Rodrigo Galdino Ximenes

Issues that Lead to Code Technical Debt in Machine Learning Systems

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor : Prof. Marcos Kalinowski
Co-advisor: Profª Tatiana Escovedo

Rio de Janeiro
April 2024

Rodrigo Galdino Ximenes

Issues that Lead to Code Technical Debt in Machine Learning Systems

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Rio de Janeiro, April 4th, 2024

Rodrigo Galdino Ximenes

Graduated in Metallurgical Engineering from the Federal University of Rio de Janeiro (UFRJ) in 2013. Completed a specialization course in Systems Analysis at PUC-Rio in 2017.

Bibliographic data

Ximenes, Rodrigo

Issues that Lead to Code Technical Debt in Machine Learning Systems / Rodrigo Galdino Ximenes; advisor: Marcos Kalinowski; co-advisor: Tatiana Escovedo. – 2024.

68 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2024.

Inclui bibliografia

1. Informática – Teses. 2. Dívida Técnica. 3. Aprendizado de Máquina. 4. Grupos Focais. I. Kalinowski, Marcos. II. Escovedo, Tatiana. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

To João Guilherme,
the little hands that brought me here.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

To CNPq and PUC-Rio, for the grants awarded, without which this work could not have been carried out.

To all the professors and staff of the Department of Informatics at PUC-Rio, for their teachings and assistance.

To Petrobras and SERPRO, for providing professionals who contributed to the work.

To my colleagues at the Tecgraf Institute and ExACTa at PUC-Rio, who were great motivators.

To professors Maria Teresa Baldassarre and Rodrigo Oliveira Spinola, who participated in the examining committee.

To Professor Marcos Kalinowski for his generosity in guiding this work and to Professor Tatiana Escovedo, who believed in the master's degree even before being accepted into the program.

To my in-laws, for helping me do what I couldn't.

To my family, for all their love and support.

To my parents, who laid the strong foundation for me to reach this point.

To Renata, this would not have been possible without you.

To João Guilherme, who taught me that there is time for everything.

Abstract

Ximenes, Rodrigo; Kalinowski, Marcos (Advisor); Escovedo, Tatiana (Co-Advisor). **Issues that Lead to Code Technical Debt in Machine Learning Systems**. Rio de Janeiro, 2024. 68p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[Context] Technical debt (TD) in machine learning (ML) systems, much like its counterpart in software engineering (SE), holds the potential to lead to future rework, posing risks to productivity, quality, and team morale. However, better understanding code-related issues leading to TD in ML systems is still a green field. [Objective] This dissertation aims to identify and discuss the relevance of code-related issues leading to TD in ML code throughout the ML life cycle. [Method] Initially, the study generated a list of potential factors that may contribute to accruing TD in ML code. This compilation was achieved by looking at the phases of the ML life cycle along with their usual tasks. Subsequently, the identified issues were refined by evaluating their prevalence and relevance in causing TD in ML code. This refinement process involved soliciting feedback from industry professionals during two focus group sessions. [Results] The study compiled a list of 34 potential issues contributing to TD in the source code of ML systems. Through two focus group sessions with nine participants, this list was refined into 30 issues leading to ML code-related TD, with 24 considered highly relevant. The *data pre-processing* phase was the most critical, with 14 issues considered highly relevant in potentially leading to severe ML code TD. Five issues were considered highly relevant in the *model creation and training* phase and four in the *data collection* phase. The final list of issues is available to the community. [Conclusion] The list can help to raise awareness on issues to be addressed throughout the ML life cycle to minimize accruing TD, helping to improve the maintainability of ML systems.

Keywords

Technical Debt; Machine Learning; Focus Groups.

Resumo

Ximenes, Rodrigo; Kalinowski, Marcos; Escovedo, Tatiana. **Problemas que Levam a Geração de Dívida Técnica de Código em Sistemas de Aprendizado de Máquina**. Rio de Janeiro, 2024. 68p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[Contexto] A dívida técnica (DT) em sistemas de aprendizado de máquina (AM), assim como sua contraparte em engenharia de software (ES), tem o potencial de levar a retrabalhos futuros, representando riscos para produtividade, qualidade e moral da equipe. No entanto, compreender melhor os problemas relacionados ao código que levam à DT em sistemas de AM ainda é um campo em aberto. [Objetivo] Este artigo tem como objetivo identificar e discutir a relevância de problemas que levam a DT no código de AM ao longo do ciclo de vida do AM. [Método] O estudo compilou inicialmente uma lista de problemas potenciais que podem levar à DT no código de AM, analisando as fases do ciclo de vida do AM e suas tarefas típicas. Posteriormente, a lista de problemas foi refinada através da avaliação da prevalência e relevância dos problemas que levam à DT no código de AM por meio de feedback coletado de profissionais da indústria em duas sessões de grupos focais. [Resultados] O estudo compilou uma lista inicial de 34 problemas que potencialmente contribuem para DT em código-fonte de sistemas de AM. Através de duas sessões de grupos focais com nove participantes, esta lista foi refinada para 30 problemas que levam à DT relacionada ao código de AM, sendo 24 considerados altamente relevantes. A fase de *pré-processamento de dados* foi a mais crítica, com 14 problemas considerados altamente relevantes em potencialmente levar a uma DT grave no código de AM. Cinco problemas foram considerados altamente relevantes na fase de *criação e treinamento do modelo* e quatro na fase de *coleta de dados*. A lista final de problemas está disponível para a comunidade. [Conclusão] A lista pode ajudar a aumentar a conscientização sobre os problemas a serem tratados ao longo do ciclo de vida do AM para minimizar a acumulação de DT, ajudando a melhorar a manutenibilidade de sistemas de AM.

