Adoption and Effects of Software Engineering Best Practices in Machine Learning

Alex Serban
a.serban@cs.ru.nl
ICIS, Radboud University
Software Improvement Group
Amsterdam, The Netherlands

Koen van der Blom
Holger Hoos
Joost Visser
LIACS, Leiden University
Leiden, The Netherlands

ABSTRACT

Background. The increasing reliance on applications with machine learning (ML) components calls for mature engineering techniques that ensure these are built in a robust and future-proof manner.
Aim. We aim to empirically determine the state of the art in how teams develop, deploy and maintain software with ML components.
Method. We mined both academic and grey literature and identified 29 engineering best practices for ML applications. We conducted a survey among 313 practitioners to determine the degree of adoption for these practices and to validate their perceived effects. Using the survey responses, we quantified practice adoption, differentiated along demographic characteristics, such as geography or team size. We also tested correlations and investigated linear and non-linear relationships between practices and their perceived effect using various statistical models.
Results. Our findings indicate, for example, that larger teams tend to adopt more practices, and that traditional software engineering practices tend to have lower adoption than ML specific practices. Also, the statistical models can accurately predict perceived effects such as agility, software quality and traceability, from the degree of adoption for specific sets of practices. Combining practice adoption rates with practice importance, as revealed by statistical models, we identify practices that are important but have low adoption, as well as practices that are widely adopted but are less important for the effects we studied.
Conclusion. Overall, our survey and the analysis of responses received provide a quantitative basis for assessment and step-wise improvement of practice adoption by ML teams.

CCS CONCEPTS

• **Software and its engineering → Software development methods**.

KEYWORDS

survey, best practices, machine learning engineering

ESEM '20, October 8–9, 2020, Bari, Italy

ACM ISBN 978-1-4503-7580-1/20/10…$15.00
https://doi.org/10.1145/3382494.3410681

ACM Reference Format:
Alex Serban, Koen van der Blom, Holger Hoos, and Joost Visser. 2020. Adoption and Effects of Software Engineering Best Practices in Machine Learning. In *ESEM '20: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '20), October 8–9, 2020, Bari, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3382494.3410681

1 INTRODUCTION

The adoption of ML components in production-ready applications demands strong engineering methods to ensure robust development, deployment and maintenance. While a wide body of academic literature acknowledges these challenges [4, 30, 32, 36, 56], there is little academic literature to guide practitioners. In fact, a large part of the literature concerning engineering practices for ML applications can be classified as grey literature [24] and consists of blog articles, presentation slides or white papers.

In this work, we aim to determine the state of the art in how teams develop, deploy and maintain software solutions that involve ML components. Towards this goal, we have first distilled a set of 29 engineering best practices from the academic and grey literature. These practices can be classified as *traditional* practices, which apply to any software application, *modified* practices, which were adapted from traditional practices to suit the needs of ML applications, and completely *new* practices, designed for ML applications.

In order to validate the adoption and relevance of the practices we ran a survey among ML practitioners, with a focus on teams developing software with ML components. The survey was designed to measure the adoption of practices and also to assess the effects of adopting specific sets of practices. We obtained 313 valid responses and analysed 29 practices and their influence on 4 different effects.

The main contributions of our work are as follows. Firstly, we summarise academic and grey literature in a collection of best practices. This body of information can be used by practitioners to improve their development process and serves as a gateway to literature on this topic. Secondly, we determine the state of the art by measuring the adoption of the practices. These results are used to rank the practices by adoption level and can serve to assess the popularity of particular practices. Thirdly, we investigate the relationship between groups of practices and their intended effects, through different lenses – by training a linear regression model to check if the intended effect is dependent on the practices and by training more sophisticated regression models, using a variety of ML approaches (including AutoML) to predict the effects from the practices. Lastly, we investigate the adoption of practices based on

the data type being processed and based on the practice categories introduced above (traditional, modified, new).

Our results suggest that the practices apply universally to any ML application and are largely independent of the type of data considered. Moreover, we found a strong dependency between groups of practices and their intended effect. Using the contribution of each practice to the desired effect (extracted from our predictive models) and their adoption rate, we outline a method for prioritising practice improvements tailored for achieving specific effects, such as increased traceability or software quality. While our study is restricted to ML, rather than the broader and less clearly delineated field of artificial intelligence (AI), many of our findings may have wider applications, as we will briefly discuss in Section 8.

The remainder of the paper is organised as follows. We first discuss background and related work (Section 2). Next, we describe the process and results of mining practices from literature (Section 3). A description of the design of our study (Section 4) is followed by a presentation of the survey results regarding the adoption of practices (Section 5) and a deeper analysis of the relationship between the practices and their effects (Section 6). Finally, we discuss interpretation and limitations of our findings (Section 7) and close with general conclusions and remarks on future work (Section 8). Our survey questions, data, and code for analysis and visualisation are publicly available [47].

2 BACKGROUND AND RELATED WORK

Engineering challenges posed by ML. As ML components are developed and deployed, several engineering challenges specific to the ML software development life-cycle emerge [4, 30, 32, 36, 56]. Arpteg et al. [4] identified a set of 12 challenges that target development, deployment and organisational issues. In particular, managing and versioning data during development, monitoring and logging data for deployed models and estimating the effort needed to develop ML components present striking differences with the development of traditional software components.

Similarly, Ishikawa and Yoshioka [30] as well as Wan et al. [56] have studied how software engineers perceive the challenges related to ML and how ML changes the traditional software development life-cycle. Both studies ran user surveys with a majority of respondents from Asia. We could not find a similar study without this regional bias. Nonetheless, both publications concluded that testing and ensuring the quality of ML components is particularly difficult, because a test oracle is missing, the components often behave nondeterministically, and test coverage is hard to define. In order to better classify the challenges raised by ML components, Lwakatare et al. introduced a comprehensive taxonomy [36].

White and grey literature analysis. In search for ways to meet the challenges presented earlier, we mined the literature and collected software engineering (SE) best practices for ML. We observed that the majority of literature on this topic consists of so called grey literature [24] – i.e., blog articles, presentation slides or white papers from commercial companies – while there is relatively little academic literature. Garousi et al. [24] showed that, if used properly, grey literature can benefit SE research, providing valuable additional information. However, this literature must be used with care, because it does not contain strong empirical evidence to support its claims [25]. We decided to included the grey literature in our literature search, using the process described by Garousi et al. [24], because: (1) coverage of the subject by academic literature is rather incomplete, (2) contextual information is important for the subject of study – i.e., practitioners may have different opinions than scientists on what qualifies as best practices – and (3) grey literature may corroborate scientific outcomes with practical experience.

Related work. We focus on peer-reviewed related work that proposes, collects or validates engineering best practices for ML. One of the initial publications on this topic is the work of Sculley et al. [45], which used the framework of technical debt to explore risk factors for ML components. In particular, they argued that ML components have a stronger propensity to incur technical debt, because they have all maintenance problems specific to traditional software as well as a set of additional issues specific to ML. They also presented a set of anti-patterns and practices aimed at avoiding technical debt in systems using ML components. Compared to [45], we introduce a broader set of practices, applicable to more effects than technical debt. Nonetheless, some of their suggestions, which are specific to engineering, are included in our catalogue of practices.

