

Titre: Quality Issues in Machine Learning Software Systems: Definition and Remedies
Title:

Auteur: Pierre-Olivier Côté
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Côté, P.-O. (2023). Quality Issues in Machine Learning Software Systems: Definition and Remedies [Mémoire de maîtrise, Polytechnique Montréal].
Citation: PolyPublie. <https://publications.polymtl.ca/57037/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/57037/>
PolyPublie URL:

Directeurs de recherche: Foutse Khomh
Advisors:

Programme: Génie informatique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Quality Issues in Machine Learning Software Systems : Definition and Remedies

PIERRE-OLIVIER CÔTÉ

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie informatique

Décembre 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

Quality Issues in Machine Learning Software Systems : Definition and Remedies

présenté par **Pierre-Olivier CÔTÉ**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Michel DESMARAIS, président

Foutse KHOMH, membre et directeur de recherche

Mohammad HAMDAQA, membre

DEDICATION

*“When you’re dead, you’re
dead... but you are not quite so
dead if you contribute
something.”*

John Dunsworth

ACKNOWLEDGEMENTS

As an old African proverb goes: "It takes a village to raise a kid". While the work presented in the following is my own, it would not have been possible without the help of many people.

I would like to start by thanking Amin Nikanjam for co-supervising me during my research. The time you spent mentoring me helped me become a better communicator. I can not stress enough how valuable this skill is; for this thesis as well as any future endeavor. You taught me that with hard work, anything can be achieved. Thank you for being available, even on weekends.

I also want to express my most sincere gratitude to my supervisor, Foutse Khomh, who made all this possible. Your contagious excitement for your work transpired in mine and pushed me to always do my best. You are an example of positive leadership to me. Thank you for welcoming me into your lab and trusting me with your projects.

Finally, this work, and many others, would not have been possible without the love and support of my relatives. First, I would like to thank my mother and my girlfriend for their unconditional trust and support. You helped me traverse the toughest times. I would also like to thank my dad who likewise supported me and nurtured my thirst to understand the world around me since my youth, which, in some ways, led me to where I am now. At last, I wish to thank my stepmother and my larger family, for everything they have done for me.

You all contributed to this work in more ways than you know.

RÉSUMÉ

Pendant les dernières décennies, l'ingénierie informatique a fait de formidables progrès. Une des avancées récentes les plus importantes est l'intégration de l'apprentissage automatique dans les systèmes informatiques. Des applications qui auraient été impossibles à construire, telles que les voitures autonomes, font leur apparition dans notre vie quotidienne. En parallèle à l'arrivée de ces nouveaux produits, un nombre croissant de nouvelles rapportent les comportements indésirables de systèmes basés sur l'apprentissage automatique (SAAs). Seulement aux États-Unis, plus de 400 accidents de véhicule autonome ont eu lieu entre juillet et mai 2022. En plus de causer des problèmes de sécurité, les SAAs de mauvaise qualité peuvent nuire à de nombreux aspects de la vie quotidienne. Par exemple, un outil expérimental de Amazon créé pour faciliter le recrutement de candidats pour un emploi rejetait les candidatures de femmes plus souvent que celles des hommes. Les SAAs modernes ont donc des problèmes de qualité, et, comme pour plusieurs problèmes, il faut les comprendre pour les résoudre.

L'objectif de ce mémoire est de comprendre et définir les problèmes de qualité des SAAs et d'identifier des remèdes potentiels à ces problèmes. Pour ce faire, nous avons réalisé des entrevues (i.e., étude qualitative) auprès de praticiens afin de connaître et comprendre les problèmes de qualité qu'ils rencontrent et les techniques qu'ils ont mises en place afin d'en atténuer les conséquences. Après avoir réalisé 42 entrevues, nous avons établi une liste de 18 problèmes de qualité, pour lesquels nous décrivons leurs causes et conséquences. Nous rapportons 21 techniques utilisées par les praticiens pour atténuer les problèmes de qualité identifiés et en proposons 12 autres basées sur nos connaissances de la littérature. Pour valider les résultats de notre recherche, nous avons partagé un questionnaire avec 21 praticiens dans lequel ils indiquaient la fréquence avec laquelle ils rencontraient les problèmes que nous avons identifiés. Nos résultats montrent que (1) l'utilisation de l'apprentissage automatique dans des systèmes informatiques pose un ensemble unique de défis et que (2) les praticiens peinent à construire des SAAs de qualité. La plupart des problèmes découlent (1) d'un manque d'outils appropriés, (2) de problèmes organisationnels, ou (3) de connaissances limitées parmi les développeurs/praticiens. Notre recherche montre aussi qu'un grand nombre de praticiens ont des problèmes de qualité des données menant à des conséquences négatives sur les SAAs. Ils rapportent souvent devoir travailler avec des ensembles de données comportant des valeurs manquantes ou incorrectes. Ainsi, dans une deuxième étude, une revue systématique de la littérature sur le nettoyage des données et l'apprentissage automatique a été menée dans le but d'identifier des remèdes potentiels aux problèmes de qualité des

données. Cette étude s'intéresse aux travaux qui proposent des techniques de nettoyage des données pour l'apprentissage automatique, ou qui utilisent l'apprentissage automatique pour automatiser le nettoyage des données. Au total, 101 articles publiés entre 2016 et 2022 et issus de diverses bases de données académiques ont été lus et analysés. Cette revue couvre six activités de nettoyage de données: le nettoyage des attributs d'ensembles de données, le nettoyage des étiquettes d'ensembles de données, la détection de doublons, l'imputation et le nettoyage holistique des données. Basées sur la revue de littérature, 24 suggestions de travaux futurs sont faites. Ces travaux futurs orbitent autour des six idées centrales suivantes: (1) d'améliorer les techniques d'augmentation des données pour le nettoyage des données, (2) de créer davantage d'ensembles de données publics pour les problèmes de nettoyage des données, (3) de fournir de meilleurs outils pour le nettoyage des données, (4) d'étudier comment les grands modèles de langage peuvent être utilisés pour le nettoyage des données, (5) d'explorer des approches de nettoyage holistique des données et (6) de développer des approches de nettoyage des données interactives. Nous espérons que notre travail permettra à la communauté scientifique d'avoir une meilleure compréhension des problèmes de qualité des SAAs en plus de servir de savoir fondateur pour les travaux futurs cherchant à résoudre ces problèmes.

ABSTRACT

The last decades have brought significant advancements in Software Engineering (SE). One of the latest and most important milestones in SE is the integration of Machine Learning (ML) into software systems. New applications that would have been almost impossible to build without ML, such as self-driving cars, are now becoming part of our daily lives. In the meanwhile, more and more instances of these undesirable behaviors from these applications are reported. In the United States alone, over 400 car crashes of automated vehicles happened between July 2022 and May 2022. In addition to causing safety issues, ML Software Systems (MLSSs) of poor quality can have pervasive effects on many other important aspects of our daily lives. For example, an experimental recruiting tool developed by Amazon was shown to prefer men over women when filtering candidates for a job role. Hence, MLSSs have quality issues, and, like many issues, they must be understood to be solved.

In this thesis, our goal is to understand and explain the quality issues of MLSSs and identify potential remedies. To achieve this objective, we first conduct a research study based on qualitative interviewing with practitioners to gather insights about their experience and practices when dealing with ML quality issues. Based on the knowledge gathered from 42 interviews, we extracted a list of 18 quality issues of MLSSs, for which we explained the causes and consequences. We report 21 mitigation techniques used by the practitioners to address the identified quality issues and propose 12 additional remedies based on the literature we reviewed. To validate our findings, we shared a questionnaire with 21 practitioners where they indicated how often they encountered the issues we identified. Our findings show that (1) using ML in software systems brings a unique set of challenges, and (2) practitioners struggle to build qualified MLSSs. Most of the issues stem from (1) a lack of appropriate tooling, (2) organizational challenges, or (3) limited knowledge of a topic among developers/practitioners. Our research also shows that many practitioners experience data quality issues leading to negative downstream effects on MLSSs. The most common data quality issues were missing values and incorrect features or labels. Therefore, in a second study, we conducted a Systematic Literature Review (SLR) on data cleaning and ML hoping to identify remedies to data quality issues. More specifically, we examined works that proposed data-cleaning techniques for ML or used ML to automate data cleaning. In total, we reviewed 101 papers published between 2016 and 2022 inclusively, from various academic databases. We covered 6 different types of data cleaning activities: feature cleaning, label cleaning, entity matching, outlier detection, imputation, and holistic data cleaning. Based on our review, we provide 24 future research directions and highlight some promising data-cleaning approaches

that can be further extended. Our future research directions revolve around 6 core ideas: (1) to improve data augmentation techniques for data cleaning, (2) to create more public data cleaning datasets, (3) to provide better tooling for cleaning datasets, (4) to study how LLMs can be used for data cleaning, (5) to explore holistic data cleaning approaches, and (6) to develop more interactive data cleaning approaches. We hope that this work will give the research community a deeper insight into the quality issues of MLSSs and serve as a foundation for future works addressing these issues.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vii
TABLE OF CONTENTS	ix
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
LIST OF SYMBOLS AND ACRONYMS	xv
LIST OF APPENDICES	xvii
CHAPTER 1 INTRODUCTION	1
1.1 Context and Problem statement	1
1.2 Research objectives	2
1.3 Thesis outline	2
CHAPTER 2 BACKGROUND	3
2.1 Machine Learning	3
2.2 Serving ML Models	4
2.3 Software Quality	5
2.4 Data Cleaning	6
CHAPTER 3 LITERATURE REVIEW	8
3.1 Requirement Engineering for MLSSs	8
3.2 Challenges in MLOps	9
3.3 Technical Debt and Anti-Patterns in MLSSs	11
3.4 Patterns and Best Practices with MLSSs	12
3.5 Quality Aspects in MLSSs	13
3.5.1 Fairness	13

3.5.2	Transparency	15
3.5.3	Robustness	17
3.5.4	Privacy	21

CHAPTER 4 ARTICLE 1: QUALITY ISSUES IN MACHINE LEARNING SOFT-

WARE SYSTEMS	24
4.1 Introduction	25
4.2 Related Work	27
4.2.1 Requirement Engineering	27
4.2.2 Technical Debt and Anti-Patterns in MLSSs	27
4.2.3 Challenges in MLOps	28
4.2.4 Tools and Techniques to Address Quality Issues	29
4.3 Study Design	29
4.3.1 Objectives and Research Questions	30
4.3.2 Industrial Partner	31
4.3.3 Participants	32
4.3.4 Interview Structure	34
4.3.5 Interview Process	35
4.3.6 Questionnaire	37
4.3.7 Analysis Plan	37
4.4 Study Results	39
4.4.1 Demographic information	39
4.4.2 RQ1: What are the quality issues encountered by practitioners when building MLSSs?	43
4.4.3 RQ2: Which quality issues are the most prevalent?	50
4.4.4 RQ3: How are the quality issues currently handled by the practitioners?	52
4.4.5 RQ4: Which data quality aspect is the most problematic in practice?	58
4.4.6 RQ5: In the case of data, which data types and collection processes are the most challenging in terms of data quality?	58
4.4.7 RQ6: What are the challenges of data quality assurance during model evolution?	61
4.5 Recommendations	64
4.5.1 Data quality	64
4.5.2 Evaluability	64
4.5.3 Explainability	66
4.5.4 Debuggability	66

4.5.5	Efficiency	67
4.5.6	Reliability	68
4.6	Threats to validity	68
4.7	Conclusion	70
CHAPTER 5 ARTICLE 2: DATA CLEANING AND MACHINE LEARNING: A SYS- THEMATIC LITERATURE REVIEW		71
5.1	Introduction	72
5.2	Methodology	73
5.2.1	Research Questions	73
5.2.2	Scope	74
5.2.3	Search Terms Selection	75
5.2.4	Paper Collection	76
5.2.5	Paper Selection	77
5.2.6	Snowballing	81
5.2.7	Data extraction	81
5.3	Selected Taxonomy	82
5.4	Statistical results	84
5.5	Review	85
5.5.1	Feature Cleaning	85
5.5.2	Label Cleaning	98
5.5.3	Entity Matching	105
5.5.4	Outlier Detection	112
5.5.5	Imputation	115
5.5.6	Holistic Data Cleaning	116
5.6	Future Directions	117
5.6.1	Data Augmentation	119
5.6.2	Data Cleaning Datasets	119
5.6.3	Tooling	120
5.6.4	Large Language Models	121
5.6.5	Holistic Data Cleaning	121
5.6.6	Interactive Data Cleaning	123
5.7	Related Works	123
5.8	Threats to Validity	124
5.9	Conclusion	125
CHAPTER 6 CONCLUSION		127

6.1	Summary of Works	127
6.2	Limitations	127
6.3	Future Research	128
REFERENCES		130
APPENDICES		171

LIST OF TABLES

Table 4.1	List of quality issues in MLSSs	44
Table 4.2	List of techniques to mitigate the quality issues' consequences.	53
Table 5.1	Number of papers retrieved from the academic databases.	78
Table 5.2	Summary of future work recommendations from Section 5.6.	118
Table A.1	Answers to our questionnaire	172

LIST OF FIGURES

Figure 4.1	Overview of the methodology of our study.	30
	Job Role	40
	World region	40
	Years of Experience	40
Figure 4.3	Practitioners' Demographics (Interviews)	40
	Number of Employees	42
	Financial Sectors	42
Figure 4.5	Interviewees' Companies' Demographics	42
	Job Role	42
	Years of Experience	42
Figure 4.7	Practitioners' Demographics (Questionnaire)	42
Figure 4.8	How common the issues are according to the practitioners' experience.	52
Figure 4.9	Percentage of answers rating an issue to be frequent	54
Figure 4.10	Number of practitioners having an issue affecting one of the quality dimensions defined by DDS [1] or DAMA [2]	59
Figure 5.1	Outline of our methodology	74
Figure 5.2	The search query used to fetch papers from all academic databases used in our study except Google Scholar.	76
Figure 5.3	The search query used to fetch papers from Google Scholar.	78
Figure 5.4	Overview of the approaches discussed in this paper	83
Figure 5.5	Distribution of data cleaning topics	85
Figure 5.6	Number of papers published per year	86
Figure 5.7	Authors' Affiliation Distribution	87
Figure 5.8	Number of papers published per venue	88

LIST OF SYMBOLS AND ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interfaces
ART	Adaptive Resonance Theory
ARX	Auto-Regressive with EXogenous inputs model
Auto-ML	Automated Machine Learning
BNN	Bayesian Neural Networks
CADQAS	Computer-Assisted Qualitative Data Analysis Software
CANFIS	Co-active Neuro-Fuzzy Inference System
CNN	Convolutional Neural Network
DAMA	Data Management community
DC4ML	Data Cleaning for Machine Learning
DCAI	Data-Centric AI
DCPG	Data Cleaning Pipeline Generation
DC&ML	Data Cleaning and Machine Learning
DDS	Dataset Definition Standard
DL	Deep Learning
EMI	Expectation-Maximization Imputation
FD	Functional Dependency
GAN	Generative Adversarial Network
GRW	Grow When Required network
GS	Google Scholar
ICA	Independent Component Analysis
KNN	K-Nearest Neighbors
LLM	Large Language Model
LSH	Local Sensitive Hashing
MCD	Monte Carlo Dropout
ML	Machine Learning
ML4DC	Machine Learning for Data Cleaning
MLN	Markov Logic Networks
MLSS	Machine Learning Software System
NN	Neural Network
PCA	Principal Component Analysis
PGM	Probabilistic Graphical Model

PPL	Probabilistic Programming Language
PSF	Point Spread Function
RL	Reinforcement Learning
RQ	Research Question
SE	Software Engineering
SLR	Systematic Literature Review
SMC	Sequential Monte Carlo
TD	Technical Debt
VAE	Variational Autoencoders

LIST OF APPENDICES

Appendix A	Answers to our questionnaire	171
Appendix B	Data quality aspects classification	173

CHAPTER 1 INTRODUCTION

1.1 Context and Problem statement

The last decades have brought significant advancements in software engineering (SE). One of the latest and significant milestones in SE is the integration of Machine Learning (ML) into software systems. New applications that would have been almost impossible to build without ML are now pervasive in our daily lives. For example, more and more people report using ChatGPT [3], OpenAI's chatbot, as a personal assistant to increase their velocity at work (e.g., writing emails, data analysis, etc.) [4]. In addition, autonomous driving companies such as Waymo have already made their product available to citizens of large cities such as San Francisco.

Behind these technologies is ML, which brings a complete paradigm shift to SE. Instead of telling a computer what to do, programmers can now tell them how to learn to do a task. Hence, behaviors that are difficult to describe programmatically but intuitive for a human can be implemented in computers because of ML. This fundamental change in how software systems are built is not without any risk, especially for safety-critical systems. For example, United States regulators have reported over 400 car crashes with automated vehicles between July 2021 and May 2022 [5]. Machine Learning Software Systems (MLSSs) of subpar quality can negatively affect different aspects of our lives. For example, an experimental ML-based recruitment tool designed by Amazon was shown to discriminate female candidates [6]. One of the reasons behind the difficulty of ensuring quality in MLSSs is the fundamental difference between traditional software systems and MLSSs. Computers now learn behavior from data instead of being told what to do by a programmer. Evaluating whether an ML model complies with the requirements of a project is much more difficult than with a traditional program. Additionally, new quality dimensions previously irrelevant to traditional software systems now emerge because of ML. For example, explaining the behavior of an MLSS (explainability) can be difficult because of the complexity of the model used, but is necessary for critical applications, such as disease detection in healthcare.

Problem

As ML is being integrated into a growing number of applications, there is an increasing number of reports of MLSSs with quality issues. The fundamental paradigm shift brought by ML requires us to rethink our quality assurance processes.

1.2 Research objectives

In light of the problem described in Section 1.1, it is obvious that there is a need for new tools and quality assurance processes for MLSSs. These tools and processes will have to build upon a clear understanding of the issues they are addressing. Hence, in this thesis, our first goal is to understand and explain the quality issues experienced by practitioners when developing or working with MLSSs. Based on the results, our second goal is to describe existing approaches that can be used to address the aforementioned quality issues. By achieving these goals, we hope to outline the issues that need to be further researched and highlight some promising research directions for future works aiming at addressing these issues.

Research objectives (ROs)

Explain the quality issues of MLSSs (RO1) and how they can be addressed in the current state of technology (RO2).

1.3 Thesis outline

As explained in Section 1.2, our research objectives are to (1) understand quality issues in MLSSs and (2) describe existing approaches to address them. In order to tackle RO1, we conducted an interview study with 42 practitioners to learn about real quality issues. Our findings notably revealed that a lot of the quality issues they experience stem from poor data quality. Thus, in a second study, we conduct a systematic literature review of data cleaning approaches to tackle RO2. More precisely, we focus on approaches that are designed for ML datasets or that use ML to clean data.

The rest of this thesis is structured as follows. In Chapter 2, we explain fundamental concepts that serve as the building blocks for the rest of the document. For example, we provide a brief overview of ML. Then, in Chapter 3, we survey the literature on quality assurance of MLSSs. Our research works are presented in Chapter 4 and Chapter 5. The interview study to understand the quality issues experienced by practitioners is in Chapter 4. The systematic literature review of data cleaning and ML is in Chapter 5. Finally, in Chapter 6, we conclude the thesis and discuss its limitations, as well as future work directions.

CHAPTER 2 BACKGROUND

In this Chapter, we explain basic concepts that will help understand the rest of the thesis.

2.1 Machine Learning

ML is a subfield of Artificial intelligence where algorithms enable computers to learn to do a task. A more formal definition of ML is provided in [7]: “A computer program is said to learn from experience E with respect to some class of tasks T , and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ”. Problems that were formerly difficult to describe with code, such as language translation, can now be implemented using ML. While there exist complex model architectures with superior learning capabilities, such as transformers [8], simple models, such as linear regression, also fall under the umbrella of ML.

To learn a desired behavior, ML algorithms “train” on datasets. A dataset is composed of individual samples. Samples can also be called “examples” or “records”. The data contained in an example can either be a feature or a label. For a house price prediction problem, the label of a record would be the house’s price. Its features would be any other information that describes the house and can be used to infer the house’s price by a model (e.g., size, location. etc.). Once an ML algorithm has finished training a model, the latter can then be used for inference (i.e., making predictions).

ML can be broadly divided into three large categories: supervised, unsupervised, and reinforcement learning. The difference between the three boils down to the type of feedback that is received by an ML algorithm. In supervised learning, an algorithm is told for each input what is the correct output. The correct output is referred to as a label. In the context of supervised learning, we usually identify two tasks: classification and regression. In classification problems, an ML model must predict the class a record belongs to. An example of a classification task is to predict whether an animal shown in a picture is a cat or a dog. In regression problems, an ML model must instead predict a numerical value for a record, such as predicting a house’s price. Contrary to supervised learning, unsupervised learning problems are not provided with any labels and instead rely on the similarity between examples to generate predictions. Thus, a common unsupervised learning problem is clustering (i.e., identifying groups of records based on their similarities). Finally, reinforcement learning problems rely on a reward to guide them in their learning process. This reward does not

indicate what is the right answer, but only how good or bad the output of a model was. For example, for a self-driving car, the model driving the car would be positively rewarded when the car stays on the road, but negatively when it goes out.

2.2 Serving ML Models

While building ML models has become increasingly easier thanks to the libraries providing good abstractions of ML concepts (e.g., Sklearn [9], TensorFlow [10], PyTorch [11]), reliably serving and maintaining these models is still considered challenging [12]. ML models are tightly coupled to the data they have been trained on, and any change in the data distribution might considerably influence the model’s performance. As most problems modeled with ML algorithms change naturally over time, ML models tend to lose accuracy as they get older [13–16]. Thus, a model must be continuously “re-trained” on the latest data for it to maintain its performance.

Shifts in the data distribution happen as the modeled problem changes but also happen because of unstable data providers (an entity providing data to an MLSS for prediction). For example, a data provider could change the units used to measure a phenomenon, which, in turn, would severely hinder a model’s performance. In addition to causing distribution shifts, data providers may be unavailable at times, which prevents the MLSS that consumes this data from generating ML predictions and makes the prediction service unavailable. Thus, a careful selection and monitoring of these data sources is a critical step toward building reliable MLSSs.

As with any software system, MLSSs sometimes have bugs. A bug is defined as a problem causing a disparity between the requirement specification of a system and its actual behavior [17, 18]. Bugs in MLSSs may be caused by incorrect predictions of their ML models as well as faults in their program or infrastructure. Identifying the root cause of a defect in an ML model can be challenging as there are many components to inspect, such as training data, program code, execution environment, and third-party frameworks [19]. To fix a bug, it is often necessary to reproduce it by having the system in the same state as when the bug happened, in addition to having the input that caused the bug. Thus, versioning all the artifacts that generated a model in production (e.g., training datasets) in addition to the models and the code is necessary to effectively debug MLSSs.

When serving models in real-world applications, aspects other than accuracy become important. For example, for safety-critical applications, it may be important to understand why a decision was taken by a model. Additionally, for applications having a significant impact

on one’s life, it is important to ensure the predictions are not undesirably biased against a group of persons sharing the same attribute (e.g., sex, race, etc.). We elaborate on this topic in the following section (Section 2.3).

2.3 Software Quality

Software Quality refers to the degree to which a program or system possesses a desired combination of attributes [20]. The quality of a software system can be evaluated along different quality dimensions. Examples of notable quality dimensions are maintainability, security, and scalability [21]. With the integration of ML in software systems, other quality aspects grew in importance as they became more challenging to achieve than previously with traditional software systems. Below, we describe such quality dimensions that can be challenging for MLSSs:

- **Explainability:** The ability to understand and explain an ML model’s prediction [22]. Understanding a model’s prediction has many uses. For users of a MLSS, providing explanations for a prediction may increase confidence in the system. For developers, it may help understand why the model made an error. Simple models, such as linear regressions, are interpretable. However, complex models, such as neural networks, are not interpretable. Thus, to understand their prediction, explainability techniques can be leveraged. These techniques generally aim to provide explanations that are comprehensible for humans and that accurately represent the underlying decision process of the model.
- **Fairness:** The capacity of a model or MLSS to not have any prejudice or favoritism toward an individual or group based on their inherent or acquired characteristics [23]. Unfair models can have disastrous consequences in critical applications. For example, a tool used by the court in the United States to help the judge decide whether a convict’s sentence should be shortened or not was shown to be prejudiced against African Americans [24]. The unfair behavior of an ML model generally stems from existing bias in a dataset. Various techniques can be used to mitigate model bias. One can modify the dataset at hand to not display bias (pre-processing algorithms). Other approaches modify the learning algorithm (in-processing algorithms) or the predictions of the model (post-processing algorithms) [23].
- **Efficiency:** In SE, efficiency refers to the aptitude of software to have appropriate performance relative to the number of resources (e.g., memory, CPU time, etc.) used [21]. State-of-the-art models in domains like NLP are notoriously known to consume a lot of resources. For example, the energy needed to train the MegatronLM model [25]

developed by Nvidia was similar to the annual consumption of three households during a full year [26]. Numerous techniques have been proposed to make models more efficient, such as quantization [27] and distillation [28].

2.4 Data Cleaning

Data cleaning is the process of detecting errors in data and repairing them [29]. It is a problem that has been addressed by the database community for a long time [30–32] and has recently gained traction with the ML community because of the importance of data quality for ML performance. A survey published in *Forbes* revealed that cleaning and organizing data takes up to 60% of data scientists’ time and is considered to be one of the least enjoyable activities [33]. Thus, there is an interest in building data cleaning solutions to make the process easier and more effective.

Data cleaning approaches generally aim to address one of three data errors: incorrect values, missing values, and duplicate records. In the following, we describe a few different types of approaches to tackle these issues.

- **Incorrect values:** Dirty data may hinder an ML model’s performance at training time as well as at inference time. Thus, fixing data errors is an important step when building ML models. Techniques to detect or repair incorrect values include outlier detection, rule-based data cleaning, and ML-based approaches. More information regarding outlier detection and ML-based approaches can be found in Chapter 5. As its name suggests, rule-based data cleaning approaches define a set of rules that clean data must abide by. Any data infringing a rule is likely to be dirty. For example, to identify an incorrect country of residence, one could define the following rule: “Two persons living in the same city must live in the same country”. If two persons in a dataset have conflicting information, one can suppose that one of them has an incorrect city or country value.
- **Missing values:** ML models can not process records with missing values. A simple solution to the problem is to “avoid” missing data by ignoring records or attributes with missing values. However, valuable information will be dropped during the process, potentially hindering a model’s performance. Thus, to address this issue, imputation approaches estimate a realistic value for missing data.
- **Duplicate records:** Generally, two records can be considered duplicates if they refer to the same real-world entity. Duplicate records could have different attribute values but refer to the same entity for many reasons. For example, a customer might be recorded multiple times in a database if he used different names when buying a product [34].

Approaches that detect duplicates, also named entity matching approaches, compare records one against another to find the ones that refer to the same real-world entity. Because comparing every record one against another can take a lot of time, entity matching is usually preceded by a blocking step, where pairs of records that are not a match are efficiently identified. As a result, entity matching approaches can avoid processing these pairs.

CHAPTER 3 LITERATURE REVIEW

In this chapter, we cover the literature on quality assurance of MLSSs. The first step toward building qualified MLSSs is to define appropriate requirements. Thus, in Section 3.1, we cover the literature on that topic. Then, we describe various papers discussing the challenges of building, deploying, and maintaining MLSSs in Section 3.2. If poorly handled, these challenges often lead to quality issues. In reaction to the growing concern about ensuring the quality of MLSSs, other works identified recurring patterns leading to quality issues, which we cover in Section 3.3. Conversely, other studies proposed best practices to follow to avoid these issues. We review this topic in Section 3.4. Finally, in Section 3.5, we discuss the challenges and current remedies for various quality aspects in MLSSs.

3.1 Requirement Engineering for MLSSs

One of the first steps when building a software system is to define good requirements. These requirements specify functional and non-functional properties that must be obtained for a system to be considered a success. A system that does not achieve these goals will most likely have quality issues. Because of ML components in MLSSs, defining appropriate requirements is more challenging. Previous requirement specification processes and tools must be adapted to the new paradigm brought by ML. The following details these challenges and describes previous work studying this issue.

The authors in [35] conducted a systematic mapping study on requirement engineering for MLSSs. They report three challenges frequently reported by the research works covered in their study. First, specifying requirements for ML models is challenging, because there is not a reliable way to express quality targets and trade-offs answering a business’s needs. For example, how one should set performance requirements for a recommender system? Second, new quality attributes must be considered because of ML, and measuring them can be difficult. For instance, how to specify an MLSS’s transparency? Third, handling customer’s expectations can be sometimes challenging because they do not understand the potential limitations of ML. Other studies identify similar challenges. For example, Vogelsang and Borg [36] describe some of the additional challenges MLSSs have compared to traditional software systems during requirement engineering. Similar works are done in [37–39]. Researchers in [38] focused on non-functional requirements and explained the difficulty of defining measurable success criteria for quality attributes. The authors in [39] conducted a Systematic Literature Review (SLR) to identify current requirement engineering challenges

and described common notations and modeling languages that are commonly used for ML projects. In reaction to the aforementioned issue, different works propose an approach to objectively specify and assess the quality attributes of MLSSs [40,41]. Similarly, the authors in [42] propose an approach to test requirements for robustness (a quality aspect) for MLSSs based on human perception.

3.2 Challenges in MLOps

In the following, research works discussing the challenges that are encountered when building and maintaining MLSSs are presented. The studies included in this category either consider the problem holistically (i.e., what are recurring problems with developing MLSSs) or focus on one step or aspect of MLSSs development.

Previous studies focused on understanding and describing MLOps holistically and described the general challenges of building MLSSs. The authors in [43] reviewed published reports of deploying MLSSs with the objective of highlighting the challenges practitioners experience during every step of building and maintaining MLSSs. In the following, we detail some of the challenges reported by the researchers for each and every phase. For data management, some of the identified issues are: having access to a sufficient quantity and quality of relevant data and the lack of tools to troubleshoot and understand datasets. The bulk of the issues for the model learning phase stem from the high resource consumption of ML algorithms when training. Similarly, for the model validation phase, most of the issues are caused by the opaque nature of ML models; assessing with high confidence that a model complies with business or legal requirements is difficult. The reported issues for the model serving phase stem from the fact that models are only performant for the data distribution they have been trained on, and any change might severely hinder their performance. The authors list various ways a model may become stale, such as feedback loops. Simply by interacting with its environment, an MLSS may modify it, rendering its model less relevant. For example, models competing against one another for stock trading will hinder another model’s performance every time they are retrained.

Other studies have done similar work. The author in [44] performed an SLR of 141 papers to uncover MLOps challenges at every step of an MLSS lifecycle. Similar to the previous works, [45] listed the challenges experienced by practitioners. To do so, they performed a Multivocal Literature Review and verified their findings through interviews with practitioners. Similarly, in [46], authors interviewed 19 practitioners and extracted a list of recurring challenges along with common anti-patterns. A similar work is done in [47] using the answers of 80 practitioners to a survey to identify challenges and best practices. Based on a

literature review, [48] built a taxonomy of 31 quality assurance challenges experienced during various phases of building an MLSS. Bosch et al. [49] conducted an interview with practitioners from 16 different companies operating in different domains (e.g., real estate, weather forecast, etc.) and of different sizes (ranging from startups to multinational companies). The authors present companies' different maturity stages when adopting ML in their organization and describe key challenges that the enterprises experienced, similar to other works [50]. Authors in [51] presented the current state of MLOps along with its challenges. Additionally, the authors proposed different ways techniques such as Automated ML (Auto-ML) and explainability could be leveraged to address some of the aforementioned challenges. In a similar vein, [52] suggests reusing practices from DevOps to facilitate building MLSSs. The authors in [53] observed that a majority of papers analyzing ML in practice are based on the experience of practitioners from large software companies (e.g., Google), which could lead to a biased representation of the domain. As a result, the authors conducted an interview study with 17 practitioners not working in these companies. Their findings show the tensions caused by the resource constraints of small companies. Crucial aspects, such as fairness and explainability, can not be considered because of a lack of time and money. The study also underlines the unreasonably high expectations of managers and consumers for the quality of the developed ML models, expecting them to compete with ML models of large technological companies (e.g., Google).

Other works focused on the challenges with a specific aspect of an MLSS. Sambasivan et al. [54] describe different challenges with data that cause negative downstream effects on MLSSs. For instance, the authors reported that practitioners sometimes do not have enough domain expertise to understand a dataset hence making bad data modeling decisions instead of consulting a domain expert for help. Researchers in [55] conducted an SLR and reviewed 24 papers to understand state-of-the-art ML deployment practices. The authors also explained current challenges in MLOps and gaps in the tooling available to practitioners. Other studies focused on the maintenance of MLSSs. The authors in [56] discuss challenges in key areas of MLSS maintenance (e.g., data drift detection and model monitoring). Similarly, in [57], authors conducted an SLR and extracted a list of 13 maintainability challenges. The authors reported the challenging aspect of ensuring compliance with minimum quality standards for some applications. Authors in [58, 59] discussed data management challenges and in [60] presented model management challenges. The challenges mentioned include selecting data validation alerts that will have the most impact in production while avoiding alert fatigue [59].

3.3 Technical Debt and Anti-Patterns in MLSSs

The authors in [61] define Technical Debt (TD) as the presence of design or implementation constructs that provide short-term advantages but create a technical context that may render future changes more costly or even impossible. Similarly, the term “anti-patterns” refers to recurring solutions to problems that yield sub-optimal results [62]. While these concepts were initially conceived for systems devoid of ML components, recent research has extended their applicability to MLSSs. One of the first works to do so is the seminal paper of [63] describing the challenges and types of TD faced by Google while building, deploying, and maintaining ML models. Namely, they explained the unnecessary usage of a large number of data sources makes the model more brittle. They also described how easily ML pipelines become overly complex and difficult to maintain. From this work stemmed many others [62, 64–70]. In the following, studies that try to uncover the types of technical debt and anti-patterns commonly found in MLSSs are presented. Studies on code smells are also discussed since they are tightly related to anti-patterns. Code smells are potential indicators of bad practices, while anti-patterns are a bad practice [62].

Previous studies have attempted to discover bad practices when building MLSSs by mining public data sources such as code repositories. Authors in [64] mined open-source GitHub repositories to aggregate a list of the most common maintenance-related modifications in Deep Learning (DL) projects. From this list, they extracted 5 code smells in DL systems. Furthermore, they measured how prevalent and problematic the code smells are from the point of view of practitioners using a survey. Similarly, researchers mined 26 open-source MLSSs and identified 14 refactorings and 7 new TD categories specific to ML [65]. Authors in [71] identified and cataloged 8 design smells that frequently occur in DL systems by reviewing the existing literature on DL systems’ design and manually inspecting 659 DL programs with performance issues and design inefficiencies. Finally, researchers in [66] examined the frequency, distribution, and impacts of code clones (a code smell) in DL systems, as well as developers’ code cloning practices. Instead of mining code repositories, researchers analyzed 66 hours of talks recorded by an MLOps community to extract a list of 17 socio-technical anti-patterns [67]. Namely, the authors have reported that requesting data from other teams is difficult because they are not incentivized to provide endpoints for the requesting team. They also indicated that data producers may cause data leaks, by providing information for historical data that will not be available online.

Other studies described anti-patterns based on their personal experience. [68] shared with the scientific community how TD appeared in the early phases of an industrial DL project. Similarly, [69] provides a catalog of common anti-patterns observed in financial MLSSs.

Facing the increasing number of papers regarding technical debt and code smells in MLSSs, [62] attempted to aggregate the knowledge on these topics by performing a systematic mapping study that presented 4 new types of technical debt, 72 anti-patterns along with 46 solutions. Similar work has been done by [70] but by consulting gray literature as well.

3.4 Patterns and Best Practices with MLSSs

Similar to the research works in Section 3.3, the papers covered in this section try to offer guidance toward building and maintaining MLSSs. However, instead of telling what one should avoid doing, they provide guidelines to follow. The studies generally do so by providing a list of best practices or patterns. A pattern can be defined as “a well-worn and known good solution to a common problem” [72].

The authors in [73] conducted a multivocal literature review to collect design patterns for MLSS. Based on the review of 51 resources, they identified 70 patterns, out of which 34 were unique to MLSSs (but often inspired by patterns in traditional software engineering). For instance, the authors describe the “deploy canary model” pattern, which amounts to deploying a model on a small number of servers in production in order to assess the performance of the model. A similar work was previously done by [74]. In addition to providing 12 architecture and 13 design patterns, the authors provided 8 anti-patterns. The authors extended their work in another contribution [75] where they selected and described three ML patterns in more detail.

Other studies likewise provided guidelines for practitioners, but named these guidelines “best practices” instead of “patterns”. The authors in [76] provide a list of best practices based on their experience at Microsoft. A similar work is done in [77]. Authors in [73] searched the academic and grey literature and identified 29 best practices when building MLSSs. Additionally, the authors conducted a survey with 312 practitioners to measure their degree of adoption and validate their perceived effectiveness. Some of the most adopted practices include versioning data, model, and configuration and monitoring models’ performance. The authors found that mature teams tend to adopt more practices. A similar work was done in [78] based on a literature review of 41 posts of the grey literature and 37 scientific papers. The authors identified 91 best practices grouped into six categories: designing, developing, testing, and deploying MLSSs. Similar to guidelines, the authors in [79] share a checklist to assess the production readiness of MLSSs at Google.

Instead of giving guidelines, other studies provide methodologies that facilitate building qualified MLSS. The authors in [80] proposed CRISP-ML(Q) a quality assurance methodology

for the development of ML applications that is inspired by CRISP-DM [81]. It details, for every step toward building an MLSS, the actions that must be taken in order to ensure the quality and success of the project. For example, as one of the first steps of their methodology, the authors encourage verifying the feasibility of an ML project and defining success criteria on different levels (e.g., business success, ML performance, etc.). A similar work based on CRISP-DM [81] is done in [82]. In addition to providing a methodology, the authors also provide a taxonomy of the types of MLOps papers. Similarly, [83] proposes a methodology tailored for IoT devices.

Finally, other papers focused on a specific aspect of MLOps when providing guidelines. Based on their observation that datasets are often opaque, poorly documented, and maintained, the authors in [84] propose a framework for dataset development that treats datasets as any other software engineering artifact. Notably, it includes rigorous documentation at every phase of a dataset’s lifecycle, from requirements to maintenance. Differently, the authors in [85] build a responsible artificial intelligence (AI) catalog based on a multivocal literature review. The provided catalog aims to address the observed issue: most responsible AI practices operate at the algorithmic level only (e.g., fairness algorithms), ignoring other ways bias can incur. For instance, the authors suggest developing an ethics committee in every organization to address responsible AI issues.

3.5 Quality Aspects in MLSSs

As ML is integrating a growing number of applications, there is growing concern among the population about its potential negative consequences on our lives. In a survey conducted in 2023 by [4], over 60% of the respondents reported being anxious about the impact of AI on society. The main causes of concern were privacy threats, corporation or government misuse of AI, or job losses. In reaction to these concerns, the trend of responsible AI emerged with the objective of ensuring an ethical development of AI systems to benefit humans, society, and the environment [85]. In the following, we cover the quality aspects of MLSSs, focusing on the four core pillars of responsible AI: fairness, transparency, robustness, and privacy. The definition, challenges, and current remedies will be discussed for every quality aspect.

3.5.1 Fairness

As ML integrates a growing number of applications, there is an increasing number of reports of unfair treatment from ML applications. For example, an experimental tool developed by Amazon to help recruiter screen resumes preferred men over women [6]. Another striking

example is an early version of a chatbot developed by Microsoft that showed racist and sexist behaviors [86]. These issues often take root in the dataset used to train the unfair models, which already display bias [87, 88]. Since the model learns from the data it has been trained on, it is bound to replicate the bias in the dataset if no corrective measure is applied. In reaction to these issues, there is a movement to develop techniques to ensure ML applications do not discriminate against groups sharing sensitive attributes (e.g., race). Public libraries provide implementations of fairness techniques [89–91]. Fairness approaches either directly mitigate bias in a training dataset (pre-processing approaches), modify the learning objective of the model to ensure fairness (in-processing approaches), or adjust the predictions of the model to be fair (post-processing approaches). In the following, we describe fairness approaches from all three categories based on existing surveys of the literature [23, 92, 93].

Pre-processing approaches

As previously mentioned, fairness pre-processing approaches mitigate fairness issues by modifying the training dataset of a model in a way that mitigates bias. A straightforward way to remove bias from a dataset is to reduce the impact of examples that are deemed unfair during training, and, inversely, give more importance to the ones that are considered fair [94–96]. Similarly, other approaches rely only on re-sampling training data (i.e., assigning weights of 0 or 1 to training examples) to remove bias from a dataset [96–98]. Instead of modifying training samples’ weights, other approaches modify their features or labels. Based on the intuition that similar samples should have similar treatment by a classifier, the authors in [99] leveraged the k-nearest neighbors algorithm to identify examples that might be discriminated against and assign them a positive outcome by changing their labels. Similar works are proposed in [95, 96]. Others altered features instead of labels [100], where authors proposed an approach that modifies the features of samples so that discriminated samples are more likely to be positively classified. The approach focuses on preserving the ordering of samples in the same group: a sample in a discriminated group that is more positively treated by a model than another one in the same group should still be favored after the transformation. Similar works are proposed in [101–103]. Instead of altering features or labels, the approach proposed in [104] represents samples of a dataset in a fair latent space.

In-Processing Approaches

In-processing approaches modify the ML algorithms so that the trained model is devoid of bias. Various approaches tried to decorrelate a model’s outcome with sensitive attributes.

The authors in [105] added a regularization term to the objective function so that the mutual information between a sensitive attribute and the model’s outcome is minimized. Similarly, the approach described in [106] proposes a split criterion for constructing decision trees such that information gain (the original objective) is maximized, but the resulting groups are homogeneous with respect to the sensitive attribute. Similar approaches were proposed as well [107–109,109]. Other approaches in this area used adversarial learning to prevent models from being unfair [110,111]. Trained models have the objective to be as accurate as possible while preventing an adversary from predicting sensitive attribute values.

Post-Processing Approaches

Instead of mitigating bias (in a dataset or model), post-processing approaches correct the predictions of a classifier for fairness. After observing that examples from a discriminated group tend to have a lower probability of positive outcomes, the authors in [112] propose to use different thresholds based on the value of sensitive attributes. As a result, samples from a discriminated group would be more likely to have a positive outcome even with low model probability scores. Similar approaches are described in [113,114]. Instead of modifying threshold values, the authors in [115] proposed an approach to ensure that the proportion of positive predictions for all groups is the same as the proportion of positive examples from the training dataset. Similar approaches are proposed in [116,117].

3.5.2 Transparency

Transparency in ML refers to the aptitude of an MLSS to explain how a prediction was made in an understandable way [118,119]. For the users of an MLSS, understanding why a decision was made can increase their confidence in its correctness. For developers, it may help find the root cause of a bug or help certify a model’s decision is not influenced by unwanted bias (e.g., sexism, racism, etc.). Having transparent MLSSs tends to be more challenging than with traditional software systems because of the opaque nature of many ML models. Indeed, for most models, understanding the factors that influenced a model’s prediction is difficult, or unintuitive for a human’s mind. For example, how can one explain, in an intelligible way, the decision process of an ensemble of decision trees? A similar challenge lies with other types of models, such as neural networks. On the contrary, explaining the decision of a single decision tree is feasible since it amounts to interpreting a series of “if” and “else” statements. This property of being naturally understandable is called interpretability, and the models bearing that quality are referred to as interpretable models. Interpretable models include linear regression, k-nearest neighbors, and rule-based learners, amongst others. Non-

interpretable models, on the other side, require special techniques for their predictions to be understood. The field devoted to crafting such techniques is called *explainability*. Various libraries have provided ready-to-use implementations of explainability techniques [120–123]. In the following, we review the literature on this topic based on existing surveys of the literature [22, 124–126]. Explainability techniques can be designed for a particular type of model or be applicable to any (model-agnostic), which we cover in this review. Model-agnostic approaches roughly fall into one of three categories: explanations by simplification, feature relevance explanations, and visual explanations [124]. In the following, we explain the main ideas behind each category and different approaches.

Explanations by Simplification

The strategy followed by these approaches to explain a model’s prediction is to approximate a black-box model with a simpler, more interpretable model and use the innate interpretability mechanisms of the second model for explainability. Generally, the simplified model can not effectively approximate all the predictions of a complex model but is relatively accurate for a region of space. For example, a polynomial of first degree can be used to approximate another polynomial of high degree in a limited region of space. Hence, some explanations by simplification describe individual predictions of a model by approximating a complex black-box model with a simple interpretable one. A well-known technique following that approach is Local Interpretable Model-Agnostic Explanations (LIME) [127]. To explain a complex model, the authors train an interpretable model (e.g., linear regression or decision tree) on examples distributed in the vicinity of a prediction that must be explained, using the predictions of the complex model as training labels. A similar approach is described in [128, 129]. Other studies approximate complex models using a set of rules. The authors in [130] find Boolean rules in Conjunctive Normal Form (i.e., AND-of-ORs) or Disjunctive Normal Form (i.e., OR-of-ANDs) that accurately describe the predictions of a model using integer programming. The generated model can be used to explain all predictions of the black-box model. In a similar vein, the authors in [131] use Generic Programming to find a set of rules and bandits algorithms are used in [132] instead.