Palavras-chave

Dívida Técnica; Aprendizado de Máquina; Grupos Focais.

Table of contents

List of figures

List of tables

List of Abreviations

TD	Technical Debts
ML	Machine Learning
SE	Software Engineering
OO	Object-Oriented
AI	Artificial Intelligence
CRISP-DM	Cross-Industry Standard Process for Data Mining
NLP	Natural Language Processing
PUC-Rio	Pontifical Catholic University of Rio de Janeiro

Barukh ata Adonai Eloheinu melekh ha-olam

1 Introduction

1.1 Context and Motivation

The concept of technical debt (TD), first introduced by Ward Cunningham in 1992 (CUNNINGHAM, 1992), draws an analogy between software engineering (SE) and financial borrowing. This analogy highlights how opting for quicker deliveries in SE can lead to accumulated 'debt' that, similar to its financial counterpart, accrues 'interest' over time and demands eventual repayment. Leveraging TD can accelerate project timelines, especially when focusing on developing features rapidly. However, it is imperative to undertake rigorous management and analysis to repay this debt, thereby minimizing the risks associated with these expedited processes.

The intersection of TD and machine learning (ML) systems has emerged as a noteworthy area of interest, as emphasized by Sculley *et al.* (SCULLEY et al., 2015). Despite this growing attention, the intricacies of identifying and managing TD within the context of ML systems remain relatively unexplored areas warranting further research.

Furthermore, Martinez *et al.* (MARTINEZ; VILES; OLAIZOLA, 2021) identified primary challenges in executing real-world ML projects, including low process maturity, reproducibility issues, absence of validation data, and inadequate quality assurance checks. Addressing these challenges underscores the importance of cohesive coordination among team and project management and efficient data information management to ensure reproducibility, reliability, and achieving desired project outcomes.

In another study, Pimentel *et al.* (PIMENTEL et al., 2019) uncovered sub-optimal practices concerning notebooks, widely used in ML for coding, data visualization, and result presentation. Their findings revealed deficiencies such as a lack of code testing, disorderly cell arrangement, unexecuted code cells, and potential hidden states. These kinds of problems can contribute to TD accumulation.

Amershi *et al.* (AMERSHI et al., 2019) observed software teams at Microsoft developing AI-based applications, employing a comprehensive nine-stage workflow process categorized into data-oriented and model-oriented phases. Kalinowski *et al.* (KALINOWSKI et al., 2023) abstracted seven generic life cycle stages based on the nine ML life cycle stages by Amershi *et al.* and the Cross-Industry Standard Process Model for Data Mining (CRISP-DM) (CHAPMAN et al., 2000) model phases.

These stages include *Data Collection*, involving gathering and analyzing data, and *Data Pre-Processing*, considered the most time-consuming phase, addressing, for instance, missing data and outliers. Subsequent stages entail listing possible models, evaluating results, and post-processing activities, culminating in result presentation and value generation. This process reflects a structured approach to ML project development, encompassing various tasks from data collection to result evaluation and presentation, ensuring a comprehensive and systematic process to achieve desired project outcomes.

Given this scenario, the motivation behind this study is to kick-start the exploration of potential causes of code-related TD accumulation in ML systems and shed light on the barriers that developers may encounter during routine tasks throughout the life cycle of an ML system. Existing studies predominantly focus on the types of TD and the emergence of new ones compared to those already known in SE. This research aims to raise awareness among ML system developers about the risks of accumulating code-related TD encountered during typical tasks throughout the life cycle of an ML system.

Hence, this research aims to identify and evaluate code-related issues leading to TD within ML code. To this end, we analyzed typical tasks across each phase of the ML life cycle and categorized problems.

1.2 Goal and Research Questions

Our main research goal is to compile and assess a list of issues that can lead to ML code TD.

The assessment goal can be stated following the GQM goal definition template (BASILI; ROMBACH, 1988) as follows: ***Analyze a candidate list of issues that can lead to ML code TD with the purpose of characterizing with respect to the occurrence in practice and perceived relevance of the issues from the point of view of ML experts in the context of developing ML systems***.

Therefore, our first research question concerns compiling such a list.

RQ1: What are potential issues that could lead to ML code TD?

We aim to establish a set of potential issues that can lead to ML code TD by analyzing problems related to typical activities conducted throughout the different ML life cycle phases.

After compiling such a list, we derive the following two research questions.

RQ2: Do the identified issues occur in practice?

This research question verifies whether the proposed issues manifest in real-world ML-enabled systems.

RQ3. Are the issues perceived as relevant by practitioners in terms of leading to TD?

This research question evaluates the perceived relevance of the suggested issues in the candidate list regarding leading to TD.

Hence, considering these research questions, we assume that frequently occurring issues perceived as relevant should be shared with practitioners to avoid them in ML system projects. To assess these characteristics from a practitioner's point of view, we conducted two focus group sessions with nine ML experts.

1.3
Methodology Overview

We initiated our study by searching the literature and targeting articles discussing various types of TD in ML systems to foster and gain insights into potential issues leading to code-related TD. We found a Systematic Mapping Study by Bogner *et al.* (BOGNER; VERDECCHIA; GEROSTATHOPOULOS, 2021), which revealed four new types of TD in AI-based systems, including data, model, configuration, and ethics debt. Moreover, it also highlights the recurring presence of established TD types such as infrastructure, architectural, code, and test debt. In addition, Tang *et al.* (TANG et al., 2021) introduced seven distinct TD categories relevant to ML, focusing on custom data types. These categories include duplicate feature extraction code, reusability, unnecessary model code, model code comprehension, modifiability, and duplicate model code.