Breck et al. [12] introduced 28 tests and monitoring practices that target different stages of the development process for ML. They also proposed a list of benefits resulting from implementing the tests and developed a model to score test practice adoption, aimed at measuring technical debt. Again, the practices dedicated to SE from [12] have been included in our catalogue. On the same topic, Zhang et al. introduced a survey on testing techniques for ML components [59], which – in contrast to the broader approach taken in [12] – only targets testing ML models.

To identify challenges faced by small companies in developing ML applications, de Souza Nascimento et al. ran interviews with 7 developers [17]. Afterwards, they proposed and validated a set of checklists to help developers overcome the challenges faced. Although the validation session is not thorough (it only included a focus group with 2 participants), some of the items in the checklists qualify as best practice candidates. Some of these practices are included in our catalogue and our survey further confirms their relevance and adoption.

Washizaki et al. [57] studied and classified software architecture design patterns and anti-patterns for ML, extracted from white and grey literature. Many of these patterns are application and context specific, i.e., they depend on the architectural style or on the type of data used. The patterns are of a general character and the ones similar to recommendations we found in literature were included in our catalogue of practices.

Amershi et al. conducted a study internally at Microsoft, aimed at collecting challenges and best practices for SE used by various teams in the organisation [3]. They reported on a broad range of challenges and practices used at different stages of the software development life cycle. Using the experience of the respondents and the set of challenges, they built a maturity model to assess each team. However, the set of challenges and reported practices are broad and often not actionable. Moreover, they represent the opinions of team members from Microsoft, where typically more resources are dedicated to ensuring adoption of best practices than

Table 1: Successful search queries. The table shows the base queries, for which any variant (described in text) led to a valid source and at least one practice.

Query	Documents
software engineering for machine learning	[3]
data labeling best practices	[2, 16, 39, 40]
machine learning engineering practices	[13, 44, 60]
software development machine learning	[31]
machine learning production	[42, 48]
machine learning production practices	[1, 5, 34, 38]
machine learning deployment	[18]
machine learning deployment practices	[43]
machine learning pipelines practices	[28]
machine learning operations	[52]
machine learning versioning	[54]
machine learning versioning practices	[26]

within smaller companies. In our work, we aim to bridge this gap by running a survey with practitioners with various backgrounds and by presenting a set of actionable, fine-grained best practices.

3 MINING PRACTICES FROM LITERATURE

Document Search. In addition to the publications discussed in Section 2, we searched the academic and grey literature on the topic of SE best practices for ML applications. We used both Google and Google Scholar, for which we compiled a common set of queries. The keywords used for querying suggest different steps in the development cycle, e.g., development, deployment, operations, etc. For each query, we also developed two variants, by (1) replacing the term 'machine learning' with 'deep learning' whenever possible, and (2) removing stop words and composing a Boolean AND query from the remaining key words. As an example of the second variant, consider the query "software engineering" AND "machine learning", stemming from the query "software engineering for machine learning". All queries were submitted to Google and Google Scholar, and the first 5 result pages were manually inspected.

A total of 64 queries, including variants, were used, and 43 of the resulting articles were selected for initial inspection. In order to avoid search engine personalisation, all queries were sent from a public network, with an extension that removes browser cookies.

Document classification. Based on criteria formulated in [24], such as authoritativeness of the outlet and author as well as objectivity of the style and content, we excluded low-quality documents and classified the remaining documents as either academic literature or grey literature. Moreover, we filtered for duplicates, because chunks of information were sometimes reused in grey literature.

After classifying and filtering the results, we identified 21 relevant documents, including scientific articles, white papers, blogs and presentation slides, that – along with the publications introduced in Section 2 – were used to mine SE best practices for ML. Other relevant sources were selected through a snowball strategy, by following references and pointers from the initial articles.

Table 1 lists the successful search terms (without variants), from which at least one document passed the final selection. Whenever the queries had common results, we only considered relevant the first query. The second column in Table 1 shows the documents selected from the base queries and their variants.

Extracting a common taxonomy for the practices. Many of the selected documents provide, or implicitly presume, a grouping of practices based on development activities specific to ML. For example, Amershi et al. [3] present a nine-stage ML pipeline. Alternatively, Sato et al. [44] partition similar activities into six pipeline stages. All processes have roots in early models for data mining, such as CRISP-DM [58].

While no single partitioning of ML activities emerged as most authoritative, we were able to reconstruct a broad taxonomy that is compatible with all partitionings found in the literature. We will use this categorisation to group ML development practices and to structure our survey and subsequent discussion of our findings:

- Data - Practices that come before training, including collecting and preparing data for training ML models.
- Training - Practices related to planning, developing and running training experiments.
- Deployment - Practices related to preparing a model for deployment, deploying, monitoring and maintaining an ML model in production.
- Coding - Practices for writing, testing, and deploying code.
- Team - Practices related to communication and alignment in a software development team.
- Governance - Practices that relate to ensuring responsible use of ML, including accountability regarding privacy, transparency, and usage of human, financial, or energy resources.

Compiling a catalogue of practices. From the selected documents we compiled an initial set of practices using the following methodology. First, we identified all practices, tests or recommendations that had similar goals. In some articles, the recommendations only suggest the final goal – e.g., ensure that trained ML models can be traced back to the data and training scripts used – without providing details on the steps needed to achieve it. In other publications, the recommendations provided detailed steps used to achieve the goals – e.g., use versioning for data, models, configurations and training scripts [38, 48, 54]. In this example, traceability is an outcome of correctly versioning all artefacts used in training. Whenever we encountered similar scenarios, we selected or abstracted actionable practices and added the high-level goals to a special group, which we call "Effects" and describe in Table 6.

Next, we assessed the resulting practices and selected those specifically related to engineering or to the organisation of engineering processes. This initial selection gave us 23 practices, which naturally fall into 4 out of the 6 classes introduced above. While this set of practices reflected the ML development process, it lacked practices from traditional SE. Given that practitioners with a strong background in ML might be unaware of the developments in SE, in a third stage, we complemented the initial set of practices with 6 practices from traditional SE – three of a strictly technical nature, falling into the "Coding" class, and three relating to social aspects,

Table 2: SE best practices for ML, grouped into 6 classes, together with the practice type, literature references and adoption ranks, where N – new practice, T – traditional practice, M – modified practice.