Feature Relevance Explanations

A second way a model’s behavior can be explained is by measuring the importance of samples’ features in its decision process. A popular feature relevance explanation approach inspired by game theory is called SHAP (SHapley Additive exPlanations) [133]. As implied by its name, SHAP is based on Shapley Values, a technique to measure the influence of a feature

on the output of a model by measuring its average marginal contribution to a model’s decision. Similar approaches based on Shapley Values are described in [134, 135]. The approach described in [135] accounts for correlated input, contrary to SHAP. Other studies follow a different approach to measuring feature importance. The authors in [136] compare the behavior of a model before and after randomizing the values in a feature of a dataset. If its behavior changes a lot after randomization, then the feature most likely has high importance. In a similar approach to explain a model’s prediction [132], beam search is used to find the minimal set of feature values that can not be randomized for the prediction to stay the same.

Visual Explanations

The third way to explain a model’s predictions is through visual explanations. Approaches based on sensitivity analysis are presented in [137, 138] to visualize the decision boundary of a model. Similarly, other approaches display the decision boundary as a function of a unique feature (by fixing the value of the other attributes) [139, 140].

Other Explainability Approaches

In this section, we describe two other approaches that are not included in the previous categories. In [141], influence functions, a technique from robust statistics, are used to find the training examples that are the most responsible for a particular prediction. Differently, the authors in [142] propose to explain a particular prediction by showing similar examples that resulted in a different prediction, i.e., counterfactuals. Counterfactuals are useful to explain how an example should change for the prediction to change as well. For example, for a loan approval system, this could explain to a user why a loan was refused and how he/she can get it approved.

3.5.3 Robustness

Robustness in ML refers to the resilience of an MLSS in the presence of perturbations [143]. More formally, the robustness of an MLSS is the difference between the expected output of an MLSS and its expected output after a perturbation to any ML component: data, model, or the software system in which the model is embedded. A sub-category that has gained much traction with the ML community is adversarial robustness, i.e., being robust to adversarial attacks to an ML model. In the following, we provide an overview of adversarial robustness. Then, we describe other studies on global robustness, i.e., not limited to adversarial robustness.

Adversarial Robustness

The adversarial robustness of a MLSS refers to its capability of being robust against adversarial attacks. Public libraries provide ready-to-use implementations of adversarial attacks and defenses [144, 145]. In the following, adversarial attacks and defenses are described and explained. This review of adversarial robustness is based on previous surveys [146–150].

Adversarial Attacks Adversarial attacks are a type of attack where test time data is altered imperceptibly (to a human’s eye) leading a model to make incorrect predictions. Previous studies have fooled models embedded in real-life applications, from road sign recognition [151] to person detection [152]. Adversarial attacks are roughly performed in one of two settings. In the white box setting, the attacker has complete access to the model. Inversely, in the black box setting, the attacker can only query the model, which is often closer to real-life scenarios. In the following, we describe white-box and black-box attacks.

A seminal work in the field of adversarial attacks is the Fast Gradient Sign Method [153]. The authors propose a white box attack that uses the gradients of a model to craft adversarial examples. Similar works were later proposed in [154–156]. Contrary to the previous approaches whose goal is to make the perturbations imperceptible, the authors in [157] visibly alter pixels in an image, hoping the perturbation will pass as a defect from the sensors capturing the image. Hence, the approach’s goal is to find the minimal set of pixels to modify an image such as the model’s prediction is changed. Instead of crafting perturbations tailored for one sample only, the white box approach proposed in [158] finds a unique perturbation that can be applied to all samples in a dataset so as to maximize the number of prediction errors by a model. Other approaches proposed attacks robust to a selected defense [159–161].

Contrary to white box attacks, black box attacks do not have access to a model’s parameters and gradients. Thus, the authors in [162] estimate models’ gradients by sending a lot of queries and observing the model’s confidence values. The estimated gradients are then used to craft adversarial samples. To reduce the number of queries necessary to estimate gradients, the authors in [163] perform gradient estimation in the latent space of an autoencoder. Since the latent space has a reduced number of dimensions, estimating gradients is more efficient. While the previous approaches leveraged the confidence scores of attacked models to craft adversarial samples, real-life applications often only provide the final prediction. Thus, the authors in [164] propose an approach able to fool a model using only its final predictions. To do so, the approach first adds sufficient noise to an image so it is misclassified, then iteratively reduces it until the sample is not misclassified anymore. The misclassified samples closer to the decision boundary are the generated adversarial samples. This approach was

later improved in [165, 166].

Adversarial Defenses In reaction to the growing number of adversarial attacks showing the weaknesses of ML models, other studies developed defense mechanisms to protect models against these attacks. Generally, the defense techniques either: (1) pre-process the data fed to the model in such a way that adversarial attacks are less effective, (2) improve the robustness of the model to adversarial attacks or (3) detect adversarial attacks to prevent their consequences. In the following, we describe defense mechanisms against adversarial attacks.

The pre-processing approaches generally adopt one of two strategies to tackle adversarial attacks. The first type of pre-processing approach adds random noise to examples such that a model’s performance is not hindered, but crafting adversarial is difficult (since the image ingested by the model differs from the one provided by the adversary). The authors in [167] randomly nullify features of records. A similar work is described in [168]. The authors in [169] propose an approach that randomly selects several transformations in a set. The second type of pre-processing approach aims to remove adversarial noise from images. The approach described in [170] removes noises with a model based on the U-net architecture [171]. Instead of directly removing noise, a generative adversarial neural network is used in [172] to generate clean examples identical to an original image without adversarial noise. A similar work is described in [173]. For images, the authors in [174] observed that JPEG compression attenuates the effect of adversarial attacks. A defense mechanism based on that approach is described in [175].

A second way MLSSs can be more robust to adversarial inputs is by making the model more robust. Previous approaches leveraged adversarial training to create robust models [153, 176, 177], which amounts to training a model on adversarial examples. While considered to be one of the most effective defenses, it is computationally expensive and often limited to a limited set of attacks [146]. Other studies added components to a model’s architecture to enhance its robustness. The approach described in [178] adds a denoising component to the feature map of a convolutional neural network in order to enhance its robustness. The authors in [179] observed that features providing a minor performance gain are often the more susceptible to adversarial attacks. Thus, in their approach, non-linear attention and L2 feature regularization are used to select only the more important features. Finally, the authors in [180] added an extra class to the training of ML models such that adversarial examples are assigned that class by the model.

The final way to protect MLSSs against adversarial attacks is through malware detection.

Two types of malware detection techniques have been proposed. The first one detects attacks by comparing the prediction of a model before and after applying transformations to the input data that should not change the model’s prediction [181, 182]. If the prediction changed after applying pre-processing techniques, then the image is likely to have been adversarially tempered. For example, the authors in [181] use rotation and translation on images to detect adversarial examples. A second way to detect adversarial examples is to observe the distribution of queries of an attacker. This defense mechanism supposes that the attacker has only black-box access to the model (i.e., can only send queries to the model, without having access to its parameters). If a user sends too many similar requests, then the person might be trying to craft an adversarial example [183].

Global Robustness

In this part, we describe studies whose goal is to measure, test, or improve the robustness of an MLSS. Studies limited to adversarial robustness are not discussed in this section and are reviewed in Section 3.5.3.

The approach described in [184] assessed the robustness of a model by clustering samples and measuring the homogeneity of predicted classes amongst the samples in the same cluster. In a similar vein, the authors in [185] described an approach providing upper and lower bounds of a model’s robustness. Similar approaches are reported in [186, 187]. Inspired by software testing, the approach described in [188] uses symbolic execution to generate samples to test a model’s robustness. Similar approaches use fuzz testing [189], combinatorial testing [187], and abstract interpretation [190] to test a model’s robustness. Other studies assessed the robustness of an MLSS to specific types of faults. For example, the approach described in [191] injects faults in the software of an MLSS to approximate hardware errors in various components of an automated vehicle. Similar approaches are described in [192–194].

The following studies proposed techniques to ensure the robustness of a model to various types of perturbations. The authors in [195] propose an approach based on Wasserstein Distributionally Robust Learning. In short, the model is optimized in a way that minimizes its worst performance in case of a shift in data distribution. To do so, the performance on all neighbor data distribution (according to Wasserstein distance) is considered when training the model. Because its computation is practically intractable for most applications, the authors relax the problem into a Lagrangian form optimizable with stochastic gradient methods, and similar works are reported in [196, 197]. For reinforcement learning agents, the authors in [198] describe an approach that considers worst-case scenarios before taking any action.

3.5.4 Privacy

In recent years, there has been an increased concern for data privacy in software applications. Policies, such as GDPR [199], are created to enforce the security of private information. In the meanwhile, studies have shown how sensitive information about a model’s training dataset can be inferred using only the models’ predictions. For example, [200] developed an approach to predict whether a record was part of the dataset used to train a model. Similar works were done by [201–203]. Knowing this information may be a severe breach of information for sensitive applications. For example, a model trained to detect a disease could leak who is part of the training dataset and has been diagnosed with that disease. This type of attack is known as model inversion attacks. Thus, protection mechanisms must be configured in MLSSs in order to prevent the model from leaking private information.

Another type of data that can be extracted from an MLSS using only the predictions of a model is the parameters of the model itself. This type of attack is known as model extraction attacks. Previous studies have shown how a model’s parameters [204] or hyperparameters [205] can be extracted under certain conditions (e.g., access to prediction probabilities). Because training ML models is an important investment for companies (e.g., the GPT-4 model [3] cost over 100 million dollars [206]), there is an incentive to build defenses against these types of attacks.

In the following, we present various attack and defense mechanisms that have been proposed in the literature for model inversion and extraction. Privacy attacks and defenses have been implemented in public libraries such as [145]. We briefly review ML privacy based on previous surveys [146, 207–209].

Model Extraction Attacks

To copy a model, the authors in [204] fetch its prediction over a group of samples and build model equations from the predictions. They show how a binary logistic regression can be recreated with only $d + 1$ queries, where d is the number of features of a sample. While described technique can be extended to neural networks, they also introduce a path-finding algorithm to extract the parameters of decision trees. Similar works are described in [210–212]. Instead of inferring parameters, the approach described in [205] steals hyperparameters. The authors highlighted that the goal of an ML algorithm is to minimize the error on a training dataset. Based on that observation, they designed a system of linear equations to find hyperparameters. Other studies removed watermarks (a model extraction defense described in Section 3.5.4) from the weights of stolen models. Based on the observation that

the watermarks proposed by [213] increase the standard deviation of the distribution of a model’s weight, the authors in [214] propose an approach capable of detecting and removing these watermarks. Differently, the authors in [215] describe how stolen watermarked models can evade watermark detection techniques, by training a second model to detect the queries of watermark detection techniques.

Model Extraction Defenses

Model extraction attacks may rely on the attacked model’s confidence scores [200, 216, 217]. Hence, defense mechanisms addressed that issue by avoiding sending confidence scores to users or by adding smart noise to the scores [204, 218]. Other approaches are proposed to detect model extraction attacks. The authors in [219] propose PRADA, a mechanism that is able to detect such types of attacks by analyzing the distribution of queries of a model’s user. Instead of preventing model extraction attacks, other researchers proposed approaches to identify stolen models. To do so, a watermark is embedded in a model’s weight, making it possible to identify the ownership of stolen models. Watermarked models can be detected by analyzing their weights [213] or their predictions for a selected set of inputs [220].

Model Inversion Attacks

Model inversion attacks’ goal is to extract information about the training dataset of a model. One piece of information that can be extracted is whether a sample was part of a model’s training dataset. They generally leverage a model’s confidence to detect samples from the training dataset and try to detect patterns in the model’s confidence. A model will generally behave differently with samples present in its training dataset. This attack was introduced by [221], where a binary classifier was trained to detect training examples. To train the classifier, the authors first generate training data that is representative of the ML problem solved by the attacked model and train multiple models on subsets of that dataset. The authors refer to these models as *shadow models*, and the generated data, *shadow dataset*. Training data for the classifier is then generated using the aforementioned shadow model’s predictions on shadow training and test data. This work was later improved in [201], to perform better on models that are not overfitted. Similar works are described in [202, 222–224]. Other works proposed approaches to infer sensitive attributes from records. Using the predictions of a model and a subset of a sample’s feature, the authors in [225] demonstrated how to infer the unknown features using a maximum a posteriori algorithm. This work was later improved in [226], where having partial access to a record’s feature is not necessary. To extract sensitive information, the approach analyzes the predictions of the model whose training data has been

poisoned (i.e., malicious samples have been added to the model’s training dataset). Similar approaches are described in [227–229].

Model Inversion Defenses

A common defense mechanism to preserve the privacy of a training dataset is Differential Privacy (DP) technique. Initially proposed by [230], DP is a formal definition of data privacy guarantees. DP ensures that the presence or absence of any single data point in a dataset has a limited impact on the output of a model, making it difficult for an adversary to infer specific information about someone’s personal information based on the algorithm’s behavior. DP techniques generally strategically add noise during the learning process of an ML algorithm, preventing the model from memorizing individual records of a dataset. Depending on where the noise is added, DP techniques can be roughly classified into one of three categories: gradient perturbation, objective perturbation, and label perturbation techniques [146].

As implied by the name, gradient perturbation techniques add noise to the gradients of a model’s parameters during the training phase. Different approaches have been proposed for various types of models: deep neural networks [231], generative adversarial neural networks [232], and autoencoders [233]. Objective perturbation techniques add noise to the loss function of various types of models: deep neural networks [234], autoencoders [235], and Convolutional Deep Belief Networks [236]. Finally, label perturbation techniques add noise to the labels generated by the teacher model during knowledge distillation [237, 238]. For example, the approach described in [237] trains an ensemble of teacher networks on different subsets of a dataset. The labels used for training the student network are aggregated from the prediction of all teacher models, and differential noise is added to the final prediction.

CHAPTER 4 ARTICLE 1: QUALITY ISSUES IN MACHINE LEARNING SOFTWARE SYSTEMS

Preface

Full Citation: P.-O. Côté¹, A. Nikanjam¹, R. Bouchoucha¹, I. Basta¹, M. Abidi¹, and F. Khomh¹, "Quality Issues in Machine Learning Software Systems", *Empirical Software Engineering Journal*, [in review] June 21 2023.

Permission: Reproduced with permission from Springer Nature.

Abstract Context: An increasing demand is observed in various domains to employ Machine Learning (ML) for solving complex problems. ML models are implemented as software components and deployed in Machine Learning Software Systems (MLSSs). **Problem:** There is a strong need for ensuring the serving quality of MLSSs. False or poor decisions of such systems can lead to malfunction of other systems, significant financial losses, or even threats to human life. The quality assurance of MLSSs is considered a challenging task and currently is a hot research topic. **Objective:** This paper aims to investigate the characteristics of real quality issues in MLSSs from the viewpoint of practitioners. This empirical study aims to identify a catalog of quality issues in MLSSs. **Method:** We conduct a set of interviews with practitioners/experts, to gather insights about their experience and practices when dealing with quality issues. We validate the identified quality issues via a survey with ML practitioners. **Results:** Based on the content of 37 interviews, we identified 18 recurring quality issues and 24 strategies to mitigate them. For each identified issue, we describe the causes and consequences according to the practitioners' experience. **Conclusion:** We believe the catalog of issues developed in this study will allow the community to develop efficient quality assurance tools for ML models and MLSSs. A replication package of our study is available on our public GitHub repository¹.

Keywords Machine Learning based Software Systems, Quality Assurance, Quality Issues, Interview, MLOps, Machine Learning

¹https://github.com/poclecoqq/quality_issues_in_MLSSs

¹Polytechnique Montréal, Québec, Canada

Email: {pierre-olivier.cote, amin.nikanjam, rached.bouchoucha, ilan.basta, mouna.abidi, foutse.khomh}@polymtl.ca

Corresponding author's e-mail address: pierre-olivier.cote@polymtl.ca

4.1 Introduction

Nowadays, Machine Learning Software Systems (MLSSs) have become a part of our daily life (e.g. recommendation systems, speech recognition, face detection). Increasing demand is observed in various companies to employ Machine Learning (ML) for solving problems in their business. Typically, an MLSS receives data as input and employs ML models to make intelligent decisions automatically based on learned patterns, associations, and knowledge from data [239]. Therefore, ML models are implemented as software components integrated into other subsystems in MLSSs, and like other software systems, quality assurance is necessary. According to the growing importance of MLSSs in today’s world, there is a strong need for ensuring their quality. False or poor decisions of such systems can lead to the malfunction of other systems, significant financial losses, or even threats to human life [240]. Recently, an automated vehicle caused a crash of 8 vehicles, leading to 2 juveniles being transported to a hospital [241]. As reported by United States safety regulators, this is not a punctual event, since more than 400 car-crash with automated vehicles have been reported between July 2021 and May 2022 [5]. Unqualified MLSSs can be harmful to society in less obvious ways. For example, an experimental job recruitment tool developed by Amazon was shown to reject women’s job applications more often than men’s [6]. ML models acted unfairly in many other critical situations [242, 243]. In addition to the dire consequences caused by unreliable or unfair models, large ML models often have a significant ecological footprint. For example, the energy needed to train the MegatronLM model [25] developed by Nvidia was almost equal to the annual consumption of three households for a year [26]. Considering the growing popularity of ML, addressing the quality issues in MLSSs becomes of increasing importance. However, to solve an issue, one must first understand it. While there are research works describing the challenges in MLOps [43, 45, 46, 58–60] and (anti-) patterns in MLSSs [54, 62, 64, 65, 67, 68, 70], we have not observed a large literature focused on discovering and understanding the quality issues themselves.

In this paper, we investigate the characteristics of real-world quality issues in MLSSs from the viewpoint of practitioners. This is a requirement of comprehensive quality assessment of MLSSs, as Zhang *et al.* already acknowledged the lack of such empirical study and asserted that conducting empirical studies on the prevalence of poor models among deployed ML models should be interesting [143]. This study attempts to cover all relevant quality factors like performance (accuracy), robustness, explainability, efficiency, maintainability, and reliability. We conduct a set of interviews with practitioners/experts, to gather insights about their experience and practices when dealing with ML quality issues.

In total, we interviewed 42 practitioners; 14 of them are employees of MoovAI. The others

were either found over GitHub using its API ² (27 practitioners) or were personal contacts (1 practitioner). To analyze the results, we performed a round of open coding, followed by a round of axial coding [244]. As a result, we extracted a list of 18 quality issues with MLSSs, for which we explain the causes and consequences. We report the current mitigation techniques used by the practitioners to address the identified quality issues. In addition to the quality issues with MLSSs, we share the data quality issues experienced by practitioners. To better understand them, we describe the data quality aspects that are the most problematic in practice according to the practitioners' experience. We also explain the challenges that occur with different data collection processes that lead to data quality issues. To validate the identified quality issues with MLSSs, we conducted a survey with practitioners found on social networks or from our personal contacts. In the survey questionnaire, the practitioners were asked to rate on a five-level Likert scale how often they are faced with the quality issues we found. In total, we obtained 21 answers. Our findings show that (1) using ML in software systems brings a unique set of challenges, and (2) practitioners struggle to build qualified MLSSs. Most of the issues stem from (1) a lack of appropriate tooling, (2) organizational challenges, or (3) limited knowledge of a topic among developers/practitioners. Through this study, we aim to contribute knowledge that will help develop efficient quality assurance tools for ML models and MLSSs. A replication package of our study is available on our public GitHub repository³.

Contributions. In summary, we make the following contributions.

- We describe 18 quality issues with MLSSs identified during our interviews. For each quality issue, we explain its causes and its consequences from the point of view of the practitioners. We also inform which ones are the most prevalent in practice.
- We share 21 strategies used by the practitioners we interviewed to mitigate the aforementioned quality issues. We also propose 12 other strategies to address the issues based on the literature and the root causes of the issues.
- In addition to the quality issues in MLSSs, we describe the data quality issues encountered by our practitioners. We show how frequent they are according to the practitioners' experience.
- We describe the challenges encountered with different data collection processes that lead to data quality issues.
- We characterize the challenges of data quality assurance during model evolution.
- Based on the quality issues (with MLSSs and with data) we found, we provide 9 future

²<https://docs.github.com/en/rest?apiVersion=2022-11-28>

³https://github.com/poclecoqq/quality_issues_in_MLSSs

work recommendations.

The rest of this paper is organized as follows. Section 4.2 reviews relevant related work to our study. In Section 4.3, we define our research questions and our methodology. Section 4.4 presents the results of our survey. Finally, in Section 4.5, we provide recommendations to practitioners (regarding handling quality issues with data and MLSSs) and to researchers (for future works).

4.2 Related Work

We organize the related works of our study into four groups. The first step toward building a qualified MLSS is defining necessary requirements. Hence, we start by presenting related works in that area. The following two parts are dedicated to topics tightly related to quality issues: technical debt and anti-patterns, and general challenges in MLOps. The final group discusses studies that propose potential solutions to quality issues.

4.2.1 Requirement Engineering

One of the first steps when building a software system is to define good requirements. These requirements specify functional and non-functional objectives that must be obtained for the MLSS to be considered a success. A system that does not achieve these goals will most likely have quality issues. [36] describes some of the additional challenges MLSSs have compared to traditional software systems during requirement engineering. Notably, new quality aspects must be considered when defining requirements for MLSSs, such as explainability, fairness, and legal aspects of data. Similar work is done by [37, 38]. [38] focused on non-functional requirements and explained the difficulty of defining measurable success criteria for quality attributes. For example, how should maintainability be measured for an ML system? In reaction to the aforementioned issue, [40] proposes an approach to objectively specify and assess the quality attributes of MLSSs.

4.2.2 Technical Debt and Anti-Patterns in MLSSs

[61] defines Technical Debt (TD) as “ [any] design or implementation constructs that are expedient in the short term, but set up a technical context that can make a future change more costly or impossible”. Similarly, anti-patterns refer to recurring solutions to a problem that leads to sub-optimal results [62]. While both were initially conceptualized for systems without ML components, recent studies transpose these concepts to MLSSs. For example, the

seminal work of [63] shared the challenges and types of TD faced by Google while building, deploying, and maintaining ML models. If left unattended, TD and anti-patterns in an MLSS may result in quality issues. For example, a model with undeclared consumers (the anti-pattern) may hinder the reliability of an MLSS (the quality issue) [62]. In the following, we present other studies that try to uncover the types of technical debt and anti-patterns commonly found in MLSSs. We also present studies on code smells, since code smells are tightly connected to anti-patterns. Code smells are potential indicators of bad practices, while anti-patterns are a bad practice [62].

[64] mined open-source GitHub repositories to aggregate a list of the most common maintenance-related modifications in Deep Learning (DL) projects. From this list, they extracted 5 code smells in DL systems. Furthermore, they measured how prevalent and problematic the code smells are from the point of view of practitioners using a survey. Similarly, [65] mined 26 open-source MLSSs and identified 14 refactorings and 7 new TD categories specific to ML. Instead of finding new types of TD, [68] shared with the scientific community how it appeared in the early phases of an industrial DL project. [71] identified and cataloged 8 design smells that frequently occur in DL systems by reviewing the existing literature on DL systems' design and manually inspecting 659 DL programs with performance issues and design inefficiencies. On the same line of work, [67] analyzed 66 hours of talks recorded by an MLOps community to extract a list of 17 socio-technical anti-patterns. Finally, [66] examined the frequency, distribution, and impacts of code clones in DL systems, as well as developers' code cloning practices. Facing the increasing number of papers regarding technical debt and code smells in MLSSs, [62] attempted to aggregate the knowledge on these topics by performing a systematic mapping study that presented 4 new types of technical debt, 72 anti-patterns along with 46 solutions. Similar work has been done by [70] but by consulting gray literature as well. [54] describes different challenges with data causing negative downstream effects on MLSSs such as technical debt. The main difference between these works and ours is that they are focused on the causes of issues (anti-patterns, technical debts, code smells), while we are focusing on the issues themselves. Furthermore, we are only interested in identifying ML quality issues and potential strategies to address them.

4.2.3 Challenges in MLOps

In the following, we present papers discussing the general topics of challenges in developing and deploying MLSSs. These challenges are often tightly coupled with an underlying quality issue. For example, [43] mentions that the hyper-parameter selection is resource-intensive (challenge), which implies an issue with efficiency. Similar to our project, [46] conducted

an interview study with practitioners to discover their pain points along with common anti-patterns. They also provide a set of best practices for successful MLSS development and maintenance. [43] performed a survey on the challenges when building and deploying MLSSs. These challenges sometimes affect quality aspects, such as maintainability, explainability, and fairness. Similar to the works above, [45] listed the challenges experienced by practitioners at different stages of the development of an MLSS. To do so, they performed a Multivocal Literature Review and verified their findings through interviews with practitioners. Other works focused on the challenges of specific aspects of an MLSS. [58, 59] discuss data management challenges and [60] presents model management challenges. The main difference between these works and ours is the scope. We are only interested in quality issues, while the aforementioned studies are interested in any recurring challenge in MLOps.

4.2.4 Tools and Techniques to Address Quality Issues

In reaction to the growing concern about quality issues in MLSSs, researchers have proposed techniques and tools to help practitioners develop qualified systems. One must define quality beforehand to detect quality issues. Therefore, for data, [1] describes different aspects that must be considered when evaluating the quality of a dataset. Some examples of quality aspects are data representativeness or data accuracy and precision. Other tools such as the DataLinter [245] or the data validation component of TFX [246] have been created to automate data quality assurance. Another group of researchers adopted a different approach and instead shared a checklist of tests to assess the production readiness of MLSSs [79]. A similar work is done by [77]. [76] presents a list of best practices when developing MLSSs. In a similar fashion, [80, 247] and TDSP⁴ propose a process for the development of ML applications where quality assurance plays a central role. Finally, [248] identifies factors that make test maintenance difficult in MLSSs and proposes ten recommendations to make it easier.

4.3 Study Design

In this section, we describe how we conducted this study. An overview of our methodology is given in Figure 4.1. We indicate the section in the paper (e.g. 3.3) where more information can be obtained for a particular activity in the figure. In the following, we start by presenting the research questions that guided our study. Then, we introduce our industrial partners who helped us conduct this study. Finally, we describe every step we followed to complete this study.

⁴<https://learn.microsoft.com/en-us/azure/architecture/data-science-process/overview>

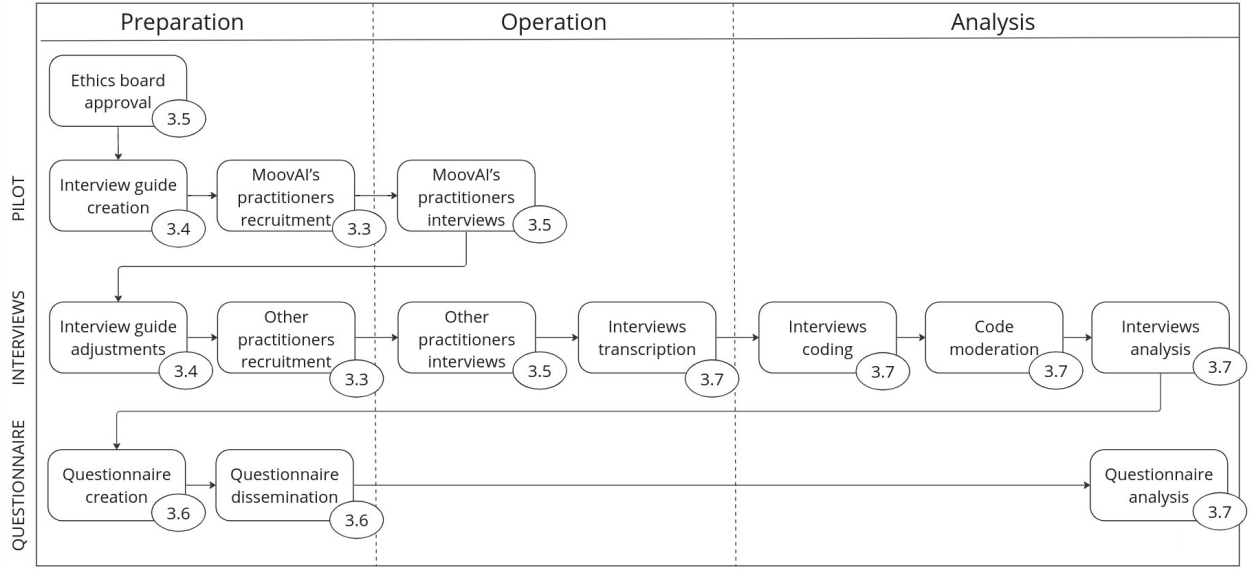


Figure 4.1 Overview of the methodology of our study.

4.3.1 Objectives and Research Questions

The goal of this study is to provide a detailed analysis of quality issues (including data and model) in MLSSs. We believe that interviewing people who really experienced these issues is an effective way to gain that knowledge. Thus, we define the following Research Questions (RQ):

RQ1: What are the quality issues encountered by practitioners when building MLSSs? For future works to solve quality issues, they must know the issues that exist. While there is some literature covering some quality issues [63, 80, 249], we believe that a lot is still unknown. In this study, we aim to share with the research community the quality issues encountered by practitioners when building ML systems. This includes issues related to data, models, and other components in MLSSs. We describe the **cause** and **consequences** of the discovered quality issues according to practitioners' experience.

RQ2: Which quality issues are the most prevalent? To help practitioners focus on the quality issues that are experienced more frequently, we also measure how frequent the discovered issues are in practitioners' experience.

RQ3: How are the quality issues currently handled by the practitioners? Once quality issues have been detected (e.g., in data or model), we expect practitioners to have put in place mechanisms to mitigate or at least attenuate their consequences. We

are interested in understanding the current mitigation approaches implemented in the industry.

RQ4: Which data quality aspect is the most problematic in practice? As it is widely known in the ML community, a large part of the practitioners’ time and effort when building ML solutions is spent on data [250]. Recent works [1, 2] have tried to build lists of desirable properties of data for ML (e.g. correct labels), to help create a common vocabulary to describe data quality. In the following, we will refer to these desirable data properties as data quality aspects. Hence, using [1]’s taxonomy, we aim to identify the data quality aspects that are the most frequently impacted by the data quality issues experienced by practitioners.

RQ5: In the case of data, which data types and collection processes are the most challenging in terms of data quality? The training data ingested by ML models come in many forms. Face recognition systems use images, while stock prediction applications can use numbers/amounts (like time-series data) or even text. Each data type comes with its own data quality challenges when training ML models. By answering this RQ, future research will be guided toward the most pressing data quality challenges faced by practitioners. Authors in [250] mention that data collection processes, the process of gathering data, may affect the quality of data. For example, one could choose to train one’s model on public datasets, or manually acquire more data with data collectors. Each of these processes has different challenges which may lead to different data quality issues. In this study, we want to identify the data collection processes that are most prone to data quality issues.

RQ6: What are the challenges of data quality assurance during model evolution? Many MLSSs encounter rapidly changing/non-stationary data, adversarial input, or differences in data distribution (concept drift, e.g., content recommendation systems and financial ML applications). Hence, the quality of the deployed model may decrease over time and consequently affects the performance of the whole system. Therefore, the robustness and the accuracy of the model’s predictions must be assessed frequently in production. Actively monitoring the quality of the deployed model in production is crucial to detect performance degradation and model staleness.

4.3.2 Industrial Partner

MoovAI⁵ is a Montreal-based company in Canada, which is active in developing AI/ML quality assurance solutions to address practical needs in various businesses. MoovAI’s experts

⁵<https://moov.ai/en/>

guide their customers (i.e., companies) to take advantage of these cutting-edge technologies, regardless of their level of maturity in data science. Nowadays, various sectors in industry are and will be developing ML models in their systems, e.g., the energy sector, the financial sector, supply chain recommendations, medical diagnosis, and treatment. As ML-based technologies become more widespread, the quality demands of ML become more important. Poor quality models are a potential barrier to exploiting ML in real-world applications and currently more companies request qualified MLSSs [251]. A study showed that 87% of ML proof of concepts have never come to production [252], usually due to lack of enough quality.

Currently, MoovAI is using a model validation tool to validate ML models prior to deployment in the real world [253]. This tool includes preliminary validation methods to assess the overall quality of an ML model. The tool evaluates the accuracy, stability, biases, and sensitivity of ML models. Now, MoovAI aims to push its tool forward to develop advanced methods for ensuring the quality of not only the ML models but also ML-based systems (i.e., software systems containing ML components) and to develop a stand-alone model validation platform. Experts at MoovAI now are looking for a comprehensive quality evaluation tool to assess and monitor the quality of ML models during their whole life cycle; from data collection to development, deployment, and maintenance.

4.3.3 Participants

In this section, we describe how we selected candidates for interviews and how we recruited them. In total, **we interviewed 42 participants**, which is more than previous similar studies [254,255].

Inclusion and exclusion criteria

In the following, we list the inclusion and exclusion criteria used to select practitioners to interview. In most cases, we could assess the exclusion criteria only after the interview has been conducted and we knew the practitioner’s experience. Hence, after conducting the interviews, we excluded 5 of them, **leaving 37 interviews for further analysis**.

Inclusion criteria:

- The interviewee is a data scientist or a machine learning engineer.
- The interviewee is a software engineer, a project manager, or a data engineer and has worked on MLSSs.

Exclusion criteria:

- The interviewee has not worked on MLSSs.
- The interviewee has not any professional experience in ML.
- The interviewee can not be understood by the interviewers because of a language barrier and the tool used to transcribe the interview is not able to provide an intelligible transcript.

Participant recruitment

We conducted two rounds of interviews. In the first round of the study, we interviewed the employees of our industrial partner (i.e., MoovAI). They are Data Scientists, Machine Learning Engineers, Data Engineers, and Project Managers (of ML projects). They have worked on many different ML projects for different clients. Thus, we think that they are able to provide a global view of quality issues encountered in the industry. We interviewed them between July 2022 and September 2022. In the second phase of the study, we interviewed practitioners from other companies. We interviewed them between November 2022 and February 2023. We adopted 4 strategies to recruit participants for the second round of interviews.

- *Q&A websites*: Questions and Answers websites are platforms on which a lot of knowledge is shared. Following similar previous work [254], we searched for practitioners with meaningful experience in ML willing to be interviewed on quality issues in ML systems. We chose to search on Stack Overflow⁶ and Data Science Stack Exchange⁷ because both these websites are significantly used by the ML community for question answering. We reached out to the top askers and answerers⁸ since they have shown significant involvement in the ML community and most likely have expertise. Since Stack Overflow is also used by the Software Engineering community to ask questions on topics unrelated to ML, we only picked the top askers and answerers on topics related to quality issues in ML systems. Similar to [254], we selected the topics *data-cleaning*, *dataset*, *machine-learning* and *artificial-intelligence*. For each topic, the Q&A websites provide the top askers and answerers of all time and the top askers and answerers of the last 30 days. We pick the top 10 practitioners from both lists. Hence, for each topic (Data Science Stack Exchange + 4 topics), we pick the top 10 askers and answerers from the ‘*All Time*’ and ‘*Last 30 Days*’ lists (10 users * 2 lists * 2 (askers and answerers) * 5 topics = 200 user profiles). We manually searched for the users’ email addresses using the

⁶<https://stackoverflow.com/>

⁷<https://datascience.stackexchange.com/>

⁸The words *askers* and *answerers* are part of the terminology used by the Q&A websites to describe people asking and answering questions

information given in the user profile. In total, we obtained 44 email addresses, but **no practitioner** answered our invitation. We conducted that search in September 2022.

- *GitHub*: GitHub is a hosting service for software development and version control using Git⁹. Its users have a public user profile they can customize. We observed that some users chose to indicate in their bio their job role. Hence, we leveraged that observation and searched users who have the keywords “Machine Learning Engineer” or “Data Scientist” in their bio. We used a script available in our replication package to automatically search the users and fetch their email addresses using the GitHub API¹⁰. In total, we managed to interview **23 practitioners** using this strategy.
- *Social networks*: Social networks are platforms in which users may engage in conversations on a wide range of topics. Some of them host discussions on ML and Artificial Intelligence in general. We believe users with important ML expertise can be found on these websites. We post an invitation for interviews on the deep learning and ML communities of Reddit¹¹ similar to our previous works [71]. The same invitation has been shared on two other MLOps communities on Slack¹² and Discord¹³, similar to [46]. We posted our message in October 2022 and got **no answer**.
- *Personal contacts*: We searched in our personal network for practitioners that have experience working with MLSSs. We managed to interview **1 practitioner**.

4.3.4 Interview Structure

Since the topic of quality issues in MLSSs is not mature and we believe there is still room for research work, we follow the principles of a research procedure suited for exploratory work, Straussian Grounded Theory [256]. It is a research method in which data collection and data analysis are executed in an iterative manner until a new theory emerges from the data. As opposed to many deductive approaches where a theory is first conceptualized and then tested through experiments, Grounded Theory goes the opposite way by inductively generating a theory from the data. Data collection stops when theoretical saturation is met: when the understanding of the subject is complete and new data does not invalidate the emerging theory. After 42 interviews, no new knowledge emerged from the interviews, so we stopped conducting interviews. Grounded Theory is often used when little is known about a phenomenon, because of its flexibility and its aptitude for the discovery of unknown concepts. Similar studies such as [249] followed Grounded Theory.

⁹<https://github.com/>

¹⁰<https://docs.github.com/en/rest?apiVersion=2022-11-28>

¹¹<https://www.reddit.com/>

¹²<https://mlops.community/>

¹³<https://mlops-discord.github.io/>

Because we are doing exploratory work, the interviews have to be structured in a way that allows the interviewees to share knowledge we might not be aware of. For this reason, we conducted semi-structured interviews similar to [254]. To help us cover every relevant topic, we devised an interview guide. It is composed mostly of open-ended questions so that the interviewee is able to direct us toward interesting information. It is the interviewer’s responsibility to ask follow-up questions when the respondent touches upon a subject relevant to the study. To avoid the interviewer being overwhelmed with his tasks and having difficulty asking the right follow-up questions, each interview is conducted by two persons. This follows the recommendation of previous works which show that interviewees share more information when two interviewers are present rather than one [257]. The role of the second interviewer is to help the primary interviewer to ask follow-up questions.

To draft an initial set of questions for the interview guide, we considered the quality issues elicited by other studies [62, 63, 70, 79, 255]. We also considered the data quality aspects defined by [1]. We completed the initial set of questions by drawing questions from a similar study [254]. We assessed the quality of our interview guide by conducting a pilot of our study. We purposefully selected 5 participants with diverse experience (3 Data scientists, a Data engineer, and an AI project manager) and interviewed them using the interview guide. By doing so, we verified that our questions are unambiguous, precise, and able to answer our research questions effectively. The questions that did not satisfy the aforementioned criteria were improved. We also added questions we believed were missing in order to address our research questions.

4.3.5 Interview Process

All the interviews were conducted in English except one that was conducted in French. After a round of introduction between the interviewee and the interviewer, we provide a brief overview of the project. Then we ask participants for some background information regarding their experience in ML. The interview officially starts with the following question: “*What are the main quality issues you have encountered with your data/model so far*”? By asking an open-ended question, we are allowing the interviewee to share experiences he is the most confident to talk about. Then we probe the interviewee’s experience in an attempt to discover quality issues. During the interview, we cover all the phases of ML workflows described by [76], so that we can cover most situations where quality issues might occur. However, we do not cover the model requirement phase. For conciseness, we merged the data cleaning phase with the feature engineering one and the data labeling phase with the data collection one in our interview guide. We describe below the questions asked in every section

of our interview.

- *Data collection*: We ask practitioners for their experience with data collection and the issues they have experienced. We also ask for the solutions they put in place to address the aforementioned issues.
- *Data preparation*: Notably, we ask about pain points when preparing data for ML and for tools to automate the process. We consider any challenge related to the cleaning and transformation of the data.
- *Model evaluation*: We probe for issues the interviewee discovered when evaluating the quality of its model. For example, we ask the respondent if he was ever surprised by the poor performance of a model when it was evaluated by the users of an MLSS.
- *Model deployment*: We gather general information about the process by which models are put into production. Then, we search for issues encountered at this step. For example, one question is: “*Did you ever deploy a model that performed well locally but poorly once deployed?*”.
- *Model maintenance*: We ask the interviewee how he ensures that the quality of its models remains the same after deployment. We specifically ask for past instances of model staleness and how it has been handled.

At the end of every section, we ask for any other issue that the interviewee may have at this step of the workflow in case we missed out on something. We conclude the interview with the open question: “*In your opinion, what is the most pressing quality issue researchers should try to solve?*”. The answer to that question might provide interesting future work directions and, by asking the question, we follow Harvard’s best practices for qualitative interviews¹⁴. The interview guide is available in our replication package.

A few days prior to the interview, we provide an overview of the interview with the participants so they can be familiar with the objectives of the study. Just before starting the interview, we ask the interviewees for permission to register and share the transcription. In order to follow ethical guidelines, we validated our research project with Research Ethics Committee at Polytechnique Montréal¹⁵ and got their approval. During interview coding, we clarify with the interviewee anything she mentioned that was ambiguous or could not be understood without further information.

¹⁴https://sociology.fas.harvard.edu/files/sociology/files/interview_strategies.pdf

¹⁵<https://www.polymtl.ca/recherche/la-recherche-polytechnique/exigences-deontologiques/travaux-de-recherche-avec-des-etres-humains>

4.3.6 Questionnaire

After interviews have been conducted and analyzed, we expect to have a set of potential quality issues. In order to validate the quality of our findings and answer RQ2 (i.e., which quality issues are the most prevalent), we share a questionnaire with other practitioners. In this questionnaire, we present to the respondents our final list of quality issues and we ask them to rate on a scale from 1 to 5 how frequently they experience the described issue. Respondents have the choice of not answering a question if they think they do not have sufficient experience to answer the question. We also leave a free text field in the form where practitioners may write any comments. We collect demographic information at the beginning of the questionnaire. The questionnaire is built using Google Forms¹⁶.

We used the same inclusion and exclusion criteria from Section 4.3.3 to select participants for our questionnaire. We sent our questionnaire to other employees of MoovAI that did not participate in the interview, personal contacts, and to practitioners identified from social networks. More details regarding these channels can be found in Section 4.3.3. We also shared the questionnaire with practitioners during an MLOps event organized by our team¹⁷. We shared the questionnaire in April and May 2023 and got **21 answers**.

4.3.7 Analysis Plan

Before analyzing any interview, we generated the transcript of the interview using the tool Descript¹⁸. It is an automated speech recognition tool that can transform the audio stream of an online meeting into text. We corrected any transcription mistake that obscured or distorted the essence of the message conveyed by the interviewee. Additionally, to preserve the interviewee’s privacy, we remove any identifier to other entities (e.g., company names, schools the person attended, etc.). We also send the anonymized transcript to the interviewee to ensure there is no private information left. The anonymized transcripts are available in our replication package.

We use the coding techniques from Straussian Grounded Theory [244] to extract knowledge from data to answer all our qualitative research questions (i.e., RQ1, RQ3, RQ6). When analyzing an interview, we start with a round of open coding [258] to break the interview’s content into discrete parts. After the interviews are conducted and we have a set of coded quality issues, we grouped them into categories so that reasoning over the issues is easier. This step is akin to axial coding in Straussian Grounded Theory [244]. While the current

¹⁶<https://www.google.ca/forms/about/>

¹⁷<https://semla.quebec/mlops/>

¹⁸<https://www.descript.com/transcription>

document has been written such as the data collection (i.e., the interviews) and analysis plan are separated, it is important to understand that these two steps happen in an iterative manner. In other words, we analyzed an interview right after we conducted it. Throughout the process, we write down interesting information mentioned during interviews and ideas for preliminary categories in “memos” [244]. They are integrated into our theory during the phase of memo sorting [244].

In order to identify the most prevalent quality issues and effectively answer RQ2, we compare the number of practitioners who answered that an issue is frequent (i.e. scores of 4 and 5 on a Likert scale) in our questionnaire.

To answer RQ4, we first perform a round of open coding for data quality issues. Each data quality issue mentioned by a practitioner is assigned a code (e.g., “incorrect labels”). Once the interviews are all conducted, the researchers individually categorized each code into one of the data quality aspects defined in [1]. We resolved the conflicts through discussion and consensus. We finally answered the question by counting the number of practitioners having experienced an issue with one of the data quality aspects. Note that some data quality issues could not be assigned to a single category (or to any). In this case, a researcher inspected each issue individually and categorized them.

