction code and for practice (24) on running automated regression tests.

All participants appreciated the depth and breath of the catalogue and did not consider any of the practices or the survey questions ill-suited or redundant. One participant, with a well-developed interest in ML engineering, noted that the catalogue is an eye opener regarding the large spectrum of engineering concerns spanned by ML. Moreover, all participants considered the survey suited for its intended purpose and did not report any comments regarding hypothetical confusing question formulations or answers that did not cover their use-cases.

Regarding the actionability of the practices, participants acknowledged that the practices are indeed actionable and sound, and could potentially be adopted in their projects. However, they all reported that complete practice adoption is project dependent and adopting the comprehensive set of practices is not always feasible given the iterative nature of ML projects. Especially since the outcome of ML projects is uncertain, and the effort of adopting some practices might not pay off. Nevertheless, using the catalogue as guidance during the different iterations of a ML project was deemed helpful, as participants could rapidly select which practices to adopt given the components which need stricter engineering during a specific iteration. Providing multiple methods to classify the practices and their difficulty, such as the ranking discussed in Section 6 and the practice importance per effect as discussed in Section 7 are viable methods to help navigate the catalogue. Developing new methods to classify the practices – such as trying to abstract common iterations within ML projects and to suggest practices constrained by the breath of each iteration – is an interesting avenue for future research. Such studies could be run using, for example, focused groups with participants while they develop different iterations of one project.

The participants also mentioned that the adoption of practices relies heavily on the tool stack used and on the maturity of the tools available. For example, (V3) and (V1) used a mature deployment framework that facilitated the adoption of deployment practices, such as automated deployment and rollback (practice 28) or continuous monitoring of deployed models (practice 30). (V2) reported on the development of

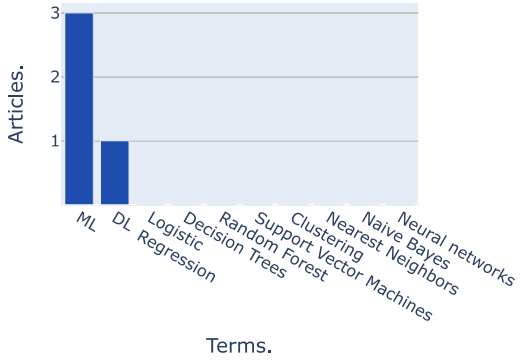


Fig. 17. Identified relevant additional articles when extending the list of terms for the SLRs.

an internal framework for feature management which, once adopted by all teams, will facilitate the adoption of feature management practices, such as documenting features (practice 12). Moreover, (V4) reported on the development of a labelling tool that could facilitate retraining and integration in other pipelines.

The engineering practices that emerged as most important are related to testing and monitoring ML models. The participants appreciated that the catalogue targets engineering practices covering these topics. Nevertheless, they still reported on the difficulty of choosing and implementing different tests. While complementing the existing catalogue with more practices for testing and monitoring ML models is feasible, the practices are project and data specific. Therefore, they will conflict with our goal of having a general catalogue of engineering practices. Moreover, the topic is well covered in the referenced related work, which serves as support for practitioners to delve deeper into the specifics of each method (Ishikawa and Yoshioka, 2019; Breck et al., 2017).

Participants also provided specific comments on the classes of practices, from which we note the importance of applying strict processes for handling, processing and labelling data (V2, V3) which were covered in depth in our catalogue, the necessity to enforce testing practices for ML systems (V1, V3, V4), the necessity to separate the two realms of code for ML and the applications surrounding ML and applying the coding practices to both realms (V1, V4) and the need to adopt frameworks for model deployment that enforce the practices from our catalogue (V4).

Regarding the practices for trustworthy ML, the participants reported general awareness and efforts spent towards adopting or reasoning more on trustworthiness, but with little practical results. For example, (V4) reported that users are requested to fill in a declaration regarding fair use of personal data, and that users do not object to this, since the problem their application solves cannot be solved without personal data. Similarly, (V1) and (V2) reported a special request must be filled in by developers to access and use personal or sensitive data for their models. However, if the performance of the models is linked to the use of sensitive data, the request is always granted. The fact that this procedure effectively ignores the sensitivity of the data in favour of performance gains (by always approving the request in this situation) emphasises the need for practice (44) from Table 3. This practice is about providing safe channels to raise concerns, which would enable team members to report this as a possible issue. The participants acknowledged the relevance of the practices for trustworthy ML in our catalogue, but also mentioned the need to enforce such practices in a top-down manner, using strict requirements for their adoption. Otherwise, engineers tend to overlook such practices.

Overall, the interviews sparked fruitful discussions and confirmed the timely need for engineering practices in ML. Moreover, the participants confirmed the relevance of the catalogue of practices developed in our study.