Nr.	Title	Class	Type	References	Rank
1	Use Sanity Checks for All External Data Sources	Data	N	[13, 42]	22
2	Check that Input Data is Complete, Balanced and Well Distributed	Data	N	[6, 12, 38, 42, 45]	18
3	Write Reusable Scripts for Data Cleaning and Merging	Data	N	[1, 13, 42]	5
4	Ensure Data Labelling is Performed in a Strictly Controlled Process	Data	N	[2, 16, 39, 40]	11
5	Make Data Sets Available on Shared Infrastructure (private or public)	Data	N	[26, 31, 32, 38]	14
6	Share a Clearly Defined Training Objective within the Team	Training	N	[9, 38, 60]	2
7	Capture the Training Objective in a Metric that is Easy to Measure and Understand	Training	N	[9, 19, 52, 60]	1
8	Test all Feature Extraction Code	Training	M	[12, 44]	23
9	Assign an Owner to Each Feature and Document its Rationale	Training	M	[60]	29
10	Actively Remove or Archive Features That are Not Used	Training	N	[45, 60]	28
11	Peer Review Training Scripts	Training	M	[11]	20
12	Enable Parallel Training Experiments	Training	N	[44, 48]	6
13	Automate Hyper-Parameter Optimisation and Model Selection	Training	N	[29, 37]	26
14	Continuously Measure Model Quality and Performance	Training	N	[18, 60]	4
15	Share Status and Outcomes of Experiments Within the Team	Training	N	[26, 34]	7
16	Use Versioning for Data, Model, Configurations and Training Scripts	Training	M	[26, 28, 34, 38, 48, 54, 57]	3
17	Run Automated Regression Tests	Coding	T	[12, 48]	27
18	Use Continuous Integration	Coding	T	[12, 44]	16
19	Use Static Analysis to Check Code Quality	Coding	T	[55]	24
20	Automate Model Deployment	Deployment	M	[20, 43, 52, 54]	15
21	Continuously Monitor the Behaviour of Deployed Models	Deployment	N	[5, 18, 20, 44, 48]	12
22	Enable Shadow Deployment	Deployment	M	[5, 20, 54, 57]	25
23	Perform Checks to Detect Skews between Models	Deployment	N	[5, 18, 44, 60]	17
24	Enable Automatic Roll Backs for Production Models	Deployment	M	[20, 44]	13
25	Log Production Predictions with the Model's Version and Input Data	Deployment	M	[28, 38, 49]	19
26	Use A Collaborative Development Platform	Team	T	[10, 50]	8
27	Work Against a Shared Backlog	Team	T	[46, 51]	9
28	Communicate, Align, and Collaborate With Multidisciplinary Team Members	Team	T	[21]	10
29	Enforce Fairness and Privacy	Governance	N	[7, 8, 12]	21

falling into the "Team" class. We selected these practices because we consider them challenging, yet essential in software development.

The resulting 29 practices are listed in Table 2 and the effects in Table 6. The practices are available to practitioners in a more elaborate format in an online catalogue[1], consisting of detailed descriptions and concise statements of intent, motivation, related practices, references and an indication of difficulty. A curated reading list with these references, further relevant literature as well as a selection of supporting tools is maintained online[2].

4 STUDY DESIGN

We validated the set of practices with both researchers and practitioners through a survey. For this, we designed a descriptive questionnaire asking respondents if the *team* they are part of adopts, in their ML projects, the practices we identified earlier. Before distributing the survey, we interviewed five practitioners with diverse backgrounds, in order to check if any information from the survey was redundant or whether important practices were missing.

Questionnaire. In designing the questionnaire used in our survey, we followed the recommendations of Kitchenham et al. [33] and Ciolkowski et al. [15]. We designed a cross-sectional observational study [33], i.e., participants were asked at the moment of filling the questionnaire if they adopted the recommended practices. Several preliminary questions were designed to specifically assign participants to groups. This renders the study a concurrent control study, in which participants are not randomly assigned to groups [33].

The target audience were *teams* of practitioners using ML components in a project. Specific preliminary questions were added to allow filtering between teams that build and deploy ML applications, use ML and do not build an application or do not use ML at all. We consider that a significant amount of engineering is also needed in research (where ML may be used without building deployed applications), especially in running large-scale deep learning experiments, and would like to verify which practices are relevant

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

Table 3: 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

in this context. Team profile (e.g., tech company, governmental organisation), team size (e.g., 1 person, 6-9 persons), team experience (e.g., most of us have 1-2 years of experience), and the types of data used (e.g., tabular data, images) were also included in the preliminaries. In total, the preliminaries contained 5 questions that were later used to group participants and filter out noise.

Then, 31 questions followed, with standard answers, mapped onto the practices from Table 2. In two cases, multiple questions map onto the same practice; for example, continuous integration is achieved by automating the build process and running it at each commit. In the questionnaire, we asked two questions, one for each action, although we compiled the answers to one best practice.

We used standard answers, on a Likert scale with four possible answers, in order to avoid the middle null-point strategy of answering [27]. The labels were chosen in order to reflect the degree of adoption, rather than the level of agreement [41]. This allowed the practices to be expressed impartially – e.g. , "our software build process is fully automated" – and the answers to express degrees of adoption – e.g., "not at all" or "completely" – instead of degrees of agreement such as "agree" or "strongly agree". This strategy eliminated confusing questions and answers, which may lead to an extreme null-point bias [27]. Whenever the answer scale did not match the full range of answers, we added specific answers which helped to avoid noisy results; for example, in the questions about data labelling, we added the answer "we do not use data labels", which accounts for unsupervised learning scenarios.

The questionnaire ended with a section on the perceived effects of adopting the practices. This enabled us to test the hypothesis that adopting a group of practices helps to achieve an effect. The four questions on perceived effects are shown in Table 6.

Although the questionnaire has 45 questions, we employ optimisation techniques, such as automatically moving to the next question once an answer is selected, to reduce the time required for completing our questionnaire to 7 minutes on average.

Pilot interviews. Before distributing the survey broadly, we invited five practitioners with distinct backgrounds – ranging from tech startups to large tech companies – to an interview. We asked them to answer a set of questions regarding the challenges they face and the most important engineering practices they adopt. All interviewees were also asked to fill out and comment on the questionnaire. Since the survey is focused on teams, in Table 3 we present the team and company profiles for each interviewee; all interviewees use ML for a project. Moreover, P4 is part of a team that builds platforms to support the ML process and uses distinct ML projects to test the platforms.

The biggest challenges faced by the interviewees were: ensuring data quality and data documentation (P5), data versioning and freshness (P2), scalability (P1, P4) and communication with other departments inside the company (P5). For each challenge mentioned, there is at least one practice addressing it in Table 2. The most important engineering practices mentioned were: using version control for data and models (P1, P4), continuous deployment (P2, P5) and model maintenance (P2). Several practices to address these challenges were already listed in Table 2.

After completing the questionnaire, all interviewees agreed with the relevance of all the practices we listed and did not consider any of them redundant. The interviewees suggested that some questions needed additional allowable answers, to cover the range of possible responses and to avoid bias. For example, for a question about the labelling process, we added "we do not use labels" to avoid forcing users of unsupervised learning to resort to "not at all".

We used the feedback from the interviews to refine the questionnaire, adding answers to four questions and rewording others.