Considering this previous research, we established our methodology in two key steps:

1. *Compilation of the Candidate Issues.* Analyzing potential problems related to typical activities conducted throughout the different ML life cycle phases to build a list of issues that may lead to code-related TD. The list was refined in discussions with an independent researcher, keeping in mind the main tasks to be performed in each of the typical stages

of the ML life cycle. Subsequently, this list was reviewed by an additional independent researcher to eliminate inconsistencies and add any overlooked issues. Further details on the compilation and the candidate list can be found in Section 3.

2. *Focus Group Sessions.* We conducted two focus group sessions, following the guidelines by Kontio *et al.* (KONTIO; BRAGGE; LEHTOLA, 2008) to facilitate in-depth discussions and obtain insights from ML experts, with the objective of assessing the occurrence and relevance of the candidate issues. The focus group sessions were conducted with nine ML experts. Further details on the focus group sessions and their results can be found in Sections 3 and 4.

1.4 Dissertation Outline

This dissertation is organized as follows. In Section 2, the background and an overview of relevant prior research are provided. Section 3 outlines the compilation and planning of the assessment of the candidate issues. Section 4 details the assessment and refinement of the list of candidate issues. Subsequently, the analysis of the results is provided in Section 5, and finally, in Section 6, the document concludes and discusses potential avenues for future research.

2
Background and Related Work

2.1
Introduction

This chapter introduces the relationship between TD and ML systems, providing a foundational understanding of both concepts. It begins by elucidating the origins of TD in SE, drawing parallels with financial debt and emphasizing its implications for productivity and code maintainability.

Moreover, it explores recent research highlighting the emergence of new types of TD within AI-based systems, shedding light on specific challenges faced in ML projects. Furthermore, the chapter delves into the ML life cycle, outlining the stages of ML project development, from data collection to model deployment and monitoring.

This chapter lays the groundwork for subsequent discussions on identifying, assessing, and managing TD in ML.

2.2
Technical Debt

In software development, Ward Cunningham (CUNNINGHAM, 1992) introduced the concept of TD, likening it to financial debt that allows for faster initial development but incurs “interest" to be paid later. Accumulating TD has far-reaching consequences.

According to Tom *et al.* (TOM; AURUM; VIDGEN, 2013), TD reduces productivity by making maintenance and code modifications more challenging. It can negatively impact developers’ morale, as they need to invest extra effort in dealing with it. Over time, developers must repay the “interest" and “principal" on the accrued debt to maintain their system.

Taking a minimally invasive approach is advisable to manage TD cost-effectively, considering potential gains and losses. Most systems with TD are already in operation and have a value that needs to be preserved. Dealing with TD carries the risk of causing more harm than good.

In a systematic literature review, Alves *et al.* (ALVES et al., 2014) gathered various types of TD. The primary focus here is on Code TD, which concerns issues within the source code that hinder code readability and maintenance. It’s typically addressed by improving coding practices and involves problems like code duplication and overly complex code structures, as defined by Li *et al.* (LI; AVGERIOU; LIANG, 2015).

2.3 Technical Debt in Machine Learning Systems

Bogner *et al.* (BOGNER; VERDECCHIA; GEROSTATHOPOULOS, 2021), in their systematic mapping study based on 21 primary studies, identified four new types of TD that emerge in Artificial Intelligence(AI)-based systems: data debt, model debt, configuration debt, and ethics debt. They also noted that AI systems commonly use established TD types like infrastructure, architectural, code, and test debt. However, they may have AI-specific aspects, such as managing and monitoring AI pipelines and models to mitigate infrastructure debt.

Tang *et al.* (TANG et al., 2021) recognized a knowledge gap regarding the evolution and maintenance of ML systems. They conducted a study across 26 projects, identifying seven new TD categories specific to ML, including custom data types, duplicate feature extraction code, model code reusability, unnecessary model code, model code comprehension, model code modifiability, and duplicate model code. It was highlighted that duplication significantly contributes to TD in ML systems, particularly in configuration and model code.

Tang *et al.* also highlight that while traditional software developers frequently use inheritance to reduce code duplication, the increasing popularity of scripting languages, mostly used in ML systems, requires model code to be written in an object-oriented manner. This transition poses challenges for ML developers to incorporate inheritance effectively. Combating code duplication debt, a common issue in ML code is crucial.

In line with this previous study, Cabral *et al.* (CABRAL et al., 2024) provided evidence that the adoption of object-oriented design principles, widely used to avoid TD in conventional software engineering, can improve code understanding within the realm of ML projects, also enhancing the maintainability of ML code.

2.4 Machine Learning Life Cycle

The Cross-Industry Standard Process Model for Data Mining (CRISP-DM) (CHAPMAN et al., 2000) process model for data mining provides an overview of the life cycle of a data mining project. It contains the phases of a project, their respective tasks, and the relationships between them. The life cycle of a data mining project consists of six phases, as shown in Figure 2.1. The sequence of the phases is not rigid. Moving back and forth between different phases is always required. It depends on the outcome of each phase and which phase or particular task has to be performed next. The arrows indicate the most important and frequent dependencies between phases.