In RQ5, we aim to identify the data types and data collection processes that are the most problematic in practice. We use the data quality issues extracted in RQ4 to find the most problematic data types. Most precisely, a researcher considers each issue and tries to determine to which data type the interviewee was referring. In case of doubt, the extract is ignored. We use the same process to determine the most problematic data collection process. Additionally, to ensure no important information is missed, we tag every issue caused by a data collection process. For example, if an MLSS is unreliable because of the data collection process, we code the extract with the data collection process’ name. Similar to RQ1, the codes are grouped into categories through axial coding [244] so that reasoning over the issues is easier.

To help us code the transcripts, we used Delve qualitative analysis tool¹⁹, because the researchers are familiar with it and it is easy to use. Delve is a computer-assisted qualitative data analysis software (CADQAS) that provides simple interfaces to code and analyze data. In order to ensure the quality of the analysis, each document is coded by two researchers. In case of a disagreement in codes, a third researcher plays the role of moderator and selects the final code for a text segment. This process is helpful for the construction of a shared understanding of the data. The coded transcripts of the coders and moderators are available

¹⁹<https://delvetool.com/>

in our replication package.

4.4 Study Results

In this section, we present the results we obtained from our interviews. We start by providing demographic information of the interviewees and the practitioners who answered our questionnaire. Then, we present the results for each of our RQs. To refer to a practitioner, we use identifiers. Each identifier has the following structure: “[*PXX*]”, where *P* is for “practitioner” and “*XX*” are digits. Note that the numbers in the identifiers are not representative of the order we interviewed the practitioners, they are random identifiers assigned to interviewees. When using pronouns, we will always use the feminine form to preserve the practitioner’s privacy. We also use identifiers to refer to the quality issues described in Section 4.4.2. These identifiers have the following structure: “[*IXX*]”, where “*I*” is for “issue” and “*XX*” are digits.

4.4.1 Demographic information

In this section, we present the demographic information we collected from the interviews we conducted and the questionnaire.

Interviews

Here, we present demographic information from the 37 interviews we conducted that were not excluded from the study. We extracted that information from what the interviewees shared with us before and during the interview. If any information was missing, we searched for the online public profiles of the person (i.e., LinkedIn²⁰). If unsuccessful, we tried to contact the person again to answer our questions. In case of failure, this information was left empty and displayed differently depending on the plot type. For waffle plots (i.e., Figures 4.2a, 4.2b, 4.4b), missing values were replaced with the “unknown” tag and displayed on the figure. For histograms (i.e., Figures 4.2c, 4.2b), missing values are not shown for clarity. The number of missing values for Figure 4.2c is 5 and 8 for Figure 4.2b.

The demographic information is presented with two sets of figures, i.e., Figure 4.3 and Figure 4.5. The former displays the demographic information of the practitioners, while the latter displays the demographic information of the company the practitioners currently work at.

As we can see from Figure 4.2c, we interviewed practitioners with varying years of professional experience. The practitioner most generally had between 3 and 5 years of experience in ML.

²⁰<https://www.linkedin.com/>

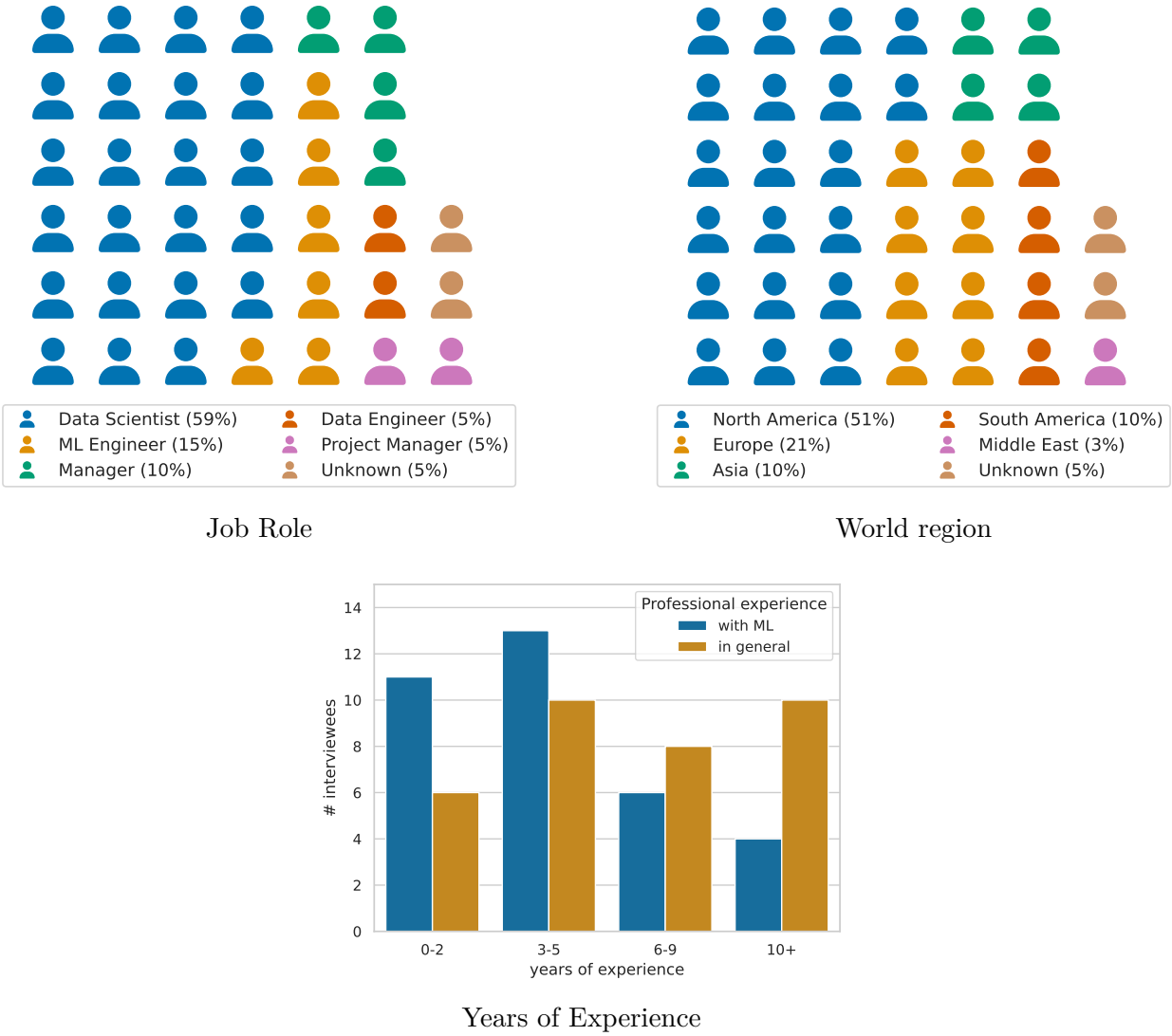


Figure 4.3 Practitioners’ Demographics (Interviews)

We interviewed 13 people with 3-5 years of experience in ML compared to 4 for people with more than 10 years of experience. Not all the interviewees started their careers in ML. As a result, the distribution of years of experience in ML is skewed towards the left (i.e., a larger amount of interviewees had a bit of experience compared to a lot of experience). We posit that the recent success of ML created job opportunities that motivated practitioners to work in that domain. We display the practitioners' job roles in Figure 4.2a. We used the title they described themselves. We included in "manager" all roles that orbited around management, such as CTO. As we can see, the majority of the interviewees (59%) are data scientists. In contrast, 20% are engineers (i.e., ML engineers or data engineers) and 15 % have management roles (i.e., manager or project manager). Figure 4.2b displays where the practitioners are based. Because we interviewed practitioners from MoovAI, the majority of interviewees are from North America (51%). Excluding the 15 practitioners we interviewed from MoovAI, we have a more balanced set: 21 % are from Europe, 13% from North America (excluding MoovAI's practitioners), 10% from Asia, 10% from South America, and 3% from the Middle East.

In Figure 4.5, we display demographic information on the company the practitioners currently work at. In Figure 4.4a, we demonstrate the number of employees. Once again, a category (11-50) has a larger count because of MoovAI's employees. Apart from that, we managed to interview practitioners from a wide range of company sizes: 4 practitioners work in startups (1-10 employees), 15 in small to medium-sized companies (11-50 and 51-500), and 7 in large companies (500-10000 and 10001+). We display the companies' industry sectors in Figure 4.4b. We used the Global Industry Classification Standard²¹ to classify companies. We added the category "government" for practitioners working for the government. The majority of our interviewees (72%) work for a company in the information technology sector: 8 % work in the financial sector, 5 % in the consumer staples sector, and 1 person for a government.

Questionnaire

In this section, we present and discuss the demographic information collected using the 21 answers to the questionnaire. In comparison to the interviews, we only collected the job roles and years of experience of the practitioners. We wanted to avoid collecting too much information to avoid deterring practitioners from answering the questionnaire. Thus, we only collected data that did not allow us to identify the practitioner or his company.

In Figure 4.6b, we display the professional experience of the practitioners who answered the questionnaire. Similar to the interviews, the interviewees most generally had 3-5 years

²¹https://en.wikipedia.org/wiki/Global_Industry_Classification_Standard

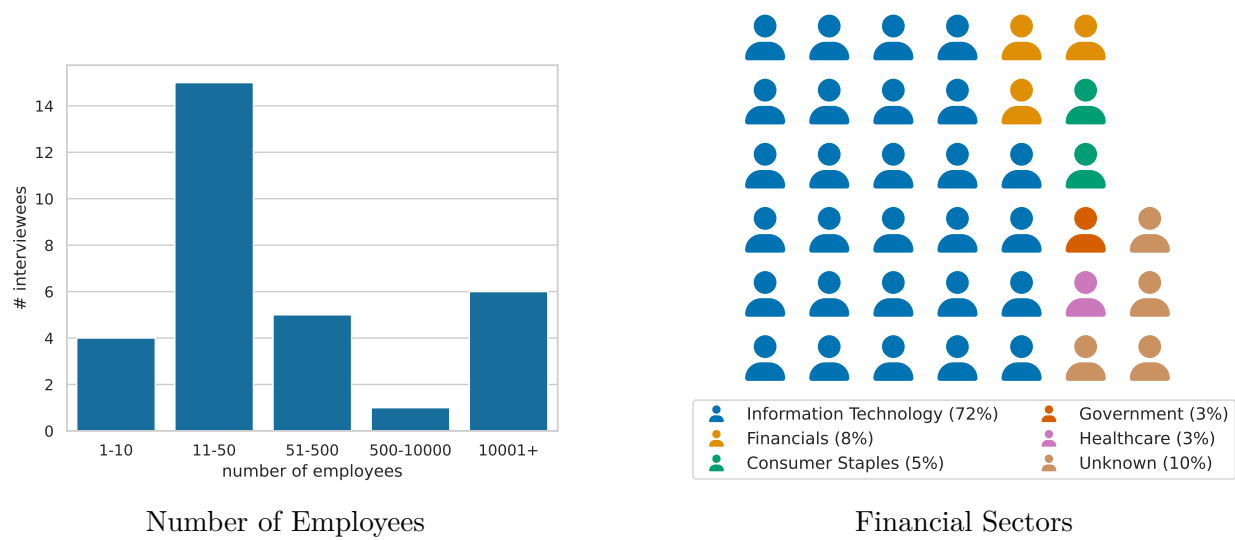


Figure 4.5 Interviewees’ Companies’ Demographics

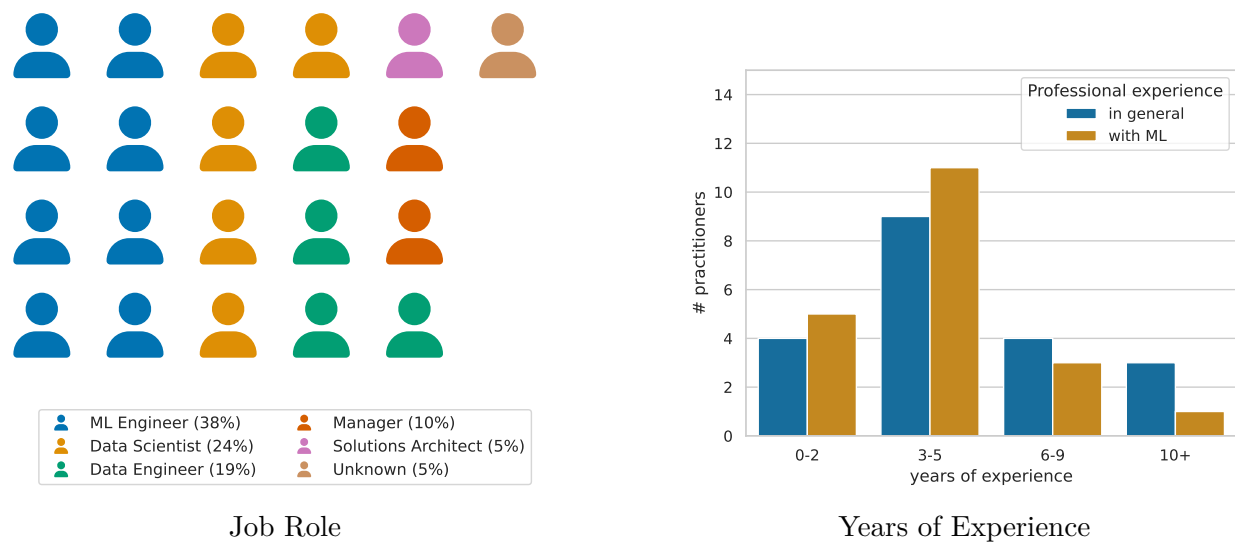


Figure 4.7 Practitioners’ Demographics (Questionnaire)

of experience. However, the distribution of experience (with ML and in general) is more heavily skewed toward the left. Thus, on average, the practitioners were generally a bit less experienced than the ones from the interviews. In Figure 4.6a, we present the job roles of the practitioners who answered our questionnaire. We obtained a diverse sample of professionals. While Data Scientist was the most common job role for the interviews, the most common one for the questionnaire is ML Engineer. Almost as many Data Engineers (4) answered our questionnaire compared to Data Scientists (5).

4.4.2 RQ1: What are the quality issues encountered by practitioners when building MLSSs?

In this section, we describe the quality issues extracted from our interviews. For each issue, we present its causes and consequences according to the experience of our interviewees. For clarity, we grouped the quality issues into quality dimensions. For example, the issues making the maintenance of an MLSS more difficult are presented under the “Maintainability” subsection (Section 4.4.2). We provide the full list of quality issues in Table 4.1. The issues’ identifiers defined in this table will be used throughout the rest of the paper to refer to an issue without having to describe it all over again.

Evaluability

Practitioners often mentioned that evaluating the quality of their models was difficult, especially offline. As [P17] said: *“There is no good enough indication to say that a model will perform as expected in the real world. [...] Before your deploy model, you cannot really know how it will perform, but at least you can guess. And guessing is not good enough”*. Similarly, evaluating the data quality of a dataset is difficult and can be time-consuming. So, we define evaluability as the ability to evaluate the quality of ML models and datasets. Generally, when evaluating models, practitioners follow one of two evaluation procedures. We describe the evaluation procedures along with their issues below.

In the research community, the most common way to evaluate the quality of a model is to measure its performance on a test dataset using a metric such as accuracy or F1-score (depending on the application). This is also how most practitioners learned to evaluate their models and, as a result, how they evaluate their models in practice. However, practitioners commonly shared that they were often disappointed with the performance of their model when deployed in a production environment [P7, P9, P16, P17, P23]. We observed two reasons that can explain why this procedure to evaluate datasets is flawed. First, the goal that is targeted when an ML metric such as accuracy is used is different from what is important for

Table 4.1 List of quality issues in MLSSs

	Id	Description
Evaluability	1.1	Evaluating the quality of a model offline (i.e., not in a production environment) is inaccurate even when the dataset used for evaluation is representative of the data distribution in production.
	1.2	Defining a good business metric for evaluating an MLSS is difficult. For MLSS, a business metric refers to the degree to which the MLSS successfully achieves the goal it has been built for.
	1.3	Trying to simulate the environment/system in which the model will operate (to evaluate the model in this simulated environment) is difficult and error-prone.
	1.4	Evaluating the quality of a dataset (e.g., presence of incorrect labels, noisy/incorrect features, wrong format data, etc.) is difficult and time-consuming.
Explainability	2.1	Explaining a model's predictions to people without ML knowledge (e.g. business stakeholders, users) using explainability techniques is challenging because these techniques require technical knowledge and interpretation.
	2.2	The explanation techniques sometimes provide explanations that do not make sense and can not be relied on.
Debuggability	3.1	Reproducing bugs in an MLSS is difficult because of unstable data sources. A data source is unstable if it returns different values for the same queried record.
	3.2	Debugging data streaming systems (e.g., Hadoop) is difficult because it is difficult to picture what data should look like at each step of the data pipeline.
	3.3	Debugging an MLSS is time-consuming when its data sources are managed by other teams, because it may require inspecting these data sources.
Efficiency	4.1	Training models consume too many resources (e.g., time, computing power, etc.).
	4.2	The queries sent to an MLSS are not answered timely (i.e., latency/delay issues).
	4.3	At inference time, models consume too much memory.
Maintainability	5.1	Maintaining an MLSS is difficult because there is not enough information describing the data the MLSS consumes.
	5.2	Maintaining a model is difficult because there is not enough information describing how the model was generated.
	5.3	Managing the dependencies (i.e., software libraries) of an MLSS is challenging and error-prone.
Reliability	6.1	Having a reliable model is difficult because of concept or data drift.
	6.2	Having a reliable model is difficult because of external data providers.
	6.3	Having a reliable MLSS is difficult because of the data pipelines which are brittle and have technical debt.

the end application. As a result, a model that is optimized for an ML metric may not answer optimally to the needs of the application (in terms of quality of predictions) [P3, P7, P9]. For example, for a recommendation engine, improving accuracy might not lead to increased revenues (i.e., the business need), if the recommended products only provide minor profit [P9]. Perhaps more profitable products should have been recommended instead. Second, the evaluation procedures do not take into consideration the system in which the model is embedded [P1, P16]. An improvement to a model in terms of accuracy on a testing dataset might not be of relevance since it does not take into consideration how the other components interact with the predictions of the model. For example, improving the component responsible for query understanding in a search engine might not improve the engine's search results overall since other components, such as the item understanding component, also influence the overall engine's performance [P16].

To address these shortcomings in the evaluation procedure, some practitioners estimate the quality of a model by simulating the environment in which it will be deployed and measuring business metrics. Business metrics are quantifiable measures that can be used to track the performance level of a business [259]. For MLSSs, business metrics give a measure of how well an MLSS achieves the goal it has been built for. For example, for an ML trading system, a business metric could be the profit made. We observed two issues making evaluating a model in a sandbox using a business metric difficult. First, reproducing the production environment is error-prone [P1, P7]. For example, [P7] did not take into account that the data used for prediction was available only after a delay and their model had to predict further in the future than expected. After reflecting on her experience with building a sandbox for a project, [P1] said that she thinks moving to A/B testing sooner (i.e., online evaluation) would have been preferable instead of improving the sandbox. The reason is that A/B testing is easier to implement and provides measures of quality that are more accurate. Second, defining good business metrics to prove the impact of a model is not an easy task for some applications, such as recommendation engines [P17, P25]. As [P17] said: *"There is no strong definition of what's the best recommendation, right? The only thing we can do is monitor how much people return. [...] So when people are happy with their purchase, they return less. So that's kind of a belief, just a belief there"*.

As a result of not being able to accurately measure the quality of a model, practitioners have ended up (1) spending a lot of time on improvements that do not provide any real value [P9], (2) deploying models with low quality [P1, P7, P9, P13, P16, P17, P23], and (3) being unable to prove the value added by an MLSS to business stakeholders [P25]. Hence, stakeholders may lose confidence that the MLSS brings value to their company. As [P25] said: *"[My manager] is skeptical if these are the right metrics [...] [As a result] you are trying*

to prove [that your system brings value], but your proof is not considered or taken forward".

In the early phases of building an MLSS, practitioners usually spend time understanding datasets and evaluating their data quality. Ensuring the quality of a dataset is especially important when it has been created by a third party. For example, [P23] mentioned that a public ML dataset contained irrelevant and harmful images by mistake. Practitioners have reported struggling to evaluate the data quality of a dataset [P8, P23, P30] and spending a lot of time doing so [P8, P30]. When asked about the most pressing issue in ML in her opinion, [P30] answered the following: *"We should find some method [to] understand faster the quality of the data. [...] I mean, just push one button and understand the data quality".* Estimating the quality of a dataset can be used to (1) assess the feasibility of applying ML to a problem, (2) estimate the effort needed to clean the dataset, and (3) select datasets. We posit that evaluating the quality of a dataset can take a lot of time because it requires a deep understanding of the domain and datasets are not always sufficiently documented (more details in Section 4.4.2). This problem is exacerbated by the size of datasets that can be quite large [P8]. [P8] reported using tools to help her estimate data quality but ended up giving up because it required too much effort: *"yes, there are tools, but most that I've tried, always require so much time investment that they become less worth than just discarding data that we find is suspicious".*

Explainability

Explainability refers to any technique that tries to explain the predictions of a model [260]. While one of the goals of explainability is to provide intelligible explanations for the predictions of an ML model [260], practitioners have reported that using explainability tools to explain the predictions of a model to people without ML knowledge (e.g., users and business stakeholders) is challenging because these tools require interpretation and technical knowledge [P3, P9, P27, P32]. As [P9] said: *"the interpretation of the outputs of the model [is] very hard to vulgarize to the business and to the stakeholders".* On the topic of explaining predictions for business stakeholders, [P3] reported: *"most of the time, [explanations] need to be in the lamest terms. If it's mathematical gibberish, it won't mean anything to them. So the challenge is also to make it intuitive and useful for them".* As a result, explanation techniques are seldom used to increase users' confidence in the MLSS. Over the 42 interviews conducted, only practitioner [P32] reported using explainability techniques in production to increase trustworthiness.

Practitioners have also reported being distrustful of the outputs of explainability tools because the explanations do not always make sense [P4, P10, P11]. Regarding one of her

previous experiences with Shapley values, [P4] shared the following: *“[I] see [that the] contribution is positive [for] this variable, but it should be negative. What’s happening? And then it produces more questions than answers. [It] is really a difficult task to have good explanations”*. Our practitioners mentioned observing that the tools’ explanations get seemingly worse as the number of features used by the model increases [P10] or when the complexity of the model (in terms of the number of parameters) increases [P4]. To show her mistrust is justified, [P11] referred us to the study made by [261]. Notably, it shows cases when Shapley-value-based feature attribution methods produce faulty explanations. Consequently, practitioners become distrustful of the explanations given by these tools and avoid using them in high-stakes scenarios. For example, [P20] preferred using interpretable models instead of explainability techniques for an application facing regulatory constraints.

Debuggability

Debugging refers to identifying and removing bugs from a computer’s hardware or software [262]. Hence, we define debuggability as the degree to which a system can be easily debugged.

In order to fix a bug, developers will generally try to reproduce it to identify its root cause. This process is difficult with MLSSs because of unstable data sources [P1, P3, P16]. To reproduce a bug, the same input data must be fed to the system. However, the data sources may return different values for the same queried object, because of unstable data sources. For example, [P16] worked on a recommendation system that ingested embeddings. These embeddings were generated daily; the new version replaced the old one. By the time [P16] was aware of a bug, the embeddings that triggered the bug were already replaced. Thus, reproducing a bug proved to be impossible at times. As a result, bugs may tend to be ignored, reducing the trust of stakeholders in the system. Similarly, data sources that are unstable because of historical revisions of data will render the MLSS less debuggable [P1, P3]. We refer the reader to Section 4.4.7 for further information on the challenges with unstable data sources.

Once the data required to reproduce the bug is obtained, the process of searching for the root cause of a bug can begin. Our practitioners have reported that this process can be difficult for two reasons. First, debugging data streaming systems is more difficult than debugging other software systems according to our practitioners’ experience [P8, P19, P25]. When reproducing a bug in order to find the root cause, one might verify the state of data at key steps in the data pipeline. However, knowing the expected state of data at these key steps is difficult. As a solution to this problem, one could compare the data against what data normally looks like after the transformations. For example, one could compare a

value against the mean or the mode. However, this is not possible with our practitioners' data streaming systems, since only one entry is processed at a time. Hence, data streaming systems are recognized to be challenging for debugging [P8, P25]. Second, debugging MLSSs can be difficult because communication across teams is often necessary to find the root cause of an issue [P27]. As explained in Section 4.4.7, MLSSs are highly coupled with the data they are consuming, and any issue with its quality might affect the MLSS's quality. Hence, in case of an issue with the MLSS, practitioners commonly have to inspect the MLSS's data sources, which are seldom managed by their team. Communicating with the team managing the other data source may be a long back-and-forth process because the other team has its own agenda and does not necessarily prioritize the bug. Additionally, the practitioners may have to reach out to more than one team, extending the process. This problem was also observed by [67].

Efficiency

The ISO/IEC 25010:2011 standard defines efficiency as: "[the] performance relative to the amount of resources used under stated conditions". Efficiency issues in MLSSs are critical because of the large volume of data MLSSs have to process. According to our interviews, efficiency has become a growing concern, as running MLSSs is expensive. [P8, P17, P23, P27, P26, P29, P30, P37]. Efficiency issues arise both at model training and inference time.

Practitioners have reported long training times, which leads to hefty spending [P8, P23, P30] and slow product development [P3]. At inference time, practitioners mentioned concerns with the latency of data pre-processing pipelines [P16, P38] and the memory consumption of the model [P31, P38]. Memory consumption can be an issue for applications that run on smaller devices, such as cellphones [P35]. [P38] observed that most latency issues come from the data pre-processing pipeline and not from the model.

Maintainability

The ISO/IEC 25010:2011 standard defines maintainability as: "[the] degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers". Practitioners have mentioned having issues with maintaining datasets, models, and libraries.

As it is explained in Section 4.4.7, the quality of ML models highly depends on the quality of the data it processes. Hence, maintaining an MLSS requires not only a good understanding of the software and the ML models but also of the data. This understanding can be communicated verbally but is more effectively shared through documentation. As practitioner [P29]

expressed: *“It’s a time-consuming process to go back and forth and ask about the correct description of the columns, and sometimes there is no clear answer”*. We have observed two types of documentation issues with datasets. First, there often is a lack of documentation describing the data in the dataset [P1, P2, P5, P7, P8, P19, P24, P29, P38]. As a result, practitioners may be unable to understand what a feature represents [P1, P5, P29], how it is measured (i.e., what are the units) [P1] and what is the meaning of special tokens [P2]. As an example of unspecified special tokens, a “none” value may mean that there is nothing for that attribute or that there has been an oversight to provide the real value [P2]. This is a problem when building MLSSs, but also when maintaining them, since modifying the system without data documentation is slower and more error-prone.

Second, practitioner [P8] expressed not being able to assess how reliable a data source is. As the interviewee shared: *“So we had [...] over 50 different sources on that project. We had a lot of issues tracking, which data came from who and which data was, let’s say, uh, production-grade, or, uh, just experimental”*. As detailed in Section 4.4.7, knowing the reliability of a data source is important during the maintenance of a system to ensure the reliability of the system.

We observed other issues that we posit are symptoms of poor data management, such as temporal data leakage (i.e., evaluating a model with data that would not be available if the model was deployed) [P20, P23] or corruption of production data with dummy data [P8, P35]. Dummy data can be mixed with production data when the production environment of an MLSS is used for testing and the dummy users created for testing are not deleted after usage. These dummy users can degrade the production model’s performance.

Similar to the datasets, practitioners have reported having issues with model maintenance. There can be poor experiment tracking [P20, P23, P25]. Experiment tracking provides the documentation necessary to understand how a model was created. As a consequence of a lack of such documentation, [P20] was unable to maintain a model, and practitioners [P20, P25] found it difficult to explain to stakeholders why a model had performance issues. As a result, stakeholders may lack trust in the system [P25]. Experiment tracking becomes of paramount importance when ML projects are handed off to other teams. Failure to do so may force the maintainers to rebuild the model from scratch since there is not enough information to retrain the models while following the same procedure, as practitioner [P20] experienced.

Dependency management has also been an issue for some practitioners [P10, P16, P26, P27, P38]. As expressed by practitioner [P38]: *“I think a big problem is, dependency hell [...] it is a big problem because [environments are easily] broken”*. Poor dependency management can break an MLSS in a production environment [P16, P26, P27, P38]. This situation may

happen if the development environment used by the practitioners is not the same as the production environment [P27] or if dependencies are not pinned and automatically upgraded by the package manager [P16, P26, P38]. Forgetting to pin library versions can become taxing when popular open-source libraries such as “Transformers” are used since they have a lot of version upgrades [P16].

Reliability

The ISO/IEC 25010:2011 standard defines reliability as: “[the] degree to which a system, product or component performs specific functions under specified conditions for a specified period of time”. Building reliable MLSSs is challenging because ML models are tightly coupled to the data they have been trained on. Any change in the distribution of data might hinder the predictive capabilities of a model. Thus, in addition to the reliability issues that may be encountered in software systems without ML, owners of MLSSs must tackle the challenges of data quality assurance during model evolution. Reliability issues with data may stem from the data providers or the data transformation pipelines.

We observed two issues with data sources that affected the reliability of MLSSs: (1) data and concept drift, and (2) unstable external data providers. We describe these issues in Section 4.4.7. As a result of unstable data sources, practitioners reported their MLSSs experienced poor ML performance [P4, P7, P10, P16, P21, P23, P25, P37] or were unable to deliver predictions [P8, P10]. Managing unreliable data sources is perceived as challenging by practitioners and, if poorly done, can lead to the failure of ML projects, as experienced by [P7]: *“If you are taking data from data sources and one of them is missing, [...] the best you can do is sending alerts and stuff like that. But if your system has too many alerts, [the clients] may just shut it down and say, whatever, let’s forget about ML”*.

In addition to data sources, data transformation pipelines can be the cause of reliability issues. A brittle data pipeline that incurs technical debt [63] may be a cause of reliability issues for MLSSs. A poorly designed pipeline may not handle well data with unexpected characteristics (e.g., wrong data type, missing values, outlier values) and create feature vectors that can not be recognized by the model. As a result, an MLSS may produce poor predictions [P13, P16, P25] or even crash [P8].

4.4.3 RQ2: Which quality issues are the most prevalent?

As explained in Section 4.3.7, we answered this RQ using a questionnaire. In the questionnaire, we asked practitioners to rate on a scale from 1 to 5 how frequently a quality issue is

according to their experience. A practitioner answers 1 if he never experienced an issue and 5 if he frequently has it. The responses are not mandatory; hence, if a practitioner does not understand a question or feels he does not have the necessary experience to answer one, he could skip it. We present the results using Figure 4.8 and Figure 4.9. For conciseness, we use the issues’ identifiers (e.g., *[I3.1]*) instead of their full description to refer to the issues. The raw results of our questionnaire (e.g., the number of ratings of 1 for an issue or the number of times a question was skipped) are given in Annex A. As mentioned earlier, 21 practitioners answered our questionnaire.

Figure 4.8 displays the distribution of the ratings (from 1 to 5) for each quality issue. The color code used in the figure maps to the scale of 1 to 5 used in the questionnaire. One can observe that some issues are more prevalent than others. For simplicity, we consider that the issues that are the most frequent are the ones crossing the 50% threshold on the right side of the figure, while the ones that are rarer are under the 50% threshold. The issues in the former group are issues *[I1.1, I1.2, I1.3, I1.4, I2.1, I3.3, I4.1, I5.3, I6.1, I6.2, I6.3]*, while the ones in the later are *[I3.1, I3.2, I4.2, I4.3, I5.1, I5.2]*, with *[I2.2]* exactly on the threshold. We identify two potential factors explaining why some issues are more frequent than others. First, there are issues that are common to a lot of ML projects, while others, only to projects with specific constraints. For example, the issues *[I4.2, I4.3]* (which refer to latency and memory issues respectively) might only be experienced for projects with speed or resource constraints. Other issues, such as *[I1.1]* (which refers to the challenging aspect of evaluating models offline), are applicable to almost any ML project. Second, some issues already have solutions or mitigation techniques designed to attenuate their consequences. For example, memory consumption issues can be attenuated using quantization methods [263], as we suggest in Section 4.5.5. As a result, practitioners who are faced with such issues will not consider it to be a problem, since a mitigation technique already exists.

As mentioned in our methodology (Section 4.3.7), we compare the ratings of considering an issue to be frequent based on the answers to our questionnaire to compare the prevalence of quality issues. We display the results in Figure 4.9. We use the identifiers defined in Table 4.1 to refer to the quality issues. The top three issues with the highest percentage are *[I1.2, I1.3, I3.3]*. Two out of the three most frequent quality issues are problems with evaluability. The three least frequent quality issues are *[I4.2, I4.3, I3.2]*. Two out of them are issues with efficiency.

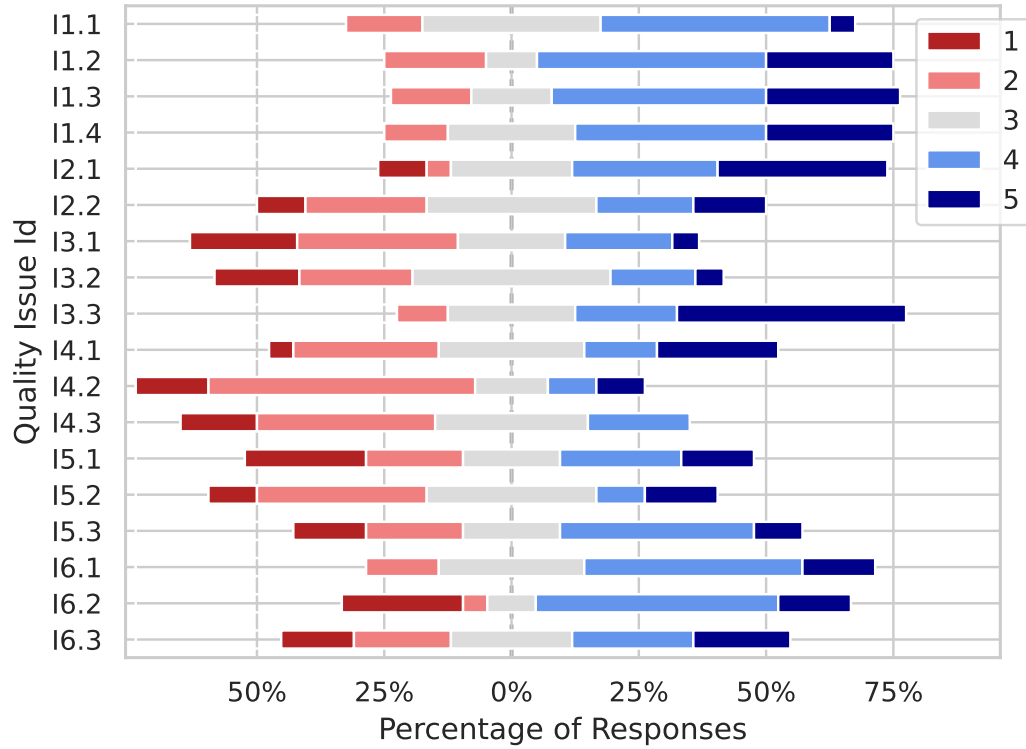


Figure 4.8 How common the issues are according to the practitioners' experience.

Key findings: Evaluability is the most frequently problematic quality dimension. Practitioners often spend a lot of time debugging MLSSs when they have external data providers. Efficiency issues are less often experienced according to our practitioners' experience.

4.4.4 RQ3: How are the quality issues currently handled by the practitioners?

In this section, we share the strategies the practitioners designed to handle the quality issues mentioned in Section 4.4.2. These strategies do not solve the issues, they only attenuate their consequences. Some issues are not covered in the following, since the practitioners from the interviews and questionnaire did not share any particular approach to address them. We provide a simplified description of the strategies used by the practitioners in Table 4.2.

Table 4.2 List of techniques to mitigate the quality issues' consequences.

	Issue's Id	Practitioners' mitigation techniques
Evaluability	1.1	Evaluate the model online using a business metric. Evaluate the model using the feedback of domain experts, users of the MLSS, or business stakeholders.
	2.1 2.2	Instead of explaining the predictions of a model, make the system more transparent to the user. Custom strategies can be developed for that purpose (e.g. presenting historical performance on similar cases to the user).
Debuggability	3.1	Temporarily save a copy of a record in a database whenever it has been used by a model (for training or inference).
Efficiency	4.1	Train models across different machines at the same time using distributed frameworks.
	4.2	Use ML architectures with a reduced number of parameters.
	4.3	Distribute computations across many computers using distributed frameworks. Use ML architectures with a reduced number of parameters.
		Optimize data processing pipelines (e.g. by using vectorized operations). Decommission features that are not important for prediction to reduce the load on the system and make it more efficient.
Maintainability	5.1	Follow rigorous project management methodology where data documentation and understanding play a central role. Use a data storage procedure that forces its users to document data when entered into the system.
	5.2	Use experiment tracking tools, namely MLflow, to automatically document the hyperparameters used to train ML models.
	5.3	Share the same development environment between developers and use that environment in production.
Reliability	6.1	Automatically retrain ML models on the latest data. Define a systematic data collection process to avoid annotation drift.
	6.2	Before using a data source, assess its reliability and, if it is not reliable, avoid using it. Fetch the same information from different data providers to have data redundancy. Monitor the data sources to address any issue in the MLSS as fast as possible. In case of an issue with a data source used at inference, inform the user that the quality of the model's predictions will be degraded. In case of an issue with data used for training, roll back to a previous version of the model.

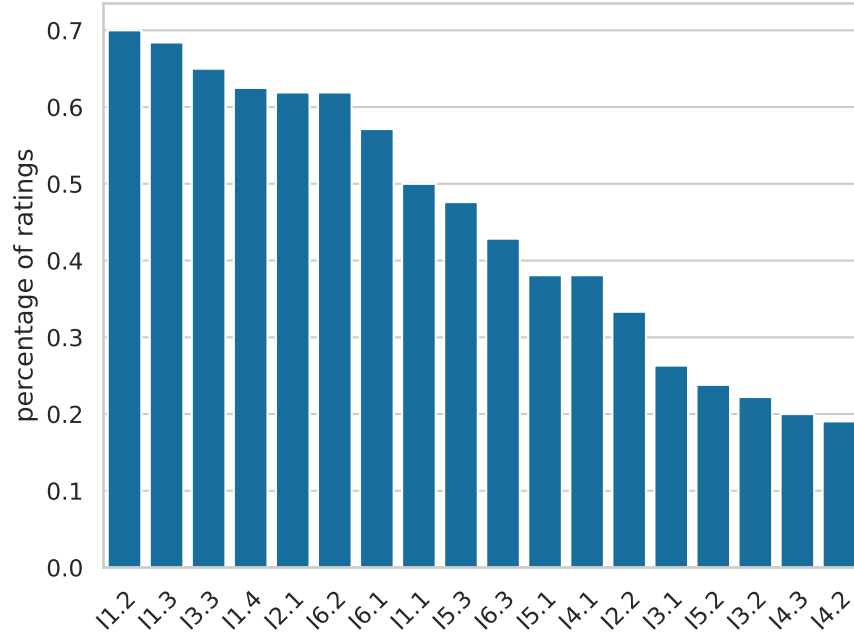


Figure 4.9 Percentage of answers rating an issue to be frequent

Evaluability

As explained in Section 4.4.2, evaluating the quality of models offline on a test dataset with an ML performance metric (e.g., accuracy) often gives poor measures of quality [I1.1]. To mitigate this problem, practitioners try to evaluate their models as frequently as possible with better proxies of quality. If the model can be evaluated online (i.e., deployed in a production environment), then the model is evaluated in a production environment using a business metric [P1, P7, P9]. For example, for a trading system, one could measure how profitable the new system is compared to the previous one. If deploying the model in a production environment is not possible, then the model can be evaluated offline using the judgment of domain experts, users of the MLSS, or business stakeholders [P3, P13, P23]. We refer to anyone with significant experience on a problem as domain experts. On the topic of model evaluation with domain experts, [P3] says: *“It’s really important to have [...] someone who can challenge [your] results and who [...] knows the business. It’s important to involve them as early as possible”*.

As mentioned in Section 4.4.2, the goal that is targeted when an ML metric such as accuracy is used is different from what is important for the end application, partially causing issue [I1.1]. Hence, [P9] suggested integrating the business perspective into the loss function of a model, so the model is optimized for what is important for the end application. For example,

for a trading system, the model could be directly optimized for profit instead of accuracy. However, the practitioner only suggested that as an idea and did not implement it.

Explainability

As mentioned in Section 4.4.2, one of the goals of explainability is to increase the users' trust in an MLSS's predictions by providing explanations of a model's predictions [260]. However, we found two issues hindering transparency, namely issues [I2.1, I2.2]. To address this problem, [P23, P32] developed custom strategies to justify the model's predictions. For example, for a house price prediction system, [P32] showed the user the price of similar houses for sale on the market. As a result, the user was able to compare the prediction of the model with similar data and decide whether the prediction could be trusted or not. More generally, [P23] recommends presenting historical performance on similar cases to the user. For example, one could show the user how many times the model was accurate on a similar example. As a concluding thought on replacing explainability tools with custom strategies, [P23] said: *"[The custom strategies] will probably solve the transparency problem [...] even though the model won't be [able to] explain. So you're really making the system usable more than you are solving the models' explainability problem"*.

Debuggability

In the following, we present possible remedies practitioners shared with us to address the issue [3.1]. As expressed in Section 4.4.2, some practitioners have issues debugging their MLSSs, because of unstable data sources that return different values for the same queried record. This makes reproducing a bug very hard and sometimes impossible. To address that problem, [P16] temporarily saves a copy of a record in a database whenever it has been used by a model (for training or inference). Similarly, [P25] mentioned trying to use data version control systems to keep a history of the data used to train her models. However, the person abandoned the tool after a few weeks, because using the tool required too much effort, and the management team did not see any value in using it.

Efficiency

In Section 4.4.2, we explained the quality issues [I4.1, I4.2, I4.3]. To reduce the time taken to train a model and partially address issue [I4.1], [P8] used Spark's ML library to train models across different machines at the same time. When facing memory-constrained environments, [P35] used ML architectures with a reduced number of parameters, such as MobileNet [264],

to address issue [I4.3]. Practitioners have reported different strategies to address latency issues with MLSSs [I4.2]. [P16, P36]’s teams typically try using a model with a reduced number of parameters, so the inference is faster. For example, [P16]’s team focuses on tree-based models instead of deep neural networks. To avoid latency issues caused by heavy traffic, [P36]’s team uses frameworks such as PySpark to distribute computing across different machines. [P36, P38] mentioned addressing latency issues by optimizing the data processing pipelines. For example, loops can be replaced by vectorized operations to make the code faster [P38]. Finally, [P16]’s team decommissions features that are not important for prediction to reduce the load on the system and make it more efficient.

Maintainability

In Section 4.4.2, we explained the quality issues [I5.1, I5.2, I5.3]. As expressed in issue [I5.1], maintenance is challenging when no proper dataset documentation is available. As a consequence of this problem, [P1, P2, P19, P29] had to communicate with domain experts who can help them understand the dataset at hand, which can be time-consuming, as explained in Section 4.4.2. To avoid that issue altogether, practitioners shared strategies used by their team to keep data well documented. [P38]’s team followed a methodology for project management in Data Science, namely TDSP²². Data understanding and documentation play a central role in the methodology, ensuring their data is sufficiently documented when developing (and maintaining) MLSSs. [P8] built a data storage procedure that forced her team members, when entering a new dataset or data source in a database, to document data to the best of their abilities. Practitioners were asked, for example, to provide the estimated level of data quality of a data source. However, the practitioner’s team ended up giving up on that solution because it was too much of a burden.

As expressed in issue [I5.2], model maintenance is challenging when no proper model documentation is available. [P25] uses experiment tracking tools, namely MLflow²³, to document the hyperparameters used to train models. [P20] considers using ML platforms with embedded model registries, namely Vertex AI²⁴, to be aware of the models already available inside her company. As a result, her team would avoid re-implementing models that already exist for a problem.

As expressed in issue [I5.3], managing the dependencies of an MLSS is challenging and error-prone. To avoid issues with dependencies, [P10]’s team shares the same development

²²<https://learn.microsoft.com/en-us/azure/architecture/data-science-process/overview>

²³<https://mlflow.org/>

²⁴<https://cloud.google.com/vertex-ai>

environment. As a result, less effort is spent on having a functional environment. This environment is used in production as well, ensuring the developed solution is functional upon deployment. Similarly, [P38]’s team avoids configuring and managing environments by using the ones provided and managed by their cloud provider.

Reliability

In Section 4.4.2, we explained the quality issues [I6.1, I6.2, I6.3]. To address concept and data drift (issue [I6.1]), our interviewees mentioned monitoring their models or data sources (e.g. [P1, P13, P15, P20, P21, P24, P36]). However, monitoring and model retraining is not always sufficient to ensure the quality of the model’s predictions. Hence, when possible, practitioners shared solutions to minimize data and concept drift. As we explain in Section 4.4.7, data and concept drift can be caused by a change in the data collection process. When data collection is manual, data and concept drift can be attenuated by standardizing the data collection process and providing clear guidelines to the people collecting data [P16, P37].