Distribution. After the pilot interviews, our survey was distributed using a snowball strategy. Initially, we reached out to our network of contacts and to the authors of the publications used to extract the practices, asking them to distribute the survey through their networks. Moreover, we openly advertised the survey through channels commonly used by practitioners, such as Twitter, Medium, HackerNoon, Dev.to and the Meetup groups for ML in several cities.

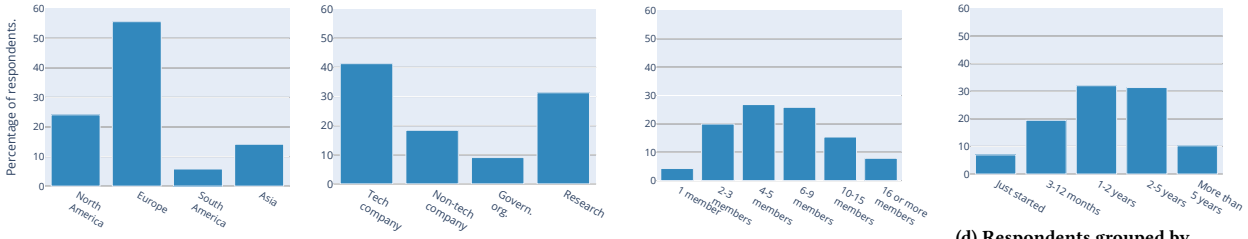
5 FINDINGS ON PRACTICE ADOPTION

In total we obtained 350 valid responses to our survey, after filtering out incomplete answers or respondents that spent too little time to have given serious answers (under 2 minutes). From this initial set, we discarded 12 answers from respondents who were not part of a team using ML. Moreover, we applied fine-grained filtering, using the percentage of questions that were answered in the prerequisites (at least 50 %) and in the practice adoption questions (at least 50 %), resulting in *313 complete responses*. Whenever not mentioned otherwise, the analysis will be based on these responses.

Demographics. Using the initial preliminary questions, we provide a demographic characterisation of the respondents in Figure 1. Firstly, we grouped the answers using the location attributes and present the results in Figure 1a. We observe that Europe has an overall higher contribution, although other regions are also well represented. This possible bias will be discussed later in this section, when analysing the answers for each region.

Figure 1b illustrates the percentage of respondents grouped by the organisation type. The higher percentages are for teams working in tech companies (e.g. social media platforms, semiconductors) and research labs. These results are not surprising, since both research and adoption of ML is driven by these two classes of practitioners. Nonetheless, non-tech companies (e.g. enterprises, banks) and governmental organisations are also well represented.

In the last two plots we show the percentage of answers grouped by team size – Figure 1c – and team experience – Figure 1d. We observe that most teams have between 4-5 and 6-9 members, corresponding to normal software development teams (as recommended, for example, in Agile development). Similarly, most teams have



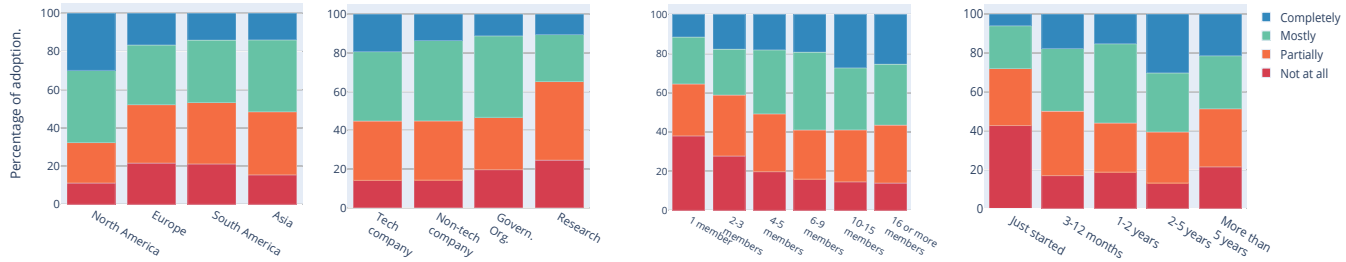
(a) Respondents grouped by regions.

(b) Respondents grouped by organisation type.

(c) Respondents grouped by team size.

(d) Respondents grouped by team experience.

Figure 1: 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 experience.

Figure 2: 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.

between 1-2 years and 2-5 years of experience, which is an anticipated result, since these intervals correspond to the recent growth in popularity of ML among practitioners. Overall, the demographic information indicates that our data is well balanced and diverse.

Next, we analysed the adoption of practices grouped by the demographic factors introduced earlier. We display the answers from the practice questions in Figure 2, grouped and normalised using the Likert scale used in the survey. Figure 2a shows the percentage of answers grouped by regions. As discussed earlier, Europe is somewhat over-represented in our data set. However, the adoption of practices for Europe does not present striking differences when compared to South America or Asia. Conversely, the respondents from North America have a significant higher number of adopted practices (corresponding to answers "Completely" or "Mostly") than other regions. Since this region is well represented in our set of responses, it is likely that practitioners from North America have a higher adoption of practices. Moreover, since Europe does not present striking differences with other regions, it is likely that little bias is introduced by its over-representation.

Figure 2b shows the adoption of practices grouped by the organisation type. We observe that tech companies have a higher rate of complete adoption than others. Research organisations tend to have lower practice adoption. This could reflect that they are aware of the practices, but only develop prototypes, for which adoption is not needed, or partial adoption is sufficient. In fact, for non-deployment practices only, adoption rates are similar.

For team size – Figure 2c – we observe a trend towards higher adoption of practices (and also lower percentage of practices that were not adopted at all) as team size increases. This could be caused by better task distribution among team members, or it could be a result of larger teams including members with different backgrounds. Similarly, for team experience, there is a trend towards higher adoption of practices as the team experience increases, as seen in Figure 2d. These results were anticipated, since more experience or a deeper involvement in technology exposes team members to the knowledge needed to adopt best practices. A contrasting trend can be observed only for teams with more than 5 years of experience, where the percentage of practices that are only partially or not at all adopted increases slightly. This result may reveal that practitioners who started very early may be unaware of practices that are developed recently.

These results confirm our questions were clear and achieved their goals, and that the answer scale did not introduce bias.

Practice adoption ranking. We now explore the adoption of practices, based on the practice types discussed in Section 1. In particular, we are interested in finding out whether traditional SE practices are equally adopted in ML engineering and which new or modified practices are popular among practitioners. Moreover, we also comment on the least and most adopted practices.

The practices are classified as follows: (1) *new* practices, designed specifically for the ML process, (2) *modified* practices, derived from

Table 4: Adoption of practices based on the practice type.

Practice Type	At least high adoption	At least medium adoption	At least low adoption
Traditional	15.6%	47.8%	76.8%
Modified	11.3%	42.0%	76.9%
New	**16.9%**	**50.0%**	**83.9%**

traditional SE practices, but adapted for the ML process and (3) *traditional* practices, applied equally in traditional SE and ML. This classification is illustrated in the "Type" column of Table 2.