10.2. Threats to validity

We identified five potential threats to the validity of our study. First, the systematic literature reviews (SLRs) may be subject to bias by missing or excluding relevant articles. To mitigate this bias, we used multiple sources for information retrieval and complemented the academic literature with grey literature. The individual researchers' bias during data extraction was limited by using a data extraction form, allowing consistency in data analysis and through discussions between all authors of this paper. To evaluate the impact of the terms selected for the first element of the queries, we ran similar queries including abbreviations for machine learning (ML) and deep learning (DL), and more ML methods such as "nearest neighbors", "decision trees", "gradient boosting" and others. The number of articles that were selected for manual inspection, for each new element of the queries is illustrated in Fig. 17. The manual inspection revealed that the recommendations from the new articles were already covered by the practices. For example, deciding on a measurable objective function (Pratt, 2019) performing code reviews (Krčah, 2017) or applying clean code principles (Krčah, 2017).

Second, the data from our surveys may be subject to bias. As discussed in Section 5, some groups are over-represented, which may introduce selection bias. Nonetheless, the analysis of the practice adoption grouped by the demographic factors from the surveys revealed that no significant bias was introduced by this over-representation.

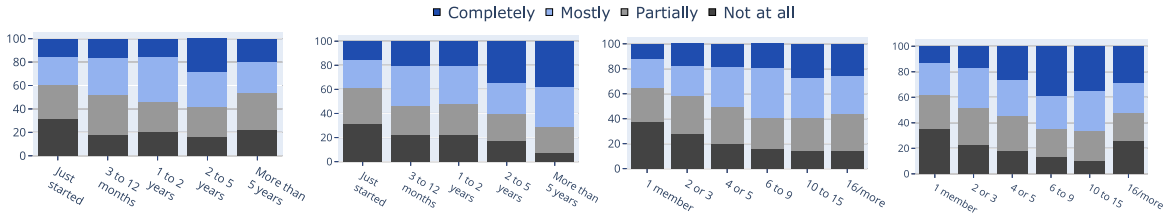
Third, the analysis of the relationship between groups of practices and their intended effects may be subject to bias, since we used the perceived effects as a proxy for the real effects. We note that establishing actual effects and determining the relationship between perceived and true effects demands a different type of study and is an important and ambitious direction for future research.

Fourth, the interview validation study may be subject to bias. To limit this bias, we invited diverse participants, both in terms of background, companies, team profiles and projects. We also used two strategies to alleviate memory bias, i.e., we asked participants to take the survey before the interviews and asked open ended questions at the beginning of the interviews. Moreover, we assured participants of confidentiality and anonymisation, removing the need to be concerned about providing answers that might reflect badly on their team or organisation.

Last, some bias may be introduced by merging the results from our surveys. To test this assumption, we illustrate in Fig. 18 the adoption of practices grouped by team experience and size for both surveys. We observe that the adoption trends are very similar, which provides clear evidence that no major bias is introduced by merging the results from the two surveys.

11. Conclusions

We studied technical and organisational practices used by teams to develop, deploy and maintain software with ML components, as well as practices used to achieve non-functional properties of ML components that fall under the umbrella of trustworthy ML. Our series of two studies consisted of three stages, organised as follows. In the first stage, a catalogue of engineering practices for ML was mined from literature through a systematic literature review (SLR). Next, in the second stage, we ran a survey with practitioners to validate the practices and measure their adoption and their effects. Last, in the third stage, we ran validation interviews to question the outcomes of the first two stages and get deeper insights into the state-of-practice. We ran this process twice: in the first study, with a focus on engineering practices for the entire ML development life-cycle, and in the second study with a focus on practices for developing trustworthy ML components. Moreover, in this study, we ran stages 2 and 3 again, for a combined analysis of the results.



(a) Team experience first study.

(b) Team experience second study.

(c) Team size first study.

(d) Team size second study.

Fig. 18. Practice adoption rates for the two studies..

Based on the survey results, we reported on the demographic factors that influence practice, adoption and on the relationships between the practices. For example, we found out that increasing team size or recruiting team members with higher experience positively impacts practice adoption, up to a point. Moreover, our results show that tech companies have more mature processes for ML than non-tech or governmental organisations, indicating substantial room for improvement for the latter two. on the effects of adopting the practices, and contrasted the adoption of practices with their contribution to the desirable effects. This analysis revealed that some highly adopted practices have a low impact on their corresponding effects and the opposite. The results suggest deeper analyses can support practice adoption plans by prioritising those with the largest expected impact.

The validation interviews added depth to the results and produced additional insights into the state of practice. The participants appreciated the relevance of the practices and mentioned the necessity of pairing practices with high-quality tool stacks.

Using the answers from our surveys, we also created a maturity model that estimates the engineering ability of teams developing software with ML components. The model allows teams to benchmark their adoption against other teams in our data set, scores the practices by difficulty and suggests possible improvements.

We also explored the robustness and representativeness of the data collected, and presented a use case in which the data can be used to support future quality models for software with ML components.

Future research. For future work, we propose to add a time dimension to our benchmarking effort. Conducting such surveys at different points in time, we would be able to monitor the practice adoption and observe adoption trends. Such analyses could be paired with observations regarding interventions, which may establish causal relationships between practice adoption and their effects.