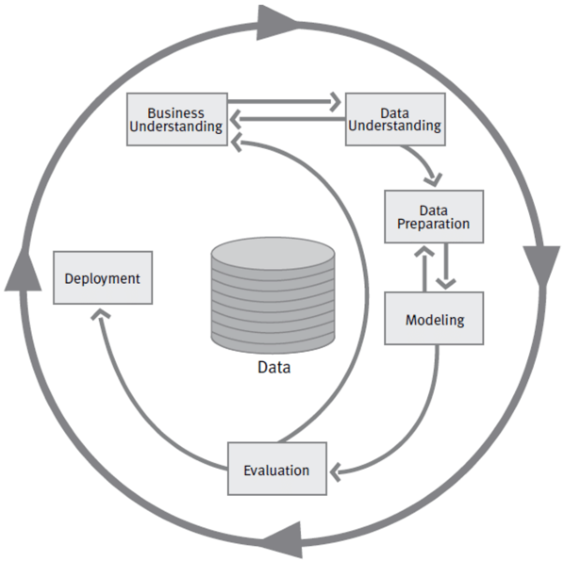


Figure 2.1: Phases of the current CRISP-DM process model for data mining (CHAPMAN et al., 2000)

In their study, Amershi *et al.* (AMERSHI et al., 2019) observed software teams at Microsoft as they developed AI-based applications. They utilized a comprehensive nine-stage workflow process, as Figure 2.2 illustrates, drawing from their prior experiences in AI application development related to search, natural language processing (NLP), and data science tools.

The nine stages fall into two main categories: data-oriented phases, which involve data collection, cleaning, and labeling, and model-oriented phases, including defining model requirements, feature engineering, model training, evaluation, deployment, and monitoring. It's worth noting that the larger feedback arrows indicate that model evaluation and monitoring can lead to a return to any of the earlier stages, allowing for iterative refinement. The smaller feedback arrow suggests that model training may loop back to the feature engineering stage in cases like representation learning.

Based on the previously mentioned nine ML life cycle stages presented by Amershi *et al.* and the CRISP-DM industry-independent process model phases (CHAPMAN et al., 2000), Kalinowski *et al.* (KALINOWSKI et al., 2023) abstracted seven generic life cycle stages, as depicted in Figure 2.3.

The *Data collection* phase involves gathering the necessary information and effectively collecting and analyzing the data to address the problems

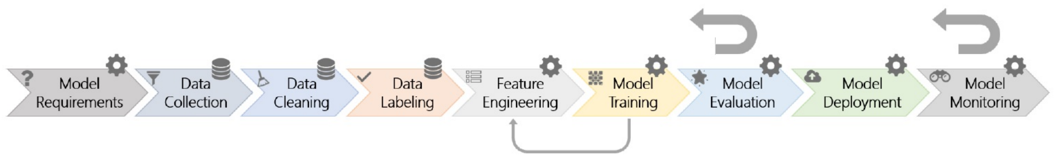


Figure 2.2: The nine stages of the ML workflow by Amershi *et al.* (AMERSHI et al., 2019)

identified in the *Problem Understanding and Requirements* phase. These data are typically, but not necessarily, organized in one or more databases, which can be relational databases, Data Warehouses, Data Marts, or Data Lakes. From there, ETL operations (Extraction, Transformation, Loading) can be performed on the source data to prepare it for future model building.

The third stage, where *Data Pre-Processing* activities are carried out, is the most time-consuming and laborious in a Data Science project, estimated to consume at least 70% of the total project time. It may be necessary to remove or complement missing data, correct or mitigate discrepant data (outliers) and class imbalances, and select the most appropriate variables and instances to compose the model(s) to be built in the next stage.

The fourth stage involves listing the possible and feasible models for each type of problem, estimating the parameters that compose the models based on the pre-processed instances and variables from the previous stage, and evaluating the results of each model using metrics and a fair comparison process.

Next, post-processing activities are carried out in the fifth stage: business heuristics are combined with the adjusted models from the previous stage, and a final evaluation is made, considering the strengths and difficulties encountered in implementing each model.

Then comes the sixth stage, the presentation of results. It is recommended to report the methodology adopted to address the solution to the managers' demands, compare the best model results with the current benchmark (if available), and plan the steps for implementing the proposed solution.

In the seventh stage, the focus is on qualitatively generating value for the enterprise (e.g., listing operational and human resources gains) and quantitatively (e.g., calculating ROI - Return on Investment).

In this work, similar to other recent research on the topic (*e.g.*, (ALVES et al., 2023), (ZIMELEWICZ et al., 2024)), we consider these generic life cycle stages.

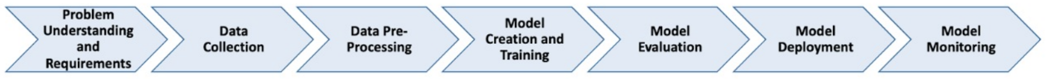


Figure 2.3: Seven typical stages of the ML life cycle

2.5 Concluding Remarks

In conclusion, the exploration of TD within ML systems reveals the multidimensional nature of TD accumulation and its implications throughout the ML life cycle. By identifying and characterizing various types of TD specific to ML, researchers and practitioners gain insights into the challenges posed by code duplication, model complexity, and evolving ML workflows.

Understanding the nuances of TD in ML systems can enable more informed decisions and developing proactive management strategies to mitigate TD and ensure the long-term sustainability of ML projects. As ML continues to advance and find applications across diverse domains, addressing TD remains paramount for building resilient and dependable ML solutions that can effectively meet the demands of real-world scenarios.

3
Compiling and Planning the Assessment of the Candidate Issues

3.1
Introduction

This chapter outlines the methodology employed to compile the list of issues that may lead to code-related TD in ML-enabled systems and the planning of the focus group sessions to investigate the occurrence and relevance of these issues. The preparation for the sessions is discussed, including creating an interactive Miro board and defining the dynamics of the discussions.

3.2
Compilation of the Candidate Issues

Practitioners employ various operational strategies to tackle the challenges identified across the ML project life cycle. For instance, during the data collection and pre-processing phases, data cleaning techniques are utilized, including detection and handling of missing values and outliers, as well as data normalization and standardization to ensure the required quality and consistency for training models. Throughout model creation and training, practitioners employ cross-validation methods and model selection techniques to ensure model generalization and adequate performance.