In order to measure the adoption rate of the practices, we devised a ranking algorithm with the following steps:

(1) Compute for each practice the percentage h of respondents with *at least high* adoption (counting "Completely" answers), the percentage m with *at least medium* adoption (counting "Completely" and "Mostly"), and the percentage l with *at least low* adoption (counting "Completely", "Mostly", and "Partially"). As an example, for practice 1 we obtained $h = 9.62\%$, $m = 34.04\%$, and $l = 65.92\%$.
(2) Convert each percentage into a rank number. For practice 1, we obtained $r_h = 22$, $r_m = 23$, and $r_l = 19$.
(3) Take the average of the three ranks for each practice and then rank the practices according to this average. For practice 1, rank 22 was obtained, as can be seen in Table 2.

Thus, the final rank is the average of: the practice rank on *at least high adoption*, its rank on *at least medium adoption*, and its rank on *at least low adoption*. By accumulating the answers in step 1, we expect to cancel out the noise stemming from fuzzy boundaries between subsequent answer types.

The results are presented in the "Rank" column of Table 2, where the highest rank corresponds to the most adopted practice. We observe that the most adopted practices (practices 6, 7) are related to establishing and communicating clear objectives and metrics for training. Versioning (practice 16), continuous monitoring of the model during experimentation (practice 14) and writing reusable scripts for data management (practice 3) complete the top 5 positions. It is interesting to note that the most adopted practices are either new or modified practices, and not traditional SE practices.

At the other end of the spectrum, we observe that the least adopted practices (practices 9, 10) are related to feature management. Writing tests (practice 17), automated hyper-parameter optimisation (practice 13) and shadow deployment (practice 22) complete the 5 least adopted practices. In general, the least adopted practices require more effort and knowledge. Some practices, related to testing (practices 8, 17) or documentation (practice 9) are also known to raise issues in traditional SE. Moreover, shadow deployment (practice 22) and automated hyper-parameter optimisation (practice 13) require advanced infrastructure.

In order to compare the adoption of practices grouped by their type, we averaged the three percentages described earlier (without transforming them into ranks), for each practice type. The results are presented in Table 4. We observe that the most adopted practices are new practices, specifically designed for ML, followed by traditional and modified practices. Traditional practices in the "Team" category are ranked highly, since collaborative development platforms have become common tools among practitioners and offer good support for information sharing and coordination inside a team. In contrast, traditional practices related to code quality, such as running regression tests (practice 17) or using static analysis tools to check code quality (practice 19), have low adoption.

Table 5: Adoption of practices based on the data type.

Data Type	Perc. of respondents	Adoption		
		At least high	At least medium	At least low
Tabular Data	31.7%	18.0%	50.1%	70.2%
Text	29.7%	19.3%	52.6%	71.4%
Images, Videos	26.4%	19.3%	50.5%	71.5%
Audio	8.8%	24.42%	55.8%	72.6%
Time Series	2.6%	28.2%	60.3%	72.6%
Graphs	0.5%	-	-	-

Influence of data type on practice adoption. The practices presented in Table 2 are general and should apply to any context. However, the type of data being processed influences the choice of algorithms and might also influence the adoption of practices. For example, when processing images or text, it is common to rely on deep neural networks (DNNs), where training is not preceded by a feature extraction step. Conversely, for other types of ML algorithms, a feature extraction step is common. Here, we investigate the influence of the type of data to be processed on the adoption of practices. Moreover, we explore the practices that have distinct adoption rates for specific data types.

The percentage of respondents per data type and the corresponding overall practice adoption rates are presented in Table 5. We employ the same percentages described earlier to assess the practice adoption rates per data type. We observe that, in our data set, tabular data, text, images and videos are predominant (each above 25%) and have very similar adoption rates. Audio and time series have lower representation (under 8%), making their adoption rates less reliable. Still, apart from the "At least high" category, adoption rates remain similar. The "Graphs" data type is used rarely (0.5%), making adoption rates too unreliable to report.

When comparing the adoption of individual practices, grouped by data type, we observed that several practices tend to have higher adoption for particular data types. For all comparisons, we used the "at least high" adoption rate. Firstly, practice 13, on automatic hyper-parameter optimisation, has an adoption rate that is more than 8% higher for tabular data than for text or images. This result could be due to the the algorithms or tools used. The tool support for automatic hyper-parameter optimisation in more traditional ML methods, such as random forests or SVMs – which are popular for tabular data – is more mature than for newer techniques, e.g., DNNs. Secondly, practice 29, on enforcing privacy and fairness, has an adoption rate for tabular data that is more than 10% higher than that for text or images. Lastly, practice 12, on the capacity to run training experiments in parallel, has adoption rates for text and images that are over 10% higher than that for tabular data. Perhaps

Table 6: Linear regression models describing the dependence of effects on the practices that were initially hypothesised to influence them. For each effect, we report the p-value from the F-test for regression and the R^2 coefficient of determination.

Effects	Description	Practices	p-value	R^2
Agility	The team can quickly experiment with new data and algorithms, and quickly assess and deploy new models	12, 18, 22, 24, 28	$7 \cdot 10^{-4}$	0.84
Software Quality	The software produced is of high quality (technical and functional)	9, 10, 11, 17, 18, 19	$5 \cdot 10^{-3}$	0.95
Team Effectiveness	Experts with different skill sets (e.g., data science, software development, operations) collaborate efficiently	6, 26, 27, 28	$1 \cdot 10^{-5}$	0.98
Traceability	Outcomes of production models can easily be traced back to model configuration and input data	3, 5, 16, 25, 27	$4 \cdot 10^{-6}$	0.75

Table 7: Mean squared error (MSE), R^2 and Spearman correlation (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.

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	12, 18, 20, 21, 22, 28	0.69 / 0.44 / 0.68	0.27 / 0.78 / 0.92	0.25 / 0.80 / 0.92	**0.24 / 0.82 / 0.92**
Software Quality	9, 10, 11, 17, 18, 19	0.35 / 0.71 / 0.83	**0.12 / 0.90 / 0.91**	0.17 / 0.87 / 0.91	0.17 / 0.87 / 0.91
Team Effectiveness	6, 26, 27, 28	0.45 / 0.63 / 0.87	0.25 / 0.80 / 0.90	0.19 / 0.84 / 0.92	**0.18 / 0.85 / 0.92**
Traceability	3, 5, 16, 21, 25, 27	0.38 / 0.69 / 0.80	0.22 / 0.82 / 0.90	**0.21 / 0.83 / 0.93**	0.22 / 0.82 / 0.93

the infrastructure needed to run experiments with text or images – where DNNs are used extensively and parallelisation is required to achieve good results – makes it easier to adopt this practice.

6 ANALYSIS OF PRACTICES AND EFFECTS

Following the practice adoption questions, in the questionnaire there were four questions about the perceived effects of adopting these practices. These questions were designed to test the hypothesis that adopting a set of practices will lead to a desired effect. A mapping between practices and effects, as hypothesised during survey design, can be found in Table 6.