Our results regarding adoption and effects of software engineering practices for ML can be used as a basis for developing various practical instruments for increasing, supporting, and auditing practice adoption. For example, developers of ML tooling and frameworks may use our results to build practice-support directly into development and deployment environments. Also, training and (self-) assessment instruments can be developed and organised to reflect the purpose, difficulty, and importance of each practice, as well as their relationships to various challenges and quality attributes. Finally, the practices in our catalogue can be used as building blocks for audit instruments, either for internal audit, or external audit by regulators or investors (in the context of transaction due diligence).

Although our focus here has been strictly on ML, it may also be interesting to expand our approach and findings to other areas within the broader field of AI.

CRediT authorship contribution statement

Alex Serban: Conceptualization, Methodology, Investigation, Validation, Software, Data curation, Writing – original draft, Writing – review & editing. **Koen van der Blom:** Conceptualization, Methodology, Investigation, Validation, Software, Data curation, Writing – original draft, Writing – review & editing. **Holger Hoos:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision. **Joost Visser:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data and code is publicly available on Zenodo and referenced in the article and a link is provided in the submission as well.

Appendix. Complete list of terms for SLR

See Tables 8 and 9.

Table 8
List of terms for SLR first study.

First terms	Second terms
Machine learning	Software engineering
Deep Learning	Software development
	Data labeling
	Data labeling practices
	Development
	Development practices
	Versioning
	Versioning practices
	Review
	Review practices
	Deployment
	Deployment practices
	Operation
	Monitoring
	Production
	Maintenance

Table 9
List of terms for SLR second study.

First terms	Second terms
Machine learning	Robustness
Deep Learning	Safety
	Privacy
	Data governance
	Transparency
	Fairness
	Bias
	Ethical
	Accountability

References

Akkiraju, Rama, Sinha, Vibha, Xu, Anbang, Mahmud, Jalal, Gundecha, Pritam, Liu, Zhe, Liu, Xiaotong, Schumacher, John, 2020. Characterizing machine learning processes: A maturity framework. In: International Conference on Business Process Management. Springer, pp. 17–31.

Algorithmia, 2019. Best practices in machine learning infrastructure. https://algorithmia.com/blog/best-practices-in-machine-learning-infrastructure. (Online; Accessed 15 December 2021).

Alkis Polyzotis Martin A Zinkevich Steven Whang Sudip Roy, 2017. Data management challenges in production machine learning. https://storage.googleapis.com/pub-tools-public-publication-data/pdf/45a9dcf23dbdfa24dbced358f825636c58518afa.pdf. (Online; Accessed 15 December 2021).

Altexsoft, 2018. How to organize data labelling for machine learning. https://www.altexsoft.com/blog/datascience/how-to-organize-data-labeling-for-machine-learning-approaches-and-tools/. (Online; Accessed 15 December 2021).

Amershi, Saleema, Begel, Andrew, Bird, Christian, DeLine, Robert, Gall, Harald, Kamar, Ece, Nagappan, Nachiappan, Nushi, Besmira, Zimmermann, Thomas, 2019. Software engineering for machine learning: A case study. In: ICSE-SEIP. IEEE, pp. 291–300.

Arpteg, Anders, Brinne, Björn, Crnkovic-Friis, Luka, Bosch, Jan, 2018. Software engineering challenges of deep learning. In: Euromicro Conference on Software Engineering and Advanced Applications. SEAA, IEEE, pp. 50–59.

Baggen, Robert, Correia, José Pedro, Schill, Katrin, Visser, Joost, 2012. Standardized code quality benchmarking for improving software maintainability. Softw. Qual. J. 20 (2), 287–307.

Baylor, Denis, Breck, Eric, Cheng, Heng-Tze, Fiedel, Noah, Foo, Chuan Yu, Haque, Zakaria, Haykal, Salem, Ispir, Mustafa, Jain, Vihan, Koc, Levent, et al., 2017. TFX: A tensorflow-based production-scale machine learning platform. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1387–1395.

Begel, Andrew, Nagappan, Nachiappan, 2007. Usage and perceptions of agile software development in an industrial context: An exploratory study. In: Empirical Software Engineering and Measurement. ESEM, IEEE, pp. 255–264.

Berges, Marc, Hubwieser, Peter, 2015. Evaluation of source code with item response theory. In: Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education. pp. 51–56.

Bogner, Justus, Verdecchia, Roberto, Gerostathopoulos, Ilias, 2021. Characterizing technical debt and antipatterns in AI-based systems: A systematic mapping study. In: 2021 IEEE/ACM International Conference on Technical Debt. TechDebt, IEEE, pp. 64–73.

Booch, Grady, Brown, Alan W., 2003. Collaborative development environments. Adv. Comput. 59 (1), 1–27.