These operational approaches are crucial for ensuring the success and sustainability of ML projects in the face of recurring challenges encountered along the way. Gathering these common activities across different ML project life cycle phases, Table 3.1 was developed. Even though not all are executed in every ML project, they reflect typical activities ML practitioners conduct throughout the project life cycle. It's important to note that, when compiling this list, the focus was not to confirm whether a specific issue leads to code-related TD; rather, the main objective is to highlight which common tasks executed along the way may accumulate TD and its relevance.

The table is structured with five columns: the first denotes the specific phase in the ML life cycle, the second column describes the activity executed during the development of an ML-enabled system, and the other three columns represent categories of problems *(e.g., Missing, Incomplete/Insufficient, or Inappropriate/Wrong)* inspired in the IEEE Standard for Software Anomalies (IEEE. . . , 2010).

The combination of the categories of problem names with the activity generates what we term as ***issues***, for instance, *'Missing identifying outliers, Incomplete/Insufficient identifying outliers, or Inappropriate/Wrong identifying outliers'*.

The principal objective of this table was to present it to experts to ensure that these issues were relevant and to warn the ML teams while developing an ML-enabled system.

Note that some of the combinations do not make sense in real-world *(e.g., Incomplete/Insufficient Rebalancing)*, so they were not considered, and it is marked with an '×' in Table 3.1. All possible combinations are marked with an '✓.'

By joining all, the final list with 34 candidate issues that may lead to TD was ready to be presented to the experts to discuss.

Table 3.1: Issues that may lead to code-related TD in ML life cycle

Phase	**Activity**	**Missing**	**Incomplete/ Insufficient**	**Inappropriate/ Wrong**
Data collection	Integrating correctly data	✓	✓	✓
	Consuming correctly data	✓	✓	✓
Data pre-processing	Identifying outliers	✓	✓	✓
	Selecting features when needed	✓	✓	✓
	Identifying missing values	✓	✓	✓
	Rebalancing (typically by resampling)	✓	×	✓
	Removing inconsistent data (remove the inconsistency)	✓	✓	✓
	Pre-processing scaling	✓	✓	✓
Model creation and training	Testing candidate algorithm possibilities	✓	✓	✓
	Splitting training/test/validation data	✓	×	✓
	Hyperparameter tuning	✓	✓	✓
Model evaluation	Choosing evaluation metric	×	✓	✓
	Properly using methods for evaluating a model's performance	×	×	✓

3.3 Focus Group Main Goal and Scope

The main goal of the focus group sessions is to confirm if the identified issues occur in practice and if they are perceived as relevant by practitioners in terms of leading to TD through experts' opinions. The GQM (Goal Question Metric) definition template (BASILI; ROMBACH, 1988) is used to set this goal as follows: ***Analyze the candidate list of issues that can lead to ML code TD with the purpose of characterizing with respect to the occurrence in practice and perceived relevance of the issues from the point of view of ML experts in the context of developing ML systems***.

3.4 Focus Group Population

We employed a targeted sampling approach for our population of ML experts to obtain more precise responses. Specifically, we selected participants who were actively engaged in ML projects at ExACTa, Petrobras, and SERPRO.

The ExACTa (Experimentation-based Agile Co-creation initiative for Digital Transformation) R&D laboratory at PUC-Rio has approximately 100 collaborators working with several industry partners.

Petrobras is a Brazilian public corporation that operates in an integrated and specialized manner in the oil, natural gas, and energy industries, with expertise in exploration and production. This company has approximately 40,000 employees.

SERPRO (Federal Data Processing Service) is a Brazilian public company providing information technology services. It is the Brazilian government's primary provider of technological solutions and currently employs approximately 10,000 staff members.

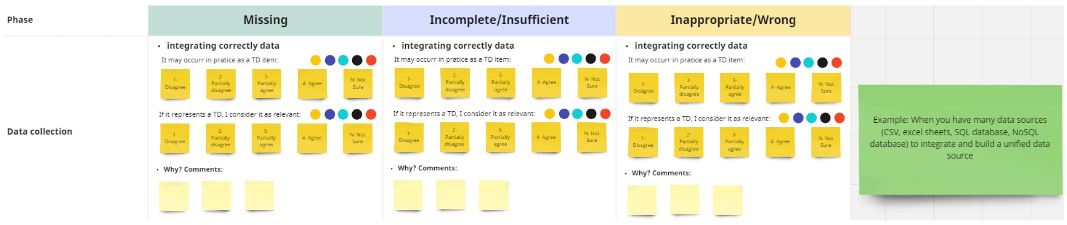
Our participant selection process was driven by convenience, as we had direct access to these professionals. We extended invitations to individuals we knew possessed relevant experience and were actively involved in developing ML-enabled systems.

3.5 Focus Group Preparation

We used the online tool Miro[1] to create the virtual template, as depicted in Figure 3.1. Leveraging this platform, we crafted an interactive board that

[1]https://miro.com/

streamlined the orchestration of the focus group session.



Example: When you have many data sources (CSV, excel sheets, SQL database, NoSQL database) to integrate and build a unified data source

Figure 3.1: Miro discussion template

The chart comprises four columns. The first indicates a phase of the ML system life cycle. The other three columns represent categories of problems *(e.g., Missing, Incomplete/Insufficient, and Inappropriate/Wrong)*, as mentioned in Section 3.2. Within each cell is the typical activity alongside two questions concerning its occurrence and relevance.

To streamline discussions, we employed the Likert scale (1- Disagree, 2- Partially Disagree, 3- Not Sure, 4- Partially Agree, and 5- Agree) to assess participants' levels of agreement for the two questions posed for all candidate issues. The first question was ***It may occur in practice***, indicating the likelihood of the issue leading to code-related TD. The second question was ***I consider it relevant***, indicating the relevance of TD resulting from the issue, assuming it occurs.