Practitioners shared different strategies to handle unstable external dependencies (issue [I6.2]). A rather straightforward one is to simply avoid using an external data source if it is an unreliable one [P4, P10, P11, P15]. Before adding a data source to an MLSS, [P11] assessed the reliability of the data source. As she mentioned in the following: *“So when we start a project, we [...] include new datasets [only] when we are really sure that they’re gonna come from a steady source. [...] You have to make sure that you’re able to feed this model [...] in a consistent manner. So we just try to be very careful with the data we include in the model. This is the main solution we have”*. When practitioners could not avoid using a data source, they adopted different strategies to mitigate the risks with reliability. [P7, P10] fetched the same information from different data sources to have data redundancy. Hence, they are able to verify the correctness of the values returned by the data sources (by comparing them) and to ensure data availability at inference time. [P7, P8, P15, P20] mentioned monitoring their data sources, doing data validation, and sending alerts in case of issues with the data sources. As a result, engineers are able to quickly detect and address a problem with data sources. However, this does not guarantee the end user will not be impacted by unreliable data sources. Hence, when a problem with a data source happens and is detected, practitioners have different strategies to attenuate the consequences. In one of her projects, [P10] informed the user whenever a model’s prediction risked being worse than usual because of an issue with a data source. Finally, when the latest training data of a model had poor quality, [P21]’s team rolled back to the previous version of a model until the issue with the data source was resolved.

4.4.5 RQ4: Which data quality aspect is the most problematic in practice?

As mentioned in Section 4.3.7, we used the Dataset Definition Standard (DDS) [1] to group the data issues. A data issue is included in a data quality aspect as defined in DDS if it hinders that aspect. For example, mentions of mislabeled data are included in the “Labels accuracy” aspect. The issues that could not be included in any of DDS’s data quality aspects were categorized by a researcher into one of the Data Management community’s (DAMA) [2] twelve most common data quality dimensions. We detail in the annex (Section B) the data issues included in each data quality aspect experienced by practitioners. We also define each of the data quality aspects or dimensions in the annex.

Figure 4.10 displays the number of practitioners having faced an issue with one of the quality aspects defined by DDS [1] or DAMA [2]. As we can observe from the figure, having accurate and complete data is challenging for practitioners. Indeed, “Data completeness”, “Data accuracy and precision”, and “Labels accuracy” are respectively the first, second, and fourth quality aspects with the highest count of issues. Data representativeness comes third in the figure, because of the large number of practitioners struggling with data and concept drift. The least affected quality aspects are “Data integrity” and “Data traceability” with only 1 practitioner having struggled with these aspects. Three of DDS’s data quality aspects are not displayed in this figure, because we have not found any issue with them in our interviews. These are: “Unintended bias in data”, “Absence of Annotation bias”, and “Independence between datasets”. The data quality dimensions from DAMA we used to complement DDS’s data quality aspects are “Data completeness”, “Data clarity”, “Data uniqueness”, and “Data validity”, and we counted respectively 16, 9, 8, and 6 practitioners having an issue with the respective data quality dimension.

Key findings: Practitioners are often faced with datasets with missing values. Additionally, they struggle to have precise and accurate features or labels. Having a dataset that is not representative of the modeled process is a recurrent issue.

4.4.6 RQ5: In the case of data, which data types and collection processes are the most challenging in terms of data quality?

We answer this RQ in two parts. In Section 4.4.6, we show, for each data type, the data quality issues our practitioners experienced. We do the same for data collection processes in Section 4.4.6.

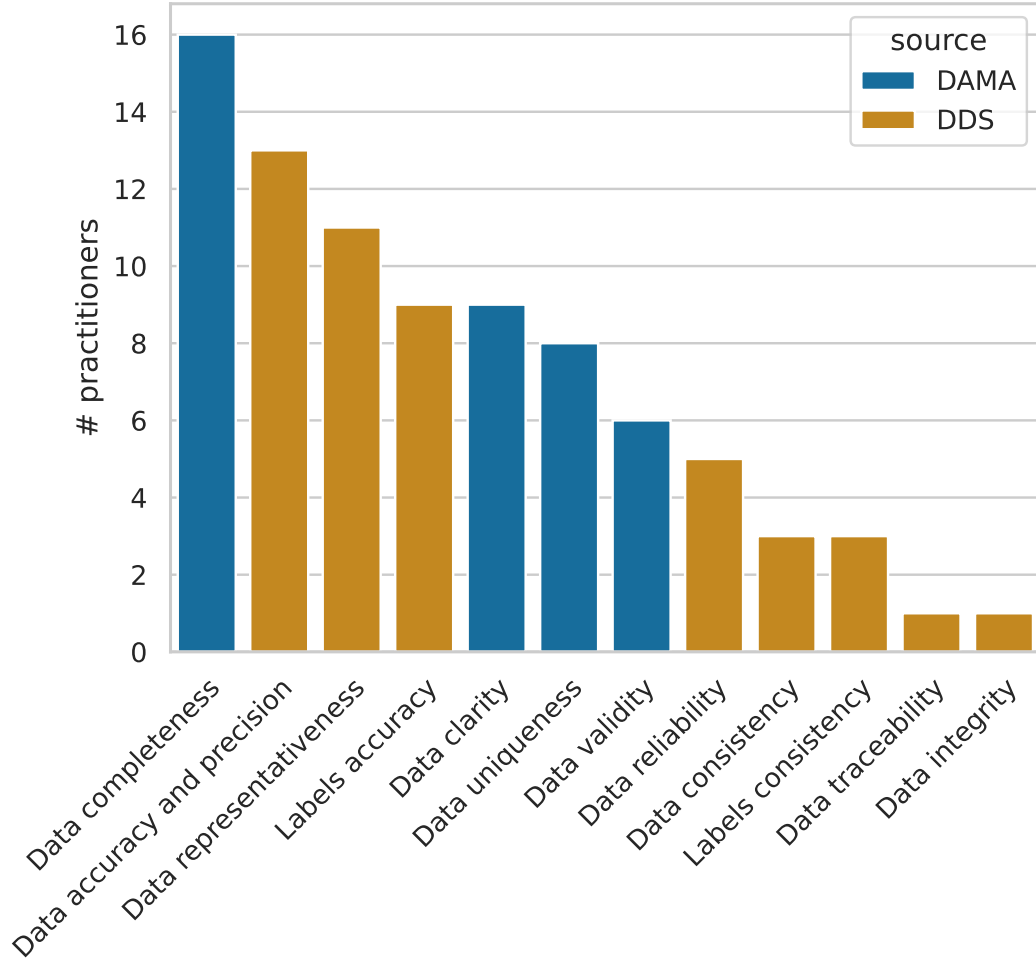


Figure 4.10 Number of practitioners having an issue affecting one of the quality dimensions defined by DDS [1] or DAMA [2]

Data types

Nine practitioners mentioned having issues with tabular data and six with image data. There was no issue we could confidently assume to belong to other data types. For tabular data, the problematic data quality aspects were: data accuracy and precision [P7, P25, P32], labels accuracy [P32], data integrity [P29], data inconsistency [P21], and data reliability [P13, P15, P34, P35]. For image data, the problematic data quality aspects were: label consistency [P21, P35] and label accuracy [P25, P32, P33, P40].

Data collection processes

In our interview, we observed shared issues between the data collection processes depending on whether they were automated or not. Hence, we present manual data collection and automated data collection separately. With manual data collection ensued data quality issues in terms of data accuracy and precision [P7, P31, P38], label accuracy [P14, P40], and label consistency [P21, P37, P43]. For automated data collection processes, we only found issues with data accuracy and precision [P7, P8, P20]. Additionally, automated data collection processes occasionally faced reliability issues [P4, P7, P10, P15, P21, P40, P43]. In the following, we detail the issues with manual and automated data collection.

Manual data collection The practitioners have mentioned that manual data collection can take a lot of time [P7, P8, P32, P38, P43]. As a result, building a dataset for ML and making sure it stays relevant (when there is data and concept drift) can be difficult. We have observed three issues with manual data collection that leads to poor data quality. First, humans occasionally make mistakes. For example, a person collecting data could misspell a word [P7, P31], inadvertently paste a wrong value [P38], or misinterpret an image and label it incorrectly [P14, P40]. As a result, practitioners may work with data that make no sense [P10] and spend a lot of time cleaning the dataset. Second, humans interpret events differently, which leads them to annotate data differently [P21, P37, P43]. As illustrated by [P37], the same temperature can be hot for someone, but mild for someone else. Third, humans write down data differently, which can lead to some inconsistency in the data. For example, data collectors may use different conventions to represent time or use commas instead of periods to separate decimals from integer numbers [P38]. Additionally, synonyms can also be used to identify the same object [P25, P31].

Automated data collection Compared to manual data collection, automated data collection has the advantage of being systematic by default. Hence, most of the issues that have been observed with manual data collection were not observed in automated data collection processes. As a result, data collection processes tend to be automatized and run periodically when possible. However, these processes may have reliability issues. Notably, practitioners have mentioned issues with web-scraping for which the slightest change to the web page might break the process [P4, P10, P15, P43]. As pointed out by [P15], maintaining web-scraping processes is time-consuming. Sensors used for data collection may experience technical issues and be unable to send accurate data [P7, P8, P20]. Batch processing scripts may also fail, as practitioners [P7, P21, P40] experienced. In [P21]’s case, it took several

days to realize there was an issue since the process was not monitored. As a result, the model got trained on old data only.

Key findings: Manual data collection can lead to data quality issues if the process to collect data is not systematic and properly defined. On the other hand, automatic data collection processes usually have reliability issues.

4.4.7 RQ6: What are the challenges of data quality assurance during model evolution?

As it is widely known in the ML community, ML models are highly sensitive to the quality of data. As a result, during the maintenance of an MLSS, a significant amount of effort must be spent on data sources in order to ensure the system’s reliability. In our study, we observed three data issues practitioners have to take into consideration during the maintenance of MLSSs. First, the problem that is modeled may change over time, which will render the deployed model useless. Second, the data collection process may change over time, which, similarly, causes a drift in the distribution of data and may render the model useless. We refer to that problem as annotation drift. Third, external data providers may not have the standards of reliability that are required to build MLSSs. As a result, practitioners must put extra care when dealing with external providers in order to ensure the reliability of their MLSSs.

Data and concept drift

Data drift refers to a change in the distribution of records over time. When modeling the function F that maps X (i.e., records) to Y (i.e., labels), data drift is a change in $P(X)$ over time. Concept drift refers to a change in the function F over time [265]. Concept and data drift are problems that must be considered by practitioners, or else the deployed model will lose predictive capabilities and become stale. According to our interviews, practitioners are already well aware of these problems and automatically retrain their models to address the issue. However, we observed two scenarios where retraining was not sufficient to attain past performances. First, sudden changes in the data, such as pandemics like COVID-19, render the past data useless for the new prediction problem. As a result, obtaining the same level of performance as with the previous models may be almost impossible, since not enough data is collected for the new problem [P9, P20]. Speaking of her experience with COVID-19, [P20] said: “*We had this weird time where we know the data has changed, but we don’t have enough data to train new models because it hasn’t been that long. We kind of just have to [wait] until*

enough time goes by and we can start retraining these models". Second, performance issues caused by concept drift can not always be solved by retraining the model [P15]. Concept drift can render features useless. If a significant number of features are affected, then predicting the target variable will be a much more difficult task for the ML model. Hence, periodically retraining a new model and deploying it will not be sufficient to preserve ML performance.

Key findings: While data and concept drift are well-known by practitioners, they are still considered hard problems. Automatically retraining models is not a panacea to these issues. Performance drops because of data or concept drift are sometimes inevitable due to insufficient data for retraining.

Annotation drift

While it is clear that data and concept drift may be caused by a change in the modeled problem (e.g., what happened for many MLSSs when COVID-19 began), it is maybe less obvious that data collection may generate the same problem. For simplicity, we refer to drift in the annotation process as annotation drift. As mentioned in Section 4.4.6, manual data collection is not always a systematic process, and, as a result, is error-prone. Humans interpret events differently and may write down information differently. Hence, annotation drift can be observed if the process for collecting data is not described exhaustively. Annotation drift naturally happens as people change, or as team members are replaced [P36]. Drastic changes in the data distribution may happen, if, for example, the annotation procedure is changed [P24, P25, P36]. In this case, one should define a procedure for transforming the data collected with the old standard into the new standard. If it is not done, a lot of data will be lost, which might hinder the training of ML models, as experienced by [P25].

Key findings: Data and concept drift sometimes creep from unexpected places, such as the data collection process. Annotation drift happens when no systematic data collection process is defined.

Unreliable external data providers

Practitioners revealed that external data sources are often the cause of many issues during maintenance. We consider any data source to be external if it is not managed by the team building the model. Because data providers are not aware that their consumers are ML models, maintainers of MLSSs face a lot of reliability issues. On the topic of the reliability of third-party APIs, [P23] said: “[...] *it’s a nightmare. [...] their job is not to provide ML-ready*

data. So the data is not formatted, [and require] quite significant post-processing". She also added: *"They often sell their data to people who don't require high uptime [...] which means that sometimes it'll break. And most of their clients don't care. But if you're trying to build an AI system, you do care quite a lot".* Overall, we observed two problems with external data providers. First, practitioners may have unreliable access to data. Second, data providers may return inconsistent information. That is, the same information accessed at different times might have different values.

We observed two causes of accessibility issues with data providers. First, they may be unable to deliver data at times because of technical issues [P10, P11, P21, P23]. For example, their API may be down. Second, they may unexpectedly change the definition of their interface [P15, P27]. As a result, the affected MLSSs are not able to fetch data until the code used to communicate with the API is adapted. Not being able to access data is a problem at any time in the model life cycle. At inference time, the model will not be able to make a prediction unless replacements for the missing values are found. The replacements are not guaranteed to be accurate; thus hindering the quality of predictions of the model. At training time, important information may be lost if records with missing values are thrown out of a dataset.

Aside from occasional accessibility issues, data sources may serve inconsistent information. More precisely, the same information fetched at different times may have different values. This is a problem for an ML model since it means it has learned from data that is not representative of the new data distribution. A data provider may return different values for the same records for various reasons. In our interviews, we have seen two. First, the data providers may choose to change the units they have used to measure some information [P4, P6, P43]. For example, dates in British format could suddenly be changed to Unix data format [P43]. Second, the data providers may voluntarily alter historical data [P4, P6, P10, P12]. This may happen if the provider believes that the data they previously served was inaccurate. For example, practitioner [P4] consumed data from a weather API. One day and without notice, the data providers corrected old weather recordings that were faulty because of a broken sensor.

Key findings: Ensuring the reliability of an MLSS when data is collected from external data providers is challenging for practitioners. External data providers may be unavailable at times, or serve inconsistent information.

4.5 Recommendations

In this section, we provide a set of recommendations for practitioners to help them tackle the quality issues mentioned in Section 4.4.2 and the data quality issues covered in Section 4.4.5. We do not repeat the ones that have already been mentioned in Section 4.4.4. Additionally, based on our results, we provide future work directions for researchers.

4.5.1 Data quality

As observed in Section 4.4.5, the data quality issues that were the most mentioned in our interviews were incorrect feature values and labels. Fixing these kinds of errors often consumes a lot of time [P8, P3] and is often done manually [P9]. According to our interviews, there is not a wide variety of tools that are used by practitioners to clean data. [P6] used Data Wrangler [266] to fasten the data cleaning process. However, that tool is limited to one cloud provider; thus limiting its usage. Hence, as future work, we recommend developing tools to facilitate the process of detecting and repairing errors in data. Similarly, we recommend developing advanced tools for data cleaning of duplicates and missing values, since a large number of practitioners had issues with it (see Section 4.4.5 for further details).

As mentioned in Section 4.4.5, we used DDS [1] to define data quality aspects. Because the DDS is not exhaustive, some quality issues could not be categorized in any quality aspect provided in DDS. Thus, we completed DDS’s definition of data quality with the twelve most common quality dimensions defined by DAMA [2]. To answer our RQs, using the later work would not have been preferable either. Some quality aspects defined in the DDS were not covered by DAMA, because the context of the latter was not ML. For example, “absence of annotation bias” is considered in DDS but not in DAMA. Hence, because either work could not capture every data issue we observed, we recommend future works to develop a taxonomy of data quality for ML applications.

4.5.2 Evaluability

As discussed in Section 4.4.2, one of the reasons why offline evaluation of models is inaccurate is because the goal targeted by accuracy is different from what is important for the user. We provide in Section 4.4.4 a set of alternatives to evaluate models to have more accurate measures of quality. As observed by [P7] in a follow-up conversation after the interview, junior practitioners rely too much on accuracy to evaluate their model and end up deploying models that perform poorly in production. We believe that a root cause for that problem is the ubiquity of these metrics in the curriculum of students. We encourage educational

programs to include techniques to evaluate a model in the context of building MLSSs. Future works could look into building loss functions that include the business perspective, as [P9] suggested. For example, for a financial trading system, the model could be rewarded based on how profitable a decision was. Training a model using these loss functions would create models that are already optimized for the end application. Alternatively, future works could look into techniques to post-process ML models to optimize the business need.

As explained in Section 4.4.2, defining a good business metric to evaluate an MLSS can be difficult for some applications, such as recommender systems (issue [1.2]). In these scenarios, practitioners may discuss with business stakeholders to conjointly develop business metrics that accurately measure how useful the model is for the application.

In Section 4.4.2, we described the problem of dataset evaluability (issue [I1.4]). Evaluating the quality of a dataset (or knowing its potential issues) is difficult. Hence, we propose two approaches for practitioners to get an estimation of data quality faster. First, data visualization tools, such as Facets²⁵, can be used to get a better understanding of the dataset at hand and maybe understand quality issues easier. The data visualization tools can be used conjointly with tools to automatically generate interesting visualizations such as SeeDB [267]. Second, error detection tools on datasets can be used to get an approximate count of the number of errors in a dataset. We recommend practitioners to explore unsupervised error detection tools, since they do not require a significant time investment before they can be applied on a dataset. Notably, [268] showed that pre-trained transformer models could detect errors in tabular data without any training examples. The models achieve even better performances when given training examples. With only 10 labeled samples, their approach surpassed state-of-the-art error detection approaches. [269] analyzed the reconstruction error of an auto-encoder model to detect dirty samples. Samples that the auto-encoder struggles to reconstruct are marked as being dirty.

One of the reasons to evaluate a dataset's quality is to estimate the feasibility of developing an ML solution using the dataset at hand. When asked about her biggest pain points, [P3] answered the following: *"the uncertainty of [knowing whether the ML solution will] work or not; maybe speeding up the process? [...] getting to a point where [I] can assess the quality of an approach faster, that would help"*. We propose two approaches for practitioners to get that information without measuring data quality. First, AutoML techniques [270] can be used to get the ML performance of a model on the dataset with minimal effort. Second, Bayes error rate estimators, such as the work of [271], can be used to directly estimate the feasibility of reaching a certain performance level on the dataset at hand.

²⁵<https://pair-code.github.io/facets/>

4.5.3 Explainability

Issue [I2.2] refers to the unreliability of explanation techniques. As expressed by [272], explainability techniques only are approximations of the underlying decision process of a model. Hence, they are bound to be wrong and they can not be relied on. [261] shows different situations where Shapley value based feature attribution methods produce faulty explanations. Similarly, [273] shows how these feature attribution methods can be manipulated with adversarial attacks to reduce the importance of targeted features. Common feature attribution techniques, such as the work done by [127], use correlation to explain a model’s prediction. They do not capture the causal influence of an input on a model’s prediction. Therefore, an interesting research direction is to develop causal explanation techniques. As pointed out by [260], there are very few numbers of studies conducted in that direction [274].

4.5.4 Debuggability

Issue [I3.1] refers to the challenge of debugging MLSS with unstable data sources. Most specifically, debugging an MLSS is difficult when two queries to an external data source for the same information return different values (see Section 4.4.2 for more details). In the following, we refer to this problem as inconsistent information. In Section 4.4.4, we shared the solution of a practitioner to address this problem. She temporarily saved any data ingested by her models, so it could be fetched later on for debugging. However, this solution has its shortcomings. First, this approach requires saving a large volume of data. Second, the older bugs can not be reproduced, since the data is eventually deleted. Hence, we highlight the usefulness of recommendations given for reliability issues in Section 4.4.4 to address the debuggability issue [I3.1]. By applying these recommendations, the data sources become more reliable, hence avoiding the issue [I3.1].

Issue [I3.2] refers to the challenging aspect of debugging data streaming systems: picturing what data should look like after a series of operations is difficult. As we explained in Section 4.4.2, one might compare data under test against the mean or the mode to detect abnormalities. However, according to our interviewees, this is not easily doable, as [P17] said the following: *“So if you want to inspect the middle steps [of a data streaming pipeline], you have to [do the] effort [of] sending all the data to another table. [...] After this [job] is done, then [you can have] a look at the table [and compare your values]”*. To avoid this long process when debugging data streaming pipelines, [275] proposes Dagger, a data debugger that can automatically provide the value of any tracked variable across any step of a data pipeline. Compared to traditional code debuggers, the tool proposes features to make data debugging easier, such as a “compare” command that enables comparing the value of a variable at a

specific time across different runs of the pipeline (i.e., for different data records). To efficiently save the value of a record throughout a data pipeline, only the changes to a value are saved. We recommend future works to develop data debuggers that can: (1) efficiently save the value of a record throughout a pipeline, (2) provide features to easily compare data values across data pipeline runs (e.g., getting the mean), and (3) easily append the debugger to existing data streaming system.

Issue [I3.3] refers to the challenging aspect of debugging MLSSs when its data sources are managed by external data providers. As explained in Section 4.4.2, this can easily become cumbersome because of long back-and-forth communications with the maintainers of the data source. Furthermore, the maintainers might not prioritize the issue because they have their own agenda. In the following, we provide a set of recommendations to practitioners to address that issue. When a data source is internal to a company, the company can raise awareness among data producers on the important role they play in ensuring the quality of ML models. For example, a company could organize showcase meetings, where ML teams present their work to data producers. As a result, data source maintainers would be more inclined to help ML model maintainers when they require assistance. A similar recommendation was made in [67]. Another way to tackle the problem is to establish clear bug prioritization measures that prioritize bugs impacting the ML model, so that the data source maintainers would be incentivized to help fix the issue. Finally, when possible, similar to full-stack developers for web applications, we recommend companies to create full-stack developers roles for ML pipelines. These practitioners could have technical knowledge of the model as well as its data sources. As a result, they would be able to fix issues with data sources without relying on the assistance of the maintainers of a data source. The above recommendations can not be applied when the data source is not managed by the company running the MLSS. Hence, before adding a data source to an MLSS, we recommend practitioners to verify the quality of the customer support that the external data providers will provide in case of an issue. This assessment can be done conjointly with the one discussed in Section 4.4.4 (i.e., assessing the reliability of a data source before using it).

4.5.5 Efficiency

In addition to the recommendations given in Section 4.4.4, we suggest using quantization methods [263] to the practitioners that use neural networks in order to reduce the models' size in memory and effectively address issue [I4.3].

4.5.6 Reliability

As discussed in Section 4.4.2, ensuring the reliability of an MLSS is difficult when its data sources are unreliable. We already provided a set of recommendations to address the problem in Section 4.4.4. Similar to the recommendation given in 4.5.4, we recommend companies to raise awareness among data producers on the important role they play in ensuring the quality of ML models. Good practices such as API versioning should be encouraged.

4.6 Threats to validity

Using some of the validity threats described in [276], we divide the analysis of the limitations of our study into five categories: internal validity, external validity, content validity, confirmability validity, construct validity, and conclusion limitations.

We identified two threats to internal validity. First, as mentioned in Section 4.3.5, an interview was conducted in French, and the others, in English. It is possible that the questions asked when the interviews are in English are not exactly the same as the ones asked when the interviews are in French. While our primary interviewers have a good understanding of both languages, their choice of words might convey slightly different meanings, leading to bias. To mitigate that threat, the second interviewer was welcome to rephrase the question of the first interviewer if it was poorly translated. Second, there is a risk that our findings for **RQ1** are inaccurate because of our data collection method. Asking practitioners about the root cause of quality issues may lead to misunderstandings or even misinformation in case they do not want to admit their mistakes. To mitigate this issue, we cross our findings with MoovAI knowledge on similar cases.

We identified three threats to external validity. First, as discussed in Section 4.4.1, a significant portion (40%) of our interviewees are practitioners of MoovAI. Thus, it is possible our findings are applicable only to small companies (11 to 50 employees) whose main service is developing MLSSs for other companies (i.e., consulting). In an attempt to mitigate this bias, we interviewed a larger number of practitioners from varying company sizes. Second, there is a chance that the practitioners who participated in our study experienced quality issues in MLSSs more than the average practitioner, leading us to believe that some issues are more important than they truly are in practice. In other words, the practitioners who participated in our study could be more interested to participate than the average practitioner, because they experienced more issues than the average practitioner. To mitigate that issue, we validated our results using a questionnaire. Because the questionnaire requires less time

to answer than participating in our interview, we expect that the people who answered our questionnaire are more likely to be representative of the general population of practitioners. Third, a limitation of our approach could come from keywords (tags) we use for finding candidates over Q&A websites and social media. For keywords, we use *data-cleaning*, *dataset*, *machine-learning* and *artificial-intelligence* which are sufficiently general to match a lot of users and are more likely to result in False Positive (users that we would end up ignoring) rather than False Negative.

We identified a threat to content validity. In our case, there is a risk that our interview guide does not enable us to find all the quality issues. The interview guide might not be comprehensive enough to cover every quality issue. If no practitioner mentioned an issue that was not probed in the interview guide, then we might have missed an issue. To mitigate that threat, we ask open questions so that the interviewee has many opportunities to share issues we did not suspect.

We have identified two threats to confirmability validity. First, the coding of the interview may be prone to researcher bias because it partially relies on subjective interpretation. To mitigate this bias, each interview is coded by two researchers, and inconsistencies in the codes are resolved by a third researcher during moderation. Second, there is a risk that some quality issues are over-represented in our interview guide. We are conducting semi-structured interviews; some conversations with the interviewees are improvised. Thus, it is possible that the findings are tainted by the preconceptions of the interviewer about the potential quality issues in MLSSs. For example, there could be more emphasis on explainability issues in the interview, which may lead to an over-representation of these issues in our findings. To mitigate that threat, we used [76]’s work to ask questions covering every step of an ML workflow.

We have identified a threat to content validity. When validating our results with the questionnaire, it is possible that the practitioners misinterpret the description we provide for each quality issue. To mitigate this, the descriptions are reviewed by two researchers. Furthermore, we provide examples when the description is ambiguous.

We have identified a threat to conclusion validity. There is a risk that researchers misinterpret an issue described by a practitioner. To avoid that issue, each transcript is coded by two researchers. If an important excerpt from a transcript required further details to be understood, we contact the respective practitioner for clarification.

4.7 Conclusion

In this paper, we investigate the characteristics of real-world quality issues in MLSSs from the viewpoint of practitioners. We presented the results of the interviews we conducted with 42 practitioners and the questionnaire answered by 21 practitioners.

We extracted a list of 18 quality issues and, for every issue, we explained the causes and consequences. We measured how common the quality issues are according to the practitioners' experience. We found that the quality dimensions that are considered to be the most problematic are: evaluability (see Section 4.4.2 for a definition), debuggability, and reliability. We share 21 strategies used by the practitioners we interviewed to mitigate the aforementioned quality issues. After consulting the literature, we proposed 12 other strategies. Regarding data quality, we describe the data quality issues encountered by our practitioners and how frequent they are according to their experience. Notably, we found that data completeness (missing values) and data and label accuracy are the most common data quality issues. We also explain the challenges encountered with different data collection processes that lead to data quality issues and we characterize the challenges of data quality assurance during model evolution. Finally, we propose some recommendations for future works.

Our study shows that (1) using ML in software systems brings a unique set of challenges, and (2) practitioners struggle to build qualified MLSSs. Based on our results, we encourage researchers to develop tools to assist practitioners when building MLSSs. We hope that our findings will serve as a solid foundation for future work to come.

Compliance with Ethical Standards

The authors declare that they have no conflict of interest.

Data availability

The datasets generated during and/or analyzed during the current study are available in the GitHub repository, https://github.com/poclecoqq/quality_issues_in_MLSSs

Acknowledgments This work is partly funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), PROMPT, and Les Technologies MoovAI Inc.

CHAPTER 5 ARTICLE 2: DATA CLEANING AND MACHINE LEARNING: A SYSTEMATIC LITERATURE REVIEW

Preface

Full Citation: P.-O. Côté¹, A. Nikanjam¹, N. Ahmed¹, D. Humeniuk¹, and F. Khomh¹, "Data Cleaning and Machine Learning: A Systematic Literature Review," *Automated Software Engineering*, [in review] Oct. 3 2023.

Permission: Reproduced with permission from Springer Nature.

Abstract Context: Machine Learning (ML) is integrated into a growing number of systems for various applications. Because the performance of an ML model is highly dependent on the quality of the data it has been trained on, there is a growing interest in approaches to detect and repair data errors (i.e., data cleaning). Researchers are also exploring how ML can be used for data cleaning; hence creating a dual relationship between ML and data cleaning. To the best of our knowledge, there is no study that comprehensively reviews this relationship.

Objective: This paper's objectives are twofold. First, it aims to summarize the latest approaches for data cleaning for ML and ML for data cleaning. Second, it provides future work recommendations. **Method:** We conduct a systematic literature review of the papers published between 2016 and 2022 inclusively. We identify different types of data cleaning activities with and for ML: feature cleaning, label cleaning, entity matching, outlier detection, imputation, and holistic data cleaning. **Results:** We summarize the content of 101 papers covering various data cleaning activities and provide 24 future work recommendations. Our review highlights many promising data cleaning techniques that can be further extended. **Conclusion:** We believe that our review of the literature will help the community develop better approaches to clean data.

Keywords Machine Learning, Data Cleaning, Systematic Literature Review, Survey, Taxonomy

This work is funded by the Fonds de Recherche du Québec (FRQ), the Canadian Institute for Advanced Research (CIFAR), and the National Science and Engineering Research Council of Canada (NSERC).

¹Polytechnique Montréal, Québec, Canada

Email: {pierre-olivier.cote, amin.nikanjam, nafisa.abdelmutalab-ali-ahmed, dmytro.humeniuk, foutse.khomh}@polymtl.ca

5.1 Introduction

Nowadays, Machine Learning (ML) integrates a growing number of industries, from transportation to healthcare and education. Recent applications of ML achieved performances similar to humans’ performance on complex tasks, such as taking the bar exam [277] or driving a car [278, 279]. Behind much of the recent success of these applications are large amounts of training data and powerful computing infrastructure [250, 280]. While a major part of the research in ML is spent on developing better modeling techniques [281], data preparation often is the most arduous and time-consuming task for practitioners. Indeed, researchers have reported that data scientists sometimes spend over 80% of their time preparing data [250, 282–284]. As a consequence, there is an emerging trend, referred to as Data-Centric AI (DCAI) to steer research to focus on improving datasets instead of models for ML problems. In the last few years, various initiatives aimed at stimulating involvement in the DCAI trend have been proposed. For example, in August 2021 the first DCAI competition [285] was organized. During this competition, participants were challenged to improve a model’s performance by only modifying the dataset it was trained on. Following this competition, a group of researchers released the DataPerf benchmark [286], a suite of data-centric benchmarks for different data tasks such as data cleaning and data debugging.

Amongst the different steps forming data preparation is a central task that consists of detecting and removing data errors, namely data cleaning [287]. Previous studies have often overlooked it assuming that the datasets used in their experiments are devoid of errors. However, as reported by a previous study [288], large public datasets considered for a long time to be error-free such as ImageNet [289] contain erroneous labels. As a consequence, there is an increasing interest in developing data-cleaning approaches to improve machine learning performance. Data cleaning is a problem tackled by the database community for a long time [30–32] and, recently, researchers started studying how ML can be leveraged to clean data.

In this study, we summarize the latest approaches in Data Cleaning for ML (DC4ML) and ML for Data Cleaning (ML4DC). For simplicity, we refer to the union of DC4ML and ML4DC as data cleaning and ML (DC&ML). To reach that goal, we adopt the Systematic Literature Review (SLR) approach, which differs from traditional literature reviews by its strict and rigorous methodology. The SLR methodology ensures that the literature review is exhaustive and devoid of biases when collecting and summarizing information [290]. With the increasing amount of work published on DC&ML (see Section 5.4), we believe that there is a need to summarize the latest approaches in the field. A replication package of our study is available on

our public GitHub repository¹. In total, we summarize the content of 101 papers and provide 24 future work directions. Our review can serve as a basis for researchers and practitioners to understand state-of-the-art DC&ML approaches and to contribute to the field.

The rest of the paper is organized as follows. In Section 5.2, the methodology followed is described. Then, in Section 5.3, the taxonomy selected to structure our review of the literature is presented. Next, statistics about the papers included in our study are detailed in Section 5.4. The review of the literature is presented in Section 5.5 and future research directions are proposed in Section 5.6. Finally, we reflect on our work and discuss threats to validity in Section 5.8. The conclusion of the paper is presented in Section 5.9.

5.2 Methodology

In this section, we describe the methodology we followed to conduct our study. We provide an outline of it in Figure 5.1. Our methodology is divided into three phases: (1) paper collection, (2) paper selection, and (3) results analysis. We collected papers on 17 October 2022. For each step in the figure, we indicate (1) the section where this step is explained (in the lower left corner of the box) and (2) the number of papers left after the step (in the lower right corner of the box). To conduct our study, we followed the guidelines defined by [290] for systematic literature reviews. This document describes best practices for conducting systematic reviews in software engineering and is based on existing guidelines from medical research, where empirical research methods are mature and rigorous.

5.2.1 Research Questions

As we mentioned earlier, the goal of this study is to conduct a systematic review of DC&ML. Thus, we define the following research questions.

RQ1: What are the latest data cleaning techniques in DC&ML? As mentioned before, a significant portion of practitioners’ time when building machine learning systems is spent on data. In parallel, ML has made significant progress in the last ten years and is applied to a growing number of applications. Applying ML to data cleaning has the potential of reducing the amount of time spent on data by practitioners, as well as improving the accuracy of data cleaning processes. Conversely, data cleaning can play an important role in improving a model’s performance. By answering this RQ, we provide researchers with an understanding of the current data cleaning approaches in DC&ML.

¹<https://github.com/poclecoqq/SLR-datacleaning>

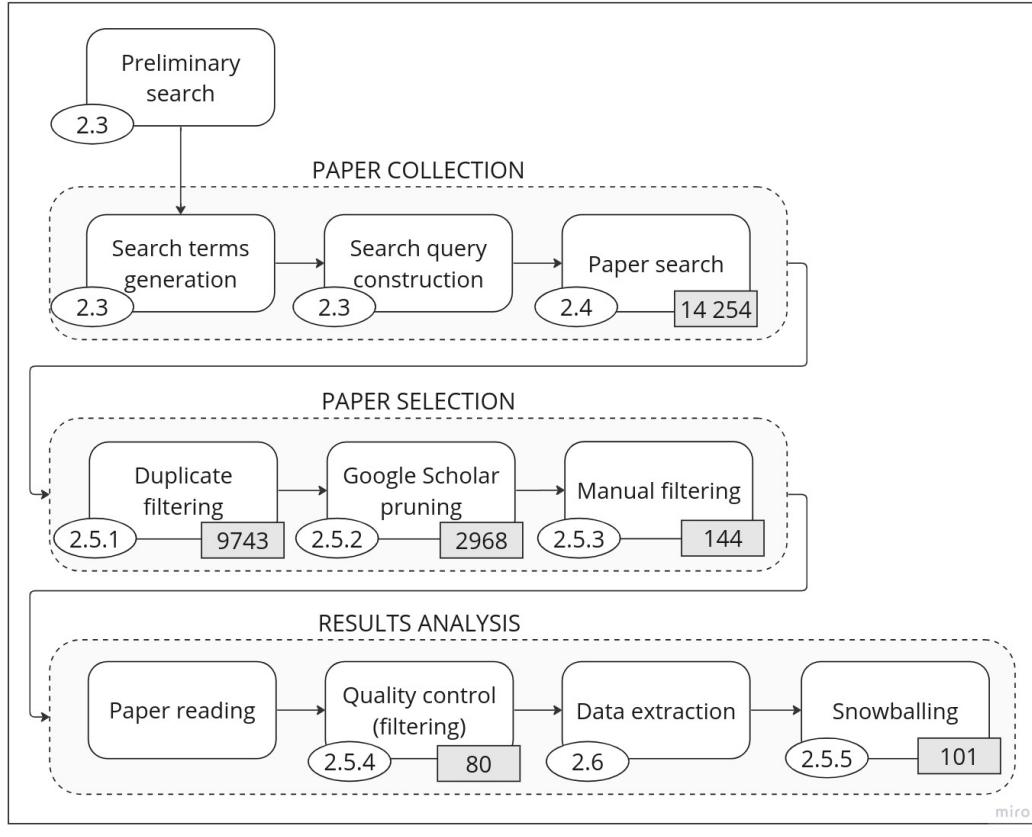


Figure 5.1 Outline of our methodology

RQ2: What are future research opportunities in data cleaning and ML? In this RQ, our objective is to identify and highlight promising research directions in the field of data cleaning with machine learning.

5.2.2 Scope

For a long time, data cleaning was mainly applied to tabular data (i.e., data stored in relational databases), and surveys on data cleaning focused primarily on this type of data [291]. Nowadays, data cleaning techniques for various data types such as images [292], text [293], graph data [294], and sound [295] are developed. The strategy used to clean data in these techniques may significantly vary from one data type to another because of the particularities of each data type. In our survey, we focused on the data types that are the most commonly processed by ML applications in practice. We used comprehensive surveys and books from the ML community to identify them [296–299]. Hence, we limited ourselves to tabular, image, and text data.

5.2.3 Search Terms Selection

As one of the first steps of the SLR, we conducted a preliminary search to help us identify the most relevant keywords to our study, as recommended by [290]. Hence, we read existing survey of the literature [34, 280, 282, 300–303] and highly cited works (e.g., [304, 305]). Then, we defined a comprehensive search query to gather the relevant literature from academic databases. Our final search query is described in Figure 5.2. In the following, we describe how we selected keywords based on the main topics of our review: ‘ML’ and ‘data cleaning’. Then, we explain how we adapted our query for one of the academic databases we used, Google Scholar (GS). Contrary to other databases, GS had peculiarities that forced us to modify our search query.

- *Machine learning*: ML is a field devoted to enabling machines to learn from data [306]. We leveraged previous SLRs [150, 307, 308] on different topics in ML to build an initial list of keywords for ML. Our final list of terms is: *machine learning*, *deep learning*, *neural network*, *reinforcement learning*, *supervised* and *unsupervised*.
- *Data cleaning*: Data cleaning refers to the activity of detecting and repairing errors in data [34]. We used the term *data cleaning* and its synonyms *data cleansing*, *data scrubbing* to conduct our searches. Additionally, we added keywords for both processes of data cleaning: error detection and error repair. For the former one, we used the keywords: *data repairing*, *data repair*, *error repairing*, *error repair*. For the latter one, we only used one keyword: *error detection*. After running the search query with the aforementioned keywords, we initially obtained a lot of irrelevant papers in the results. The reason is that error detection is a very broad topic that is not only limited to the detection of errors in data² Thus, in an attempt to filter out studies that do not propose an approach to detect data errors, we limited the results to papers that are also mentioning a data type. As mentioned in Section 5.2.2, we are focusing on three common data types: tabular, text, and image data. Hence, we used the following terms to limit the papers on error detection: *tab**, *cell**, *row**, *image** and *text**. Finally, because we are interested in data cleaning for ML (and ML datasets have labels), we added the following keywords: *label cleaning* and *confident learning*.

²For example, "A modular edge-/cloud-solution for automated error detection of industrial hairpin weldings using convolutional neural networks" is a paper that was included in the results but not relevant to our study since it is not a data cleaning approach.

```

("machine learning" OR "deep learning" OR "neural network" OR "neural networks" OR "reinforcement learning" OR
supervised OR unsupervised)
AND
(
  ("data cleaning" OR "data cleansing" OR "data scrubbing")
  OR ("data repairing" OR "data repair" OR "error repair" OR "error repairing")
  OR ("confident learning" OR "label cleaning")
  OR ("error detection" AND (tab* OR cell* OR row* OR image* OR text*))
)

```

Figure 5.2 The search query used to fetch papers from all academic databases used in our study except Google Scholar.

Adaptations for Google Scholar

As opposed to other academic databases, GS has many peculiarities that require us to adapt our search pattern. Our final search pattern can be visualized in Figure 5.3. Below, we describe GS’s peculiarities and explain how we adapted the query.

1. All queries are truncated to 256 characters.
2. Parentheses are not a priority operator.
3. *OR* operators have a higher precedence than *AND*.
4. *AND* operators are implicitly added between terms that are not operators (*e.g.*, terms that are not *OR* or *AND*). Hence, the search query *data cleaning* is equivalent to *data AND cleaning*.

In order to address the first constraint, we split the query in two and ran each part separately. For the same reason, we also removed any *AND* operators, since they are redundant and added automatically (constraint 4). As parenthesis do not have their usual effect (constraint 2), we also removed them from our query. This prevents any nested Boolean operator. Hence, we removed the conjunction we described in Section 5.2.3 to limit the number of results on error detection.

5.2.4 Paper Collection

In the following, we describe how we collected papers from academic databases using the query described in Subsection 5.2.3. Then, we describe the separate paper collection process we followed for GS, because of the limitations of that database. Similarly to the other works [150,307,309], we conducted the search in the following academic databases:

- Google Scholar³

³<https://scholar.google.com/>

- Engineering Village⁴
- Web of Science⁵
- Science Direct⁶
- ACM Digital Library⁷
- IEEE Xplore⁸

As shown by the recent publications [282, 300], DC&ML is a relatively new topic. Hence, when running our search query, we limited the results to papers published between 1 January 2016 and 17 October 2022 (the later date being when we ran the query). On Engineering Village, we used the duplicate removal feature, by keeping the results from Compendex over Inspec.

Adaptations for Google Scholar

Unlike other databases, GS does not provide a way to export records in bulk, which discourages mining techniques. Thus, to alleviate the task of collecting paper, we used the tool “Publish or Perish”⁹. Similarly to [150], we searched 1000 records for each year. As mentioned in Section 5.2.3, our initial query had to be divided into two parts because of GS’s limitations. Thus, we separately ran both queries, for every year for a period of six years, and collected 1000 papers per year, for a total of 12 000 papers (2 queries $\times$ 6 years $\times$ 1000 papers = 12 000 papers).

5.2.5 Paper Selection

In this section, we explain the steps we followed to filter out irrelevant results. Table 5.1 shows the number of papers obtained from each academic database prior to any kind of filtering. We provide the list of papers retained after each step of the paper selection process in our replication package¹⁰.

⁴<https://www.engineeringvillage.com>

⁵<https://webofknowledge.com>

⁶<https://www.sciencedirect.com>

⁷<https://dl.acm.org>

⁸<https://ieeexplore.ieee.org>

⁹Harzing, A.W. (2007) Publish or Perish, available from <https://harzing.com/resources/publish-or-perish>

¹⁰<https://github.com/poclecoq/SLR-datacleaning>

```
"machine learning" OR "deep learning" OR "neural network" OR "reinforcement learning" OR "supervised" OR
"unsupervised"
"confident learning" OR "label cleaning" OR "error detection"

"machine learning" OR "deep learning" OR "neural network" OR "reinforcement learning" OR "supervised" OR
"unsupervised"
"data cleaning" OR "data cleansing" OR "data scrubbing" OR "data repairing" OR "data repair" OR "error repair"
```

Figure 5.3 The search query used to fetch papers from Google Scholar.

Table 5.1 Number of papers retrieved from the academic databases.

Database	Number of papers
Google Scholar	12,000
Engineering village	1224
Web of Science	511
Science Direct	135
ACM Digital Library	89
IEEE Xplore	295
Total	14,254

Duplicate Filtering

We uploaded all the titles of papers obtained from the various databases in Zotero¹¹. We leveraged the duplicate removal functionality of the tool and found 3,742 duplicates in GS' papers and 769 in the results of the other databases. Thus, after this step, 8,258 papers from GS were retained, and 1,485 for the other databases.

Google Scholar Pruning

While GS returned a lot of results, a considerable amount of them are not relevant to our study, for two reasons. First, a lot of irrelevant papers are included because of the search engine's limitation which we describe in Section 5.2.3. Second, GS only searches keywords in the full text of the paper or in the title. Thus, papers irrelevant to our study that mentioned data cleaning and ML in the papers' body (e.g., as key/index works) matched our search query. While we could not address the first issue in any other way than manually inspecting and then filtering papers, we devised an approach for the second issue. We managed to

¹¹<https://www.zotero.org/>

only keep papers whose title, abstract, and author keywords matched the search query, similar to what was done by a similar study [150]. We did so using a Python script we implemented which is available in our replication package¹². After this step, 1,862 results from GS remained. We merged this set of papers together with the 1,485 papers coming from the other databases. We removed duplicated papers from the merged set, resulting in a total of **2,968 papers**.