Bosch, Jan, Olsson, Helena Holmström, Crnkovic, Ivica, 2021. Engineering AI systems: A research agenda. In: Artificial Intelligence Paradigms for Smart Cyber-Physical Systems. IGI Global, pp. 1–19.

Bouwers, Eric, Deursen, Arie van, Visser, Joost, 2014. Towards a catalog format for software metrics. In: Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics. pp. 44–47.

Branke, Jürgen, Deb, Kalyanmoy, Miettinen, Kaisa, Slowiński, Roman, 2008. Multiobjective Optimization: Interactive and Evolutionary Approaches. In: Lecture Notes in Computer Science, vol. 5252, Springer.

Braun, Virginia, Clarke, Victoria, 2006. Using thematic analysis in psychology. Qual. Res. Psychol. 3 (2), 77–101.

Breck, Eric, Cai, Shanqing, Nielsen, Eric, Salib, Michael, Sculley, D., 2016. What's your ML test score? A rubric for ML production systems. In: Reliable Machine Learning in the Wild - NeurIPS Workshop.

Breck, Eric, Cai, Shanqing, Nielsen, Eric, Salib, Michael, Sculley, D., 2017. The ML test score: A rubric for ML production readiness and technical debt reduction. In: International Conference on Big Data (Big Data). IEEE, pp. 1123–1132.

Breuel, Cristiano, 2019. ML Ops: Machine learning as an engineered disciplined. https://towardsdatascience.com/ml-ops-machine-learning-as-an-engineering-discipline-b86ca4874a3f. (Online; Accessed 15 December 2021).

Brundage, Miles, Avin, Shahar, Wang, Jasmine, Belfield, Haydn, Krueger, Gretchen, et al., 2020. Toward trustworthy AI development: Mechanisms for supporting verifiable claims. arXiv:2004.07213.

Chakraborty, Joymallya, Majumder, Suvodeep, Menzies, Tim, 2021. Bias in machine learning software: Why? How? What to do? arXiv:2105.12195.

Chatila, Raja, Havens, John C., 2019. The IEEE global initiative on ethics of autonomous and intelligent systems. Robot. Well-Being 11–16.

Chawla, Nitesh V., Bowyer, Kevin W., Hall, Lawrence O., Kegelmeyer, W. Philip, 2002. SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.

Ciolkowski, Marcus, Laitenberger, Oliver, Vegas, Sira, Biffl, Stefan, 2003. Practical experiences in the design and conduct of surveys in empirical software engineering. In: Empirical Methods and Studies in Software Engineering. Springer, pp. 104–128.

Cloudfactory, 2019. The ultimate guide to data labeling for ML. https://www.cloudfactory.com/data-labeling-guide. (Online; Accessed 15 December 2021).

Cruz, Luis, Abreu, Rui, 2019. Catalog of energy patterns for mobile applications. Empir. Softw. Eng. 24 (4), 2209–2235.

Cruzes, Daniela S., Dyba, Tore, 2011. Recommended steps for thematic synthesis in software engineering. In: International Symposium on Empirical Software Engineering and Measurement. ESEM, IEEE, pp. 275–284.

De Souza Nascimento, Elizamary, Ahmed, Iftekhar, Oliveira, Edson, Palheta, Márcio Piedade, Steinmacher, Igor, Conte, Tayana, 2019. Understanding development process of machine learning systems: Challenges and solutions. In: ESEM. IEEE, pp. 1–6.

Dean, Jeff, 2019. An overview of google's work on automl and future directions. https://slideslive.com/38917526/an-overview-of-googles-work-on-automl-and-future-directions. (Online; accessed 11-05-2021).

Dekleva, Sasa, Drehmer, David, 1997. Measuring software engineering evolution: A rasch calibration. Inf. Syst. Res. 8 (1), 95–104.

Dunning, Ted, Friedman, Ellen, 2019. Machine learning logistics. https://mapr.com/ebook/machine-learning-logistics/. (Online; Accessed 15 December 2021).

Easterbrook, Steve, Singer, Janice, Storey, Margaret-Anne, Damian, Daniela, 2008. Selecting empirical methods for software engineering research. In: Guide To Advanced Empirical Software Engineering. Springer, pp. 285–311.

Embretson, Susan E., Reise, Steven P., 2013. Item Response Theory. Psychology Press.

Fandel, Günter, 1990. Group decision making: Methodology and applications. In: Readings in Multiple Criteria Decision Aid. Springer, pp. 569–605.

Faraj, Samer, Sproull, Lee, 2000. Coordinating expertise in software development teams. Manage. Sci. 46 (12), 1554–1568.

Feurer, Matthias, Klein, Aaron, Eggensperger, Katharina, Springenberg, Jost, Blum, Manuel, Hutter, Frank, 2015. Efficient and robust automated machine learning. In: NeurIPS 2015. pp. 2962–2970.

Fowler, Martin, 2018. Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional.

Gamma, Erich, Helm, Richard, Johnson, Ralph, Vlissides, John, 1994. Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education.