Each participant was represented by a colored bullet, which allowed them to move freely to their preferred answer on a Likert scale yellow box. Additionally, a green box containing explanations was provided on the right-hand side for clarity and to eliminate any uncertainties regarding the questions posed. Moreover, the Miro platform enabled participants to comment by adding light-yellow sticky notes to each proposed TD item. This collaborative and interactive approach facilitated an in-depth and insightful discussion during the focus group session.

A Participant Characterization Form was developed to gather information on each participant, enabling a more accurate interpretation of the results.

3.6 Focus Group Session Dynamics

Following the recommendation of Menary *et al.* (MENARY et al., 2021) for online focus-group sessions, we organized multiple sessions with limited participants. To accommodate this, we divided the participants into two focus group sessions based on availability and preferences.

The two remote focus group sessions took place via video conferences on June 15, 2023, and September 22, 2023. The video conferences were held by

Zoom platform [2] and recorded to facilitate getting all relevant participants' comments. Both sessions were approximately 2 hours in duration.

These sessions included nine industry practitioners, with four participants in the first and five in the second. Additionally, two researchers served as facilitators throughout both sessions.

The sessions were organized following the steps:

1. The Participant Characterization Form was distributed and filled by the participants;

2. A brief presentation was included to provide participants with a clear definition of TD;

3. Subsequently, the facilitators introduced the Miro board and how the session would be conducted. They explained the approach that would be taken during the discussions;

4. Finally, the facilitators proceeded to individually introduce each issue to discuss and register the opinions and comments of the experts.

3.7 Concluding Remarks

This chapter detailed the methodology employed in conducting the focus group sessions to investigate the relevance and occurrence of issues that may lead to code-related TD in ML-enabled systems. Initially, 34 candidate issues that could contribute to TD across various phases of the ML life cycle were compiled. These issues were derived from analyzing typical tasks in each phase and categorized into three main problem categories: Missing, Incomplete/Insufficient, and Inappropriate/Wrong. The chapter also detailed the preparations for the focus group sessions and the session dynamics. The results will be described in the next chapter.

[2] https://zoom.us/

4
Assessing and Refining the List of Candidate Issues

4.1
Introduction

In this chapter, we present the characterization of the participants and the results obtained from the focus group sessions. To summarize the discussions, we showcase the final votes from each session, provide visual representations, and report expert comments to enhance the understanding of the results.

4.2
Participant Characterization

Table 4.1 offers an overview of the participants' backgrounds. As mentioned, the data for this table was gathered through an online Participant Characterization Form via Google Forms and distributed to participants shortly before the beginning of the focus group session. It was composed of four questions:

- **Q1**. What is your highest education level?
- **Q2**. How many years of experience do you have developing machine learning-enabled systems?
- **Q3**. How many machine learning-enabled system projects have you worked on?
- **Q4**. How do you classify your level of knowledge concerning technical debt? (B: Beginner, M: Mid-level, A: Advanced)

Table 4.1: Participant Background Collected via Characterization Form

	1º Session				2º Session				
Participant	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**
Q1	PhD	MSc	MSc	MSc	PhD	MSc	BSc	BSc	BSc
Q2	14	3	3	4	10	4	4	1	2
Q3	15	10	3	5	15	1	3	3	1
Q4	B	M	M	M	A	B	B	M	M

As previously mentioned, we conducted two focus group sessions. The first session included four participants, P1 to P4, while the second involved five participants, identified as P5 to P9.

It's noteworthy, as indicated in Table 4.1, that the recruited participants are experienced data scientists, predominantly holding either an MSc or a PhD in Computer Science.

To more objectively communicate our results, in the section 4.3, we present the detailed analyses only for the data collection ML life cycle phase issues. Similar analyses for issues of all other ML life cycle phases can be found in Appendix A.

4.3 Issues of the Data Collection ML Life Cycle Phase

The presentation of results follows a structured approach: we enumerate typical activities, offer examples, and detail the discussion and outcomes for each candidate issue. During these discussions, we exhibit the final votes cast in each session, offer visual representations, and incorporate expert comments to enhance comprehension of the results.

The visual representation follows a Likert scale with the following colors: dark red for *disagree*, light red for *partially disagree*, gray for *not sure*, light green for *partially agree*, and dark green for *agree*.

Please note that the names of the issues were designated to make the content more understandable to the reader, replacing the initial concatenated name presented in the sessions solely to explain the main idea to the participants. For example, the combined name *Inappropriate/Wrong identifying outliers* was renamed to *Inaccurate outlier detection*.

The criteria for determining occurrence is if the majority of votes fall between *partially agree* or *agree*. Regarding relevance, it is considered high if the majority of votes are *agree*; otherwise, it is considered low. Additionally, it's worth noting that the median was used to determine the final result.

It is also important to mention that the final votes for each session can be found in our open science repository Zenodo[1]. Here, to be concise, only the aggregated picture is presented.

– **Integrating data correctly** - Example: When you have multiple data sources (CSV, Excel sheets, SQL databases, NoSQL databases) to integrate and create a unified data source.

[1]https://zenodo.org/doi/10.5281/zenodo.10035700

1. ***Missing data integration:***

   In the first session, the first question related to the occurrence; two participants reported agreeing, one reported partially agreeing, and one reported partially disagreeing. For the second question related to relevance, two participants reported partially agree, one reported partially disagree, and one wrote disagree.

   In the second session, all participants indicated agreement in the first question. One participant reported partial agreement about the second question, while four fully agreed.