Inclusion and Exclusion Criteria

From this step onward, papers were filtered manually. One author went through the whole set of papers and filtered out the ones irrelevant to our study. A second author verified the resulting set of papers. Any leftover duplicate was removed. For this step, the following inclusion and exclusion criteria were considered:

Inclusion criteria:

- Papers proposing an approach to detect or fix errors in an ML dataset.
- Papers presenting an approach to detect or fix errors in data while leveraging ML techniques.

Exclusion criteria:

- Papers whose data cleaning approach is specific to their problem (i.e., the technique does not generalize).
- Papers on data cleaning for other data types than tabular, image, or text data, such as sound data.
- Papers on detecting or fixing any type of attacks on an ML model through data (i.e., data poisoning and adversarial examples).
- Papers that can not be accessed free of charge through our institution's subscriptions.
- Secondary works (e.g., surveys and review papers), research proposal, workshop, letters, undergrad theses, or position papers.
- Papers that are not written in English.
- Papers on text data cleaning for other languages than English.

After this step, **124 papers remained**.

¹²<https://github.com/poclecoqq/SLR-datacleaning>

Quality Control Assessment

While most papers are relevant to our study, not all of them are able to answer our RQs. Thus, based on a previous study [150], we defined a set of questions to filter out papers with low quality. We also defined a second set of questions to filter out any remaining paper that is irrelevant to our study.

Quality control questions:

- Q1.1 Is the objective of the research clearly defined?
- Q1.2 Is the context of research clearly defined?
- Q1.3 Does the study bring value to academia or industry?
- Q1.4 Are the findings clearly stated and supported by results?
- Q1.5 Are limitations explicitly mentioned and analyzed?
- Q1.6 Is the methodology clearly defined and justified?
- Q1.7 Is the experiment clearly defined and justified?

Relevance control questions:

- Q2.1 Does the paper present an approach to detect or fix errors in an ML dataset?
- Q2.2 Does the paper present an approach to detect or fix errors in data using ML techniques?

Each paper is considered by a reviewer who assigns it a score based on how well it complies with the questions. Papers under a certain threshold are considered by a second reviewer who repeats the process followed by the first reviewer. If the second reviewer's score is also under the threshold, the paper under consideration is excluded from the study. However, if the second reviewer assigns a score above the threshold, the reviewers have to discuss to decide whether the paper should be included or not in our study. If they are unable to reach a consensus, a third reviewer makes the final decision.

In order to evaluate a paper, reviewers rated on a scale from 0 to 2 how much they agreed with the quality and relevance control questions. For the quality control questions (i.e., Q1.X), a paper is excluded by a reviewer if the sum of the 7 questions is lower than 7. For the relevance control question, a paper is excluded if it has a score of zero for both questions (i.e., Q2.1 and Q2.2).

As a result, we excluded 44 papers. Our ratings are available in our replication package¹³. After this step, **80 papers remained**.

¹³<https://github.com/poclecoqq/SLR-datacleaning>

5.2.6 Snowballing

Snowballing refers to the act of discovering new papers through paper references [310]. In this study, to read relevant studies we miss, we include any paper cited by at least two papers from the filtered list of papers from Section 5.2.5. We partially automated this process with a script available in our replication package¹⁴. In this script, we used the Python library Scholarly [311] to fetch paper citations from GS. We evaluate the inclusion and exclusion criteria from Section 5.2.5 along with the control assessment questions from Section 5.2.5 before including a snowballed paper in our study. After this step, we found 21 new papers. Hence, in total, we collected **101 papers for examination** in our study.

5.2.7 Data extraction

In order to reason efficiently over the large amount of information contained in the set of papers we read, we extracted the following information from each paper:

General information

- Title
- URL to the paper
- Authors
- Year
- Publication venue

Questions

- What does the data cleaning approach try to fix (e.g. missing values, dirty data, inconsistent data, etc.)?
- What are the datasets used in the experiment?
- What are the ML models used in the experiment?
- What are the quality measurements (i.e., how is the data cleaning approach evaluated)?
- Against what is the approach compared (e.g., against baselines, other approaches, etc.)?
- What is the performance of the approach, in terms of ML performance (e.g., accuracy, F1-score, etc.)?
- What is the performance of the approach, in terms of engineering aspects (e.g., resource consumption, time to completion, etc.)?
- What are the limitations of the approach?

¹⁴<https://github.com/poclecoqq/SLR-datacleaning>

Additionally, reviewers are asked to summarize the papers they have reviewed, in order to give a general picture of the study and to highlight any other interesting information. All extracted data is available in our replication package¹⁵.

5.3 Selected Taxonomy

Based on the papers included in our study and using existing studies [34, 300], we built a taxonomy of DC&ML approaches, which we describe in this section. We use that taxonomy to structure our review of the literature in Section 5.5. In the taxonomy, illustrated in Figure 5.4, data cleaning approaches are first grouped based on the type of error they try to address (e.g., incorrect feature value, duplicates, etc.). Bold font is used in the figure to indicate the number of papers included in a category. The number of papers in a category may not be equal to the sum of the number of papers in its subcategories because a paper may be classified to zero or many subcategories of a category it belongs to.

In the following, we introduce the different types of data cleaning activities, which correspond to the categories on the first level of the taxonomy in Figure 5.4. The other categories will be introduced later in their respective subsections during the review (in Section 5.5).

- **Feature cleaning:** Feature cleaning approaches try to detect or repair errors in the features of a record. It is a problem that has been addressed by the database community for a long time [30–32] and is generally loosely referred to as data cleaning. It should be noted that data cleaning is not limited to feature cleaning and that other activities are also considered to be data cleaning activities (e.g., entity matching or outlier detection [34]). Thus, we use the term “feature cleaning” instead of “data cleaning” to refer to that data cleaning activity. Feature errors may happen for a large variety of reasons. For example, a noun may be misspelled by the person collecting data [299].
- **Label cleaning:** Similar to feature cleaning, label cleaning approaches try to detect or repair label errors. Label errors may happen for a variety of reasons. For example, a record might be mislabeled because of a lack of domain knowledge [54].
- **Entity matching:** Entity matching refers to the process of finding tuples in a database that refer to the same real-world entity [34]. Duplicates can be introduced in a dataset for many reasons. For example, a customer could be recorded multiple times in a database because the customer used different names at checkout [34].
- **Outlier detection:** An outlier can be defined as an observation that deviates so much from the other records in a dataset that it arouses suspicions that it has been generated

¹⁵<https://github.com/poclecoqq/SLR-datacleaning>

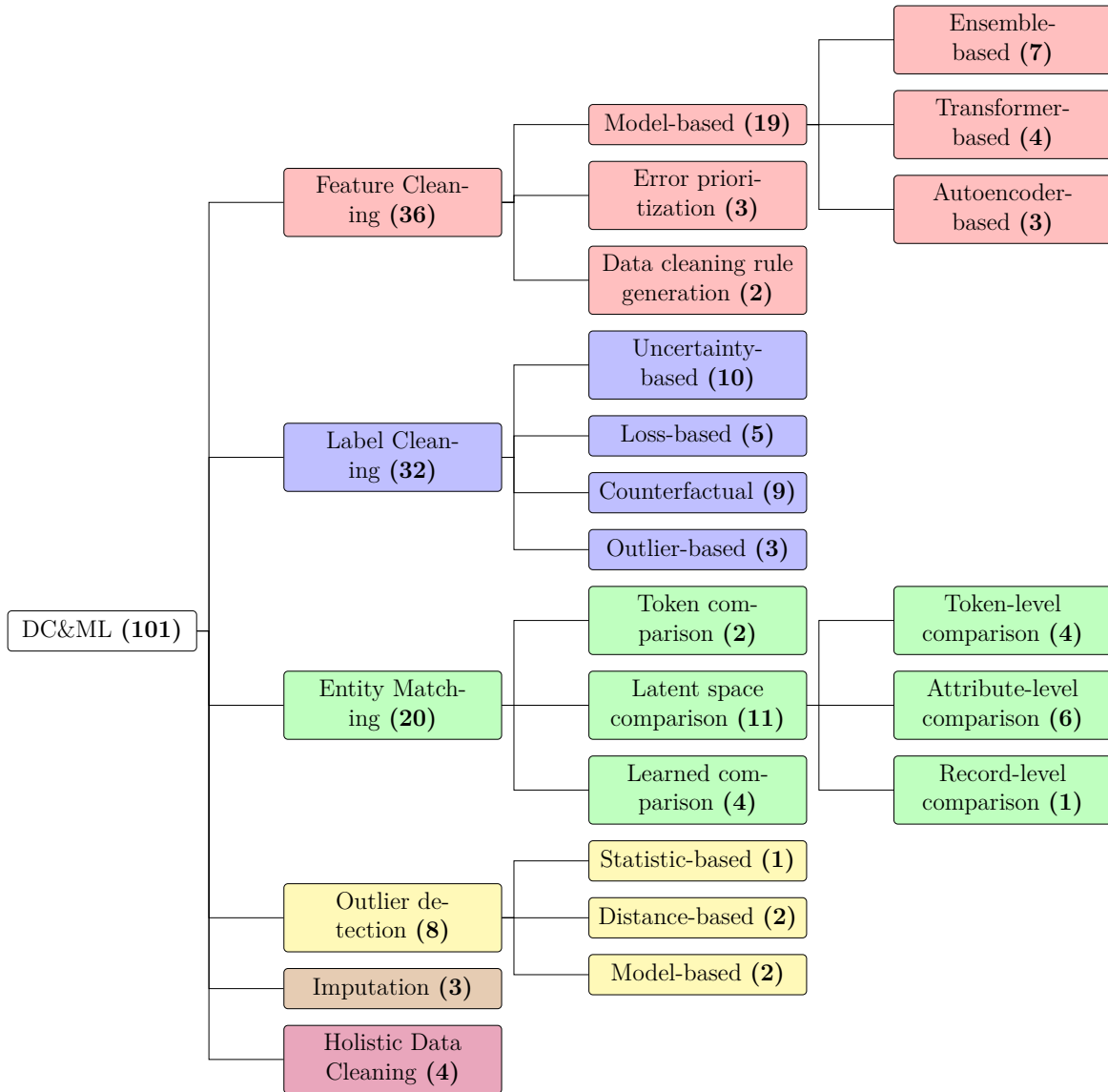


Figure 5.4 Overview of the approaches discussed in this paper

by a different process [312]. Outliers may hinder an ML model’s performance if they have been generated by a different process. Hence, outliers may be purged from a dataset during data cleaning.

- **Imputation:** Imputation refers to assigning a value to a record’s feature whose value is missing [313]. Having missing values is a frequent problem [314] and is often experienced when data collection is manual [315]. An ML model can only ingest complete records; thus records with missing values have to be imputed otherwise they are removed from datasets.
- **Holistic data cleaning:** We use the term “holistic data cleaning” to refer to the approaches that try to clean more than one type of error at a time. For example, [316] designed an RL agent that learns to clean a dataset using a set of data cleaning tools that can clean different types of errors.

5.4 Statistical results

In this section, we provide a statistical description of the papers included in this study. Figure 5.5 displays the number of papers included in our study per data cleaning activity (which we previously defined in Section 5.3). Studies proposing an approach to clean more than one error at a time are in the category “More than one”. Generally, the papers that were included in that category proposed an approach that could be adapted for different cleaning goals (e.g., [317]). We can observe three main categories of papers, along with four minor ones. The main categories are “Feature Cleaning” (34%), “Label Cleaning” (32%), and “Entity Matching” (18%). The minor categories are “Outlier Detection” (8%), “Holistic” (4%), “Imputation” (3%), and “More than one” (i.e., approaches that individually fix more than one type of error) (2%). Figure 5.6 shows the distribution of the papers’ publication year. We can observe a generally growing trend. The number of papers in 2022 is less than in 2021 because we collected the papers during this year, in October 2022. In Figure 5.6, we display the authors’ affiliation distribution. The majority of the papers included in our study have been written by researchers from academia (74%). We observe that 21% of the papers included in our study have been written by teams composed of a mix of researchers from academia and the industry. Finally, only 5% of the papers are attributed to teams from the industry. In Figure 5.8, we display the number of papers published per venue. We filtered out venues that counted only one paper for readability purposes. The most recurrent venues are the ACM SIGMOD conference, the Association for the Advancement of Artificial Intelligence conference (AAAI), the Expert Systems with Applications journal, and the VLDB Endowment conference.

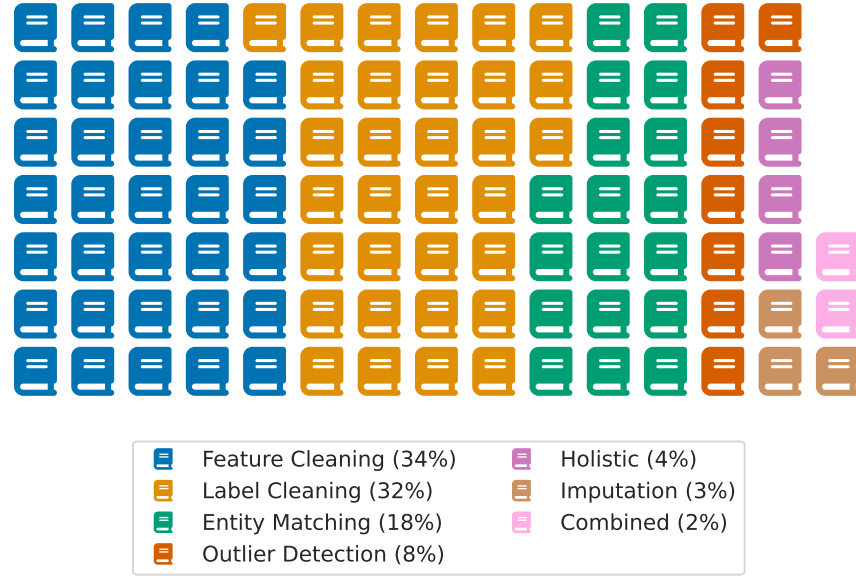


Figure 5.5 Distribution of data cleaning topics

5.5 Review

In this section, we summarize current data cleaning approaches for tabular, text, and image data. We use the taxonomy described in Section 5.3 to structure our review. Hence, the following subsections discuss about feature cleaning, label cleaning, entity matching, outlier detection, imputation, and holistic data cleaning. We provide a brief description of the existing approaches along with future work recommendations at the end of every subsection.

5.5.1 Feature Cleaning

In this section, we cover data cleaning approaches to detect or repair feature errors. Because of the long-lasting efforts of the database community to clean tabular data [30–32], the majority of the papers in this section process this data type. Thus, the following subsections present feature cleaning approaches for tabular data, except for the last one (Section 5.5.1). In the first one (Section 5.5.1), we present model-based approaches. Then, we follow with ensemble-based (Section 5.5.1), transformer-based (Section 5.5.1), and autoencoder-based approaches (Section 5.5.1), which all are subcategories of model-based approaches. We continue with error prioritization (Section 5.5.1) and data-cleaning rule generation approaches (Section 5.5.1). Then, we present other feature-cleaning approaches that do not belong to any of the aforementioned groups (Section 5.5.1). Finally, we describe some approaches commonly used to improve the data cleaning approaches performance (Section 5.5.1) and we finish with

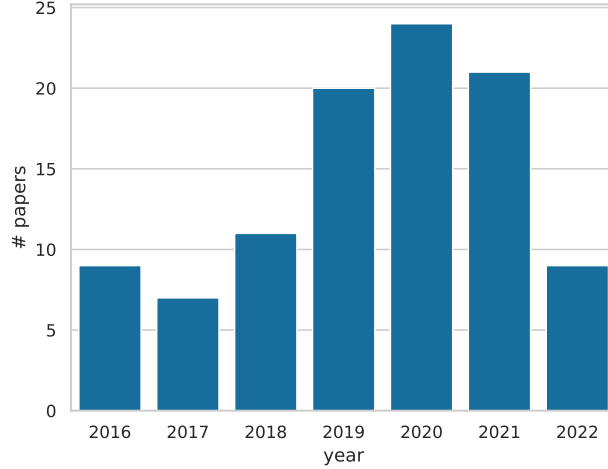


Figure 5.6 Number of papers published per year

feature cleaning approaches for other data types than tabular (Section 5.5.1).

Model-Based Approaches

The approaches presented in this category view data cleaning as a prediction problem. That is, a model must predict whether a value is clean or not (i.e., error detection) or what is the clean value of a dirty cell (i.e., error repair). Formally, for error detection, a model m must predict whether a record's feature value $r[A_i]$ (where r refers to a record and A_i to the i -th attribute) is dirty or not ($y \in [0, 1]$) given the full record r . Similarly, for error repair, a model m must predict the clean value y for a record's feature $r[A_i]$ given the full record r , where y 's domain is not necessarily restricted. Note that the following sections (i.e., ensemble-based, transformer-based, and autoencoder-based approaches) are also model-based approaches. In the following, we refer to the clean features of a record as records' labels. Indeed, the clean values are the labels used by the feature cleaning approaches.

One of the major differentiating factors between error detection approaches is their choice of feature engineering. We discerned three categories of features: (1) frequency-based features, (2) format-based features, and (3) metadata features. Similar to outlier detection approaches (see Section 5.5.4), the engineered features are generally designed to highlight the abnormal characteristics of data. A commonly used feature-engineering practice for error detection is to create features that measure how frequent a value is in a dataset (i.e., frequency-based features) [318–321]. For example, the authors in [319] measured the TF-IDF score of n -grams inside a cell to encode how common a cell's value is. The less common it is, the higher the chances of it being an error. Frequency-based features can be measured using the information

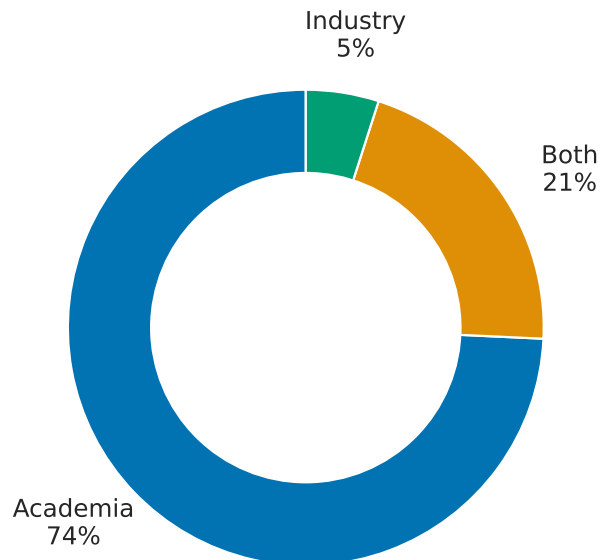


Figure 5.7 Authors' Affiliation Distribution

contained in other columns [319, 320]. For example, [319] used the co-occurrence of values among different attributes to facilitate the error detection process. Another category of engineered features focuses on the format of the values (i.e., format-based features) [318–321]. Corrupted cells might not follow the syntactic format that is expected for a feature. For example, a value that represents a name should not have numbers. [321] replaces all numbers and characters with symbols that are unique to them; thus focusing only on the shape of the data. For example, given the value “400\$”, the encoded format could be “nnns”, where “n” represents numbers and “s”, symbols. Finally, the data cleaning approaches sometimes complement their feature set with metadata information (i.e., metadata features) [318, 319]. For example, [319] indicates the data type and the string length of a cell in the feature set. For error repair, ML can be used to predict values to repair corrupted cells, similar to data imputation (see Section 5.5.5), as it was done in [322].

Ensemble-Based Approaches

Coined by [282], ensemble-based approaches to data cleaning comprise every approach where a model uses the output of data cleaning tools to clean data. As it was pointed out in [323], no single data cleaning tool is the best in every situation and no tool can detect all errors. Leveraging ensembles of data cleaning tools opens up the possibility of powerful approaches to clean errors since they could potentially detect and repair a larger diversity of

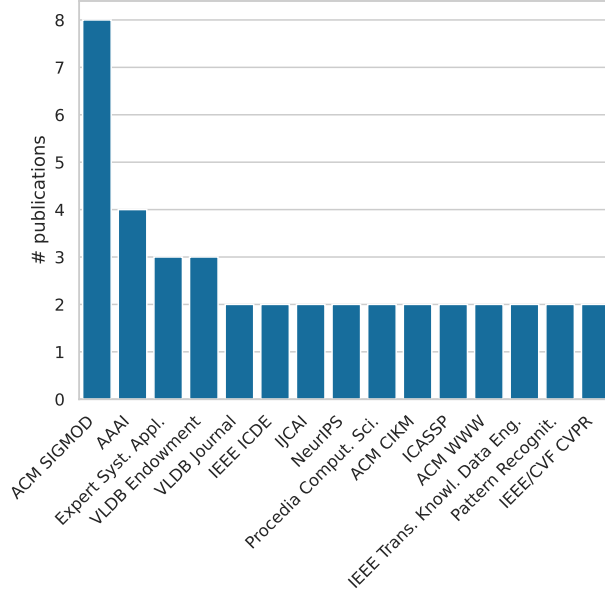


Figure 5.8 Number of papers published per venue

errors [304, 324, 325]. Similarly to ensembling techniques with ML models, ensemble-based approaches for feature cleaning have an ensemble of data cleaning tools, which we refer to as the “base” data cleaning tools. The outputs of the base data cleaning tools are ingested by one singular model making the final data cleaning prediction, which we refer to as the meta-model.

Usually, the base data cleaning tools are simple cleaning tools developed without using ML techniques. The most commonly used ones are integrity constraints [304, 318, 324, 325], matching dependencies [304, 325], outlier detection [304, 318, 325], and pattern violation detection [318, 325]. For data repair, simple algorithms that memorize transformations between dirty and clean data are sometimes used [317, 326]. These are designed specifically for ensemble-based approaches and can only suggest good repairs collectively. Hence, they are generally not viable data repair tools independently. Mahdavi et al. define three types of such tools: (1) value-based models, (2) vicinity-based models, and (3) domain-based models [326]. Models in the first category only consider the corrupted value when suggesting repairs [317, 326]. For example, given a misspelled string such as “a dg”, the tool could suggest “a dog”. On the contrary, vicinity-based models leverage the context of the corrupted value to predict its repair. For example, given the “city” attribute of a record is set to “Tokyo”, then we know that the “country” attribute should be “Japan”. Finally, domain-based models repair a value based on the domain of the attribute. For example, a domain-based repair strategy could replace a corrupted value with the attribute’s mode.

Once the base data cleaning tools have processed an example, one is left with a set of data cleaning predictions (error detection or repair) that are ingested by the meta-model for the final data cleaning prediction. We observed different ways to include the cleaning signal of the base data cleaning tools in the prediction process of the meta-model. When the meta-model of the cleaning approach is a Probabilistic Graphical Model (PGM), the base data cleaning tools are transformed into inference rules for the meta-model [304, 324]. For error detection, [325] trains, for each feature, a model that ingests the base data cleaning tools' binary output (i.e., dirty or not) and predicts whether a record is dirty or not. Compared to error detection, error repair with ensemble-based approaches has the additional challenge of having base data cleaning tools that may output a wide variety of values. Thus, error repair tools adopt a different approach to combine the base data cleaning tool predictions. Instead of generating a repair, the meta-model selects the repair that is the most likely to be correct in a pool of repairs suggested by the base data cleaning tools [317, 326]. In [326], each base error repair tool suggests values to repair corrupted cells. Then, for each repair, a vector holding the confidence that the repair is correct according to each base error repairer is generated. This vector is then fed to a Neural Network (NN) that will predict how likely a repair is given the confidence of the base error repair tools. The repair with the highest score becomes the repaired version of a dirty record. Instead of evaluating repairs using the confidence score of the base tools, Wang et al. let the meta-model select the most probable repair by comparing the repaired versions of a record with its dirty version [317]. To facilitate the comparison process, the records are transformed into embeddings. Then, an NN ingests the difference between the embeddings, along with the embedding resulting from the concatenation of both records. This strategy of encoding records captures both the differences and similarities between the compared records. For every considered pair, the NN will output a score, stating how likely is the proposed repair. The most likely repair for a record is then selected. This approach was originally designed for entity matching but showed to be apt for error repair as well.

While ensemble-based approaches may seem appealing, they have a few challenges of their own. To begin with, the predictions of a meta-model can be skewed if the predictions of the base tools are correlated [318]. This is simply due to the fact that a larger feature space is allocated to what is essentially the same information. The authors in [318] address this challenge by pruning out correlated tools and keeping the most effective ones. To achieve that, k-means clustering is run on the predictions of the base tools. Then, using a validation dataset, the best tool for each cluster is selected. In addition to the aforementioned issue, a large pool of base cleaning tools results in two additional problems. First, it may lead to a longer run time for the whole data cleaning process. Thus, [327] prunes out tools

that proved to be ineffective on similar datasets. The dataset similarity is measured by comparing the semantic and syntactic similarity between columns. Second, significant human labor can ensue if a large number of tools require manual parametrization. To address this problem, [327] generates, for every tool, all configurations in a predefined set.

Transformer-Based Approaches

In this section, we present approaches that clean the features of a record using transformer models [328]. Because of their impressive performances in the last years, there has been an interest in the research community to apply this kind of model to a wide variety of tasks [329–331]. Two issues must be circumvented in order for these models to be used for feature cleaning. The first hurdle is in the very nature of the task: feature cleaning is different from text generation. Effectively, transformer models are designed to generate text, while feature cleaning is usually conceptualized as a classification problem. Three approaches have been suggested to address this problem. The first solution transforms feature cleaning into a text generation problem. More precisely, [268] prompts the foundation models with data-cleaning questions. For example, for error detection, the user may ask the model if there is an error in a given sample, to which the model will answer “yes” or “no”. The authors further show that very few training samples are required for the model to detect errors, from a few (few-shot learning) to none (zero-shot learning). For error repair, [332] uses a transformer to predict the value of every cell of a record and replace it if it differs from the shown value. The transformer model is trained using the masked data model objective. That is, the model ingests a record with a masked value and outputs a record with the mask replaced with a value. If the value generated by the model differs from the actual value that was masked, then the example is considered to be corrupted. Instead of transforming data cleaning into a text generation task, [333] adapts a transformer model to a classification task. A linear layer with softmax activation is added to the output of the transformer and the model is trained to minimize cross-entropy loss. Before being fine-tuned for this binary classification task, the transformer is trained using the masked data model objective, so it learns the structure of data. The third way to use transformers for data cleaning is by using them to generate embeddings that can be used by another model for data cleaning. [317] pre-trained a transformer using SimCLR contrastive learning framework [334], so similar records have similar embeddings. Then, an NN ingests these embeddings and predicts how likely a repair is given a dirty sample. The most likely repair is selected.

The second issue that must be addressed before using transformers for data cleaning is the format of the data itself. While the data cleaning approaches in this section are applied to

tabular data, transformers are trained with natural language. A naive approach would feed the record as it is to the model (i.e., transform the record into a sentence by concatenating its features using spaces). However, precious information about the features’ semantics would be lost, since the model does not know what a value represents (e.g., a number does not make sense if we do not know what it counts). Hence, to address that problem, the tabular records must first be transformed into sentences, and then the model must learn to understand the structure of these sentences. To inject the tabular structure of data into strings, [268] prepends each attribute value with its name before concatenating it into a string. For a record that has a feature named “City” and a value of “Montreal”, then the attribute would be represented by “City: Montreal”. The latter would be concatenated into a sentence with the substrings generated for the other features. To inject the semantic difference between attributes’ names and their values, [317, 332] further prepend attributes’ names with the special token “[A]” and the values with “[V]”. Extending on the previous example, we would have “[A]City [V]Montreal”. Once tabular data is converted into strings, the model must learn to understand the format of the data. Similar to what is usually done with large language models (LLM), the masked data model objective is used in [332, 333] to train transformers in an unsupervised way. The model is trained to predict the missing value in a record’s entry, which enables it to unlearn its assumption of processing natural language data and learn the structure of the dataset at hand. [317] leveraged contrastive learning to train the encoder of a transformer to generate embeddings that are close in the latent space for similar records, and far for different records.

Autoencoder-Based Approaches

The data cleaning approaches in this section use autoencoders to detect or repair records. Autoencoders are a type of ML model trained to output the data they receive as input while being constrained to represent the data in a latent space with fewer dimensions than the input [335].

[336] argues that autoencoders have denoising capabilities because of the compression process. Representing the input data into a lower dimension forces the autoencoder’s intermediate representations to contain only the most important information necessary to recreate the input data. As a result, noise is removed from the data. Hence, they use the innate noise removal capability of autoencoders to repair records. To do so, they first transform records into probabilistic ones (i.e., records whose features have a probability distribution over potential values). Then, they feed the record to the autoencoder, which outputs another probabilistic record. The result can be transformed back to non-probabilistic data by picking

the most likely value for each feature. [337] proposes a variant of variational autoencoders (VAE) architecture for noise removal that does not suffer from the posterior collapse problem (when the model confidently suggests a small number of reconstructions). Authors employ a three-step process to clean samples using their autoencoder. First, an example to clean is ingested by the model and a latent distribution is generated by the model. Second, multiple latent vectors are sampled from that latent distribution. Third, examples are generated from each latent vector and averaged to generate the final clean value. Instead of using the autoencoder to generate values to repair errors, Liu et al. use the reconstruction loss of an autoencoder to detect corrupted records [269]. Records that have a high loss during the first epochs of training are flagged as erroneous. The study further shows that poisoned samples generally have the lowest loss; hence their method can also be used to detect such records.

Error Prioritization Approaches

Instead of directly cleaning data, the approaches covered in this section try to identify and prioritize the records that should be cleaned by a human expert.

Authors in [338] leverage the knowledge stored in Wikipedia to clean data. More precisely, by generating questions for a QA engine that understands natural language and whose answers are Wikipedia pages, they are able to verify the values in a dataset by comparing them with what is on Wikipedia. If a value conflicts with what the engine has returned, the records are sent to a human for review. Instead of identifying records that are potentially erroneous, authors in [339] search for records that would have the largest impact on a model's performance if cleaned. They argue that data cleaning efforts are wastefully spent on samples that do not alter a model's performance. Thus, the authors propose an approach that tries to find samples that, once cleaned, will render the next iterations of cleaning less useful (since the next iterations will not alter a model's performance). Similarly, Krishnan et al. [340] propose an approach to prioritize the samples sent to a human for review. A model is first trained on a dataset (for the prediction task of the dataset, not for data cleaning). Then, samples that would trigger the largest weight update if the model was retrained on them after being cleaned are prioritized. The weight updates are estimated using previous examples with similar feature values.

Data Cleaning Rule Generation Approaches

The approaches included in this category use ML to infer data cleaning rules that can be used to clean a dataset. Data cleaning rules are typically expressed using integrity constraints, a technique from database schema design [341]. Integrity constraints can express a wide

variety of logical clauses, such as: “two persons living in the same city must live in the same country” or “two persons can not have the same phone number” [34].

[342] proposes a hybrid data cleaning framework, named MLNClean, that leverages Markov Logic Networks (MLNs) to supplement integrity constraints and makes it possible to clean unknown data errors. MLNClean first uses MLN to generate a set of probable data cleaning rules with their corresponding probabilities. These rules are then leveraged to generate repair candidates. The repair candidate that is the most likely (based on the likeliness of the data cleaning rules that generated it) and that differs the least from the original value is then used to repair an observed value. In another work [343], authors propose AutoFD, a framework for Functional Dependency (FD) (a type of integrity constraint) discovery based on structure learning techniques in PGMs [343]. A PGM is introduced to capture dependencies that FDs introduce among attributes in a dataset, revealing that learning the graph structure of this model is equivalent to discovering FDs.

Other Approaches

In this section, we cover other feature cleaning approaches for tabular data that are not included in the previous categories.

- [305]: Similar to ensemble-based feature cleaning, their work uses an ensemble of feature cleaning tools. However, contrary to the aforementioned category, their goal is not to clean data but rather to directly generate a model that would have been trained on clean data. To do so, all feature cleaning tools are individually applied to a dataset, generating different versions of it. Then an ML algorithm trains a model for each processed dataset. The models’ predictions are combined together using boosting [344]. As a result, the predictions of the ensemble are influenced by all the data cleaning tools.
- [345]: Authors propose a new domain-specific Probabilistic Programming Language (PPL), namely PClean that is tailored for data cleaning. Similar to [304], PClean leverages dataset-specific knowledge. Authors introduce a domain-general nonparametric prior on the number of latent objects and their link structure in the dataset. Code written by PClean customizes this prior probability via a relational schema (for relational dataset) and via generative models for objects’ attributes. Inference in PClean is based on a novel Sequential Monte Carlo (SMC) algorithm, to initialize a latent object database with plausible guesses, and novel restoring updates to fix mistakes.
- [346]: Their paper proposes an improvement over HoloClean [304]. Most of the data cleaning approaches consider that data cleaning is only applied once, before training a model. This does not reflect how data is cleaned with real ML software systems since

new training data is continuously generated. Hence, they propose a few enhancements to HoloClean [304] to make it more efficient in that context. Notably, they avoid retraining the data cleaning model if the distribution of incoming data does not change.

Common Techniques to Improve Performance

In this section, we cover various techniques commonly used to improve a model’s performance for model-based approaches, namely, data augmentation, semi-supervised techniques, and active learning.

Data Augmentation Data augmentation is a technique to generate synthetic new examples by applying transformations to existing data [347]. Data augmentation is used in feature cleaning approaches to address the class imbalance problem; training datasets for feature cleaning have a larger amount of labeled clean records than labeled dirty records [320, 333]. In other words, in labeled datasets, there is generally more clean data than dirty data and dirty data is needed for the ML approach to learn to detect dirty data (error detection) or to learn to clean it (error repair). Hence, data augmentation tries to address that problem by generating more labeled dirty data. To do so, data augmentation techniques transform clean data into dirty data by applying plausible corruption mechanisms (i.e., the error induced into the record is plausible given the dataset). In order to define a data augmentation strategy, two components must be defined: the process that transforms clean data into dirty data and the process that selects data transformations (i.e., assigning a probability to each transformation). Instead of manually crafting data transformation rules, [320, 321, 333] learn them from the data. In [333], Gestalt Pattern Matching [348] is used to find the non-matching substrings between the clean and dirty versions of a text feature. These substrings are then used to form a new mapping, from the clean substring to the dirty one. Similarly, [320, 321] adopt a hierarchical pattern matching approach to learn valid transformations. Once data transformation functions are defined, the data augmentation approach must build the function that selects the data transformation that will be applied to a record. In other research works [320, 321, 333], authors sample a valid transformation following the empirical distribution of transformations (i.e., how often each transformation was used empirically).

Semi-Supervised Techniques Semi-supervised learning is a branch of ML that uses unlabeled data (in addition to labeled data) to build better models [349]. The semi-supervised techniques presented in this section use the labeled data to label the unlabeled data. In order to generate training data for its data augmentation approach, [320] fits a Naive-Bayes

model that, similar to imputation, predicts a clean value for each feature of a record. The most confident predictions (i.e., with over 90% confidence) become the clean versions of the observed values. [321, 325] use label propagation to increase a labeled dataset size for error detection. Label propagation consists of assigning the label of a record to a similar unlabeled record [350]. The challenge with this approach is to devise an accurate measure of similarity between records to avoid mislabels. [321] considered two feature vectors to be similar if all their attributes did not differ by more than a threshold. Similarly, [325] used hierarchical clustering [351] to find groups of similar feature vectors. Records in the same cluster share the same label.

Active Learning Active learning refers to a situation where a ML algorithm can iteratively query an oracle (e.g., a human) to label new data points [352]. In the following, we show how some studies used active learning to achieve superior performance. [319] relies on the confidence score of the data cleaning models to select the cells to send to a reviewer. Because they ask reviewers to clean cells, not tuples, their acquisition function (i.e., the function that selects records) starts by selecting a feature to clean (i.e., a column of a dataset), then picks a batch of cells inside that column to clean. The column selected in the first step is the one where the model is, on average, less confident of its predictions. The cells are then selected using the query-by-committee algorithm [353]. [321] uses two different models for data cleaning and active learning. The model used for active learning is implemented using Probabilistic Soft Logic [354] and, similar to the data cleaning model, tries to predict whether cells are dirty or clean. Records that are more likely to be dirty are sent for review. Finally, for error repair, [326] prioritizes tuples with a lot of errors that are common in the dataset. The goal is to create a training dataset that covers most types of errors in a dataset. Note that the approach assumes that an error-detection tool already marked the dirty cells.

Other Data Types

In this section, we cover feature cleaning approaches for other data types than tabular. We chose to present data cleaning approaches for each data type in a dedicated part because each data type has a peculiarity that makes the data cleaning task significantly different. Thus, in the following, we describe feature cleaning approaches for time series, text, and image data, respectively.

Time Series Time series are a collection of observations made in chronological order [355]. The feature cleaning approaches in this category have to consider the ordinal nature of time

series data in order to clean it.

[356] combined auto-regressive models with the minimum repairing principle of data cleaning [34] to iteratively clean data. Auto-regressive models are used to suggest values to repair anomalous observations. In their approach, an auto-regressive model suggests repairs for every observation. Only one repair is applied: the one that differs the less from the original observation. The auto-regressive model is retrained on the cleaned time series, and the procedure is repeated until the difference between the original and clean value is below a threshold or the maximum number of iterations is reached. Similarly, [357] presents an iterative approach for the detection and repair of anomalies in time series data. First, Auto-regressive with Exogenous inputs (ARX) and NN models are trained on a time series and used to predict repair values for every value of a time series. Then, hypothesis testing over the difference between observed values and predicted values is used to detect outliers. The most abnormal observed value is replaced by the predicted value, and the process is repeated.

Text In this section, we cover techniques for detecting errors in text data. The addressed errors include grammatical, syntactical, and semantic errors.

NNs models are used for detecting and correcting grammatical errors in text. For example, in [358], an RNN-based model is used for detecting grammatical errors in English verbs. The model consists of two sub-RNNs such that one processes the sentence from the beginning to the target verb and the other processes the sentence from the end to the target verb. An additional layer is used to process the output of the RNNs and predict the verb which is then compared with the original verb to detect any potential error. Knill et al. approach error detection as a sequence labeling task where a Bi-directional LSTM is trained to classify each token in the input sentence as correct or incorrect within the sequence [359]. [360] utilize word contextual embeddings generated by BERT and concatenate them to character embeddings generated by an LSTM. They then use the BERT model, in particular, the first six layers of BERT to classify each word as correct or incorrect. [361] investigates different NN architectures (CNN, Bi-RNN, Bi-LSTM, and their multi-layer variants). Provided with a sequence of tokens, the model calculates the probability for each token to be correct or incorrect. Among the compared models, Bi-LSTM outperforms all other models including the multi-layer Bi-LSTM variant.

To detect syntactic errors, [362] proposes to train two LSTM models, one is forward and the other is backward, to predict tokens. The disagreement between the two models is used as an indication of a possible syntax error. They also attempted to propose a correction by identifying an alternative token sequence derived from the models.

The model introduced by [363] detects semantic errors in terms of inconsistencies between

a numerical value (e.g., clinical test result) and its interpretation. When a document is received, sets of words with similar meanings are employed to generate all possible word substitutions within the document, creating a set of candidate repaired documents. Each of the candidate documents is fed to an LM to predict which one is the most likely to be correct. Likelihood ratios between candidate and original documents are then computed. Candidates with higher scores than the original indicate errors; the highest score is accepted as the correction.

Image A common distortion that can be encountered in images, particularly in those captured by autonomous vehicle perception systems is a radial blur. This type of blur arises due to the motion of the imaging system during image acquisition. [364] proposes a DL-based pipeline to estimate and correct the radial blur in order to improve the detection of traffic signboards in images. The proposed pipeline consists of estimation and enhancement modules. The authors designed a convolutional neural network (CNN) named CuratorNet, to estimate the point spread function (PSF) of an image with a precision of up to a second decimal point. PSF is a mathematical description of how light gets spread out or blurred across the image due to various factors in the imaging system. By applying the inverse operation of the PSF to the observed image, it can be possible to recover the original scene. To correct the image, the authors propose a convolutional autoencoder-based enhancement module that eliminates the radial blur, based on the obtained PSF estimation.

Feature Cleaning: Summary and Research Opportunities

General existing approaches: We presented approaches to clean tabular, text, and image data. We identified the following feature cleaning approaches for tabular data.

- *Model-based approaches:* Approaches that train an ML model to directly clean data. One may observe the following three subcategories.
 - *Ensemble-based approaches:* An ML model uses the predictions of base data cleaning tools for cleaning. The base data cleaning tools are generally simple.
 - *Transformer-based approaches:* Approaches that use a transformer to clean the features of a record. They rely on the capabilities of pre-trained LLMs.
 - *Autoencoder-based approaches:* Approaches that use autoencoders to clean data. They rely on the denoising capability of auto-encoders.
- *Error prioritization approaches:* Approaches that prioritize the records that should be reviewed by an expert.
- *Data cleaning rule generation approaches:* Approaches that generate data cleaning rules and use them to clean data.

Research opportunities:

- Develop feature cleaning approaches that consider the label of a record when cleaning its features. The approaches covered in our study do not leverage the special status of labels when cleaning a feature (they are not mentioned nor treated differently than other features). Potentially, labels could help clean the features of a record more efficiently than if treated as another feature. Thus, we encourage future works to explore how the records' labels can be used for feature cleaning.
- Extend the advances done with ensemble-based approaches. Future works should experiment with a wider number of base data cleaning tools. For example, other feature cleaning approaches could themselves be included in the ensemble.
- Continue exploring how LLMs can be adapted to data cleaning tasks. We extend on this topic in Section 5.6.4.
- Explore how techniques from outlier detection can be transposed to error detection. Outlier detection and error detection are similar challenges since corrupted records often are outliers. Future studies could leverage the existing literature on outlier detection and ML [365, 366].

5.5.2 Label Cleaning

In this section, we cover data cleaning approaches to detect or correct label errors. For simplicity, we refer to label errors as mislabels. We do not differentiate the label cleaning approaches based on type of the data being cleaned, since the main ideas are generic. That is, any approach can be adapted to other data types if an adequate representation learning technique is used.

In the following, we describe the different categories of label cleaning techniques seen in our review. The first category uses the confidence score of an ML model to detect corrupted samples. The second one adopts a similar approach but uses the loss of a model instead of its confidence scores. The third one detects mislabels by searching for instances that degrade a model's performance. The fourth one uses outlier techniques to find mislabels. We also present label cleaning approaches that did not belong to any of the aforementioned categories at the end of this section.

Uncertainty-Based Approaches

A straightforward approach to label cleaning is to use the confidence score of an ML model to detect corrupted samples. After having been trained on a dataset, one can suppose that an ML model has learned the representative features of each class. Hence, if the model's

predictions disagree with the assigned label for an instance (i.e., the model assigns a low probability for the labeled class), one can assume that the instance has a higher chance of being corrupted compared to other samples. However, the model may predict something different than a sample’s label because of corrupted features, mislabels, or more simply because the model made a mistake. Thus, the predictions of the model can be used as a weak signal for detecting mislabels.

Once a sample is marked as being mislabeled, it must be cleaned. The cleaning strategies sometimes assume that someone will manually clean the flagged instances. When they do, either all the instances are sent for review [367,368] or only the most important ones. In the latter case, one must find criteria to prioritize samples over others. [369] prioritizes records for which the model’s predictions have the highest entropy. Similarly, [370,371] prioritize the records where the model is highly uncertain about the labeled class. In order to avoid uselessly spending humans’ time, [372] proposed to also consider the difficulty of labeling an instance for a human. Thus, the approach considers two things when selecting the next sample to clean: (1) how much the model disagrees with the label and (2) how difficult it is to clean for a human. To measure the latter, the authors considered the entropy of a model’s predictions (the higher the entropy, the more difficult labeling). If different people cleaned a record differently, then we can suppose that it is difficult to clean, and we should avoid sending it again for cleaning. Other uncertainty-based approaches [288,292,373,374] do not assume the presence of a human expert for cleaning instances; hence, they must automatically handle the cleaned instances. The straightforward strategy is to drop all the flagged instances [292,373]. As a result, the dataset is purged from every potentially corrupted instance. Instead of removing all potentially corrupted instances, [374] iteratively removes the instances with the lowest confidence and retrains the model on the cleaned dataset until the model’s confidence for all instances is above a threshold. [288] builds a class-conditional noise matrix and suggests different methods for record pruning using that matrix. This approach uses the confidence of the model to estimate class-conditional noise. An instance might belong to another class if the model confidence for that other class is above the per-class threshold. The per-class threshold of a class j is the model’s average confidence over the samples labeled j for the same class j . Having a threshold defined for every class allows the approach to be more robust to heterogeneous class probability distributions and class imbalance. When an instance is above the threshold for more than one class, the approach selects the class the model is the most certain about. The authors propose five methods to prune out records while considering class-conditional noise. One of them prunes out the records that are predicted to belong to another class (when using the per-class thresholds).