Garousi, Vahid, Felderer, Michael, Mäntylä, Mika V., 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Inf. Softw. Technol. 106, 101–121.

Giray, Görkem, 2021. A software engineering perspective on engineering machine learning systems: State of the art and challenges. J. Syst. Softw. 180, 111031.

Golendukhina, Valentina, Lenarduzzi, Valentina, Felderer, Michael, 2022. What is software quality for AI engineers? towards a thinning of the fog. In: Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI. pp. 1–9.

Google AI Blog, 2019a. Fairness indicators. https://ai.googleblog.com/2019/12/fairness-indicators-scalable.html. (Online; Accessed 15 December 2021).

Google AI Blog, 2019b. Responsible AI practices. https://ai.google/responsibilities/responsible-ai-practices/. (Online; Accessed 15 December 2021).

Google Devs, 2019. Testing and debugging in machine learning. https://developers.google.com/machine-learning/testing-debugging/pipeline/production. (Online; Accessed 15 December 2021).

Hardt, Moritz, 2020. Fairness. https://www.youtube.com/watch?v=Igq_S_7IfOU. (Online; Accessed 15 December 2021).

Hébert-Johnson, Úrsula, Kim, Michael, Reingold, Omer, Rothblum, Guy, 2018. Multicalibration: Calibration for the (computationally-identifiable) masses. In: ICML. PMLR, pp. 1939–1948.

Herron, David, 2019. Principled machine learning: Practices and tools for efficient collaboration. https://dev.to/robogeek/principled-machine-learning-4eho. (Online; Accessed 15 December 2021).

High-Level Expert Group on AI, 2021. Ethics guidelines for trustworthy AI. https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai. (Online; Accessed 15 December 2021).

Hopkins, Aspen, Booth, Serena, 2021. Machine learning practices outside big tech: How resource constraints challenge responsible development. In: Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society. AIES '21, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450384735, pp. 134–145. http://dx.doi.org/10.1145/3461702.3462527.

Hove, Siw Elisabeth, Anda, Bente, 2005. Experiences from conducting semi-structured interviews in empirical software engineering research. In: International Software Metrics Symposium. IEEE, pp. 10–20.

Humbatova, Nargiz, Jahangirova, Gunel, Bavota, Gabriele, Riccio, Vincenzo, Stocco, Andrea, Tonella, Paolo, 2020. Taxonomy of real faults in deep learning systems. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. pp. 1110–1121.

Hummer, Waldemar, Muthusamy, Vinod, Rausch, Thomas, Dube, Parijat, El Maghraoui, Kaoutar, Murthi, Anupama, Oum, Punleuk, 2019. ModelOps: Cloud-based lifecycle management for reliable and trusted AI. In: 2019 IEEE International Conference on Cloud Engineering. IC2E, IEEE, pp. 113–120.

Hutter, Frank, Kotthoff, Lars, Vanschoren, Joaquin (Eds.), 2019. Automated Machine Learning: Methods, Systems, Challenges. Springer.

International Organization for Standardization, 2016. Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): Measurement of System and Software Product Quality. ISO.

Ishikawa, Fuyuki, Yoshioka, Nobukazu, 2019. How do engineers perceive difficulties in engineering of machine-learning systems?: questionnaire survey. In: Proceedings of the Joint 7th International Workshop on Conducting Empirical Studies in Industry and 6th International Workshop on Software Engineering Research and Industrial Practice. IEEE, pp. 2–9.

John, Per, 2019. Software development best practices in a deep learning environment. https://towardsdatascience.com/software-development-best-practices-in-a-deep-learning-environment-a1769e9859b1. (Online; Accessed 15 December 2021).

Kearns, Michael, Neel, Seth, Roth, Aaron, Wu, Zhiwei Steven, 2018. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In: ICML. PMLR, pp. 2564–2572.

Khomh, Foutse, Adams, Bram, Cheng, Jinghui, Fokaefs, Marios, Antoniol, Giuliano, 2018. Software engineering for machine-learning applications: The road ahead. IEEE Softw. 35 (5), 81–84.

Kitchenham, Barbara, Charters, Stuart, 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Citeseer.

Kitchenham, Barbara A., Pfleeger, Shari L., 2008. Personal opinion surveys. In: Guide To Advanced Empirical Software Engineering. Springer, pp. 63–92.

Kleinberg, Jon, Mullainathan, Sendhil, Raghavan, Manish, 2016. Inherent trade-offs in the fair determination of risk scores. arXiv:1609.05807.

Krčah, Marcel, 2017. From a model to production like a pro: Software-engineering best-practices. https://www.youtube.com/watch?v=MKrPXfvIWoc&t=465s. (Online; accessed 15-04-2023).

Kumeno, Fumihiro, 2019. Sofware engneering challenges for machine learning applications: A literature review. Intell. Decis. Technol. 13 (4), 463–476.

Kuwajima, Hiroshi, Yasuoka, Hirotoshi, Nakae, Toshihiro, 2020. Engineering problems in machine learning systems. Mach. Learn. 109 (5), 1103–1126.