   Participant P5 gave an example: *"In the context of predictive models, a lack of data integration can lead to the absence of relevant information for the model, hampering the achievement of high accuracy. For example, in practice, a crucial explanatory column may be missing, undermining the model's ability to adequately capture relationships between the available variables"*.

   Participant P8 reaffirmed the previous comment: *"There will be a possibility of future rework because, when evaluating the results, it may be challenging to explain or find a justification for the inadequate model due to the absence of a portion of the data"*.

   It becomes evident that, in the second session, as in the first session, it was confirmed that it occurs in practice and can lead to code-related TD. Its relevance, to varying degrees, is acknowledged, but it can be classified as an issue of low relevance using the median. Figure 4.1 illustrates the total for this issue.

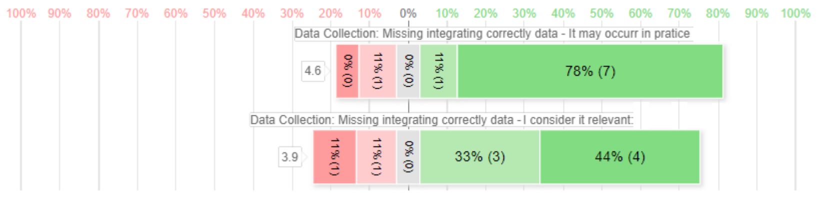


Figure 4.1: Total - Missing data integration

2. ***Insufficient data integration:***

   In the first session, in both questions, all participants reported agreeing.

   In the second session, all participants agreed on the first statement. In the second question, one participant reported partial agreement and four reported agreement.

   Participant P5 stated: *"You can be unlucky not to consider an important part of the data; on the other hand, you can also be lucky* *not to consider a part that isn't so crucial"*. After, gave an example: *"In one scenario, for instance, when building a model to predict diesel prices, data scientists may have no idea which variables influence diesel prices. It's possible not to know whether a particular piece of data is important or to be unaware of the existence of some available data. This ultimately poses a challenge during the model generation process"*. And then conclude: *"It is possible that a lack of understanding of the business and the available data can result in insufficient data integration"*.

Participant P7 argued: *"I think it's quite common and happens more often than simply missing (the previous issue). You consider one part later and then realize that something is missing. I believe it's more common than the first scenario because when you do nothing, you immediately see that a step in the process is missing."*.

Figure 4.2 provides a visual representation of the total for this issue, leaving no doubt about its status as a highly relevant issue.

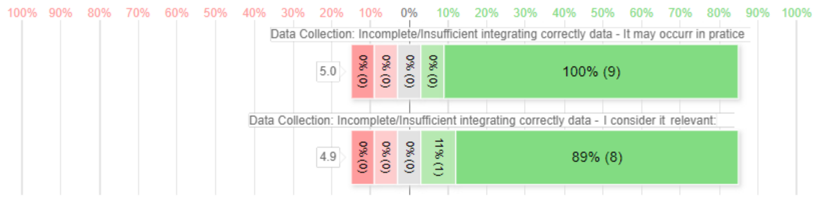


Figure 4.2: Total - Insufficient data integration

3. ***Improper data integration:***

   In both sessions, all participants reported agreement in response to both questions.

   Participant P6 provided an example from a project's outset: *"In my team, we utilize a template that streamlines data integration. Before adopting this approach, we encountered challenges as multiple team members were tackling the same issue independently, resulting in redundant efforts"*.

   Participant P5 stated: *"When data integration is done incompletely or inadequately, considering that the machine learning model will learn from this input data, it may not learn effectively, leading to poor accuracy. However, the impact can be even worse, as the model can learn incorrectly if data integration is done incorrectly"*.

   Upon analyzing the results from both sessions, Figure 4.3 unmistakably establishes the classification of this issue as highly relevant.

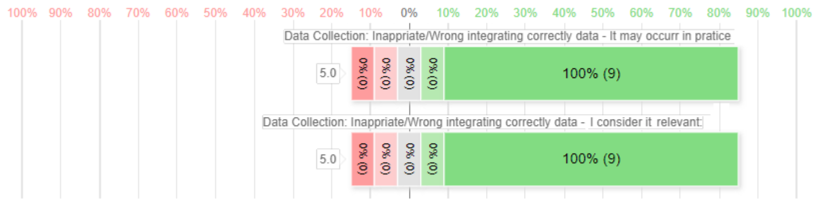


Figure 4.3: Total - Improper data integration

– **Consuming data correctly** - Example: When you have a large unified data source, consume only part of it.

1. ***Missing consuming data correctly:***

   In the first session, the first question about occurrence; two participants reported partially disagreeing, one reported disagreeing, and one said to be not sure. In the second question about relevance, three reported disagreeing, and one participant reported partially disagreeing.

   Participant P1 expressed skepticism about the feasibility of continuing development in this scenario, stating: *"I do not see the possibility of continuing development in this case"* while Participant P2 asserted that *"this is a crucial step missing; it would be a bug"*.

   In the second session, three participants reported disagreeing in the first question, and one said to be unsure. In the second question, one reported disagreeing, and four said were not sure.

   Counting the votes from both sessions, it is evident that this issue can be discarded.

2. ***Insufficient data consumption:***

   In the first session, the question about occurrence, one participant reported partially agreeing, and three said agreeing. Regarding the question about relevance, one participant reported partially disagreeing, and three reported agreeing.

   Participant P3 pointed out that *"I disagree with this statement because when data arrives incomplete, it is important to adopt an approach that allows us to handle this gap efficiently. This doesn't necessarily lead to TD. It's more a matter of recognizing the situation and establishing a policy to address it. It doesn't need to become a problem automatically.*

   This is a challenge we often encounter, and we deal with it by filling in the gaps by repeating the previous value, typically the standard procedure, or by applying appropriate calculations for the situation".