Correlations among practices. Firstly, we report results from an analysis of the correlation between practices. We employ the Spearman rank correlation coefficient, ρ, in light of the ordinal nature of the Likert scale used in our questionnaire. In order to determine the statistical significance of the observed correlations, we perform t-tests with a significance level of 0.01.

In total we found 244 statistically significant, moderate to strong correlations ($\rho \geq 0.35$), of which we report on the most informative ones. For example, writing reusable scripts for data management (practice 3) correlates positively with testing for skews between different models (practice 23, $\rho = 0.35$). This suggests that the ability to reuse code for data transformation can facilitate model evaluation. Furthermore, sharing the training objectives within the team (practice 6) correlates positively with using a shared backlog (practice 27, $\rho = 0.38$) and using relevant metrics to measure the training objective (practice 7, $\rho = 0.43$). Testing the feature extraction code (practice 8) correlates positively with practices 9 ($\rho = 0.35$) and 10 ($\rho = 0.56$), on feature documentation and management. This indicates that practitioners tend to use advanced feature management methods concomitantly and that the feature management practices complement each other. As expected, practice 8 correlates positively with practice 17, on running regression tests ($\rho = 0.37$).

Performing peer review on training scripts (practice 11) correlates positively with all team practices – using collaborative development platforms (practice 26, $\rho = 0.40$), working against a backlog (practice 27, $\rho = 0.44$) and good team communication (practice 28, $\rho = 0.44$). This result is in line with our expectations, since collaborative platforms provide features for code review, and this is further enhanced by good communication within the team. Peer review also correlates positively with using static analysis tools for code quality (practice 19, $\rho = 0.48$), which suggests that teams prioritising code quality apply various techniques for this purpose.

The practices for deployment correlate positively between themselves, suggesting that teams with advanced deployment infrastructures tend to adopt all practices. For example, automated model deployment (practice 20) correlates positively with shadow deployment (practice 22, $\rho = 0.48$) and automated roll backs (practice 24, $\rho = 0.51$). Moreover, continuous monitoring of deployed models (practice 21) correlates positively with logging predictions in production (practice 25, $\rho = 0.51$). These results indicate that the deployment practices are complementary and that adopting some enables the adoption of others.

Linear relationship between practices and effects. Secondly, we used the initial mapping from practices to effects (presented in Table 6) to investigate the hypothesis that adopting a set of practices leads to each desired effect. For the analysis, we trained four simple, linear regression models, one for each set of practices and effects in Table 6. For each model, we used the F-test for linear regression to

test the null hypothesis that none of the practices are significant in determining the effect, with a significance level of 0.01. Since some of the data sets were imbalanced, i.e., contained substantially more examples for the positive or negative effect, we applied random under-sampling to balance those sets.

The null hypothesis was rejected for all effects; the respective p-values from the F-test are shown in Table 6. We also performed t-tests to assess whether any of the coefficients in the regression models were statistically significantly different from zero, and found evidence that (at significance level 0.01) this was the case. For example, the t-value of practice 25 for traceability is 6.29. Moreover, the R^2 values, also shown in the table, are high for all effects, which indicates that the observed effects are rather well described by a linear model of the degree of adoption of the associated practices.

Non-linear relationship between practices and effects. Lastly, we report the results from training statistical models to predict each perceived effect from sets of practices. Unlike the linear regression models described earlier, here, we additionally considered ML models that do not assume a linear relationship between the practices and effects. Moreover, in order to strengthen the evaluation, we performed hold-out testing, using a test set of 25% of the data for each effect, which was only used for the final assessment of our models. We also revised the sets of practices associated with two of the effects (agility and traceability), in order to enhance the prediction accuracy of the models as assessed on validation data.

Following practice 13 from Table 2, we considered four types of models with increasing sophistication: (1) simple linear regression models, (2) random forest (RF) regression models resulting from manual hyper-parameter and feature engineering, (3) RF regression models whose hyper-parameters were optimised using grid search, and (4) models obtained from an AutoML system that performed automatic model selection and hyper-parameter optimisation [22].

During training, we used 5-fold cross-validation on the training data (i.e., the 75% of the data retained after setting aside the test sets). For all experiments we used under-sampling on the training data to remove class imbalance. We also experimented with the SMOTE over-sampling algorithm for regression [14, 53], but did not observe significant increases in performance of our models. For the grid search used for hyper-parameter optimisation of our RF models, we used 384 candidate configurations for each of the five folds. For our AutoML approach, we used auto-sklearn [22] with a relatively modest configuration time budget of 1 hour and 5-fold cross-validation for internal evaluation of candidate models.

The performance of our predictive models on test data is shown in Table 7. For all effects, we used three standard evaluation metrics: mean squared error (MSE), the R^2 coefficient of determination, and the Spearman correlation coefficient (ρ) for predicted *vs* true outcomes. We observe that, in all scenarios, the effects can be predicted from the practices with very low error and a high coefficient of determination. Moreover, the RF models always outperform linear regression. This clearly indicates that at least some practices have non-linear impact on the effects we studied.

Models created using AutoML yielded the highest accuracy for two of the effects. For the two other effects, we observed slight overfitting to the training data for the models obtained from AutoML and hyper-parameter-optimised RFs. Nonetheless, the Spearman

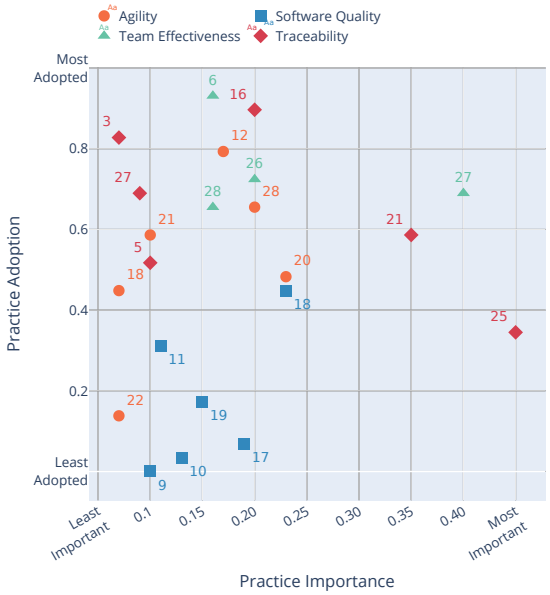


Figure 3: Practice adoption and importance, for each effect and practice. The practice importance is the Shapley value extracted from the grid search RF models in Table 7, using the test data set.

rank correlation between predicted and true outcomes is consistently high ($\rho \geq 0.90$) across all models, except those obtained from linear regression. This indicates that, in all cases, the effects studied can be accurately predicted from the associated sets of practices.