Lämmel, Ralf, Visser, Joost, 2002. Design patterns for functional strategic programming. In: Proceedings of the 2002 ACM SIGPLAN Workshop on Rule-Based Programming. pp. 1–14.

Lavin, Alexander, Gilligan-Lee, Ciarán M, Visnjic, Alessya, Ganju, Siddha, Newman, Dava, Ganguly, Sujoy, Lange, Danny, Baydin, Atılım Güneş, Sharma, Amit, Gibson, Adam, et al., 2021. Technology readiness levels for machine learning systems. arXiv:2101.03989.

Le, James, 2019. 10 Best practices for deep learning. https://nanonets.com/blog/10-best-practices-deep-learning/. (Online; Accessed 15 December 2021).

Lenarduzzi, Valentina, Lomio, Francesco, Moreschini, Sergio, Taibi, Davide, Tamburri, Damian Andrew, 2021. Software quality for AI: Where we are now? In: International Conference on Software Quality. Springer, pp. 43–53.

Lewis, Grace A., Bellomo, Stephany, Ozkaya, Ipek, 2021. Characterizing and detecting mismatch in machine-learning-enabled systems. In: 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI. WAIN, IEEE, pp. 133–140.

Li, Shuyue, Guo, Jiaqi, Lou, Jian-Guang, Fan, Ming, Liu, Ting, Zhang, Dongmei, 2022. Testing machine learning systems in industry: an empirical study. In: Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice. pp. 263–272.

Liu, Lydia T., Simchowitz, Max, Hardt, Moritz, 2019. The implicit fairness criterion of unconstrained learning. In: ICML. PMLR, pp. 4051–4060.

Lu, Qinghua, Zhu, Liming, Xu, Xiwei, Whittle, Jon, Douglas, David, Sanderson, Conrad, 2022. Software engineering for responsible AI: An empirical study and operationalised patterns. In: 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice. ICSE-SEIP, IEEE, pp. 241–242.

Lwakatare, Lucy Ellen, Raj, Aiswarya, Bosch, Jan, Olsson, Helena Holmström, Crnkovic, Ivica, 2019. A taxonomy of software engineering challenges for machine learning systems: An empirical investigation. In: International Conference on Agile Software Development. Springer, pp. 227–243.

Martínez-Fernández, Silverio, Bogner, Justus, Franch, Xavier, Oriol, Marc, Siebert, Julien, Trendowicz, Adam, Vollmer, Anna Maria, Wagner, Stefan, 2021. Software engineering for AI-based systems: A survey. arXiv:2105.01984.

Mayo, Matthew, 2017. The current state of automated machine learning. https://www.kdnuggets.com/2017/01/current-state-automated-machine-learning.html. (Online; Accessed 15 December 2021).

McGraw, Gary, 2004. Software security. IEEE Secur. Priv. 2 (2), 80–83.

McGraw, Gary, Chess, Brian, 2009. The building security in maturity model (bsimmss).

Megler, V.M., 2019. Managing machine learning projects. https://d1.awsstatic.com/whitepapers/aws-managing-ml-projects.pdf. (Online; Accessed 15 December 2021).

Microsoft Blog, 2019. How do teams work together on automated ML projects. https://azure.microsoft.com/en-us/blog/how-do-teams-work-together-on-an-automated-machine-learning-project/. (Online; Accessed 15 December 2021).

Mikkonen, Tommi, Nurminen, Jukka K, Raatikainen, Mikko, Fronza, Ilenia, Mäkitalo, Niko, Männistö, Tomi, 2021. Is machine learning software just software: A maintainability view. In: International Conference on Software Quality. Springer, pp. 94–105.

Mitchell, Margaret, Wu, Simone, Zaldivar, Andrew, Barnes, Parker, Vasserman, Lucy, Hutchinson, Ben, et al., 2019. Model cards for model reporting. In: Conference on Fairness, Accountability, and Transparency. PMLR, pp. 220–229.

Mojica-Hanke, Anamaria, Bayona, Andrea, Linares-Vásquez, Mario, Herbold, Steffen, González, Fabio A, 2023. What are the machine learning best practices reported by practitioners on stack exchange? arXiv preprint arXiv:2301.10516.

Molnar, Christoph, 2020. Interpretable Machine Learning. Lulu.com.

Muccini, Henry, Vaidhyanathan, Karthik, 2021. Software architecture for ML-based systems: What exists and what Lies ahead. arXiv preprint arXiv:2103.07950.

Myllyaho, Lalli, Raatikainen, Mikko, Männistö, Tomi, Nurminen, Jukka K, Mikkonen, Tommi, 2022. On misbehaviour and fault tolerance in machine learning systems. J. Syst. Softw. 183, 111096.