For the uncertainty-based approaches to be successful, two things must be taken into con-

sideration. First, the model should not overfit the dataset, or else the mislabeled instances will be correctly predicted (with respect to the labeled class, not the true class) and mislabeled instances will have a high confidence score. Second, an accurate measure of the model confidence must be used. Prior work [375,376] has shown that the softmax score should not be used to measure the model’s confidence, since it only acts as a normalization factor.

The first problem can be addressed using techniques that are robust to label noise, for example, [372] used Co-Teaching [377] and BYOL [378]. Another solution to prevent the ML model from predicting the mislabeled class of an instance is to stop the training process before the model overfits the data. [379] made the observation that the mislabeled records are learned later in the model’s learning process compared to the correctly labeled ones, and, as a result, stopping the training early would enable the model to have low confidence on mislabeled instances. Leveraging that observation, [292] proposed an approach to automatically stop the training process. After each epoch, the model’s confidence for each instance’s labeled class is recorded and clustered using the k-means algorithm with a k value of 2. Hence, after each epoch, a cluster with the most certain records and another with the least certain records are generated. When less than 1% of the records change the cluster from one epoch to another, the model stops training.

The second problem can be addressed using modeling techniques able to provide uncertainty scores, such as Bayesian Neural Networks (BNN) [292]. The output of these NNs is a posterior distribution over the classes. Therefore, BNNs give a good measure of confidence when adequately trained. They are, however, complicated to implement and hard to train due to the high number of hyperparameters and computational cost [292]. Thus, uncertainty-based approaches use different approximations of BNNs when concerned with providing a good measure of confidence. Inspired by the work of [379], which also leverages Deep Ensemble [380], [292] approximates BNNs with Monte Carlo Dropout (MCD) [381]. MCD provides confidence measures that are numerically tractable and easier to implement than BNNs.

Loss-Based Approaches

Similar to uncertainty-based approaches, loss-based approaches posit that a model trained on a dataset will have learned the representative features of each class and, as a result, will generate predictions that are not aligned with incorrect labels. However, instead of relying on a model’s confidence, loss-based approaches will detect mislabels using the loss of a model. Records for which the model has a high loss are considered more likely to be mislabeled than other records.

The authors in [382,383] first train a model on a dataset, then flag any record with a high

loss value as mislabeled. Flagged records are then relabeled with the model’s predictions. A new model is then trained on the modified dataset, and the process is repeated using the new model and the modified dataset. The authors in [384] made the observation that NNs tend to overfit on noisy examples later in the training process than clean ones. Thus, corrupted examples are identified by looking for examples with a higher loss than average during training. However, the loss values of noisy and clean samples might vary depending on how the NN is initialized, which makes the approach less reliable. To address that concern, model training is repeated several times and the loss values are averaged. The weights of the NN are periodically re-initialized by suddenly increasing the learning rate and taking large gradient steps, which makes the model unlearn. The authors in [385] address the challenge of detecting records not belonging to any of the classes of a dataset (noisy class). The approach adds a new class to the dataset for these samples. Then, a probability of 0.5 is assigned to the labeled class and the noisy class for every sample of the dataset. The approach iteratively trains a model and modifies the labels in such a way that the model’s loss is decreased. Similarly to [384], the authors use a high learning rate to avoid overfitting on noisy labels. Similarly to Picket [269] (see Section 5.5.1), the authors in [386] measure the reconstruction error of autoencoders to detect corrupted samples. Examples with a high reconstruction error are flagged as mislabeled instances. Contrary to Picket [269], [386] trains an autoencoder for each class. To repair a mislabeled instance, they measure the reconstruction error of all the autoencoders for that sample and assign the class associated with the autoencoder with the lowest error. Similarly, the authors in [387] used and compared three ML techniques, namely principal component analysis (PCA), independent component analysis (ICA), and autoencoders for unsupervised label noise detection. With each technique, they constructed a binary classifier that identifies instances with a high reconstruction error as anomalies. Based on their results, the autoencoder-based classifier obtained the best label noise detection score.

Counterfactual Approaches

A reason to do data cleaning is to improve a model’s performance [388]. This implicitly assumes that data errors reduce a model’s performance. Thus, by finding dataset repairs that improve a model’s performance, it is possible to find corrupted instances. In this study, we observed two types of dataset repairs: instance removal (i.e., dropping a record from a dataset) and attribute modification (i.e., modifying a record’s features values, or labels). Counterfactual approaches try to find or repair corrupted records by comparing the performance of a model trained on a dataset with the performance of a model trained on the counterfactual dataset (i.e., the same dataset with some features or labels modified). In the following, we describe how the counterfactual modifications (i.e., instance removal and at-

tribute modification) are used to detect mislabels and how they are implemented. Then, we cover the metrics used to quantify the impact of a counterfactual modification on a model’s performance.

The first category of counterfactual approaches detects mislabels by comparing the performance of a model when trained on a dataset including a sample with the performance of a model when trained on the same dataset without that sample. A naive approach would train the model twice, once with the sample and another time without it. However, this can quickly become intractable as the size of the dataset increases. To address that problem, the authors in [389] used parallel computing. Two other techniques have been proposed to address this problem in other works. The first technique is to use ML algorithms to predict the impact of including a record in the training dataset of a model [390]. In other words, an ML model is in charge of predicting the change in validation loss when a record is added to the training dataset of another ML model. To achieve that, [390] records a model’s validation loss at every step of the training process. Hence, each time a model learns from a sample, the model is evaluated on a validation dataset. The validation loss is averaged per sample and the average becomes each record’s label. Using this new dataset, ML algorithms can be used to predict the change of validation loss for any sample added to a dataset. The number of samples used to train the model predicting deltas in validation loss is a subset of the whole dataset to clean. The second technique to approximate the impact of removing a sample from a model’s training dataset is using influence functions [141,391,392]. Influence functions are a technique from robust statistics that can be used to estimate how much a model’s parameters change in reaction to an infinitesimal change in a training point’s weight in a dataset. The influence function implementation of [141] can estimate the effect of removing an instance from a dataset needing only an oracle access to gradients and Hessian-vector products of the ML algorithms. However, it is theoretically sound only for convex-loss models. To address this limitation, [388] designed a novel estimator that, similar to influence functions, can estimate the change in performance if an instance is removed from the training dataset for models trained with stochastic gradient descent. In an experiment where the instances with the worst influence on a model’s performance are progressively dropped, [388] have shown that the approach improves the model performance more effectively than [141]. Similarly to [388], [391] and [392] provide improvements in terms of efficiency to [141]’s implementation of influence functions, so that it can be used more efficiently for label cleaning.

The second category of counterfactual approaches detects mislabels by comparing the performance of a model when trained on a dataset including a sample with the performance of a model when trained on the same dataset with a modified version of that sample. In addition to detecting potentially corrupted records, this counterfactual approach also provides

repair suggestions (the counterfactual sample). [393,394] formulate the search for mislabeled instances as an optimization problem where the goal is to maximize the model accuracy on an error-free dataset, by correcting the training samples' labels. Both approaches can automatically correct mislabels given an error-free test dataset.

Except for one paper in our selected papers, the counterfactual approaches included in this review evaluated modifications to a dataset using a loss function. However, as pointed out by [395], loss functions are not always a good objective function to optimize for, since influence functions can not find records corrupted by systematic noise. Indeed, removing a record corrupted with systematic noise will not significantly impact the model's performance, since other samples are affected by the same noise. Hence, records corrupted with systematic noise are not influential per se. To address this issue, [395] designed an approach that uses complaints (filed by a human) to find corrupted instances. Complaints can be filed regarding tuples (e.g., "that record should not have this label") or aggregate values (e.g., "there should be a fewer number of records that have a label"). The counterfactual approach then addresses that complaint using influence functions to find instances that, once repaired, will address the complaint.

Outlier-Based Approaches

Outliers are records that are significantly dissimilar from other records, to a point that one can believe that they have not been generated by the same process (as the rest of the data) [396]. Thus, to detect mislabeled instances, outlier-based approaches for label cleaning search for records that differ significantly from the other records from a class. In [397], a record is compared against every other clean record from its class using cosine distance. If the average distance is too far, then the record is marked as dirty. Because this approach does not scale to classes with many records, [398] compares instances against class prototypes; a vector representing all the instances from a class. The class prototype is generated by encoding a subset of the most representative records from a class using self-attention. Self-attention allows the encoder to focus on the most important instances when generating the final embedding. Records whose cosine distance with the class prototype is above a selected threshold are flagged as dirty. The approach iteratively runs the cleaning technique and trains a model on the cleaned dataset. The first layers of the trained NN are used to generate embeddings for the data cleaning steps. Thus, every time the NN improves performance, the data cleaning technique uses improved embeddings. [399] proposes a method for label denoising based on Bayesian aggregation. This approach addresses the limitations of kNN-based approaches, which are not robust to high levels of label noise rate since they only

have a local view of instances. A local view refers to considering the immediate neighbors or similar instances of a data point within a certain vicinity. In contrast, a global view refers to comparing a record to others that are not in its neighbor. The proposed approach combines both views to detect mislabels. The global view of data is obtained by measuring the distance of an instance to its class distribution or by evaluating the value of the probability density function of each class at an instance. The aggregation leads to a robust detection of instances with noisy labels even in the presence of high levels of label noise.

Other Approaches

In this section, we cover other label cleaning approaches that are not included in the previous categories.

- [400]: The authors argue that records near the decision boundary are more likely to be mislabeled by a human because they are more difficult to label. Thus, the authors propose to review support vector examples of an SVM by a human. Because there might be a lot of samples to review, they prioritize the ones that are on the wrong side of the decision boundary of another model trained on the dataset devoid of the support vector records.
- [401]: Instead of detecting and repairing errors, they generate a synthetic dataset devoid of mislabels. To do so, the authors leverage the phenomenon of mode collapse [402] in Generative Adversarial Networks (GANs). They train a GAN to generate realistic images for a class. Because GANs are subject to mode collapse, they expect the synthetic dataset to be free of mislabeled instances, since the GAN will not have memorized the less common instances of a class (i.e., mislabels).
- [403]: Instead of detecting errors and repairing them in two different steps, the authors propose an approach that directly maps an example's label to the correct value. An NN is then trained to predict the correct label given an instance and its label. An identity skip-connection is between the input label and the output of the NN. Thus, its task is to map the input label to the correct value. Similar to [288], the authors argue that label noise is often class-conditional. Thus, learning to map labels from one class to another is sensible.
- [293]: This study evaluates existing methods for detecting annotation errors and inconsistencies in text. Inconsistencies in this context refer to instances that should be labeled in the same manner to indicate the same type but are assigned different labels instead. The authors re-implemented 18 such methods and assessed their performance on several datasets for text classification, token labeling, and span labeling. The ex-

periments indicate that inconsistencies are more challenging to detect than annotation errors.

Label Cleaning: Summary and Research Opportunities

General existing approaches:

- *Uncertainty-based approaches*: Approaches relying on a model’s confidence to detect mislabels. Records for which a trained model predicts a different class than the one that is labeled have a higher chance of being mislabeled than other records.
- *Loss-based approaches*: Approaches relying on a model’s loss to detect mislabels. Similar to uncertainty-based approaches, records that are mislabeled are more likely to be incorrectly predicted by the model than other records (and have a high loss).
- *Counterfactual approaches*: Mislabels may hinder a model’s performance. Thus, if the performance of a model improves after a record has been removed or relabeled, then the record is most likely mislabeled.
- *Outlier-based approaches*: Records that are incorrectly labeled might differ significantly from the other records in the labeled class. These approaches rely on that observation to detect mislabeled.

Research opportunities:

- Develop efficient and accurate methods to estimate the impact of removing a record from a dataset on ML performance. Current solutions may consume a lot of resources [388, 391] or be limited to a specific type of model [141]. Improving these techniques will allow for more performant counterfactual approaches (covered in Section 5.5.2) while enriching the knowledge of ML explainability.
- Combine label cleaning methods using ensembling techniques for improved performance. In Section 5.5.1, we described approaches that combine feature cleaning tools using ensembling techniques. Reusing the same ideas but for label cleaning could be an interesting research direction.
- Develop accurate model confidence measure. As we covered in Section 5.5.2, uncertainty-based approaches rely on a model’s confidence to detect mislabels. However, using a model’s softmax score is not a good measure of model confidence as previous research pointed out [375, 376]. Future works could develop techniques to better measure a model’s confidence.

5.5.3 Entity Matching

In this section, we cover approaches to detect records referring to the same real-world entity. For ML, entity matching can be framed as a binary problem, where the model receives two

records and must predict if they are the same entity or not. Formally, given a pair of records (t_1, t_2) , a model M predicts if the pair of records is a match or not ($y \in [0, 1]$). For conciseness, we use the word “pair” to refer to a pair of records. Furthermore, for simplicity, we call pairs referring to the same real-world identity as positive pairs, and negative pairs if they do not.

Deciding whether records refer to the same real-world entity essentially comes down to comparing records and looking for similarities and differences. Hence, when designing entity-matching approaches, researchers may bias the model into comparing records, which we refer to as the comparison bias. For example, [404] compare records by looking for shared and unique words between two records. The sets of shared and unique words are used by the model to decide whether they refer to the same entity or not. Because the comparison bias plays a central role in the design of the approaches, we grouped the approaches based on the way this bias is injected. Before the record-matching step, pairs that are most certainly not a match are filtered out of the dataset for efficiency purposes. This step is referred to as blocking.

In the following, we explain entity matching and present the approaches included in our study in three parts. We will first cover the blocking strategies. Then we present the different entity-matching approaches using ML. Finally, we show common techniques used to improve a model’s performance on entity-matching tasks (e.g., data augmentation). Note that all the entity-matching approaches in our review operate over tabular data.

Blocking

As mentioned earlier, blocking is applied before entity matching. The goal is to make the whole process of finding matching records more efficient. Instead of comparing each and every instance using the matching model, blocking places records into blocks so that entity matching is only applied to the tuples in the same blocks. This process effectively filters out pairs of records across different blocks, which have a low probability of being a match, making the whole process more efficient.

In the simplest case, hand-crafted rules can be used to filter out pairs of records that most likely are not a match. For example, for an entity matching task of companies’ names, [405] exclude pairs of records that do not share the same zip code or were not one of the top 20 nearest neighbors based on the TF-IDF cosine similarity on the name and address attributes. A downside of that approach is that designing blocking rules requires prior domain knowledge and can be time-consuming. Hence, [406] proposed an approach that is able to filter out pairs of records with minimal human involvement. To do so, Local Sensitive Hashing (LSH) [407] is applied to the distributed representations (i.e., embeddings) of the tuples. Initially designed

to efficiently approximate the k-nearest neighbors problem in sublinear time, LSH is used to filter out improbable matches. It hashes the distributed representations and uses the hash function to place each record into blocks, where each block holds records that are spatially close. Only pairs inside the same block can be considered for entity matching. [406] generate the records' distributed representations by transforming the record's tokens into embeddings using the Glove embedding model [408] and merging them together with a bi-LSTM [409]. The token pre-trained token embeddings are fine-tuned on the domain of the task using vocabulary retrofitting. Similarly, [410] applies LSH on records' embeddings. To the difference of [406], they generate more than one embedding per record. The records having at least one embedding in the same block as another record will be forwarded to the entity-matching model. The other pairs will be filtered out. The embeddings are generated by fusing fastText [411] embeddings using attention mechanisms. Similarly to this work, [412] uses the FAISS library [413] to efficiently compute the k-nearest neighbors and then applies an algorithm to improve recall so that pairs of tuples that are a match are not filtered out. Essentially, the algorithm resizes the k-nearest neighbors set for each record (i.e., adjust "k") so that the likeliness of the k neighbor being a match is significantly more probable than the k+1 neighbor.

Token Comparison

The first category of entity-matching approaches compares records at the feature level. These approaches first extract a set of shared and unique words from the compared records, transform the words into embedding, and feed them to the model. [404] creates, for each attribute, a set of shared and different words. For example, for the values "Montreal, Canada" and "Quebec, Canada", the set of shared words would be "{Canada}" and the set of different words would be "{Quebec, Montreal}". Similarly, [414] creates a set of shared words and two other sets containing the words unique to each record. The words in each set are then transformed into embeddings using a pre-trained language model [404] or word embeddings [414]. Each set of word embeddings is then condensed down to an embedding using summation [414] or attention mechanisms [404, 414]. As a result, an embedding is created per set of words. These embeddings are then concatenated together and fed to the last layers of the model for prediction. In addition to the aforementioned strategies to combine embeddings, [414] proposes to use the embedding resulting from the set of shared words to serve as the "query" key in the attention mechanisms applied to the sets of unique words. Their intuition is that information from the shared words is useful to know the important words in the set of unique words.

Latent Space Comparison

The second type of entity-matching approach compares records in a latent space using a distance measure such as cosine distance before predicting whether a pair of records is a match or not. The records can be compared at the token level, the attribute level, or at the record level. We will refer to the embeddings used for comparison at each of these levels as token embeddings, attribute embeddings, and record embeddings respectively. Effectively, these approaches use one of the aforementioned embeddings for each record, compare the embeddings, aggregate the results of the comparison, and feed it to the entity-matching model.

Token-Level Comparison First, embeddings must be generated for each token. They can be generated from pre-trained embedding models [415–417], transformers’ encoders [412], or any other suitable technique. To generate token embedding, [418] uses graph convolutional neural networks. The structured dataset is transformed into a graph by taking into consideration how frequent a word is for an attribute and the co-occurrence of words inside the same attribute. Once token embeddings are generated, they can be compared against the embeddings of the paired record. [412] compares each token embedding against the most similar token embedding from the other record and generates similarity (with scalar product) and difference vectors (with element-wise difference). Similarly, [415] uses attention mechanisms to find the most relevant token to compare against and performs element-wise difference for comparing. Instead of comparing word embeddings against each other, [418] compares the word embeddings of an instance against a representation of the other record’s attribute. Attention mechanisms are used to generate the attribute representations and comparison is done with an element-wise difference. For every approach, after the token embeddings have been compared, they are condensed together in a unique representation for the record using an NN. The representation of each record is concatenated together and fed to the entity-matching model.

Attribute-Level Comparison When records are compared at the attribute level, embeddings must be generated for each attribute. [406, 416, 419] generate word embeddings for each word inside an attribute, then create an attribute embedding by merging the word embeddings together using variants of recurrent neural networks (e.g., LSTM [409]) or any other method (e.g., summation). [417] use attention mechanisms to strategically combine the records’ token embeddings. [420, 421] use sentence encoders such as the universal sentence encoder [422] to directly output embeddings for attributes. Once the attribute embeddings

are generated for both records of a pair, they can be compared using cosine distance [421], or element-wise difference [406, 416, 419]. For every approach, the result of the comparison for each attribute is concatenated into a vector and fed to the last layers of the model for conducting the prediction task.

Record-Level Comparison [317] generates record embeddings using a transformers' encoder. Tabular data is transformed into text using the serialization method of Ditto [405]. The embeddings are compared using element-wise differences and fed to the model for the final prediction task. Because important information might be lost in the comparison vector (e.g., what the records have in common), they also give to the model the embedding of the concatenated records.

Learned Comparison

The third category groups all entity-matching approaches that do not have a comparison bias and lets the model learn how to compare instances by itself. These approaches generally adopt superior modeling techniques, such as attention. [416] designed a NN with attention mechanisms and trained the model for entity-matching tasks. [423] follows a three-step process to detect records referring to the same real-world entity. First, a model predicts the semantic type (e.g., company name, person name) for each attribute of a record. Second, an entity-matching model trained specifically for the semantic type predicted in the previous step is selected and predicts whether the attribute's value of both records refers to the same real-world entity (e.g., "Bill Gates" versus "William Gates"). The binary predictions (which are generated for each attribute) are fed to a one-layer NN to predict if the records are a match. As discussed in Section 5.5.1, the authors in [268] evaluate LLMs on data cleaning tasks, notably entity-matching. To perform entity-matching, they first transform the tabular records into text format (see Section 5.5.1 for more details on the serialization procedure), show the serialized record to the model, then prompt it to predict whether the records refer to the same real-world entity or not. [405, 424] added a classification layer on top of a pre-trained transformer model and fine-tuned it for the entity matching problem. Similarly to the feature cleaning approaches with transformers (see Section 5.5.1), both approaches transform tabular data into text format by concatenating the values of a record. Additionally, [405] adds special tokens to differentiate the columns' names from the values of a record. For example, for a record of a person living in Montreal, Canada, a serialized entry could look like the following: [ATTRIBUTE NAME] city [ATTRIBUTE VALUE] Montreal [ATTRIBUTE NAME] country [ATTRIBUTE VALUE] Canada. In addition, authors added special tokens

in the string for special types (e.g., phone number, zip code).

Other Approaches

In this section, we cover other entity-matching approaches that are not included in the previous categories.

- [425]: The authors propose an unsupervised approach for entity matching (i.e., no training data is required). Essentially, two Gaussian distributions are fit to the data, one to the positive pairs, and the other, to the negative pairs. The Gaussian distribution can then be used to estimate the likelihood of a pair of a record being a match or not. They propose optimizations to (1) reduce the number of parameters of the Gaussian distributions, (2) improve the prediction accuracy of their approach, and (3) address the performance issues caused by features with zero variance (i.e., the singularity problem in pattern recognition [426]).
- [427]: Instead of predicting if two records refer to the same entity, they predict to which entity each record refers. In other words, each class in their approach corresponds to a real-world entity. To classify records, they transformed each record into a sequence of word embeddings and then fed it to a convolutional NN for classification.

Common Techniques to Improve Performance

In this section, we cover various techniques commonly used to improve the entity-matching approaches' performance. Namely, these are data augmentation, semi-supervised techniques, active learning, and transfer learning.

Data Augmentation Because entity matching is an unbalanced problem (i.e., there are more records that do not refer to the same real-world entity than do), generating negative pairs is fairly easy. Two records randomly selected from a dataset will most likely not refer to the same real-world entity. [317] randomly select two records that are similar but are not a match and assign them the negative label. The authors argue that selecting similar instances is superior to random selection since it makes the prediction problem more difficult for the model. Generating positive pairs is, however, more difficult than generating negative pairs. [317] applies task-specific data transformations that preserve the label of a pair of records. [317] picks any two records whose embeddings have a cosine distance inferior to a fixed threshold. We describe in Section 5.5.3 how the embeddings are generated. Instead of selecting instances, [405, 412] apply transformations to a record that should preserve the record's identity (e.g., deleting an attribute's value). The modified record can replace the

initial record in an already existing pair of records [405], or the modified record and the initial record can become a new pair [412]. To minimize the risk that the pair is not truly a match, [405] proposes MixDA, a technique inspired by MixUp [428]. The approach interpolates the initial record embedding with its transformed version and adds the interpolated version to the training dataset.

Semi-Supervised Techniques To create positive pairs, [419] assigns a positive label to any pair that is rated as very likely to be a positive pair according to the entity-matching model. The model is then re-trained on these newly labeled samples and the process can be repeated. [412] used adversarial learning to generate additional training samples. The goal of the generator model is to fool the discriminator model (i.e., the entity-matching model) into believing that a generated record refers to the same real-world entity as a real record.

Active Learning [419] observed that entity matching is an imbalanced problem (i.e., there exist more negative pairs than positive pairs), and if it is not considered in the active learning process, more negative pairs than positive pairs might be labeled. Hence, the approach tries to send to the oracle an equal amount of positive and negative pairs by referring to the prediction of the entity-matching model, so only uncertain pairs are sent (i.e., when the probability that a pair is a match is close to 0.5). The oracle receives an equal amount of pairs slightly under and over the threshold (of 0.5). [405]’s approach prioritizes samples with the highest informativeness. They start by grouping unlabeled pairs into clusters based on a custom similarity measure. A pair’s informativeness is based on the number of samples that will be labeled if the current pair is labeled positively or negatively. [429] benchmarked different active-learning-based approaches for entity matching. They tried different combinations of models (i.e., linear, non-convex, tree-based, and rule-learning) with different acquisition functions to select pairs (i.e., query-by-committee [353], how close a sample is from the decision boundary, and likely false positives/negatives [430]). They adapt blocking (see Section 5.5.3) to sample selection in active learning to fasten the process.

Transfer Learning Transfer learning refers to reusing a model trained on a different but related task hoping some of the knowledge will be transferable for the task at hand [431]. To have better performances, some entity-matching approaches reuse the models trained for another entity-matching problem for the task at hand. For example, a model used to match citation records between two databases could be used for two other databases with different schema [419]. Here, we discuss the techniques used to make the transfer of knowledge learned from one task to another more efficient. As discussed in Section 5.5.3, [419] predicts if two

records are a match by comparing the records' tuple embedding. In order to maximize the benefits of transfer learning, they trained the model generating tuple embeddings to be dataset-invariant. In other words, the goal is for the embeddings from the target domain to be similar to the ones of the source domain. The authors use adversarial learning to train the embedding model to generate dataset-invariant embeddings. The role of the discriminator model in the adversarial setting is to predict the provenance of a record (i.e., to predict whether a record comes from the source or target domain). Similarly, [404] tries to increase domain adaptation of their approach (i.e., transfer learning) by minimizing the KL divergence between embeddings from the source domain with the ones from the target domain.

Entity Matching: Summary and Research Opportunities

General existing approaches: The approaches can be grouped based on how the records are compared (for the purpose of entity matching). We identified the following three categories.

- *Token comparison:* A set of common and different words between two records are generated and fed to an ML model that will predict whether the records match or not.
- *Latent space comparison:* The records are compared in a latent space using a distance measure such as cosine distance. The distance measure can then be used by an ML model to predict whether the records match or not. The embeddings used for comparison can be token, attribute, or record embeddings.
- *Learned comparison:* A model ingests both records and predicts whether the records match or not. Contrary to other approaches, the model is entirely responsible for learning how to compare records.

Research opportunities:

- Explore how comparing records using different latent representations at the same time can improve entity matching. In Section 5.5.3, we described three types of latent-space representation approaches (token-level, attribute-level, and record-level). Combining all three representation approaches could enable the detection of low-level differences between records as well as high-level ones, which could improve performances.

5.5.4 Outlier Detection

In Section 5.5.2, we presented approaches to detect mislabels using ideas from outlier detection. Here, we cover approaches to detect outliers in a dataset (not for the purpose of detecting mislabels). Borrowed from [34], we identified three categories of outlier detection approaches: statistic-based, distance-based, and model-based approaches. Note that an ap-

proach may belong to more than one category. For example, the approach in [432] uses a model to detect outliers (model-based), but also preprocess data before feeding it to the model using distance-based approaches.

Statistic-Based Approaches

Statistic-based approaches detect outliers by looking for records that are in low-probability regions according to a stochastic model [34]. To detect outliers, the authors in [433] model a dataset using generative models and use them to detect outliers. Any record with a low probability according to the generative models is marked as an outlier. To improve the performance of their approach, the records are enriched with metadata (e.g., the string length of an attribute).

Distance-Based Approaches

Distance-based approaches measure the distance between a record and other records to detect outliers [34]. For example, a record that is far from any other data points could be declared an outlier. [434] compares a record against its neighbors to determine if it is an outlier. If a significant portion of its neighbors come from another class, the record might be an outlier. The authors argue that previous works often incorrectly detect legitimate records from a class with a low number of records as outliers since the records are often in regions with a lot of records from other classes. To mitigate this problem, a weighted average of the labels in the neighborhood of a record is performed to detect outliers and give a larger weight to samples that are close and that are from the minority class. Similarly, [432] uses the neighbor of a sample to determine if it is an outlier. The authors make the observation that outliers often are in clusters of samples with high heterogeneity of classes, or in clusters where a large proportion of the samples have the same class. Thus, to detect outliers, both aspects are measured and fed to a model for prediction.

Model-Based Approaches

Model-based approaches primarily rely on a classifier to detect outliers [34]. Detecting outliers is formulated as a binary prediction task. [435] proposes an approach to detect outliers that is inspired by the GAN architecture [436]. Their approach consists of a discriminator model and k generator models whose goal is to generate synthetic samples. Each generator is trained on a different subset of similar data points to increase the diversity of generated samples. Once trained, the discriminator model can be used to detect outliers. The authors in [437] use

the reconstruction error of an autoencoder (similar to autoencoder-based feature cleaning, see Section 5.5.1) and the distance between a record and the closest class distribution to detect outliers (similar to distance-based approaches, see Section 5.5.4). They conducted experiments with two distance measures, namely Mahalanobis and Euclidean distance. The more a record is poorly reconstructed and far from any class distribution, the more likely it is an outlier.

Other Approaches

In this section, we cover outlier detection approaches that were not included in the previous categories.

- [438]: Several unsupervised detecting approaches are proposed to detect outliers; however, according to the authors they rely predominantly on a single metric (e.g., distance, density, etc.), and each has its shortcomings. In order to address this issue, a voting ensemble-based approach of multiple outlier detection methods relying on different metrics is introduced. To improve performances, non-informative methods are filtered from the ensemble using an unsupervised spectral feature selection algorithm [439].
- [440]: The authors conduct a study in which they compare outlier detection methods. The compared methods are grouped into probabilistic-based (e.g., robust kernel density estimators [441]), neighbors-based (e.g., Subspace outlier detection [442]), distance-based (e.g., Mahalanobis distance [443]), information theory-based (e.g., Kullback-Leibler divergence [444]), NN-based (e.g., Grow When Required (GWR) network [445]), domain-based methods (e.g., One-class SVM [446]) and isolation methods (e.g., Isolation forest [447]). The study shows that isolation forest [447] is an efficient method for detecting outliers in terms of average precision, training and prediction time, and scalability.
- [314]: Similar to [440], the work focuses on evaluating techniques for detecting outliers in time-series data. The evaluated techniques include ML-based techniques (e.g., one-class SVM [446], statistical-based techniques (e.g., z-score [448] and regression-based techniques (e.g., polynomial fit [449]. The experiments show that KNN is the optimal choice in terms of ML performance, simplicity of configuration, and computation time.

Outlier Detection: Summary

General existing approaches:

- *Statistic-based approaches*: Records are marked as outliers if they are in a low proba-

bility region according to a probability model.

- *Distance-based approaches:* The distance between a record and others is measured to determine if the record is an outlier.
- *Model-based approaches:* A model plays a central part in determining if a record is an outlier.

5.5.5 Imputation

In this section, we cover approaches to impute values in datasets with missing values. [450] compares the performance of three ML algorithms (i.e., decision trees, k-nearest neighbor, and Bayesian networks) for imputation on 10 datasets. They found that Bayesian networks are the most accurate for most datasets, but they are expensive to train. [451] proposes two techniques, kEMI and kEMI+ to impute categorical and numerical missing data. Both techniques first utilize the K-Nearest Neighbors (KNN) algorithm as a local search algorithm to find the K-top similar records to a record with missing values. Then kEMI invokes the Expectation-Maximization Imputation (EMI) algorithm which uses the feature correlation among the K-top similar records to impute missing values. The idea is to use conditional probability distribution of a feature given the other features using only the K-top similar records. In contrast, kEMI+ repeatedly invokes EMI to gather a collection of estimates for the missing value and combines them using Dempster-Shafer fusion [452, 453]. Although kEMI+ can be more accurate than kEMI, it is less scalable. A novel technique proposed in [454] that combines Adaptive Resonance Theory (ART) algorithm [455] with Co-active Neuro-Fuzzy Inference System (CANFIS) [456] to impute missing data. CANFIS integrates the reasoning capabilities of fuzzy systems and the computational capabilities of NN such that the fuzzy logic provides a linguistic representation of variables and performs fuzzy inference, while the NN performs computations on the fuzzy rule outputs and maps them to crisp output values. ART complements CANFIS by providing unsupervised learning and clustering of the input data which helps the fuzzy rules construction and enhances the model representation capabilities.

Imputation: Summary

General existing approaches: The approaches covered in our study impute missing data by using feature correlation or by combining NNs with fuzzy systems.

5.5.6 Holistic Data Cleaning

In this section, we present data cleaning approaches that try to clean more than one type of error at a time.

Data Cleaning Pipeline Generation

Instead of directly cleaning data, the approaches in this category try to find the optimal sequence and configuration of data cleaning tools. The process of building the optimal cleaning pipeline can be formulated as a search problem. Given a metric to optimize, the goal is to find the optimal ordering and configuration of cleaning tools. These kinds of approaches are sometimes referred to as a cleaning optimizer [282]. The main factor that differentiates approaches from one another is the objective function and the optimization algorithm used, which we describe below.

Optimally, one would want to decrease the total number of errors in the dataset. However, that information is not always known a priori. Thus, a common approach when cleaning for ML is to use the performance of the model trained on the cleaned dataset as a proxy of data quality [316, 457]. If the ultimate goal of cleaning the dataset is to improve the model’s performance, this objective can be preferred to any other proxy of dataset quality. [458] showed that not all data cleaning tools improve a model’s performance. Thus, cleaning a dataset using ML performance as a metric to maximize can avoid cleaning operations that decrease model performance. Other metrics can be used as well. For example, [457]’s approach can minimize the number of outliers in a dataset or the number of integrity constraints violated.

Technically, ML hyperparameter optimizers, such as Python Hyperopt [459], could be used to configure the data pipelines. However, as [457] pointed out, these optimizers do not leverage the incremental nature of data cleaning. Effectively, instead of evaluating the impact of a data cleaning tool on a model’s ML performance right after it has been applied to a dataset, ML optimizers evaluate complete solutions (i.e., a fully configured data cleaning pipeline). To address that problem, [457] uses beam search to find optimal pipeline configurations. The least promising pipelines are periodically pruned out based on the performance of the model trained on the current dataset. Another strategy proposed by [316] is to use Reinforcement Learning (RL) to find the optimal configuration for data-cleaning pipelines. In their work, an agent iteratively selects a data cleaning tool that will be applied to a dataset. The change in a model’s performance after applying the data cleaning tool serves as the reward function of the agent, and the state is the last cleaning tool applied to the dataset. In addition to the data cleaning tools, pre-processing operations, such as normalization, can also be selected by

the agent.

Instead of searching for new pipeline configurations tailored to a dataset, [460] reuses data cleaning pipelines that were shown to be effective on similar datasets. For this to be effective, a good measure of similarity between datasets must be used. The authors chose to represent datasets by vectors of 22 dimensions composed of common meta-features from the literature, such as mean feature skew. The datasets' vectors are compared against one another using L1 distance. While the results were inconclusive, they argue that their approach could open the path to new ways of generating data-cleaning pipelines.

Other Approach

[458] benchmark different data cleaning techniques on real-world datasets and observe their impact on the performance of ML classification tasks. To perform their experiments, they used data cleaning tools from all the data cleaning categories discussed in this review (i.e., Section 5.5.1 to Section 5.5.5). Using different datasets, the authors measured the impact of using each tool individually on ML performance, then they repeated the experiments with different combinations of tools. Some of the findings of the study are that an improvement to a model's ML performance because of data cleaning will generalize to other models and that no single data cleaning tool is the best for all datasets.

Holistic Data Cleaning: Summary and Research Opportunities

General existing approaches:

- *Data cleaning pipeline generation*: Instead of directly cleaning data, these approaches try to find the best sequence and configuration of data cleaning tools to optimize a metric (e.g. ML accuracy of a model trained on the cleaned dataset).

Research opportunities: We elaborate on future research directions for holistic data cleaning in Section 5.6.5.

5.6 Future Directions

In Section 5.5, we proposed future work directions for each data cleaning activity. In this section, we suggest higher-level future work directions based on the full set of reviewed papers. We provide a simplified description of the research directions proposed in this section in Table 5.2.

Table 5.2 Summary of future work recommendations from Section 5.6.

Section	Research opportunities
5.6.1	Develop approaches to find an optimal parametrization of data augmentation techniques. Develop approaches to automatically filter out invalid augmented samples.
5.6.2	Create datasets repositories for all data cleaning activities (e.g., feature cleaning). Build a taxonomy of data errors. Publish data cleaning datasets that are labeled. Facilitate the process of correcting errors in public ML datasets and keep track of the changes.
5.6.3	When proposing an approach, a researcher should implement its approach in a public data cleaning library. This should be enforced by publishers' reviewers.
5.6.4	Develop new data-cleaning pipeline generation approaches. Develop approaches that can clean more than one type of error at a time. Share LLMs pre-trained on data cleaning tasks.
5.6.5	Explore how LLMs can be leveraged for data cleaning tasks. Explore how tabular data can be transformed into text for data cleaning with LLMs. Develop tools combining data cleaning with other data pre-processing activities. Analyze how different data cleaning activities affect one another when applied to the same dataset.
5.6.6	Develop interactive data cleaning approaches.

5.6.1 Data Augmentation

As mentioned in Section 5.5.1 and Section 5.5.3, datasets of data cleaning activities are often imbalanced. The datasets used for feature cleaning generally only have a few dirty (labeled) records [320, 333]. Similarly, for entity matching, finding pairs of records referring to the same-world entity is more difficult than finding records not referring to the same entity [419]. Hence, it is common for data cleaning techniques to leverage data augmentation to generate more training samples for the class that is underrepresented. While it is true that data augmentation can improve a model’s performance, it can also reduce it, if not properly tuned [461, 462]. For example, a data augmentation operator could transform a record in a way that its initial label does not apply anymore. Thus, when using data augmentation strategies, it is important to ensure they are correctly parameterized. Past studies in computer vision have tried to design search algorithms to automatically find the best parametrization of data augmentation techniques [463, 464]. Future works could continue in this direction and evaluate their approach for data cleaning tasks. While a good parametrization increases the chances that good samples are generated from the data augmentation tool, all samples are not guaranteed to be valid. [461] proposes an approach to automatically filter out augmented samples that are invalid. Additionally, the approach automatically weights examples based on their likeliness to improve a model’s performance. Thus, the model learns more from samples that are weighted more. While not a complete replacement for a good parametrization of data augmentation tools, this strategy makes configuring data augmentation tools less critical. We encourage researchers to develop approaches to automatically filter out augmented records that hinder a model’s performance.

5.6.2 Data Cleaning Datasets

To evaluate a data cleaning approach or train an ML model to be used in a data cleaning approach, one needs a labeled dataset. As building a labeled dataset is a laborious process [465], researchers are interested in using the existing public datasets. Additionally, using public datasets provides a baseline for researchers to compare their approaches. The release of large public datasets has been known to propel research in ML [465]. For instance, the release of ImageNet datasets [466] led to the birth of new neural architectures such as AlexNet [467] and VGG [468]. For data cleaning tasks, there exist popular datasets used across many papers such as the Hospital dataset, the Flights dataset, and the Beers dataset (all of which are described in [325]). However, for researchers, finding and selecting a dataset to evaluate their data cleaning approaches can be a laborious process as it entails searching through many data cleaning papers for datasets. Furthermore, because data cleaning papers sometimes

forget to provide the source of a dataset they used (and instead reference the paper they extracted the dataset from), obtaining the actual dataset can be arduous. To address this problem, researchers have published a dataset repository for entity matching tasks [469]. A similar work could be done for other data cleaning activities, such as feature cleaning.

Providing a data repository for other data cleaning activities would open up other new research directions. For feature cleaning, one could build a taxonomy of errors in ML datasets using the dataset repository. This taxonomy would help researchers evaluate their feature-cleaning approach in a systematic way. Instead of solely selecting datasets based on previous works, the researchers would be able to select datasets that have the error types their approach tries to address.

Finally, in the same effort to provide data-cleaning datasets to researchers, future works could target publishing labeled data-cleaning datasets. To address the lack of labeled datasets, previous works [269, 319, 395] have used datasets with artificial errors (i.e., errors injected into clean data). Thus, in an attempt to enable researchers to evaluate their approach against real-world errors, we encourage future works to publish labeled data-cleaning datasets. Additionally, maintainers of any public ML dataset could facilitate the process of identifying data errors and repairs, so as to organically create labeled data-cleaning datasets.

5.6.3 Tooling

Data cleaning is known to be tedious and time-consuming [54]. Researchers have reported that data cleaning can take up to 80% of data scientists' time [33]. However, as highlighted in [54], data cleaning is one of AI's most under-valued and de-glamorized aspects. To clean data efficiently, practitioners might look for data-cleaning tools. If the practitioner wants the latest advances in data cleaning, he may look for the studies' replication packages. However, not all papers provide a replication package. Out of the 101 papers included in our study, only 20 provided one. Additionally, using replication packages is not convenient for practitioners. They must familiarize themselves with the application programming interfaces (API), which might not be user-friendly and differ from one replication package to another. Additionally, setting up the tool (e.g., installing dependencies) can be tedious and documentation may be lacking. Thus, to address this issue, [470] proposes OpenClean, an open-source library where all data-cleaning approaches can be centralized and made available via a unique API. Similar initiatives have gained a lot of traction in ML, such as Scikit-learn [9]. On the contrary, OpenClean has not received any support apart from its creators two years ago. Thus, a practitioner looking for data-cleaning tools might be deterred from using this open-source library. We encourage future works on data cleaning to provide a ready-to-use implementation

of their approach on open-source libraries such as OpenClean [470]. Additionally, we suggest considering contributions to open-source libraries when evaluating data-cleaning approaches so as to incentivize researchers to contribute.

5.6.4 Large Language Models

We presented in Section 5.5.1 and Section 5.5.3 data cleaning techniques that used LLMs to clean data. These approaches generally obtained state-of-the-art results on their respective tasks. Recently, OpenAI released ChatGPT ¹⁶, an LLM which has shown impressive performances on diverse tasks, from solving programming bugs [471] to passing the Bar exam [472]. We believe that future work should continue proposing data cleaning approaches that leverage LLMs, so as to benefit from the improvements with these models and achieve state-of-the-art performances [268].

As shown in Section 5.5.1 and Section 5.5.3, [268] ask questions in natural language to an LLM in order to clean data. For example, [268] uses the following sentence to clean data: "Is there an error in attribute X" (where X is the name of an attribute), to which the model answers "yes" or "no". Before asking the question, the tabular record is presented to the model in a special format that we describe in Section 5.5.1 and Section 5.5.3. Their approach achieved state-of-the-art performances. We believe that an interesting future research direction would be to explore and compare different ways tabular data can be formatted for data cleaning with LLMs. Similarly, we encourage researchers to explore and compare different ways to prompt an LLM for data-cleaning tasks. This could build upon the current knowledge in prompt engineering [473].

Finally, we encourage future works to train and share LLMs trained on data cleaning tasks. Other researchers and practitioners could use these LLM models for their data-cleaning problems. A pre-trained model could leverage the knowledge it gained from previous data cleaning tasks for any other one. Similar contributions have been made in other fields [474, 475]. For example, in finance, [475] developed FinBert, a language model fine-tuned on finance texts.

5.6.5 Holistic Data Cleaning

Except for the papers in Section 5.5.6, the papers included in our study considered each data cleaning task individually. However, in the real world, practitioners find themselves cleaning more than one type of error in a dataset. For example, a practitioner could have to remove

¹⁶<https://openai.com/blog/chatgpt>

outliers, and then clean the features in a dataset. Cleaning each data error individually could lead to inefficiencies. For example, [458] has shown that cleaning all data error types in a dataset does not necessarily improve a model’s performance. Thus, a careful selection, configuration, and ordering of different data cleaning tools are important for better ML performance. In Section 5.5.6, we presented two approaches to perform holistic data cleaning [316,457]. These approaches search for the best order and configuration of data cleaning tools. Because we have not found so many works on data cleaning pipeline generation (DCPG) approaches, we believe there is space in the literature for more research in that direction. Future works could build upon the work done in AutoML [476], since the latter shares many similarities with DCPG. Indeed, DCPG could be seen as a subset of AutoML, where instead of considering the whole ML pipeline (as with AutoML), the search algorithm only focuses on a specific step in the pipeline (data cleaning).

More broadly, we encourage researchers to develop tools that try to clean more than one type of error in a dataset. A previous study [317] has demonstrated how two different types of errors (i.e., duplicates and feature errors) could be cleaned following a similar strategy. The authors [317] were able to adapt their entity-matching approach for feature cleaning with only minor modifications. Potentially, other data cleaning activities share similarities that are still uncovered. Thus, designing a learning approach that cleans more than one type of error at a time could clean a dataset more effectively than using each type of tool independently.

On an even broader scale, future work may concentrate on developing tools combining data cleaning with other data pre-processing activities. For example, in Section 5.5.6, we presented the work of [316], who combined other data preparation tasks (e.g., feature normalization and selection) with data cleaning techniques (e.g., outlier detection). [477] proposed a framework that combines data cleaning with unfairness mitigation and poisoned sample removal.

To effectively design tools to holistically clean a dataset, one must understand how these tools interact with one another. However, we found little knowledge about that in our review of the literature. As we covered in Section 5.5.6, [458] studied how different types and combinations of data cleaning tools impacted the performance of a model trained on the cleaned dataset. Instead of only considering an ML model’s performance, researchers could study how applying one type of data-cleaning activity affects the performance of another one. For example, it could be interesting to study how feature cleaning influences an entity-matching model. Maybe cleaning data errors would make entity matching easier. Additionally, future works could also replicate [458]’s study but with a larger set of tools, since [458] used only one tool for some of the data cleaning activities that were considered (e.g., label cleaning). This work

could use some of the tools reviewed in this paper.