Importance of practices. We also studied the contribution of each practice to the final effect, in order to determine the practices that are the most important for each effect. Towards this end, we used a well-known concept from cooperative game theory called the *Shapley value* to quantify the contributions of individual practices [23, 35]. In our case, the Shapley value intuitively reflects the increase in predictive accuracy caused by a single practice, averaged over all possible subsets of practices already considered in a given model. In order to maintain consistency across all effects, and because the models obtained from AutoML are ensembles that are more difficult to analyse, we performed all Shapley value computations for the hyper-parameter-optimised RF algorithms. We have computed Shapley values on training and test data and obtained consistent results for all effects.

In order to showcase the importance of each practice for an effect, we contrast it with the adoption ranking of the practices from Section 5. We plot the Shapley values and the normalised ranks in Figure 3. The plot indicates, given our data, which practices are most important for achieving a desired effect. We observe that some very important practices have low adoption rates, while some less important practices have high adoption rates. For example, practice 25 is very important for "Traceability", yet relatively weakly adopted. We expect that the results from this type of analysis can, in the future, provide useful guidance for practitioners in terms of aiding them to assess their rate of adoption for each practice

and to create roadmaps for improving their processes. We note that our analysis currently does not take into account functional dependencies between the practices.

7 DISCUSSION

We now comment on the relation between practice adoption and the challenges from Section 2, and discuss threats to the validity of our results.

Engineering challenges vs. practice support. When comparing practice adoption (Table 2) with the engineering challenges referenced in Section 2, we observe that many challenges are supported by well adopted engineering practices.

In particular, versioning the artefacts related to ML projects, considered a challenge by [4] and corresponding to practice 16 in our study, has a high adoption rate (rank 3). The challenges raised by experiment management [4] and prototyping [36], such as clearly specifying desired outcomes or formulating a problem (practices 6, 7), as well as monitoring experiments and sharing their outcomes (practices 14, 15), also have high adoption rates. These results suggest that these challenges have been met by practitioners.

In contrast, the challenge of testing ML artefacts [4, 30, 36], corresponds to practices 8 and 17, which have low adoption in our study. Although we do not detail all testing methods for ML, as done in [59], the adoption rates for the two testing practices in our study suggests that testing remains challenging.

Several practices presented in this study have low adoption and are not mentioned in previous studies as challenging; this is particularly the case for the practices related to feature management (practices 8, 9 and 10) as well as automating hyper-parameter optimisation and model selection (practice 13). Although these practices have been recommended in the literature, we plan to further validate their relevance through future participant validation (member check) interviews and by collecting additional data.

Threats to validity. We identify three potential threats to the validity of our study and its results. Firstly, the data extracted from literature may be subject to bias. To limit this bias, several authors with different backgrounds have been involved in the extraction process. Also, the pilot interviews and survey produced no evidence suggesting that any of the practices we identified are not recognised by practitioners, nor did we find any indications that important practices were missing from our list. Nevertheless, in the future, we intend to further test completeness and soundness of our catalogue of practices through participant validation interviews.

Secondly, the survey answers may be subject to bias. As shown in Section 5, some groups of respondents are over-represented and may introduce selection bias. In particular, although the adoption rates for respondents in Europe do not present striking differences when compared to those in South America or Asia, Europe remains over-represented. Also, some bias may stem from respondents in North America, for which the adoption patterns are different, while they are not equally represented to other groups. This bias can be removed by gathering more data, as we plan to do in the future.

Lastly, the measurements used to investigate the relationship between groups of practices and their intended effects may be subject to bias. Rather than measurements of *actual* effects, we used the *perceived* effects as evaluated by the survey respondents. We have not established that perceived effects indeed reflect actual effects, which is an important and ambitious topic for future research.

8 CONCLUSIONS AND FUTURE RESEARCH

We studied how teams develop, deploy and maintain software solutions that involve ML components. For this, we mined both academic and grey literature and compiled a catalogue of 29 SE best practices for ML, grouped into 6 categories. Through a survey with 313 respondents, we measured the adoption of these practices as well as their perceived effects.

Contributions. We reported on the demographic characteristics of respondents and the degree of adoption of (sets of) practices per characteristic. For example, we found that larger teams tend to adopt more practices, and that traditional SE practices tend to have lower adoption than practices specific to ML. We also found that tech companies have higher adoption of practices than non-tech companies, governmental organisations or research labs.

Further analysis revealed that specific sets of practices correlate positively with effects such as traceability, software quality, agility and team effectiveness. We were able to train predictive models that can predict, with high accuracy, these perceived effects from practice adoption.

We contrasted the importance of practices, i.e., their impact on desirable effects as revealed by these predictive models, with practice adoption, and thus indicating which practices merit more (or less) attention from the ML community. For example, our results suggest that traceability would benefit most from increased adoption of practice 25, the logging of production predictions with model versions and input data. At the level of teams or organisations, these same results can be used to critically assess current use of practices and to prioritise practice adoption based on desired effects. For example, a team with a strong need for *agility* and low adoption of associated practices may plan to increase adoption of those practices.

Future work. We plan to further increase the number of respondents of our survey, so we can perform even more fine-grained analyses. We may also add more questions, for example to better measure the effects of practices related to AutoML, a relatively new direction that is receiving sharply increasing attention in academia and industry. We also plan to better cover the traditional best practices from SE, using a process similar to the other practices. Through validation interviews with respondents, we plan to add depth to the interpretation of our findings, especially regarding the relationships between practices and their effects. We also intend to develop and test a data-driven assessment instrument for ML teams, to assess and plan their adoption of engineering practices. While our study is restricted to ML we may also investigate to which extent our findings are applicable for other domains within the broader field of AI. Overall, our hope is that this line of work can facilitate the effective adoption of solid engineering practices in the development, deployment and use of software with ML components, and thereby more generally contribute to the quality of AI systems. Furthermore, we are convinced that other areas of AI would benefit from increased attention to and adoption of such practices.

REFERENCES

[1] Algorithmia. 2019. Best Practices in Machine Learning Infrastructure. https://algorithmia.com/blog/best-practices-in-machine-learning-infrastructure. [Online; accessed 15-12-2019].

[2] 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 15-12-2019].

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

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

[5] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, 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*. 1387–1395.

[6] Denis Baylor, Kevin Haas, Konstantinos Katsiapis, Sammy Leong, Rose Liu, Clemens Menwald, Hui Miao, Neoklis Polyzotis, Mitchell Trott, and Martin Zinkevich. 2019. Continuous Training for Production ML in the TensorFlow Extended (TFX) Platform. In *2019 USENIX Conference on Operational Machine Learning (OpML 19)*. 51–53.

[7] Google AI Blog. 2019. Fairness Indicators. https://ai.googleblog.com/2019/12/fairness-indicators-scalable.html. [Online; accessed 15-12-2019].

[8] Google AI Blog. 2019. Responsible AI practices. https://ai.google/responsibilities/responsible-ai-practices/. [Online; accessed 15-12-2019].

[9] 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 15-12-2019].

[10] Grady Booch and Alan W Brown. 2003. Collaborative development environments. *Advances in computers* 59, 1 (2003), 1–27.