Nakamichi, Koji, Ohashi, Kyoko, Namba, Isao, Yamamoto, Rieko, Aoyama, Mikio, Joeckel, Lisa, Siebert, Julien, Heidrich, Jens, 2020. Requirements-driven method to determine quality characteristics and measurements for machine learning software and its evaluation. In: 2020 IEEE 28th International Requirements Engineering Conference. RE, IEEE, pp. 260–270.

Nashimoto, Kane, Wright, F.T., 2007. Nonparametric multiple-comparison methods for simply ordered medians. Comput. Stat. Data Anal. 51 (10), 5068–5076.

National Science and Technology Council (US). Select Committee on Artificial Intelligence, 2021. The national artificial intelligence research and development strategic plan: 2019 update. https://www.nitrd.gov/news/National-AI-RD-Strategy-2021.aspx. (Online; Accessed 15 December 2021).

Palinkas, Lawrence A., Horwitz, Sarah M., Green, Carla A., et al., 2015. Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. In: Administration and Policy in Mental Health and Mental Health Services Research, Vol. 42, No. 5. Springer, pp. 533–544.

Pratt, Lorien, 2019. Machine learning software engineering: Top five best practices. https://www.youtube.com/watch?v=MKrPXfvIWoc&t=465s. (Online; accessed 15-04-2023).

Prendki, Jennifer, 2018. The curse of big data labeling and three ways to solve it. https://aws.amazon.com/blogs/apn/the-curse-of-big-data-labeling-and-three-ways-to-solve-it/. (Online; Accessed 15 December 2021).

Raji, Inioluwa Deborah, Smart, Andrew, White, Rebecca N, Mitchell, Margaret, Gebru, Timnit, et al., 2020. Closing the AI accountability gap: defining an end-to-end framework for internal algorithmic auditing. In: Conference on Fairness, Accountability, and Transparency. PMLR, pp. 33–44.

Roh, Yuji, Heo, Geon, Whang, Steven Euijong, 2019. A survey on data collection for machine learning: a big data-ai integration perspective. IEEE Trans. Knowl. Data Eng. 33 (4), 1328–1347.

Ruhe, Günther, 2002. Software engineering decision support–a new paradigm for learning software organizations. In: International Workshop on Learning Software Organizations. Springer.

Sapp, Carlton, 2017. Preparing and architecting for machine learning. https://www.gartner.com. (Online; Accessed 15 December 2021).

Sato, Danilo, Wider, Arif, Windheuser, Christoph, 2019. Continuous delivery for machine learning. https://martinfowler.com/articles/cd4ml.html. (Online; Accessed 15 December 2021).

Schnellbach, Adam, Griessnig, Gerhard, 2019. Development of the ISO 21448. In: European Conference on Software Process Improvement. Springer, pp. 585–593.

Sculley, David, Holt, Gary, Golovin, Daniel, Davydov, Eugene, Phillips, Todd, Ebner, Dietmar, Chaudhary, Vinay, Young, Michael, Crespo, Jean-Francois, Dennison, Dan, 2015. Hidden technical debt in machine learning systems. In: NeurIPS 2015. pp. 2503–2511.

Sedano, Todd, Ralph, Paul, Péraire, Cécile, 2019. The product backlog. In: ICSE. IEEE, pp. 200–211.

Serban, Alex, van der Blom, Koen, Hoos, Holger, Visser, Joost, 2020a. Adoption and Effects of Software Engineering Best Practices in Machine Learning - Supplementary Material. Zenodo, http://dx.doi.org/10.5281/zenodo.3946453.

Serban, Alex, Poll, Erik, Visser, Joost, 2020b. Towards using probabilistic models to design software systems with inherent uncertainty. In: European Conference on Software Architecture. ECSA, Springer.

Serban, Alex, van der Blom, Koen, Hoos, Holger, Visser, Joost, 2020c. Adoption and effects of software engineering best practices in machine learning. In: International Symposium on Empirical Software Engineering and Measurement. ESEM, ACM, ISBN: 9781450375801, pp. 1–12. http://dx.doi.org/10.1145/3382494.3410681.

Serban, Alex, van der Blom, Koen, Hoos, Holger, Visser, Joost, 2021. Practices for engineering trustworthy machine learning applications. In: IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI. WAIN, IEEE, pp. 97–100.

Serban, Alex, Visser, Joost, 2022. Adapting software architectures to machine learning challenges. In: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering. SANER, IEEE, pp. 152–163.

Seyffarth, Roman, 2019. Machine learning: Moving from experiments to production. https://blog.codecentric.de/en/2019/03/machine-learning-experiments-production/. (Online; Accessed 15 December 2021).

Shahin, Mojtaba, Babar, Muhammad Ali, Zhu, Liming, 2017. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. IEEE Access 5, 3909–3943.

Shrikanth, N.C., Menzies, Tim, 2020. Assessing practitioner beliefs about software defect prediction. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice. ICSE-SEIP, IEEE, pp. 182–190.

Shrikanth, N.C., Nichols, William, Fahid, Fahmid Morshed, Menzies, Tim, 2021. Assessing practitioner beliefs about software engineering. Empir. Softw. Eng. 26 (4), 1–32.