5.6.6 Interactive Data Cleaning

Most of the papers covered in this SLR (92/101 papers) provide standalone data-cleaning solutions that do not need someone’s efforts to function (except for dataset labeling). In other words, the approach automatically cleans data once sufficient training data is provided. However, in practice, human experts are often involved in the cleaning process, since it is usually impossible for machines to clean datasets flawlessly [34]. We refer to cleaning data using an expert’s feedback as interactive data cleaning. These experts can provide valuable input to the data-cleaning process by solving various kinds of uncertainties. For example, for feature error repair (see Section 5.5.1), experts can validate the repairs proposed by the approach to ensure that no errors are introduced in the dataset. Considering the low number of interactive data-cleaning approaches covered in this SLR and their relevance in practice, we encourage the community to develop such solutions.

5.7 Related Works

Previous works partially cover DC&ML. The authors in [34] explain data cleaning and describe different data cleaning activities (e.g., entity matching, outlier detection). The authors reserved a full chapter to ML-based techniques for data cleaning. However, their work is not as exhaustive (in terms of approaches covered) as the work presented in this paper, and has a broader scope: all of data cleaning (not limited to ML). The authors in [282] present their vision of a holistic data-cleaning framework for ML applications. They also provide an overview of recent DC4ML or ML4DC approaches. Contrary to our work, they only provide an overview of the topic and do not examine the different techniques systematically and in detail as we do in this work. Similarly, [300] review the relationship between data cleaning and ML (i.e., DC4ML and ML4DC) and present current challenges in data cleaning for ML applications. In addition, they describe robust ML approaches. Their work is different from ours because their goal is not to systematically review the literature. The authors in [301] explain how DL techniques can be leveraged for cleaning data and present other studies that leverage DL to clean data. As a vision paper, it proposes some research directions to the community. In [280], authors conducted a survey of data collection for ML. They argue that data collection is composed of different processes, which feature cleaning, label cleaning, and entity matching are part of. Similar to ours, the data cleaning approaches presented are part of ML4DC or DC4ML. Contrary to our work, their scope is broader and they are not performing a systematic literature review. In the extended version of their study [302], in

addition to covering data collection for ML, the authors cover robust model training and fair model training. Similar to the two previous works, a survey on data management for ML is reported in [303] and includes a section to describe data cleaning for ML. [29, 478, 479] are tutorials that briefly discuss how ML can be used for data cleaning.

Other related works focus on one of the data-cleaning activities covered in our review (e.g., feature cleaning, label cleaning, etc.). For example, [480] reviews the literature on entity matching including a section on entity matching using ML. [481] performs a survey on neural networks for entity matching. Authors in [482–484] review techniques to handle labeling errors in ML datasets including label cleaning. Authors in [396, 485] perform a survey of outlier detection techniques and cover ML-based techniques. The works of [486, 487] survey techniques to impute missing data and discuss how ML can be used to impute missing data.

5.8 Threats to Validity

We use the threats to validity described in [276, 488] to analyze the limitations of our works. We detected three types of threats to validity in our work: construct validity, internal validity, and conclusion validity.

Construct validity. First, there is a risk that our search query does not allow us to exhaustively cover the field of interest of this study, preventing us from effectively answering our RQs. As mentioned in Section 5.2.4, we did preliminary searches to build a list of keywords that most likely capture the majority of the papers we are interested in. However, there is a risk that we missed keywords that would have led us to papers relevant to our study. To mitigate that threat, we performed snowballing, as described in Section 5.2.6. The second threat relates to the existence of papers relevant to the study but published after our data collection date, and hence, not included in this paper. Future works should consider replicating our study. There is also a risk that the academic databases used in this study might be incomplete, i.e., missing some papers relevant to our study. To mitigate this threat, we searched on Google Scholar, which allowed us to find papers from other databases (which we had access to but were not on our list). Another threat to construct validity relates to the risk that our inclusion and exclusion criteria might have excluded works relevant to our study. However, similar criteria have been used successfully in previous studies [150]. Also, papers relevant to our study might have been excluded because they processed a different data type than image, text, or tabular data (i.e., the data types considered in our study). However, as mentioned in Section 5.2.2, we selected three of the most common data types. Thus, we believe our scope allowed us to cover the most relevant papers for our study.

Internal validity. There is a risk that papers relevant to our study have been incorrectly excluded from our study while (1) applying inclusion and exclusion criteria (Section 5.2.5) or while (2) applying quality control questions (Section 5.2.5). To avoid excluding relevant papers, in the first filtering step, we included papers in case of doubt. As mentioned in Section 5.2.5, a second researcher reviewed the list to exclude any paper that should not be included. To avoid excluding relevant papers in the second filtering step, a second researcher read every paper excluded because of the quality control questions and verified that the exclusion of a paper is justified, as mentioned in Section 5.2.5. There is also a risk that one of the authors misunderstood the approach proposed in a paper leading to an incorrect explanation of the paper in our review. To mitigate this threat, each reviewer asked other reviewers to read the paper and share their understanding whenever they felt they did not fully understand the approach. Furthermore, a researcher verified that the papers’ descriptions in our review were coherent with the papers’ abstracts.

External validity. External limitations of our study are related to the generalization of the presented results. We attempted to cover different topics in DC4ML and ML4DC as much as possible. Nevertheless, given the scope and selection criteria used in our study, we believe that our findings can be generalized.

Conclusion validity. There is a risk that our study can not be reproduced. To mitigate it, we provide a replication package containing the list of papers retained after each step in the filtering process described in Section 5.2.5, along with their quality ratings (see Section 5.2.5) and extracted data (see Section 5.2.7) when applicable. Any script used to help us conduct the study is also provided in the replication package.

5.9 Conclusion

This study provides a comprehensive systematic review of the literature on DC4ML and ML4DC, i.e., data cleaning with and for ML. Our study includes 101 papers published between 2016 and 2022 inclusively, from various academic databases. We covered 6 different types of data cleaning activities: feature cleaning, label cleaning, entity matching, outlier detection, imputation, and holistic data cleaning. Based on our review, we provide 24 future research directions and we highlight some exciting data-cleaning approaches that can be further extended. Our future research directions revolve around 6 core ideas: (1) to improve data augmentation techniques for data cleaning, (2) to create more public data cleaning datasets, (3) to provide better tooling for cleaning datasets, (4) to study how LLMs can be used for data cleaning, (5) to explore holistic data cleaning approaches, and (6) to develop more interactive data cleaning approaches. We hope that this paper will serve as a solid

foundation for future works on this important topic.

Acknowledgments This work is funded by the Fonds de Recherche du Quebec (FRQ), the Canadian Institute for Advanced Research (CIFAR), and the National Science and Engineering Research Council of Canada (NSERC). We would like to thank Dr. Hyacinth Ali for contributing to improving this SLR with his valuable comments.

Declarations

Data availability

All data generated or analyzed during this study are available in the GitHub repository to help reproduce our results: <https://github.com/poclecoqq/SLR-datacleaning>.

Competing interests

The authors declare that they have no conflict of interest.

CHAPTER 6 CONCLUSION

6.1 Summary of Works

The objective of this thesis is to shed light on the quality issues in MLSSs and their remedies. To achieve this objective, we first conducted an interview study with 42 practitioners in which we investigated the characteristics of real-world quality issues in MLSSs. We extracted a list of 18 quality issues and provided 33 potential remedies to these problems based on the practitioners' experience and our review of the literature. Notably, our results show that practitioners struggle to evaluate the quality of their models and ensure the reliability of their MLSSs. Furthermore, they frequently reported having issues with data quality. Thus, in our second study, we reviewed the literature on data cleaning and machine learning. More specifically, we conducted an SLR on DC4ML and ML4DC. We reviewed 101 papers published between 2016 and 2022 inclusively and identified 6 different data cleaning activities: feature cleaning, label cleaning, entity matching, outlier detection, imputation, and holistic data cleaning. We also provide 24 future research directions, centered around 6 core ideas: (1) to improve data augmentation techniques for data cleaning, (2) to create more public data cleaning datasets, (3) to provide better tooling for cleaning datasets, (4) to study how LLMs can be used for data cleaning, (5) to explore holistic data cleaning approaches, and (6) to develop more interactive data cleaning approaches.

6.2 Limitations

In this section, we describe the limitations of our work with respect to the objectives of this thesis. As a reminder, our first goal is to understand and explain the quality issues of real MLSSs (RO1); our second goal is to describe remedies to quality issues in MLSSs (RO2). In the following, we list limitations with respect to the aforementioned objectives.

Limitations with respect to RO1:

- L1.1 We conducted an interview study with practitioners and stopped conducting interviews once the information collected was redundant. Potentially, we missed relevant information that could have been captured with a larger number of interviews.
- L1.2 Other information could have been missed because of our distribution of participants: our interviewees mainly worked in small and medium-sized companies. However, as explained in Section 3.2, a significant portion of existing studies described the experience of practitioners from large companies.

L1.3 Finally, relevant information could have been missed because of the data collection process selected (interviews). Indeed, as explained in Section 4.6, various threats could lead us to have biased or inaccurate results. For example, preconceptions of the interviewee on quality issues could steer conversations away from relevant findings.

Limitations with respect to RO2:

L2.1 We describe remedies for quality issues in our interview study (Section 4). These solutions are based on the interviewees' answers and our knowledge of the literature and are not exhaustive. To address that problem, we systematically reviewed the literature on a potential remedy for one type of quality issue (data quality). We have not conducted a review of remedies for the other quality issues. As a result, they are not covered exhaustively in our work

L2.2 To find a solution to address the data quality issues we discovered in our first study (Section 4), we chose to review automated data cleaning techniques. Potentially, other strategies can be leveraged to handle these issues, which we have not reviewed. For example, data can be manually cleaned with the help of data visualization tools.

L2.3 We conducted a systematic literature review on DC4ML and ML4DC. This does not cover all data cleaning techniques. These techniques could help solve data quality issues and help achieve the second objective of our work.

L2.4 Potentially, we could have missed relevant data cleaning techniques because we only surveyed the scientific literature. For example, existing data cleaning tools (e.g., Cleanlab¹) could provide effective solutions to address data quality issues, but they are not covered in our review.

L2.5 While we did a preliminary search to find relevant keywords, our search query could miss some important terms, leading us to miss relevant studies.

6.3 Future Research

In this section, we provide future work directions that emerged from considering our work holistically.

- Apart from struggling with data quality, the practitioners we interviewed in our first study (Section 4) also found it difficult to evaluate, debug, and ensure the reliability of their systems. Similar to our work, future works could survey the literature to find remedies to these issues.
- As expressed in limitation *L2.2*, there could exist other remedies to data quality issues

¹<https://cleanlab.ai/>

than data cleaning. Future works could explore different ways that data quality issues can be handled. This knowledge could be gathered from the literature and from interviews. The pros and cons of each approach could be discussed, providing a principled way for practitioners to tackle these issues.

- One surprising observation of our studies is the absence of a standardized library or framework for data cleaning. The interviewees of our first study often described data cleaning as a manual and ad-hoc process, which leads to issues being missed or poorly repaired. Rarely they mentioned using a tool to clean data systematically. Similarly, we have not discovered any publicly available data cleaning library that is actively maintained during our SLR. Future works could explore data cleaning tools by conducting a review of gray and scientific literature [489]. This work could help practitioners select the right tool for their needs. Based on this effort, future work could improve existing data cleaning libraries or develop new ones.
- To the best of our knowledge, our SLR is the first to provide an unified view on DC&ML. We describe and explain many different strategies employed to clean data. An exciting future work direction is to develop new data cleaning techniques based on the newly consolidated knowledge in this review.
- In our SLR on data cleaning and ML, a varying number of papers was reviewed from one data cleaning activity to another. Some data cleaning activities, such as outlier detection or imputation, could have been less covered compared to other data cleaning activities since we only found a few papers (8 and 3 respectively). As expressed with limitation *L2.5*, we potentially missed some keywords in our search query. Thus, future work could extend our work and review the literature on these two topics more thoroughly.

REFERENCES

- [1] C. Cappi *et al.*, “Dataset definition standard (dds),” *arXiv preprint arXiv:2101.03020*, 2021.
- [2] A. Black and P. van Nederpelt, “Dimensions of data quality (ddq),” Sept. 2020. [Online]. Available: <https://www.dama-nl.org/wp-content/uploads/2020/09/DDQ-Dimensions-of-Data-Quality-Research-Paper-version-1.2-d.d.-3-Sept-2020.pdf>
- [3] OpenAI, “Gpt-4 technical report,” 2023.
- [4] J. K. Vincent, “Hope, fear, and ai,” Jun 2023. [Online]. Available: <https://www.theverge.com/c/23753704/ai-chatgpt-data-survey-research>
- [5] T. Krisher, “Us report: Nearly 400 crashes of automated tech vehicles,” Jun 2022. [Online]. Available: <https://apnews.com/article/self-driving-car-crash-data-ae87cadec79966a9ba56e99b4110b8d6>
- [6] J. Dastin, “Amazon scraps secret ai recruiting tool that showed bias against women,” Oct 2018. [Online]. Available: <https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G>
- [7] T. M. Mitchell, *Machine Learning*. Burr Ridge, IL: McGraw Hill, 1997, vol. 45, no. 37.
- [8] A. Vaswani *et al.*, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: <http://arxiv.org/abs/1706.03762>
- [9] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [10] M. Abadi *et al.*, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: <https://www.tensorflow.org/>
- [11] A. Paszke *et al.*, “Automatic differentiation in pytorch,” in *NIPS-W*, 2017.
- [12] D. Sculley *et al.*, “Hidden technical debt in machine learning systems,” *Advances in neural information processing systems*, vol. 28, 2015.
- [13] C. Huyen, *Designing machine learning systems*. " O'Reilly Media, Inc.", 2022.

- [14] A. Mallick *et al.*, “Matchmaker: Data drift mitigation in machine learning for large-scale systems,” in *Proceedings of Machine Learning and Systems*, D. Marculescu, Y. Chi, and C. Wu, Eds., vol. 4, 2022, pp. 77–94. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2022/file/069a002768bcb31509d4901961f23b3c-Paper.pdf
- [15] B. Sahiner *et al.*, “Data drift in medical machine learning: implications and potential remedies,” *The British Journal of Radiology*, p. 20220878, 2023.
- [16] A. L’heureux *et al.*, “Machine learning with big data: Challenges and approaches,” *Ieee Access*, vol. 5, pp. 7776–7797, 2017.
- [17] M. M. Morovati *et al.*, “Bug characterization in machine learning-based systems,” *Empirical Software Engineering*, vol. 29, no. 1, p. 14, 2024.
- [18] “Systems and software engineering vocabulary,” p. 410, 2010.
- [19] H. B. Braiek and F. Khomh, “On testing machine learning programs,” *Journal of Systems and Software*, vol. 164, p. 110542, 2020.
- [20] [Online]. Available: <https://standards.ieee.org/ieee/729/967/#:~:text=Its%20purpose%20is%20to%20identify,the%20vocabulary%20of%20software%20engineering>.
- [21] J. P. Miguel, D. Mauricio, and G. Rodríguez, “A review of software quality models for the evaluation of software products,” *arXiv preprint arXiv:1412.2977*, 2014.
- [22] R. Dwivedi *et al.*, “Explainable ai (xai): Core ideas, techniques, and solutions,” *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–33, 2023.
- [23] N. Mehrabi *et al.*, “A survey on bias and fairness in machine learning,” *ACM computing surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.
- [24] J. Angwin *et al.*, “Machine bias,” May 2016. [Online]. Available: <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>
- [25] M. Shoenybi *et al.*, “Megatron-lm: Training multi-billion parameter language models using model parallelism,” *arXiv preprint arXiv:1909.08053*, 2019.
- [26] M. Labbe, “Energy consumption of ai poses environmental problems: Techtarget,” Aug 2021. [Online]. Available: <https://www.techtarget.com/searchenterpriseai/feature/Energy-consumption-of-AI-poses-environmental-problems>

- [27] B. Jacob *et al.*, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.
- [28] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015.
- [29] X. Chu *et al.*, “Data cleaning: Overview and emerging challenges,” in *Proceedings of the 2016 international conference on management of data*, 2016, pp. 2201–2206.
- [30] E. Rahm, H. H. Do *et al.*, “Data cleaning: Problems and current approaches,” *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3–13, 2000.
- [31] T. L. Fox *et al.*, “Maintaining quality in information systems,” *Journal of Computer Information Systems*, vol. 40, no. 1, pp. 76–80, 1999.
- [32] C. Mayfield, J. Neville, and S. Prabhakar, “Eracer: a database approach for statistical inference and data cleaning,” in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010, pp. 75–86.
- [33] G. Press, “Cleaning big data: Most time-consuming, least enjoyable data science task, survey says,” Oct 2022. [Online]. Available: <https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/>
- [34] I. Ilyas and X. Chu, *Data Cleaning*. Association for Computing Machinery and Morgan & Claypool Publishers, 2019. [Online]. Available: <https://books.google.ca/books?id=RxieDwAAQBAJ>
- [35] H. Villamizar, T. Escovedo, and M. Kalinowski, “Requirements engineering for machine learning: A systematic mapping study,” in *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2021, pp. 29–36.
- [36] A. Vogelsang and M. Borg, “Requirements engineering for machine learning: Perspectives from data scientists,” in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2019, pp. 245–251.
- [37] W. Maalej, Y. D. Pham, and L. Chazette, “Tailoring requirements engineering for responsible ai,” *Computer*, vol. 56, no. 4, pp. 18–27, 2023.

- [38] J. Horkoff, “Non-functional requirements for machine learning: Challenges and new directions,” in *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 2019, pp. 386–391.
- [39] K. Ahmad *et al.*, “What’s up with requirements engineering for artificial intelligence systems?” in *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE, 2021, pp. 1–12.
- [40] J. Siebert *et al.*, “Construction of a quality model for machine learning systems,” *Software Quality Journal*, vol. 30, no. 2, pp. 307–335, 2022.
- [41] X. Wang and M. Miao, “A framework for requirements specification of machine-learning systems,” in *Conference on Software Engineering and Knowledge Engineering (SEKE)*, vol. 143, 2022.
- [42] B. C. Hu *et al.*, “Towards requirements specification for machine-learned perception based on human performance,” in *2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, 2020, pp. 48–51.
- [43] A. Paleyes, R.-G. Urma, and N. D. Lawrence, “Challenges in deploying machine learning: a survey of case studies,” *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–29, 2022.
- [44] G. Giray, “A software engineering perspective on engineering machine learning systems: State of the art and challenges,” *Journal of Systems and Software*, vol. 180, p. 111031, 2021.
- [45] M. Steidl, M. Felderer, and R. Ramler, “The pipeline for the continuous development of artificial intelligence models—current state of research and practice,” *Journal of Systems and Software*, p. 111615, 2023.
- [46] S. Shankar *et al.*, “Operationalizing machine learning: An interview study,” *arXiv preprint arXiv:2209.09125*, 2022.
- [47] M. S. Rahman *et al.*, “Machine learning application development: practitioners’ insights,” *Software Quality Journal*, pp. 1–55, 2023.
- [48] M. A. Al Alamin and G. Uddin, “Quality assurance challenges for machine learning software applications during software development life cycle phases,” in *2021 IEEE International Conference on Autonomous Systems (ICAS)*. IEEE, 2021, pp. 1–5.
- [49] J. Bosch, H. H. Olsson, and I. Crnkovic, “Engineering ai systems: A research agenda,” *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*, pp. 1–19, 2021.

- [50] L. Baier, F. Jöhren, and S. Seebacher, “Challenges in the deployment and operation of machine learning in practice.” in *ECIS*, vol. 1, 2019.
- [51] G. Symeonidis *et al.*, “Mlops-definitions, tools and challenges,” in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022, pp. 0453–0460.
- [52] L. E. Lwakatare, I. Crnkovic, and J. Bosch, “Devops for ai—challenges in development of ai-enabled applications,” in *2020 international conference on software, telecommunications and computer networks (SoftCOM)*. IEEE, 2020, pp. 1–6.
- [53] A. Hopkins and S. Booth, “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*, 2021, pp. 134–145.
- [54] N. Sambasivan *et al.*, ““everyone wants to do the model work, not the data work”: Data cascades in high-stakes ai,” in *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–15.
- [55] A. B. Kolltveit and J. Li, “Operationalizing machine learning models: a systematic literature review,” in *Proceedings of the 1st Workshop on Software Engineering for Responsible AI*, 2022, pp. 1–8.
- [56] J. Klaise *et al.*, “Monitoring and explainability of models in production,” *arXiv preprint arXiv:2007.06299*, 2020.
- [57] K. Shivashankar and A. Martini, “Maintainability challenges in ml: A systematic literature review,” in *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2022, pp. 60–67.
- [58] N. Polyzotis *et al.*, “Data management challenges in production machine learning,” in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1723–1726. [Online]. Available: <https://doi.org/10.1145/3035918.3054782>
- [59] —, “Data lifecycle challenges in production machine learning: a survey,” *ACM SIGMOD Record*, vol. 47, no. 2, pp. 17–28, 2018.
- [60] S. Schelter *et al.*, “On challenges in machine learning model management,” Tech. Rep., 2015.

- [61] P. Avgeriou *et al.*, “Managing technical debt in software engineering (dagstuhl seminar 16162),” in *Dagstuhl reports*, vol. 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
- [62] J. Bogner, R. Verdecchia, and I. Gerostathopoulos, “Characterizing technical debt and antipatterns in ai-based systems: A systematic mapping study,” in *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE, 2021, pp. 64–73.
- [63] D. Sculley *et al.*, “Hidden technical debt in machine learning systems,” in *Advances in Neural Information Processing Systems*, C. Cortes *et al.*, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: <https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcdf2674f757a2463eba-Paper.pdf>
- [64] B. Van Oort *et al.*, “The prevalence of code smells in machine learning projects,” in *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*. IEEE, 2021, pp. 1–8.
- [65] M. Dilhara, A. Ketkar, and D. Dig, “Understanding software-2.0: A study of machine learning library usage and evolution,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 4, pp. 1–42, 2021.
- [66] H. Jebnoun *et al.*, “Clones in deep learning code: What, where, and why?” *Empirical Softw. Engg.*, vol. 27, no. 4, jul 2022. [Online]. Available: <https://doi.org/10.1007/s10664-021-10099-x>
- [67] A. Mailach and N. Siegmund, “Socio-technical anti-patterns in building ml-enabled software,” Tech. Rep., 2023.
- [68] M. Alahdab and G. Çalıkli, “Empirical analysis of hidden technical debt patterns in machine learning software,” in *International Conference on Product-Focused Software Process Improvement*. Springer, 2019, pp. 195–202.
- [69] N. Muralidhar *et al.*, “Using antipatterns to avoid mlops mistakes,” *arXiv preprint arXiv:2107.00079*, 2021.
- [70] H. Washizaki *et al.*, “Studying software engineering patterns for designing machine learning systems,” in *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*. IEEE, 2019, pp. 49–495.
- [71] A. Nikanjam and F. Khomh, “Design smells in deep learning programs: An empirical study,” in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021, pp. 332–342.

- [72] R. C. Martin, “Design principles and design patterns,” *Object Mentor*, vol. 1, no. 34, p. 597, 2000.
- [73] L. Heiland, M. Hauser, and J. Bogner, “Design patterns for ai-based systems: A multi-vocal literature review and pattern repository,” *arXiv preprint arXiv:2303.13173*, 2023.
- [74] H. Washizaki *et al.*, “Machine learning architecture and design patterns,” *IEEE Software*, vol. 8, 2020.
- [75] —, “Software engineering patterns for machine learning applications (sep4mla) part 2,” in *Proceedings of the 27th conference on pattern languages of programs*, 2020, pp. 1–10.
- [76] S. Amershi *et al.*, “Software engineering for machine learning: A case study,” in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 291–300.
- [77] G. Fujii *et al.*, “Guidelines for quality assurance of machine learning-based artificial intelligence,” *International journal of software engineering and knowledge engineering*, vol. 30, no. 11n12, pp. 1589–1606, 2020.
- [78] L. Leite *et al.*, “Practices for managing machine learning products: a multivocal literature review,” 2023.
- [79] E. Breck *et al.*, “The ml test score: A rubric for ml production readiness and technical debt reduction,” in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 1123–1132.
- [80] S. Studer *et al.*, “Towards crisp-ml (q): a machine learning process model with quality assurance methodology,” *Machine Learning and Knowledge Extraction*, vol. 3, no. 2, pp. 392–413, 2021.
- [81] R. Wirth and J. Hipp, “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*, vol. 1. Manchester, 2000, pp. 29–39.
- [82] M. Testi *et al.*, “Mlops: A taxonomy and a methodology,” *IEEE Access*, vol. 10, pp. 63 606–63 618, 2022.
- [83] M. Antonini *et al.*, “Tiny-mlops: A framework for orchestrating ml applications at the far edge of iot systems,” in *2022 IEEE international conference on evolving and adaptive intelligent systems (EAIS)*. IEEE, 2022, pp. 1–8.

- [84] B. Hutchinson *et al.*, “Towards accountability for machine learning datasets: Practices from software engineering and infrastructure,” in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021, pp. 560–575.
- [85] Q. Lu *et al.*, “Responsible ai pattern catalogue: A multivocal literature review,” *arXiv preprint arXiv:2209.04963*, 2022.
- [86] J. Vincent, “Twitter taught microsoft’s ai chatbot to be a racist asshole in less than a day,” Mar 2016. [Online]. Available: <https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist>
- [87] R. Binns, “Fairness in machine learning: Lessons from political philosophy,” in *Conference on fairness, accountability and transparency*. PMLR, 2018, pp. 149–159.
- [88] D. Boyd and K. Crawford, “Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon,” *Information, communication & society*, vol. 15, no. 5, pp. 662–679, 2012.
- [89] R. K. E. Bellamy *et al.*, “AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias,” Oct. 2018. [Online]. Available: <https://arxiv.org/abs/1810.01943>
- [90] H. Weerts *et al.*, “Fairlearn: Assessing and improving fairness of ai systems,” 2023.
- [91] Y. Ahn and Y. Lin, “Fairsight: Visual analytics for fairness in decision making,” *CoRR*, vol. abs/1908.00176, 2019. [Online]. Available: <http://arxiv.org/abs/1908.00176>
- [92] S. Caton and C. Haas, “Fairness in machine learning: A survey,” *ACM Computing Surveys*, 2020.
- [93] D. Pessach and E. Shmueli, “A review on fairness in machine learning,” *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–44, 2022.
- [94] T. Calders and I. Žliobaitė, “Why unbiased computational processes can lead to discriminative decision procedures,” in *Discrimination and Privacy in the Information Society: Data mining and profiling in large databases*. Springer, 2013, pp. 43–57.
- [95] T. Calders and S. Verwer, “Three naive bayes approaches for discrimination-free classification,” *Data mining and knowledge discovery*, vol. 21, pp. 277–292, 2010.
- [96] F. Kamiran and T. Calders, “Data preprocessing techniques for classification without discrimination,” *Knowledge and information systems*, vol. 33, no. 1, pp. 1–33, 2012.

- [97] L. E. Celis *et al.*, “How to be fair and diverse?” *arXiv preprint arXiv:1610.07183*, 2016.
- [98] S. Shekhar *et al.*, “Adaptive sampling for minimax fair classification,” in *Advances in Neural Information Processing Systems*, M. Ranzato *et al.*, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 24 535–24 544. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/cd7c230fc5deb01ff5f7b1be1acef9cf-Paper.pdf
- [99] B. T. Luong, S. Ruggieri, and F. Turini, “k-nn as an implementation of situation testing for discrimination discovery and prevention,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 502–510.
- [100] M. Feldman *et al.*, “Certifying and removing disparate impact,” in *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 259–268.
- [101] P. Gordaliza *et al.*, “Obtaining fairness using optimal transport theory,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2357–2365. [Online]. Available: <https://proceedings.mlr.press/v97/gordaliza19a.html>
- [102] S. Hajian and J. Domingo-Ferrer, “A methodology for direct and indirect discrimination prevention in data mining,” *IEEE transactions on knowledge and data engineering*, vol. 25, no. 7, pp. 1445–1459, 2012.
- [103] J. E. Johndrow and K. Lum, “An algorithm for removing sensitive information,” *The Annals of Applied Statistics*, vol. 13, no. 1, pp. 189–220, 2019.
- [104] R. Zemel *et al.*, “Learning fair representations,” in *International conference on machine learning*. PMLR, 2013, pp. 325–333.
- [105] T. Kamishima *et al.*, “Fairness-aware classifier with prejudice remover regularizer,” in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23*. Springer, 2012, pp. 35–50.
- [106] F. Kamiran, T. Calders, and M. Pechenizkiy, “Discrimination aware decision tree learning,” in *2010 IEEE international conference on data mining*. IEEE, 2010, pp. 869–874.
- [107] B. Woodworth *et al.*, “Learning non-discriminatory predictors,” in *Conference on Learning Theory*. PMLR, 2017, pp. 1920–1953.

- [108] M. B. Zafar *et al.*, “Fairness constraints: Mechanisms for fair classification,” in *Artificial intelligence and statistics*. PMLR, 2017, pp. 962–970.
- [109] Y. Bechavod and K. Ligett, “Penalizing unfairness in binary classification,” *arXiv preprint arXiv:1707.00044*, 2017.
- [110] L. E. Celis and V. Keswani, “Improved adversarial learning for fair classification,” *arXiv preprint arXiv:1901.10443*, 2019.
- [111] B. H. Zhang, B. Lemoine, and M. Mitchell, “Mitigating unwanted biases with adversarial learning,” in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018, pp. 335–340.
- [112] M. Hardt, E. Price, and N. Srebro, “Equality of opportunity in supervised learning,” *Advances in neural information processing systems*, vol. 29, 2016.
- [113] A. K. Menon and R. C. Williamson, “The cost of fairness in binary classification,” in *Conference on Fairness, accountability and transparency*. PMLR, 2018, pp. 107–118.
- [114] S. Corbett-Davies *et al.*, “Algorithmic decision making and the cost of fairness,” in *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*, 2017, pp. 797–806.
- [115] Z. Zhang and D. B. Neill, “Identifying significant predictive bias in classifiers,” *arXiv preprint arXiv:1611.08292*, 2016.
- [116] G. Pleiss *et al.*, “On fairness and calibration,” *Advances in neural information processing systems*, vol. 30, 2017.
- [117] I. Chen, F. D. Johansson, and D. Sontag, “Why is my classifier discriminatory?” *Advances in neural information processing systems*, vol. 31, 2018.
- [118] H. Felzmann *et al.*, “Towards transparency by design for artificial intelligence,” *Science and Engineering Ethics*, vol. 26, no. 6, pp. 3333–3361, 2020.
- [119] [Online]. Available: <https://engineering.linkedin.com/blog/2019/fairness-privacy-transparency-by-design>
- [120] [Online]. Available: <https://eli5.readthedocs.io/en/latest/overview.html>
- [121] Shap, “Shap/shap: A game theoretic approach to explain the output of any machine learning model.” [Online]. Available: <https://github.com/slundberg/shap>

- [122] EthicalML, “Ethicalml/xai: Xai - an explainability toolbox for machine learning.” [Online]. Available: <https://github.com/EthicalML/xai>
- [123] Oracle, “Oracle/free: Free oracle technologies for developers.” [Online]. Available: <https://github.com/oracle/free>
- [124] V. Belle and I. Papantonis, “Principles and practice of explainable machine learning,” *Frontiers in big Data*, p. 39, 2021.
- [125] C. Molnar, G. Casalicchio, and B. Bischl, “Interpretable machine learning—a brief history, state-of-the-art and challenges,” in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2020, pp. 417–431.
- [126] A. B. Arrieta *et al.*, “Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai,” *Information fusion*, vol. 58, pp. 82–115, 2020.
- [127] M. T. Ribeiro, S. Singh, and C. Guestrin, ““ why should i trust you?” explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.
- [128] S. Mishra, B. L. Sturm, and S. Dixon, “Local interpretable model-agnostic explanations for music content analysis.” in *ISMIR*, vol. 53, 2017, pp. 537–543.
- [129] M. T. Ribeiro, S. Singh, and C. Guestrin, “Nothing else matters: Model-agnostic explanations by identifying prediction invariance,” *arXiv preprint arXiv:1611.05817*, 2016.
- [130] G. Su *et al.*, “Interpretable two-level boolean rule learning for classification,” *arXiv preprint arXiv:1511.07361*, 2015.
- [131] U. Johansson, R. König, and L. Niklasson, “The truth is in there-rule extraction from opaque models using genetic programming.” in *FLAIRS conference*, vol. 1. Miami Beach, FL, 2004.
- [132] M. T. Ribeiro, S. Singh, and C. Guestrin, “Anchors: High-precision model-agnostic explanations,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
- [133] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” *Advances in neural information processing systems*, vol. 30, 2017.

- [134] E. Strumbelj and I. Kononenko, “An efficient explanation of individual classifications using game theory,” *The Journal of Machine Learning Research*, vol. 11, pp. 1–18, 2010.
- [135] A. Datta, S. Sen, and Y. Zick, “Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems,” in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 598–617.
- [136] A. Henelius *et al.*, “A peek into the black box: exploring classifiers by randomization,” *Data mining and knowledge discovery*, vol. 28, pp. 1503–1529, 2014.
- [137] P. Cortez and M. J. Embrechts, “Opening black box data mining models using sensitivity analysis,” in *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE, 2011, pp. 341–348.
- [138] —, “Using sensitivity analysis and visualization techniques to open black box data mining models,” *Information Sciences*, vol. 225, pp. 1–17, 2013.
- [139] A. Goldstein *et al.*, “Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation,” *journal of Computational and Graphical Statistics*, vol. 24, no. 1, pp. 44–65, 2015.
- [140] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.
- [141] P. W. Koh and P. Liang, “Understanding black-box predictions via influence functions,” in *International conference on machine learning*. PMLR, 2017, pp. 1885–1894.
- [142] S. Wachter, B. Mittelstadt, and C. Russell, “Counterfactual explanations without opening the black box: Automated decisions and the gdpr,” *Harv. JL & Tech.*, vol. 31, p. 841, 2017.
- [143] J. M. Zhang *et al.*, “Machine learning testing: Survey, landscapes and horizons,” *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 1–36, 2020.
- [144] J. Rauber, W. Brendel, and M. Bethge, “Foolbox: A python toolbox to benchmark the robustness of machine learning models,” in *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. [Online]. Available: <http://arxiv.org/abs/1707.04131>
- [145] M.-I. Nicolae *et al.*, “Adversarial robustness toolbox v1.0.0,” 2019.

- [146] X. Liu *et al.*, “Privacy and security issues in deep learning: A survey,” *IEEE Access*, vol. 9, pp. 4566–4593, 2020.
- [147] S. H. Silva and P. Najafirad, “Opportunities and challenges in deep learning adversarial robustness: A survey,” *arXiv preprint arXiv:2007.00753*, 2020.
- [148] N. Akhtar and A. Mian, “Threat of adversarial attacks on deep learning in computer vision: A survey,” *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018.
- [149] X. Yuan *et al.*, “Adversarial examples: Attacks and defenses for deep learning,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.
- [150] F. Tambon *et al.*, “How to certify machine learning based safety-critical systems? a systematic literature review,” *Automated Software Engineering*, vol. 29, no. 2, pp. 1–74, 2022.
- [151] K. Eykholt *et al.*, “Robust physical-world attacks on deep learning visual classification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1625–1634.
- [152] S. Thys, W. Van Ranst, and T. Goedemé, “Fooling automated surveillance cameras: adversarial patches to attack person detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019, pp. 0–0.
- [153] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.
- [154] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” *arXiv preprint arXiv:1611.01236*, 2016.
- [155] A. Madry *et al.*, “Towards deep learning models resistant to adversarial attacks,” *arXiv preprint arXiv:1706.06083*, 2017.
- [156] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: a simple and accurate method to fool deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.
- [157] N. Papernot *et al.*, “The limitations of deep learning in adversarial settings,” in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

- [158] S.-M. Moosavi-Dezfooli *et al.*, “Universal adversarial perturbations,” 2017.
- [159] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.
- [160] A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” in *International conference on machine learning*. PMLR, 2018, pp. 274–283.
- [161] A. Athalye *et al.*, “Synthesizing robust adversarial examples,” in *International conference on machine learning*. PMLR, 2018, pp. 284–293.
- [162] P.-Y. Chen *et al.*, “Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models,” in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 15–26.
- [163] C.-C. Tu *et al.*, “Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 742–749.
- [164] W. Brendel, J. Rauber, and M. Bethge, “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models,” *arXiv preprint arXiv:1712.04248*, 2017.
- [165] T. Brunner *et al.*, “Guessing smart: Biased sampling for efficient black-box adversarial attacks,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4958–4966.
- [166] J. Chen, M. I. Jordan, and M. J. Wainwright, “Hopskipjumpattack: A query-efficient decision-based attack,” in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1277–1294.
- [167] Q. Wang *et al.*, “Adversary resistant deep neural networks with an application to malware detection,” in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1145–1153.
- [168] A. Prakash *et al.*, “Deflecting adversarial attacks with pixel deflection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8571–8580.

- [169] E. Raff *et al.*, “Barrage of random transforms for adversarially robust defense,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6528–6537.
- [170] F. Liao *et al.*, “Defense against adversarial attacks using high-level representation guided denoiser,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1778–1787.
- [171] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.
- [172] P. Samangouei, M. Kabkab, and R. Chellappa, “Defense-gan: Protecting classifiers against adversarial attacks using generative models,” *arXiv preprint arXiv:1805.06605*, 2018.
- [173] R. Bao, S. Liang, and Q. Wang, “Featurized bidirectional gan: Adversarial defense via adversarially learned semantic inference,” *arXiv preprint arXiv:1805.07862*, 2018.
- [174] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, “A study of the effect of jpg compression on adversarial images,” *arXiv preprint arXiv:1608.00853*, 2016.
- [175] N. Das *et al.*, “Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression,” *arXiv preprint arXiv:1705.02900*, 2017.
- [176] T.-J. Chang, Y. He, and P. Li, “Efficient two-step adversarial defense for deep neural networks,” *arXiv preprint arXiv:1810.03739*, 2018.
- [177] H. Kannan, A. Kurakin, and I. Goodfellow, “Adversarial logit pairing,” *arXiv preprint arXiv:1803.06373*, 2018.
- [178] C. Xie *et al.*, “Feature denoising for improving adversarial robustness,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 501–509.
- [179] C. Liu and J. JaJa, “Feature prioritization and regularization improve standard accuracy and adversarial robustness,” *arXiv preprint arXiv:1810.02424*, 2018.
- [180] H. Hosseini *et al.*, “Blocking transferability of adversarial examples in black-box learning systems,” *arXiv preprint arXiv:1703.04318*, 2017.
- [181] S. Tian, G. Yang, and Y. Cai, “Detecting adversarial examples through image transformation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

- [182] B. Liang *et al.*, “Detecting adversarial image examples in deep neural networks with adaptive noise reduction,” *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 72–85, 2018.
- [183] S. Chen, N. Carlini, and D. Wagner, “Stateful detection of black-box adversarial attacks,” in *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*, 2020, pp. 30–39.
- [184] D. Gopinath *et al.*, “Deepsafe: A data-driven approach for assessing robustness of neural networks,” in *Automated Technology for Verification and Analysis: 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings 16*. Springer, 2018, pp. 3–19.
- [185] W. Ruan *et al.*, “Global robustness evaluation of deep neural networks with provable guarantees for the l_0 norm,” *arXiv preprint arXiv:1804.05805*, 2018.
- [186] S. S. Banerjee *et al.*, “Towards a bayesian approach for assessing fault tolerance of deep neural networks,” in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks—Supplemental Volume (DSN-S)*. IEEE, 2019, pp. 25–26.
- [187] R. Mangal, A. V. Nori, and A. Orso, “Robustness of neural networks: A probabilistic and practical approach,” in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2019, pp. 93–96.
- [188] D. Gopinath *et al.*, “Symbolic execution for deep neural networks,” *arXiv preprint arXiv:1807.10439*, 2018.
- [189] J. Guo *et al.*, “Dlfuzz: Differential fuzzing testing of deep learning systems,” in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 739–743.
- [190] L. Ma *et al.*, “Combinatorial testing for deep learning systems,” *arXiv preprint arXiv:1806.07723*, 2018.
- [191] S. Jha *et al.*, “Avfi: Fault injection for autonomous vehicles,” in *2018 48th annual ieee/ifip international conference on dependable systems and networks workshops (dsn-w)*. IEEE, 2018, pp. 55–56.

- [192] —, “Kayotee: A fault injection-based system to assess the safety and reliability of autonomous vehicles to faults and errors,” *arXiv preprint arXiv:1907.01024*, 2019.
- [193] —, “ML-based fault injection for autonomous vehicles: A case for bayesian fault injection,” in *2019 49th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2019, pp. 112–124.
- [194] C. E. Tuncali *et al.*, “Simulation-based adversarial test generation for autonomous vehicles with machine learning components,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1555–1562.
- [195] A. Sinha *et al.*, “Certifying some distributional robustness with principled adversarial training,” *arXiv preprint arXiv:1710.10571*, 2017.
- [196] A. Kandel and S. J. Moura, “Safe zero-shot model-based learning and control: A wasserstein distributionally robust approach,” *arXiv preprint arXiv:2004.00759*, 2020.
- [197] J. Liu *et al.*, “Invariant adversarial learning for distributional robustness,” *arXiv preprint arXiv:2006.04414*, 2020.
- [198] M. Everett, B. Lütjens, and J. P. How, “Certifiable robustness to adversarial state uncertainty in deep reinforcement learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4184–4198, 2021.
- [199] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council. [Online]. Available: <https://data.europa.eu/eli/reg/2016/679/oj>
- [200] R. Shokri *et al.*, “Membership inference attacks against machine learning models,” in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.
- [201] Y. Long *et al.*, “Understanding membership inferences on well-generalized learning models,” *arXiv preprint arXiv:1802.04889*, 2018.
- [202] A. Salem *et al.*, “ML-leaks: Model and data independent membership inference attacks and defenses on machine learning models,” *arXiv preprint arXiv:1806.01246*, 2018.
- [203] B. Hitaj, G. Ateniese, and F. Perez-Cruz, “Deep models under the gan: information leakage from collaborative deep learning,” in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

- [204] F. Tramèr *et al.*, “Stealing machine learning models via prediction {APIs},” in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 601–618.
- [205] B. Wang and N. Z. Gong, “Stealing hyperparameters in machine learning,” in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 36–52.
- [206] W. Knight, “Openai’s ceo says the age of giant ai models is already over,” Apr 2023. [Online]. Available: <https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/>
- [207] H. C. Tanuwidjaja, R. Choi, and K. Kim, “A survey on deep learning techniques for privacy-preserving,” in *Proc. Int. Conf. Mach. Learn. Cyber Secur.* Cham, Switzerland: Springer, 2019, pp. 29–46.
- [208] A. Boulemtafes, A. Derhab, and Y. Challal, “A review of privacy-preserving techniques for deep learning,” *Neurocomputing*, vol. 384, pp. 21–45, 2020.
- [209] M. Rigaki and S. Garcia, “A survey of privacy attacks in machine learning,” *ACM Computing Surveys*, 2020.
- [210] H. Yu *et al.*, “Cloudleak: Large-scale deep learning models stealing through adversarial examples.” in *NDSS*, 2020.
- [211] S. Pal *et al.*, “Activethief: Model extraction using active learning and unannotated public data,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 865–872.
- [212] V. Chandrasekaran *et al.*, “Exploring connections between active learning and model extraction,” in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1309–1326.
- [213] Y. Uchida *et al.*, “Embedding watermarks into deep neural networks,” in *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, 2017, pp. 269–277.
- [214] T. Wang and F. Kerschbaum, “Attacks on digital watermarks for deep neural networks,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2622–2626.
- [215] D. Hitaj and L. V. Mancini, “Have you stolen my model? evasion attacks against deep neural network watermarking techniques,” *arXiv preprint arXiv:1809.00615*, 2018.