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

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

[13] Cristiano Breuel. 2019. ML Ops: Machine Learning as an engineered disciplined. https://towardsdatascience.com/ml-ops-machine-learning-as-an-engineering-discipline-b86ca4874a3f. [Online; accessed 15-12-2019].

[14] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[15] Marcus Ciolkowski, Oliver Laitenberger, Sira Vegas, and Stefan Biffl. 2003. Practical experiences in the design and conduct of surveys in empirical software engineering. In *Empirical methods and studies in software engineering*. Springer, 104–128.

[16] Cloudfactory. 2019. The ultimate guide to data labeling for ML. https://www.cloudfactory.com/data-labeling-guide. [Online; accessed 15-12-2019].

[17] Elizamary de Souza Nascimento, Iftekhar Ahmed, Edson Oliveira, Márcio Piedade Palheta, Igor Steinmacher, and Tayana Conte. 2019. Understanding Development Process of Machine Learning Systems: Challenges and Solutions. In *ESEM*. IEEE, 1–6.

[18] Google Devs. 2019. Testing and Debugging in Machine Learning. https://developers.google.com/machine-learning/testing-debugging/pipeline/production. [Online; accessed 15-12-2019].

[19] Microsoft Docs. 2019. Team Data Science Process Documentation. https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/. [Online; accessed 15-12-2019].

[20] Ted Dunning and Ellen Friedman. 2019. Machine Learning Logistics. https://mapr.com/ebook/machine-learning-logistics/. [Online; accessed 15-12-2019].

[21] Samer Faraj and Lee Sproull. 2000. Coordinating expertise in software development teams. *Management science* 46, 12 (2000), 1554–1568.

[22] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *NeurIPS 2015*. 2962–2970.

[23] Alexandre Fréchette, Lars Kotthoff, Tomasz P. Michalak, Talal Rahwan, Holger H. Hoos, and Kevin Leyton-Brown. 2016. Using the Shapley Value to Analyze Algorithm Portfolios. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*. AAAI Press, 3397–3403. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12495

[24] Vahid Garousi, Michael Felderer, and Mika V Mäntylä. 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology* 106 (2019), 101–121.

[25] Vahid Garousi, Michael Felderer, Mika V Mäntylä, and Austen Rainer. 2019. Benefitting from the Grey Literature in Software Engineering Research. *arXiv:1911.12038* (2019).

[26] David Herron. 2019. Principled Machine Learning: Practices and Tools for Efficient Collaboration. https://dev.to/robogeek/principled-machine-learning-4eho. [Online; accessed 15-12-2019].

[27] Joeri Hofmans, Peter Theuns, Sven Baekelandt, Olivier Mairesse, Niels Schillewaert, and Walentina Cools. 2007. Bias and changes in perceived intensity of verbal qualifiers effected by scale orientation. In *Survey Research Methods*, Vol. 1. 97–108.

[28] Waldemar Hummer, Vinod Muthusamy, Thomas Rausch, Parijat Dube, Kaoutar El Maghraoui, Anupama Murthi, and Punleuk Oum. 2019. Modelops: Cloud-based lifecycle management for reliable and trusted ai. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 113–120.

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

[30] Fuyuki Ishikawa and Nobukazu Yoshioka. 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, 2–9.

[31] Per John. 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 15-12-2019].

[32] Foutse Khomh, Bram Adams, Jinghui Cheng, Marios Fokaefs, and Giuliano Antoniol. 2018. Software Engineering for Machine-Learning Applications: The Road Ahead. *IEEE Software* 35, 5 (2018), 81–84.

[33] Barbara A Kitchenham and Shari Lawrence Pfleeger. 2002. Principles of survey research part 2: designing a survey. *ACM SIGSOFT Software Engineering Notes* 27, 1 (2002), 18–20.

[34] James Le. 2019. 10 Best Practices for Deep Learning. https://nanonets.com/blog/10-best-practices-deep-learning/. [Online; accessed 15-12-2019].

[35] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence* 2, 1 (2020), 2522–5839.

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

[37] Matthew Mayo. 2017. The Current State of Automated Machine Learning. https://www.kdnuggets.com/2017/01/current-state-automated-machine-learning.html. [Online; accessed 12-May-2020].

[38] V.M. Megler. 2019. Managing Machine Learning Projects. https://d1.awsstatic.com/whitepapers/aws-managing-ml-projects.pdf. [Online; accessed 15-12-2019].

[39] Jennifer Prendki. 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 15-12-2019].

[40] Yuji Roh, Geon Heo, and Steven Euijong Whang. 2019. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[41] Bernd Rohrmann. 2007. Verbal qualifiers for rating scales: Sociolinguistic considerations and psychometric data. *University of Melbourne, Australia* (2007).

[42] 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 15-12-2019].

[43] Carlton Sapp. 2017. Preparing and Architecting for Machine Learning. https://www.gartner.com. [Online; accessed 15-12-2019].

[44] Danilo Sato, Arif Wider, and Christoph Windheuser. 2019. Continuous Delivery for Machine Learning. https://martinfowler.com/articles/cd4ml.html. [Online; accessed 15-12-2019].

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

[46] Todd Sedano, Paul Ralph, and Cécile Péraire. 2019. The product backlog. In *ICSE*. IEEE, 200–211.

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

[48] Roman Seyffarth. 2019. Machine learning: Moving from experiments to production. https://blog.codecentric.de/en/2019/03/machine-learning-experiments-production/. [Online; accessed 15-12-2019].

[49] Vinay Sridhar, Sriram Subramanian, Dulcardo Arteaga, Swaminathan Sundararaman, Drew Roselli, and Nisha Talagala. 2018. Model governance: Reducing the anarchy of production ML. In *2018 USENIX Annual Technical Conference (USENIX*

ATC 18). 351–358.

[50] Margaret-Anne Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M German. 2016. How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering* 43, 2 (2016), 185–204.

[51] Jeff Sutherland and Ken Schwaber. 2013. The scrum guide. *The definitive guide to scrum: The rules of the game. Scrum. org* 268 (2013).

[52] Nisha Talagala. 2018. Operational Machine Learning. https://www.kdnuggets.com/2018/04/operational-machine-learning-successful-mlops.htmls. [Online; accessed 15-12-2019].

[53] Luís Torgo, Rita P Ribeiro, Bernhard Pfahringer, and Paula Branco. 2013. Smote for regression. In *Portuguese conference on artificial intelligence*. Springer, 378–389.

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

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

[56] Zhiyuan Wan, Xin Xia, David Lo, and Gail C Murphy. 2019. How does Machine Learning Change Software Development Practices? *IEEE Transactions on Software Engineering* (2019).

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

[58] Rüdiger Wirth and Jochen Hipp. 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, 29–39.

[59] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* (2020).

[60] Martin Zinkevich. 2019. Rules of Machine Learning: Best Practices for ML Engineering. https://developers.google.com/machine-learning/guides/rules-of-ml. [Online; accessed 15-12-2019].