Sridhar, Vinay, Subramanian, Sriram, Arteaga, Dulcardo, Sundararaman, Swaminathan, Roselli, Drew, Talagala, Nisha, 2018. Model governance: Reducing the anarchy of production ML. In: 2018 USENIX Annual Technical Conference. USENIX ATC 18, pp. 351–358.

Storey, Margaret-Anne, Zagalsky, Alexey, Figueira Filho, Fernando, Singer, Leif, German, Daniel M., 2016. How social and communication channels shape and challenge a participatory culture in software development. IEEE Trans. Softw. Eng. 43 (2), 185–204.

Sutherland, Jeff, Schwaber, Ken, 2013. The scrum guide. In: The Definitive Guide to Scrum: The Rules of the Game. Scrum.org, Vol. 268.

Talagala, Nisha, 2018. Operational machine learning. https://www.kdnuggets.com/2018/04/operational-machine-learning-successful-mlops.htmls. (Online; Accessed 15 December 2021).

Tezza, Rafael, Bornia, Antonio Cezar, De Andrade, Dalton Francisco, 2011. Measuring web usability using item response theory: Principles, features and opportunities. Interact. Comput. 23 (2), 167–175.

Torgo, Luís, Ribeiro, Rita P., Pfahringer, Bernhard, Branco, Paula, 2013. SMOTE for regression. In: Portuguese Conference on Artificial Intelligence. Springer, pp. 378–389.

Tunguz, Bojan, 2020. Six levels of auto ML. https://medium.com/@tunguz/six-levels-of-auto-ml-a277aa1f0f38. (Online; accessed 11-05-2021).

Usman, Muhammad, Sun, Youcheng, Gopinath, Divya, Dange, Rishi, Manolache, Luca, Pasareanu, Corina S, 2022. An overview of structural coverage metrics for testing neural networks. arXiv:2208.03407.

van der Blom, Koen, Serban, Alex, Hoos, Holger, Visser, Joost, 2021. AutoML adoption in ML software. In: 8th ICML Workshop on Automated Machine Learning. AutoML.

van der Weide, Tom, Papadopoulos, Dimitris, Smirnov, Oleg, Zielinski, Michal, van Kasteren, Tim, 2017. Versioning for end-to-end machine learning pipelines. In: Proceedings of the 1st Workshop on Data Management for End-To-End Machine Learning. pp. 1–9.

Villamizar, Hugo, Escovedo, Tatiana, Kalinowski, Marcos, 2021. Requirements engineering for machine learning: A systematic mapping study. In: 2021 47th Euromicro Conference on Software Engineering and Advanced Applications. SEAA, IEEE, pp. 29–36.

Visser, Joost, Rigal, Sylvan, Wijnholds, Gijs, van Eck, Pascal, van der Leek, Rob, 2016. Building Maintainable Software, C# Edition: Ten Guidelines for Future-Proof Code. O'Reilly Media, Inc..

Wan, Zhiyuan, Xia, Xin, Lo, David, Murphy, Gail C., 2019. How does machine learning change software development practices? IEEE Trans. Softw. Eng. 47 (9), 1857–1871.

Washizaki, Hironori, Khomh, Foutse, Guéhéneuc, Yann-Gaël, Takeuchi, Hironori, Natori, Naotake, Doi, Takuo, Okuda, Satoshi, 2022. Software-engineering design patterns for machine learning applications. Computer 55 (3), 30–39.

Washizaki, Hironori, Uchida, Hiromu, Khomh, Foutse, Guéhéneuc, Yann-Gaël, 2019. Studying software engineering patterns for designing machine learning systems. In: 10th International Workshop on Empirical Software Engineering in Practice. IWESEP, IEEE, pp. 49–495.

Weiss, Michael, Tonella, Paolo, 2021. Fail-safe execution of deep learning based systems through uncertainty monitoring. In: 2021 IEEE 14th International Conference on Software Testing, Validation and Verification. ICST, IEEE.

Wendler, Roy, 2012. The maturity of maturity model research: A systematic mapping study. Inf. Softw. Technol. 54 (12), 1317–1339.

Wirth, Rüdiger, Hipp, Jochen, 2000. CRISP-DM: Towards a standard process model for data mining. In: Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining. Springer, pp. 29–39.

ZelrosAI, 2019. The Four Maturity Levels of Automated Machine Learning: Towards Domain Specific AutoML. https://tinyurl.com/2p8aw746. (Online; accessed 11-05-2021).

Zhang, Jie M., Harman, Mark, Ma, Lei, Liu, Yang, 2020. Machine learning testing: Survey, landscapes and horizons. IEEE Trans. Softw. Eng. 48 (1), 1–36.

Zinkevich, Martin, 2019. Rules of machine learning: Best practices for ML engineering. https://developers.google.com/machine-learning/guides/rules-of-ml. (Online; Accessed 15 December 2021).