- [216] X. Gong *et al.*, “Inversenet: Augmenting model extraction attacks with training data inversion.” in *IJCAI*, 2021, pp. 2439–2447.
- [217] Y. Chen *et al.*, “D-dae: Defense-penetrating model extraction attacks,” in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 382–399.
- [218] T. Lee *et al.*, “Defending against machine learning model stealing attacks using deceptive perturbations,” *arXiv preprint arXiv:1806.00054*, 2018.
- [219] M. Juuti *et al.*, “Prada: protecting against dnn model stealing attacks,” in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 512–527.
- [220] J. Zhang *et al.*, “Protecting intellectual property of deep neural networks with watermarking,” in *Proceedings of the 2018 on Asia conference on computer and communications security*, 2018, pp. 159–172.
- [221] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.
- [222] B. Jayaraman and D. Evans, “Evaluating differentially private machine learning in practice,” in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1895–1912.
- [223] Y. He *et al.*, “Segmentations-leak: Membership inference attacks and defenses in semantic image segmentation,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer, 2020, pp. 519–535.
- [224] M. A. Rahman *et al.*, “Membership inference attack against differentially private deep learning model.” *Trans. Data Priv.*, vol. 11, no. 1, pp. 61–79, 2018.
- [225] M. Fredrikson *et al.*, “Privacy in pharmacogenetics: An {End-to-End} case study of personalized warfarin dosing,” in *23rd USENIX security symposium (USENIX Security 14)*, 2014, pp. 17–32.
- [226] S. Hidano *et al.*, “Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes,” in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2017, pp. 115–11509.

- [227] Z. Yang *et al.*, “Neural network inversion in adversarial setting via background knowledge alignment,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 225–240.
- [228] Y. Zhang *et al.*, “The secret revealer: Generative model-inversion attacks against deep neural networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 253–261.
- [229] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.
- [230] C. Dwork, “Differential privacy: A survey of results,” in *International conference on theory and applications of models of computation*. Springer, 2008, pp. 1–19.
- [231] M. Abadi *et al.*, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2016. [Online]. Available: <https://doi.org/10.1145%2F2976749.2978318>
- [232] L. Mescheder, S. Nowozin, and A. Geiger, “Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks,” in *International conference on machine learning*. PMLR, 2017, pp. 2391–2400.
- [233] G. Acs *et al.*, “Differentially private mixture of generative neural networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 6, pp. 1109–1121, 2018.
- [234] N. Phan *et al.*, “Adaptive laplace mechanism: Differential privacy preservation in deep learning,” in *2017 IEEE international conference on data mining (ICDM)*. IEEE, 2017, pp. 385–394.
- [235] —, “Differential privacy preservation for deep auto-encoders: an application of human behavior prediction,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.
- [236] N. Phan, X. Wu, and D. Dou, “Preserving differential privacy in convolutional deep belief networks,” *Machine learning*, vol. 106, no. 9-10, pp. 1681–1704, 2017.
- [237] N. Papernot *et al.*, “Semi-supervised knowledge transfer for deep learning from private training data,” *arXiv preprint arXiv:1610.05755*, 2016.
- [238] A. Triastcyn and B. Faltings, “Generating differentially private datasets using gans,” 2018.

- [239] D. Marijan, A. Gotlieb, and M. K. Ahuja, “Challenges of testing machine learning based systems,” in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 2019, pp. 101–102.
- [240] H. Foidl and M. Felderer, “Risk-based data validation in machine learning-based software systems,” in *proceedings of the 3rd ACM SIGSOFT international workshop on machine learning techniques for software quality evaluation*, 2019, pp. 13–18.
- [241] “Tesla behind eight-vehicle crash was in full self-driving mode, says driver,” Dec 2022. [Online]. Available: <https://www.theguardian.com/technology/2022/dec/22/tesla-crash-full-self-driving-mode-san-francisco>
- [242] E. Martinez and L. Kirchner, “The secret bias hidden in mortgage-approval algorithms – the markup,” Aug 2021. [Online]. Available: <https://themarkup.org/denied/2021/08/25/the-secret-bias-hidden-in-mortgage-approval-algorithms>
- [243] J. Angwin *et al.*, “Machine bias,” May 2016. [Online]. Available: <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>
- [244] A. Strauss and J. Corbin, “Grounded theory methodology: An overview.” 1994.
- [245] N. Hynes, D. Sculley, and M. Terry, “The data linter: Lightweight, automated sanity checking for ml data sets,” in *NIPS MLSys Workshop*, vol. 1, 2017.
- [246] E. Breck *et al.*, “Data validation for machine learning.” in *MLSys*, 2019.
- [247] A. Lavin *et al.*, “Technology readiness levels for machine learning systems,” *Nature Communications*, vol. 13, no. 1, p. 6039, 2022.
- [248] L. Berglund *et al.*, “Test maintenance for machine learning systems: A case study in the automotive industry,” Ph.D. dissertation, Master’s thesis, University of Gothenburg, 2022, available from [https ...](https://www.gu.se), 2022.
- [249] N. Nahar *et al.*, “Collaboration challenges in building ml-enabled systems: Communication, documentation, engineering, and process,” *Organization*, vol. 1, no. 2, p. 3, 2022.
- [250] S. E. Whang *et al.*, “Data collection and quality challenges in deep learning: A data-centric ai perspective,” *arXiv preprint arXiv:2112.06409*, 2021.
- [251] D. Sato, A. Wider, and C. Windheuser, “Continuous delivery for machine learning,” 2019. [Online]. Available: <https://martinfowler.com/articles/cd4ml.html>

- [252] S. Azimi and C. Pahl, “Root cause analysis and remediation for quality and value improvement in machine learning driven information models.” in *ICEIS (1)*, 2020, pp. 656–665.
- [253] O. Blais, “Validate and monitor your machine learning models,” 2020. [Online]. Available: https://github.com/moovai/model_validation_tutorial/blob/master/ODSC_East_2020_Validation_Monitoring_Training.pdf
- [254] N. Humbatova *et al.*, “Taxonomy of real faults in deep learning systems,” in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 1110–1121.
- [255] A. Serban and J. Visser, “An empirical study of software architecture for machine learning,” *arXiv preprint arXiv:2105.12422*, 2021.
- [256] A. Strauss, J. Corbin, and J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 1998. [Online]. Available: <https://books.google.ca/books?id=wTwYUnHYsmMC>
- [257] S. Hove and B. Anda, “Experiences from conducting semi-structured interviews in empirical software engineering research,” in *11th IEEE International Software Metrics Symposium (METRICS’05)*, 2005, pp. 10 pp.–23.
- [258] C. B. Seaman, “Qualitative methods in empirical studies of software engineering,” *IEEE Transactions on software engineering*, vol. 25, no. 4, pp. 557–572, 1999.
- [259] D. Luther, “What are business metrics? 35 metrics businesses need to track,” 2022. [Online]. Available: <https://www.netsuite.com/portal/resource/articles/business-strategy/business-metrics.shtml#:~:text=Business%20metrics%20are%20quantifiable%20measures,businesses%2C%20with%20many%20different%20processes>.
- [260] U. Bhatt *et al.*, “Explainable machine learning in deployment,” *CoRR*, vol. abs/1909.06342, 2019. [Online]. Available: <http://arxiv.org/abs/1909.06342>
- [261] L. Merrick and A. Taly, “The explanation game: Explaining machine learning models using shapley values,” in *Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4*. Springer, 2020, pp. 17–38.

- [262] Oxford Languages, 2023. [Online]. Available: <https://languages.oup.com/>
- [263] A. Gholami *et al.*, “A survey of quantization methods for efficient neural network inference,” *arXiv preprint arXiv:2103.13630*, 2021.
- [264] A. G. Howard *et al.*, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [265] P. Tannor, “Data drift vs. concept drift,” Mar 2023. [Online]. Available: <https://deepchecks.com/data-drift-vs-concept-drift-what-are-the-main-differences/>
- [266] D. Hudgeon and R. Nichol, “Machine learning for business: Using amazon sagemaker and jupyter,” 2020. [Online]. Available: <https://aws.amazon.com/sagemaker/data-wrangler/>
- [267] M. Vartak *et al.*, “Seedb: Efficient data-driven visualization recommendations to support visual analytics,” in *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, vol. 8. NIH Public Access, 2015, p. 2182.
- [268] A. Narayan *et al.*, “Can foundation models wrangle your data?” *arXiv preprint arXiv:2205.09911*, 2022.
- [269] Z. Liu, Z. Zhou, and T. Rekatsinas, “Picket: guarding against corrupted data in tabular data during learning and inference,” *The VLDB Journal*, pp. 1–29, 2022.
- [270] A. Doke and M. Gaikwad, “Survey on automated machine learning (automl) and meta learning,” in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2021, pp. 1–5.
- [271] C. Renggli *et al.*, “Ease.ml/snoopy: Towards automatic feasibility studies for ml via quantitative understanding of "data quality for ml",” *arXiv preprint arXiv:2010.08410*, 2022.
- [272] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *arXiv preprint arXiv:1811.10154*, 2018. [Online]. Available: <https://arxiv.org/abs/1811.10154>
- [273] G. Laberge, U. Aïvodji, and S. Hara, “Fooling shap with stealthily biased sampling,” *arXiv preprint arXiv:2205.15419*, 2022.
- [274] A. Chattopadhyay *et al.*, “Neural network attributions: A causal perspective,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings

- of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 981–990. [Online]. Available: <https://proceedings.mlr.press/v97/chattopadhyay19a.html>
- [275] E. K. Rezig *et al.*, “Dagger: a data (not code) debugger,” in *CIDR 2020, 10th Conference on Innovative Data Systems Research, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*, 2020.
- [276] R. Feldt and A. Magazinius, “Validity threats in empirical software engineering research-an initial survey.” in *Seke*, 2010, pp. 374–379.
- [277] OpenAI, March 2023. [Online]. Available: <https://openai.com/research/gpt-4>
- [278] C. Badue *et al.*, “Self-driving cars: A survey,” *Expert Systems with Applications*, vol. 165, p. 113816, 2021.
- [279] A. Gitnux, “Self driving cars safety statistics and trends in 2023 • gitnux,” Mar 2023. [Online]. Available: <https://blog.gitnux.com/self-driving-cars-safety-statistics/>
- [280] Y. Roh, G. Heo, and S. E. Whang, “A survey on data collection for machine learning: a big data-ai integration perspective,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, 2019.
- [281] A. Ng, “A chat with andrew on mlops: From model-centric to data-centric ai,” Mar 2021. [Online]. Available: https://www.youtube.com/watch?v=06-AZXmWHjo&ab_channel=DeepLearningAI
- [282] F. Neutatz *et al.*, “From cleaning before ml to cleaning for ml.” *IEEE Data Eng. Bull.*, vol. 44, no. 1, pp. 24–41, 2021.
- [283] D. Deng *et al.*, “The data civilizer system.” in *Cidr*, 2017.
- [284] A. Agrawal *et al.*, “Cloudy with high chance of dbms: A 10-year prediction for enterprise-grade ml,” *arXiv preprint arXiv:1909.00084*, 2019.
- [285] A. Ng, L. He, and D. Laird, “Data-centric ai competition,” 2021. [Online]. Available: <https://https-deeplearning-ai.github.io/data-centric-comp/>
- [286] M. Mazumder *et al.*, “Dataperf: Benchmarks for data-centric ai development,” *arXiv preprint arXiv:2207.10062*, 2022.
- [287] D. Zha *et al.*, “Data-centric artificial intelligence: A survey,” *arXiv preprint arXiv:2303.10158*, 2023.

- [288] C. G. Northcutt, L. Jiang, and I. L. Chuang, “Confident learning: Estimating uncertainty in dataset labels,” 2019. [Online]. Available: <https://arxiv.org/abs/1911.00068>
- [289] O. Russakovsky *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, pp. 211–252, 2015.
- [290] B. Kitchenham, “Procedures for performing systematic reviews,” *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.
- [291] X. Chu *et al.*, “Data cleaning: Overview and emerging challenges,” in *Proceedings of the 2016 International Conference on Management of Data*, ser. SIGMOD ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 2201–2206. [Online]. Available: <https://doi.org/10.1145/2882903.2912574>
- [292] F. Ponzio *et al.*, “W2wnet: a two-module probabilistic convolutional neural network with embedded data cleansing functionality,” *arXiv preprint arXiv:2103.13107*, 2021.
- [293] J.-C. Klie, B. Webber, and I. Gurevych, “Annotation error detection: Analyzing the past and present for a more coherent future,” *Computational Linguistics*, pp. 1–42, 2022.
- [294] K. Cheng *et al.*, “Pge: Robust product graph embedding learning for error detection,” 2022. [Online]. Available: <https://arxiv.org/abs/2202.09747>
- [295] P. Hwang and Y. Kim, “Data cleaning of sound data with label noise using self organizing map,” in *2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2022, pp. 1–5.
- [296] I. H. Sarker, “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions,” *SN Computer Science*, vol. 2, no. 6, pp. 1–20, 2021.
- [297] S. Pouyanfar *et al.*, “A survey on deep learning: Algorithms, techniques, and applications,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.
- [298] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [299] P.-O. Côté *et al.*, “Quality issues in machine learning software systems,” *Empirical Software Engineering*, [under consideration] 2023.

- [300] I. F. Ilyas and T. Rekatsinas, “Machine learning and data cleaning: Which serves the other?” *J. Data and Information Quality*, vol. 14, no. 3, jul 2022. [Online]. Available: <https://doi.org/10.1145/3506712>
- [301] S. Thirumuruganathan *et al.*, “Data curation with deep learning.” in *EDBT*, 2020, pp. 277–286.
- [302] S. E. Whang *et al.*, “Data collection and quality challenges in deep learning: A data-centric ai perspective,” *The VLDB Journal*, vol. 32, no. 4, pp. 791–813, 2023.
- [303] C. Chai *et al.*, “Data management for machine learning: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4646–4667, 2022.
- [304] T. Rekatsinas *et al.*, “Holoclean: Holistic data repairs with probabilistic inference,” *arXiv preprint arXiv:1702.00820*, 2017.
- [305] S. Krishnan *et al.*, “Boostclean: Automated error detection and repair for machine learning,” *arXiv preprint arXiv:1711.01299*, 2017.
- [306] Wikipedia, Jun 2023. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning
- [307] Z. Kang, C. Catal, and B. Tekinerdogan, “Machine learning applications in production lines: A systematic literature review,” *Computers & Industrial Engineering*, vol. 149, p. 106773, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S036083522030485X>
- [308] M. I. Azeem *et al.*, “Machine learning techniques for code smell detection: A systematic literature review and meta-analysis,” *Information and Software Technology*, vol. 108, pp. 115–138, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950584918302623>
- [309] H. Alsolai and M. Roper, “A systematic literature review of machine learning techniques for software maintainability prediction,” *Information and Software Technology*, vol. 119, p. 106214, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950584919302228>
- [310] C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

- [311] S. A. Cholewiak *et al.*, “SCHOLARLY: Simple access to Google Scholar authors and citation using Python,” 2021. [Online]. Available: <https://github.com/scholarly-python-package/scholarly>
- [312] D. M. Hawkins, *Identification of outliers*. Springer, 1980, vol. 11.
- [313] Wikipedia, Feb 2023. [Online]. Available: [https://en.wikipedia.org/wiki/Imputation_\(statistics\)](https://en.wikipedia.org/wiki/Imputation_(statistics))
- [314] H. Alimohammadi and S. N. Chen, “Performance evaluation of outlier detection techniques in production timeseries: A systematic review and meta-analysis,” *Expert Systems with Applications*, vol. 191, p. 116371, 2022.
- [315] K. Lakshminarayan, S. A. Harp, and T. Samad, “Imputation of missing data in industrial databases,” *Applied intelligence*, vol. 11, no. 3, pp. 259–275, 1999.
- [316] L. Berti-Equille, “Learn2clean: Optimizing the sequence of tasks for web data preparation,” in *The World Wide Web Conference*, 2019, pp. 2580–2586.
- [317] R. Wang, Y. Li, and J. Wang, “Sudowoodo: Contrastive self-supervised learning for multi-purpose data integration and preparation,” *arXiv preprint arXiv:2207.04122*, 2022.
- [318] L. Visengeriyeva and Z. Abedjan, “Metadata-driven error detection,” in *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, 2018, pp. 1–12.
- [319] F. Neutatz, M. Mahdavi, and Z. Abedjan, “Ed2: Two-stage active learning for error detection—technical report,” *arXiv preprint arXiv:1908.06309*, 2019.
- [320] A. Heidari *et al.*, “Holodetect: Few-shot learning for error detection,” in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 829–846.
- [321] M. Pham *et al.*, “Spade: A semi-supervised probabilistic approach for detecting errors in tables.” in *IJCAI*, 2021, pp. 3543–3551.
- [322] M. Ataeyan and N. Daneshpour, “A novel data repairing approach based on constraints and ensemble learning,” *Expert Systems with Applications*, vol. 159, p. 113511, 2020.
- [323] Z. Abedjan *et al.*, “Detecting data errors: Where are we and what needs to be done?” *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 993–1004, 2016.

- [324] L. Visengeriyeva *et al.*, “Improving data quality by leveraging statistical relational learning,” in *ICIQ*, 2016, pp. 220–236.
- [325] M. Mahdavi *et al.*, “Raha: A configuration-free error detection system,” in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 865–882.
- [326] M. Mahdavi and Z. Abedjan, “Baran: Effective error correction via a unified context representation and transfer learning,” *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 1948–1961, 2020.
- [327] —, “Semi-supervised data cleaning with raha and baran.” in *CIDR*, 2021.
- [328] A. Vaswani *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [329] M. Chen *et al.*, “Evaluating large language models trained on code,” *arXiv preprint arXiv:2107.03374*, 2021.
- [330] A. Dosovitskiy *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [331] M. E. Chasmai, “Cubetr: Learning to solve the rubiks cube using transformers,” *arXiv preprint arXiv:2111.06036*, 2021.
- [332] N. Tang *et al.*, “Relational pretrained transformers towards democratizing data preparation [vision],” *arXiv preprint arXiv:2012.02469*, 2020.
- [333] M. Nashaat *et al.*, “Tabreformer: Unsupervised representation learning for erroneous data detection,” *ACM/IMS Transactions on Data Science*, vol. 2, no. 3, pp. 1–29, 2021.
- [334] T. Chen *et al.*, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
- [335] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” *arXiv preprint arXiv:2003.05991*, 2020.
- [336] R. Mauritz *et al.*, “A probabilistic database approach to autoencoder-based data cleaning,” *arXiv preprint arXiv:2106.09764*, 2021.
- [337] F. Tonolini *et al.*, “Tomographic auto-encoder: Unsupervised bayesian recovery of corrupted data,” *arXiv preprint arXiv:2006.16938*, 2020.

- [338] X. Zhang *et al.*, “Deepclean: data cleaning via question asking,” in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2018, pp. 283–292.
- [339] B. Karlaš *et al.*, “Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions,” *arXiv preprint arXiv:2005.05117*, 2020.
- [340] S. Krishnan *et al.*, “Activeclean: Interactive data cleaning for statistical modeling,” *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 948–959, 2016.
- [341] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of databases*. Addison-Wesley Reading, 1995, vol. 8.
- [342] C. Ge *et al.*, “A hybrid data cleaning framework using markov logic networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 5, pp. 2048–2062, 2020.
- [343] Z. Guo and T. Rekatsinas, “Learning functional dependencies with sparse regression,” 2019.
- [344] Wikipedia, “Boosting (machine learning),” Feb 2023. [Online]. Available: [https://en.wikipedia.org/wiki/Boosting_\(machine_learning\)](https://en.wikipedia.org/wiki/Boosting_(machine_learning))
- [345] A. Lew *et al.*, “Pclean: Bayesian data cleaning at scale with domain-specific probabilistic programming,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 1927–1935.
- [346] P. H. Oliveira *et al.*, “Batchwise probabilistic incremental data cleaning,” *arXiv preprint arXiv:2011.04730*, 2020.
- [347] A. Hernández-García and P. König, “Data augmentation instead of explicit regularization,” *arXiv preprint arXiv:1806.03852*, 2018.
- [348] N. S. Tawfik and M. R. Spruit, “Evaluating sentence representations for biomedical text: Methods and experimental results,” *Journal of biomedical informatics*, vol. 104, p. 103396, 2020.
- [349] N. N. Pise and P. Kulkarni, “A survey of semi-supervised learning methods,” in *2008 International conference on computational intelligence and security*, vol. 2. IEEE, 2008, pp. 30–34.
- [350] X. Zhu and Z. Ghahramani, “Learning from labeled and unlabeled data with label propagation,” *ProQuest Number: INFORMATION TO ALL USERS*, 2002.

- [351] C. Aggarwal Charu and K. Reddy Chandan, “Data clustering: algorithms and applications,” 2013.
- [352] Wikipedia, “Active learning (machine learning),” Mar 2023. [Online]. Available: [https://en.wikipedia.org/wiki/Active_learning_\(machine_learning\)](https://en.wikipedia.org/wiki/Active_learning_(machine_learning))
- [353] Y. Freund *et al.*, “Selective sampling using the query by committee algorithm,” *Machine learning*, vol. 28, no. 2-3, p. 133, 1997.
- [354] S. H. Bach *et al.*, “Hinge-loss markov random fields and probabilistic soft logic,” *Journal of Machine Learning Research*, 2017.
- [355] X. Wang and C. Wang, “Time series data cleaning: A survey,” *Ieee Access*, vol. 8, pp. 1866–1881, 2019.
- [356] A. Zhang *et al.*, “Time series data cleaning: From anomaly detection to anomaly repairing (technical report),” *arXiv preprint arXiv:2003.12396*, 2020.
- [357] H. N. Akouemo and R. J. Povinelli, “Data improving in time series using arx and ann models,” *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3352–3359, 2017.
- [358] Y. He *et al.*, “Automatic detection of grammatical errors in english verbs based on rnn algorithm: Auxiliary objectives for neural error detection models,” *Computational Intelligence and Neuroscience*, vol. 2021, 2021.
- [359] K. M. Knill *et al.*, “Automatic grammatical error detection of non-native spoken learner english,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 8127–8131.
- [360] Q. Wang and Y. Tan, “Grammatical error detection with self attention by pairwise training,” in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.
- [361] M. Rei and H. Yannakoudakis, “Compositional sequence labeling models for error detection in learner writing,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1181–1191. [Online]. Available: <https://aclanthology.org/P16-1112>
- [362] E. A. Santos *et al.*, “Finding and correcting syntax errors using recurrent neural networks,” *PeerJ PrePrints*, vol. 5, p. e3123v1, 2017.

- [363] G. P. Spithourakis, I. Augenstein, and S. Riedel, “Numerically grounded language models for semantic error correction,” *arXiv preprint arXiv:1608.04147*, 2016.
- [364] V. Hurakadli *et al.*, “Deep learning based radial blur estimation and image enhancement,” in *2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE, 2019, pp. 1–5.
- [365] G. Pang *et al.*, “Deep learning for anomaly detection: A review,” *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.
- [366] A. B. Nassif *et al.*, “Machine learning for anomaly detection: A systematic review,” *Ieee Access*, vol. 9, pp. 78 658–78 700, 2021.
- [367] M. Rottmann and M. Reese, “Automated detection of label errors in semantic segmentation datasets via deep learning and uncertainty quantification,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3214–3223.
- [368] S. Teso *et al.*, “Interactive label cleaning with example-based explanations,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 966–12 977, 2021.
- [369] I. Rehbein and J. Ruppenhofer, “Detecting annotation noise in automatically labelled data,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1160–1170.
- [370] G. Atkinson and V. Metsis, “Tsar: a time series assisted relabeling tool for reducing label noise,” in *The 14th PErvasive Technologies Related to Assistive Environments Conference*, 2021, pp. 203–209.
- [371] —, “Identifying label noise in time-series datasets,” in *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*, 2020, pp. 238–243.
- [372] M. Bernhardt *et al.*, “Active label cleaning for improved dataset quality under resource constraints,” *Nature communications*, vol. 13, no. 1, p. 1161, 2022.
- [373] D. Liu, Y. Meng, and L. Wang, “Data cleaning of irrelevant images based on transfer learning,” in *2020 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*. IEEE, 2020, pp. 450–456.

- [374] X. Ke *et al.*, “Multi-index dialogue data cleaning model,” in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. IEEE, 2019, pp. 672–676.
- [375] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” 2015. [Online]. Available: <https://arxiv.org/abs/1506.02142>
- [376] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” 2016. [Online]. Available: <https://arxiv.org/abs/1610.02136>
- [377] B. Han *et al.*, “Co-teaching: Robust training of deep neural networks with extremely noisy labels,” 2018. [Online]. Available: <https://arxiv.org/abs/1804.06872>
- [378] J.-B. Grill *et al.*, “Bootstrap your own latent - a new approach to self-supervised learning,” in *Advances in Neural Information Processing Systems*, H. Larochelle *et al.*, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 21 271–21 284. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/file/f3ada80d5c4ee70142b17b8192b2958e-Paper.pdf>
- [379] J. M. Köhler, M. Autenrieth, and W. H. Beluch, “Uncertainty based detection and relabeling of noisy image labels.” in *CVPR Workshops*, 2019, pp. 33–37.
- [380] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” 2016. [Online]. Available: <https://arxiv.org/abs/1612.01474>
- [381] Y. Gal, “Uncertainty in deep learning,” 2016.
- [382] Z. Huang *et al.*, “Misabeled samples adjustment based on self-paced learning framework,” in *2021 7th International Conference on Computer and Communications (ICCC)*. IEEE, 2021, pp. 1659–1659.
- [383] Z. Li, W. Du, and N. Rao, “Research on error label screening method based on convolutional neural network,” in *2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)*. IEEE, 2021, pp. 1020–1024.
- [384] J. Huang *et al.*, “O2u-net: A simple noisy label detection approach for deep neural networks,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3326–3334.

- [385] Q. Yu and K. Aizawa, “Unknown class label cleaning for learning with open-set noisy labels,” in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 1731–1735.
- [386] W. Zhang and X. Tan, “Combining outlier detection and reconstruction error minimization for label noise reduction,” in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2019, pp. 1–4.
- [387] Z. Salekshahrezaee, J. L. Leevy, and T. M. Khoshgoftaar, “A reconstruction error-based framework for label noise detection,” *Journal of Big Data*, vol. 8, pp. 1–16, 2021.
- [388] S. Hara, A. Nitanda, and T. Maehara, “Data cleansing for models trained with sgd,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [389] M. Dolatshah *et al.*, “Cleaning crowdsourced labels using oracles for supervised learning,” *PVLDB*, vol. 12, no. 4, pp. 376–389, 2018.
- [390] L. Smyth, “Training-valuenet: A new approach for label cleaning on weakly-supervised datasets,” *University of Exeter*, 2020.
- [391] K. Suzuki, Y. Kobayashi, and T. Narihira, “Data cleansing for deep neural networks with storage-efficient approximation of influence functions,” *arXiv preprint arXiv:2103.11807*, 2021.
- [392] Y. Wu, J. Weimer, and S. B. Davidson, “Chef: a cheap and fast pipeline for iteratively cleaning label uncertainties (technical report),” *arXiv preprint arXiv:2107.08588*, 2021.
- [393] S. Xiang *et al.*, “Interactive correction of mislabeled training data,” in *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 2019, pp. 57–68.
- [394] X. Zhang, X. Zhu, and S. Wright, “Training set debugging using trusted items,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
- [395] L. Flokas *et al.*, “Complaint-driven training data debugging at interactive speeds,” in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 369–383.
- [396] H. Wang, M. J. Bah, and M. Hammad, “Progress in outlier detection techniques: A survey,” *IEEE Access*, vol. 7, pp. 107 964–108 000, 2019.

- [397] G. Guo, D. Adjeroh, and X. Li, “Automated cleaning of identity label noise in a large-scale face dataset using a face image quality control,” 2018.
- [398] K.-H. Lee *et al.*, “Cleannet: Transfer learning for scalable image classifier training with label noise,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5447–5456.
- [399] P. Bagherzadeh and H. Sadoghi Yazdi, “Label denoising based on bayesian aggregation,” *International Journal of Machine Learning and Cybernetics*, vol. 8, pp. 903–914, 2017.
- [400] R. Ekambaram *et al.*, “Active cleaning of label noise,” *Pattern Recognition*, vol. 51, pp. 463–480, 2016.
- [401] J. Su *et al.*, “Correcting corrupted labels using mode dropping of acgan,” in *2021 15th International Symposium on Medical Information and Communication Technology (ISMICT)*. IEEE, 2021, pp. 98–103.
- [402] “Common problems,” Jul 2022. [Online]. Available: <https://developers.google.com/machine-learning/gan/problems>
- [403] A. Veit *et al.*, “Learning from noisy large-scale datasets with minimal supervision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 839–847.
- [404] D. Jin *et al.*, “Deep transfer learning for multi-source entity linkage via domain adaptation,” *arXiv preprint arXiv:2110.14509*, 2021.
- [405] Y. Li *et al.*, “Deep entity matching with pre-trained language models,” *arXiv preprint arXiv:2004.00584*, 2020.
- [406] M. Ebraheem *et al.*, “Distributed representations of tuples for entity resolution,” *Proceedings of the VLDB Endowment*, vol. 11, no. 11, pp. 1454–1467, 2018.
- [407] P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, ser. STOC ’98. New York, NY, USA: Association for Computing Machinery, 1998, p. 604–613. [Online]. Available: <https://doi.org/10.1145/276698.276876>

- [408] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [409] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [410] W. Zhang *et al.*, “Autoblock: A hands-off blocking framework for entity matching,” in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 744–752.
- [411] P. Bojanowski *et al.*, “Enriching word vectors with subword information,” *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.
- [412] J. Huang *et al.*, “Deep entity matching with adversarial active learning,” *The VLDB Journal*, vol. 32, no. 1, pp. 229–255, 2023.
- [413] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.
- [414] Z. Wang *et al.*, “Cordel: a contrastive deep learning approach for entity linkage,” in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1322–1327.
- [415] C. Fu *et al.*, “Hierarchical matching network for heterogeneous entity resolution,” in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 3665–3671.
- [416] S. Mudgal *et al.*, “Deep learning for entity matching: A design space exploration,” in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 19–34.
- [417] H. Nie *et al.*, “Deep sequence-to-sequence entity matching for heterogeneous entity resolution,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 629–638.
- [418] B. Li *et al.*, “Grapher: token-centric entity resolution with graph convolutional neural networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8172–8179.
- [419] J. Kasai *et al.*, “Low-resource deep entity resolution with transfer and active learning,” *arXiv preprint arXiv:1906.08042*, 2019.

- [420] A. Bogatu *et al.*, “Cost-effective variational active entity resolution,” in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1272–1283.
- [421] H. Lattar, A. B. Salem, and H. H. B. Ghezala, “Does data cleaning improve heart disease prediction?” *Procedia Computer Science*, vol. 176, pp. 1131–1140, 2020.
- [422] D. Cer *et al.*, “Universal sentence encoder,” *arXiv preprint arXiv:1803.11175*, 2018.
- [423] C. Zhao and Y. He, “Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning,” in *The World Wide Web Conference*, 2019, pp. 2413–2424.
- [424] U. Brunner and K. Stockinger, “Entity matching with transformer architectures-a step forward in data integration,” in *23rd International Conference on Extending Database Technology, Copenhagen, 30 March-2 April 2020*. OpenProceedings, 2020.
- [425] R. Wu *et al.*, “Zeroer: Entity resolution using zero labeled examples,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1149–1164.
- [426] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.
- [427] R. D. Gottapu, C. Dagli, and B. Ali, “Entity resolution using convolutional neural network,” *Procedia Computer Science*, vol. 95, pp. 153–158, 2016.
- [428] H. Zhang *et al.*, “mixup: Beyond empirical risk minimization,” *arXiv preprint arXiv:1710.09412*, 2017.
- [429] V. V. Meduri *et al.*, “A comprehensive benchmark framework for active learning methods in entity matching,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1133–1147.
- [430] K. Qian, L. Popa, and P. Sen, “Active learning for large-scale entity resolution,” in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1379–1388.
- [431] Wikipedia, “Transfer learning,” Apr 2023. [Online]. Available: https://en.wikipedia.org/wiki/Transfer_learning

- [432] O. R. Terrades, A. Berenguel, and D. Gil, “A flexible outlier detector based on a topology given by graph communities,” *Big Data Research*, vol. 29, p. 100332, 2022.
- [433] C. Pit-Claudel *et al.*, “Outlier detection in heterogeneous datasets using automatic tuple expansion,” MIT - Computer Science and Artificial Intelligence Laboratory (MIT-CSAIL-TR-2016-002), Tech. Rep., 2016.
- [434] H. Guan *et al.*, “Wenn for individualized cleaning in imbalanced data,” in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 456–461.
- [435] Y. Liu *et al.*, “Generative adversarial active learning for unsupervised outlier detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1517–1528, 2019.
- [436] I. J. Goodfellow *et al.*, “Generative adversarial networks,” 2014.
- [437] W. Jiang *et al.*, “Read: Aggregating reconstruction error into out-of-distribution detection,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, 2023, pp. 14 910–14 918.
- [438] V. Papastefanopoulos, P. Linardatos, and S. Kotsiantis, “Unsupervised outlier detection: a meta-learning algorithm based on feature selection,” *Electronics*, vol. 10, no. 18, p. 2236, 2021.
- [439] Z. Zhao and H. Liu, “Spectral feature selection for supervised and unsupervised learning,” in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 1151–1157.
- [440] R. Domingues *et al.*, “A comparative evaluation of outlier detection algorithms: Experiments and analyses,” *Pattern recognition*, vol. 74, pp. 406–421, 2018.
- [441] J. Kim and C. D. Scott, “Robust kernel density estimation,” *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2529–2565, 2012.
- [442] H.-P. Kriegel *et al.*, “Outlier detection in axis-parallel subspaces of high dimensional data,” in *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13*. Springer, 2009, pp. 831–838.
- [443] I. Ben-Gal, “Outlier detection in: Data mining and knowledge discovery handbook: A complete guide for practitioners and researchers,” 2005.

- [444] M. Filippone and G. Sanguinetti, “Information theoretic novelty detection,” *Pattern Recognition*, vol. 43, no. 3, pp. 805–814, 2010.
- [445] S. Marsland, J. Shapiro, and U. Nehmzow, “A self-organising network that grows when required,” *Neural networks*, vol. 15, no. 8-9, pp. 1041–1058, 2002.
- [446] B. Schölkopf *et al.*, “Support vector method for novelty detection,” *Advances in neural information processing systems*, vol. 12, 1999.
- [447] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.
- [448] B. Rosner, “Percentage points for a generalized esd many-outlier procedure,” *Technometrics*, vol. 25, no. 2, pp. 165–172, 1983.
- [449] H. J. Motulsky and R. E. Brown, “Detecting outliers when fitting data with nonlinear regression—a new method based on robust nonlinear regression and the false discovery rate,” *BMC bioinformatics*, vol. 7, no. 1, pp. 1–20, 2006.
- [450] N. Z. Abidin, A. R. Ismail, and N. A. Emran, “Performance analysis of machine learning algorithms for missing value imputation,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 6, 2018.
- [451] R. Razavi-Far *et al.*, “Similarity-learning information-fusion schemes for missing data imputation,” *Knowledge-Based Systems*, vol. 187, p. 104805, 2020.
- [452] D. AP, “Upper and lower probabilities induced by a multivalued mapping,” *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325–339, 1967.
- [453] A. P. Dempster *et al.*, “Upper and lower probabilities induced by a multivalued mapping,” *Classic works of the Dempster-Shafer theory of belief functions*, vol. 219, no. 2, pp. 57–72, 2008.
- [454] E.-L. Silva-Ramírez and J.-F. Cabrera-Sánchez, “Co-active neuro-fuzzy inference system model as single imputation approach for non-monotone pattern of missing data,” *Neural Computing and Applications*, vol. 33, pp. 8981–9004, 2021.
- [455] G. A. Carpenter, S. Grossberg, and D. B. Rosen, “Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system,” *Neural networks*, vol. 4, no. 6, pp. 759–771, 1991.

- [456] S. S. Tfwala *et al.*, “Prediction of missing flow records using multilayer perceptron and coactive neurofuzzy inference system,” *The Scientific World Journal*, vol. 2013, 2013.
- [457] S. Krishnan and E. Wu, “Alphaclean: Automatic generation of data cleaning pipelines,” *arXiv preprint arXiv:1904.11827*, 2019.
- [458] P. Li *et al.*, “Cleanml: A benchmark for joint data cleaning and machine learning [experiments and analysis],” *arXiv preprint arXiv:1904.09483*, p. 75, 2019.
- [459] J. Bergstra *et al.*, “Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms,” in *Proceedings of the 12th Python in science conference*, vol. 13. Citeseer, 2013, p. 20.
- [460] I. Gemp, G. Theocharous, and M. Ghavamzadeh, “Automated data cleansing through meta-learning,” in *Twenty-Ninth IAAI Conference*, 2017.
- [461] Z. Miao, Y. Li, and X. Wang, “Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond,” in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1303–1316.
- [462] J. Wei and K. Zou, “Eda: Easy data augmentation techniques for boosting performance on text classification tasks,” *arXiv preprint arXiv:1901.11196*, 2019.
- [463] E. D. Cubuk *et al.*, “Autoaugment: Learning augmentation strategies from data,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 113–123.
- [464] S. Lim *et al.*, “Fast autoaugment,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [465] K. Gauen *et al.*, “Comparison of visual datasets for machine learning,” in *2017 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, 2017, pp. 346–355.
- [466] J. Deng *et al.*, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [467] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

- [468] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [469] S. Das *et al.*, “The magellan data repository,” <https://sites.google.com/site/anhaidgroup/projects/data>, 2016.
- [470] H. Müller *et al.*, “From papers to practice: the openclean open-source data cleaning library,” *Proceedings of the VLDB Endowment*, vol. 14, no. 12, pp. 2763–2766, 2021.
- [471] N. M. S. Surameery and M. Y. Shakor, “Use chat gpt to solve programming bugs,” *International Journal of Information Technology & Computer Engineering (IJITC) ISSN: 2455-5290*, vol. 3, no. 01, pp. 17–22, 2023.
- [472] 2023. [Online]. Available: <https://www.cnet.com/tech/chatgpt-can-pass-the-bar-exam-does-that-actually-matter/>
- [473] J. White *et al.*, “A prompt pattern catalog to enhance prompt engineering with chat-gpt,” *arXiv preprint arXiv:2302.11382*, 2023.
- [474] I. Beltagy, K. Lo, and A. Cohan, “Scibert: A pretrained language model for scientific text,” *arXiv preprint arXiv:1903.10676*, 2019.
- [475] D. Araci, “Finbert: Financial sentiment analysis with pre-trained language models,” *arXiv preprint arXiv:1908.10063*, 2019.
- [476] X. He, K. Zhao, and X. Chu, “Automl: A survey of the state-of-the-art,” *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.
- [477] K. H. Tae *et al.*, “Data cleaning for accurate, fair, and robust models: A big data-ai integration approach,” in *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*, 2019, pp. 1–4.
- [478] X. L. Dong and T. Rekatsinas, “Data integration and machine learning: A natural synergy,” in *Proceedings of the 2018 international conference on management of data*, 2018, pp. 1645–1650.
- [479] B.-E. Laure, B. Angela, and M. Tova, “Machine learning to data management: A round trip,” in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1735–1738.
- [480] V. Christophides *et al.*, “An overview of end-to-end entity resolution for big data,” *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–42, 2020.

- [481] N. Barlaug and J. A. Gulla, “Neural networks for entity matching: A survey,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 3, pp. 1–37, 2021.
- [482] J. M. Johnson and T. M. Khoshgoftaar, “A survey on classifying big data with label noise,” *ACM Journal of Data and Information Quality*, vol. 14, no. 4, pp. 1–43, 2022.
- [483] W. Feng *et al.*, “A review of addressing class noise problems of remote sensing classification,” *Journal of Systems Engineering and Electronics*, vol. 34, no. 1, pp. 36–46, 2023.
- [484] D. Karimi *et al.*, “Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis,” *Medical image analysis*, vol. 65, p. 101759, 2020.
- [485] A. Boukerche, L. Zheng, and O. Alfandi, “Outlier detection: Methods, models, and classification,” *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–37, 2020.
- [486] W.-C. Lin and C.-F. Tsai, “Missing value imputation: a review and analysis of the literature (2006–2017),” *Artificial Intelligence Review*, vol. 53, pp. 1487–1509, 2020.
- [487] D. Adhikari *et al.*, “A comprehensive survey on imputation of missing data in internet of things,” *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–38, 2022.
- [488] X. Zhou *et al.*, “A map of threats to validity of systematic literature reviews in software engineering,” in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2016, pp. 153–160.
- [489] H. R. Rothstein and S. Hopewell, “Grey literature,” *The handbook of research synthesis and meta-analysis*, vol. 2, pp. 103–125, 2009.

APPENDIX A ANSWERS TO OUR QUESTIONNAIRE

We present in Table A.1 the raw results obtained from our questionnaire. The column N/A (i.e., not answered) refers to the number of times the question has not been answered. We identify the quality issues using the identifiers assigned in Table 4.1.

			Likert Score				
	Id	N/A	1	2	3	4	5
Evaluability	1.1	1	0	3	7	9	1
	1.2	1	0	4	2	9	5
	1.3	2	0	3	3	8	5
	1.4	13	0	1	2	3	2
Explai- nability	2.1	0	2	1	5	6	7
	2.2	0	2	5	7	4	3
Debugga- bility	3.1	2	4	6	4	4	1
	3.2	3	3	4	7	3	1
	3.3	1	0	2	5	4	9
Efficiency	4.1	0	1	6	6	3	5
	4.2	0	3	11	3	2	2
	4.3	1	3	7	6	4	0
Maintain- ability	5.1	0	5	4	4	5	3
	5.2	0	2	7	7	2	3
	5.3	0	3	4	4	8	2
Reliability	6.1	0	0	3	6	9	3
	6.2	0	5	1	2	10	3
	6.3	0	3	4	5	5	4

Table A.1 Answers to our questionnaire

APPENDIX B DATA QUALITY ASPECTS CLASSIFICATION

In this section, we provide the details regarding how the results shown in section 4.4.5 were compiled. More specifically, for each of DDS’ data quality aspects or DAMA’s data quality dimensions, we specify the data quality issues included in the category along with the practitioners that faced these issues.

DDS’ data quality aspects

- **Data representativeness:** DDS [1] defines data representativeness as: “[data that] contains key characteristics in proportions similar to the ones of the population”. We included this category concept and data drift [P3, P4, P8, P9, P15, P20, P25, P30, P32, P36, P43].
- **Data traceability:** DDS [1] defines data representativeness as: “a way to maintain a connection between the different development artifacts, including requirement specifications, dataset, models, etc.”. In this RQ, we are interested only in data. Hence we exclude requirements specifications and models from the data traceability definition. As mentioned in Section 4.4.2, [P8] once could not know the provenance of data and its lineage, which, in turn, made the system difficult to maintain and unreliable.
- **Data accuracy and precision:** DDS [1] defines data accuracy and precision as follows: “Data are accurate if they represent with high fidelity those that will be acquired in operation. Precision refers to the closeness of data related to the same operation situation.” [P7, P8, P13, P14, P15, P20, P25, P31, P32, P33, P35, P38, P43] mentioned facing noisy features once.
- **Data reliability:** DDS [1] defines data reliability as follows: “Data are reliable if they are considered credible and relevant for the operational domain”. [P23] finds it difficult to get relevant data for ML problems. [P38] doubt the credibility of public data sources such as Wikipedia when scraping data. [P13, P15, P34] mentioned dealing with outliers in their datasets.
- **Data consistency:** DDS [1] defines data consistency as follows: “Data consistency is [...] the absence of discrepancy between data concerning the same object (e.g., two different birthdays for the same person).” [P7, P21] faced that issue when having to integrate data coming from different sources. [P35] mentions that data consistency is an issue she recurrently faces.
- **Data integrity:** DDS [1] defines data integrity as follows: “Data integrity is defined

as the assurance of the accuracy and integrity of data over its entire life cycle". In a past project, [P29] could not ensure the integrity of its data because it was not version-controlled and it was shared between team members with a spreadsheet.

- **Labels consistency:** DDS [1] defines label consistency as follows: "Consistency refers to the fact that the same label is assigned to the same object of interest when it is represented several times in the dataset". [P21, P35, P37, P43] mentioned having issues with inconsistent labels.
- **Labels accuracy:** DDS [1] defines label accuracy as follows: "Accuracy of labeling is defined as the correctness of the labeling with respect to the true value (or "ground truth")". [P14, P21, P23, P24, P25, P30, P32, P40, P43] mentioned having issues with noisy labels.

DAMA's data quality dimensions

- **Data completeness:** DAMA defines data completeness (of data values) as follows: "the degree to which all required data values are present". [P2, P3, P4, P6, P8, P12, P13, P16, P19, P27, P30, P31, P32, P34, P35, P43] mentioned having a dataset with missing values.
- **Data validity:** DAMA defines data validity as follows: "the degree to which data values comply with rules". [P1, P3, P6, P25, P31, P38] mentioned having a dataset with values that were not formatted correctly. For example, commas could be used interchangeably with periods to separate decimals from integer numbers in a dataset.
- **Data clarity:** DAMA defines data clarity as follows: "the ease with which data consumers can understand the metadata". As described in Section 4.4.2 [P1, P2, P5, P7, P8, P19, P24, P29, P38] used datasets with missing metadata.
- **Data uniqueness:** DAMA defines data uniqueness as follows: "The degree to which records occur only once in a data file". [P5, P19, P20, P21, P25, P31, P35, P38] mentioned handling duplicates (i.e. records that refer to the same real-world entity).