In the second session, all participants agreed regarding the first question. In the second question, one participant reported partial agreement, while four reported full agreement.

Participant P5 stated: *"For me, this problem is very similar to incomplete data integration, but there it involves columns, and here it relates to rows. It's a similar comment to the one I made in Incomplete/Insufficient integrating data items correctly. If, by chance, the most relevant lines are the ones I forgot, this could potentially become a problem."*.

Figure 4.4 illustrates the results from both sessions. Most participants reported full agreement, leading us to conclude that this issue can be categorized as highly relevant.

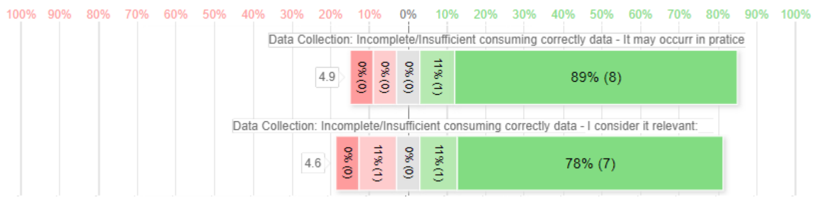


Figure 4.4: Total - Insufficient data consumption

3. ***Improper data consumption:***

   In the first session, in both questions, all participants reported agreeing.

   All participants agreed with the first statement about the occurrence during the second session. In the second question, one participant reported partially agreeing, and four said agreeing.

   Participant P8 cited a previous experience: *"This has happened to me. I used a different version of the dataset to compare it with a model generated from another dataset. In theory, we wanted to compare models built with the same dataset. When we evaluated the performance, it was very poor. It wasn't immediately obvious because the model was working. I only realized much later when evaluating the performance by comparing the models because it was very poor. The comparison with the other model wasn't a fair one"*.

   Since nearly all participants agreed with both statements and accordingly by our focus group sessions, we can confidently conclude that this issue can be classified as high relevance. The results are in Figure 4.5.

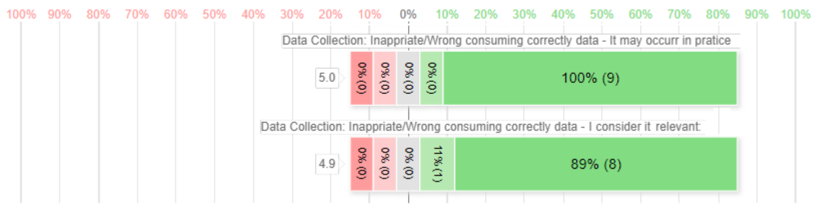


Figure 4.5: Total - Improper data consumption

As mentioned previously, we presented only the results of the data collection ML life cycle phase issues above, and all other ML life cycle phase issues can be found in Appendix A.

4.4 Synthesis of the Results

Table 4.2 provides a comprehensive list of issues that may lead to code-related TD associated with typical activities throughout the development of ML-enabled systems. These issues are categorized based on relevance and occurrence across different ML life cycle phases.

Our study identified 34 potential issues (*cf.* Table 3.1), then refined to 30 through the focus group sessions. Among these, 24 were deemed highly relevant, with particular emphasis on the data pre-processing phase, where 14 issues were considered highly relevant.

The results underscore the significance of code-related TD in ML systems, highlighting challenges such as improper data integration, inaccurate feature selection, and incorrect identification of missing values and outliers.

4.5 Concluding Remarks

The focus group sessions delved into identifying and assessing potential code-related TD issues across various stages of the ML life cycle. The participants highlighted critical concerns such as improper data integration and consumption, insufficient data handling, and sub-optimal model training practices as significant contributors to code-related TD in ML-enabled systems. By presenting a comprehensive list of these issues and their relevance levels, we provide valuable insights for practitioners aiming to code-related TD in ML systems, emphasizing the importance of proactive mitigation strategies. The next chapter presents a more detailed discussion of the results.

Table 4.2: Final result

Phase	**Issue**	**Relevance**
Data Collection	Improper data integration	High
	Improper data consumption	High
	Insufficient data consumption	High
	Insufficient data integration	High
	Missing data integration	Low
Data pre-processing	Inaccurate feature selection	High
	Inaccurate missing value identification	High
	Inaccurate outlier detection	High
	Inaccurate pre-processing scaling	High
	Incomplete data removal	High
	Incomplete missing value identification	High
	Incomplete pre-processing scaling	High
	Incorrect rebalancing	High
	Missed features selection	High
	Missed identification of outliers	High
	Missed pre-processing scaling	High
	Not identifying missing values	High
	Not removing inconsistent data	High
	Removing inconsistent data inappropriately	High
	Incomplete feature selection	Low
	Incomplete outlier identification	Low
	Missed rebalancing	Low
Model creation and training	Inappropriate splitting of training/test/validation data	High
	Incorrect hyperparameter adjustment	High
	Neglected splitting training/test/validation data	High
	Neglected hyperparameter adjustment	High
	Neglected testing of candidate algorithm possibilities	High
	Insufficient exploration of candidate algorithm options	Low
	Suboptimal hyperparameter adjustment	Low
Model evaluation	Suboptimal evaluation metric selection	High

5
Discussion

5.1
Introduction

This chapter discusses the outcomes of the focus group sessions. Its primary aim is to understand critical issues that may arise throughout the ML system development life cycle, shedding light on their perceived relevance and practical incidence.

First, we discuss the issues raised in each of the four ML life cycle phases considered with issues presented to the practitioners. Thereafter, we address each of the three main research questions. Additionally, we discuss the study's limitations and threats to validity.

The analysis presented in this research underscores the complexity and importance of addressing code-related TD in ML systems. By examining various phases of the ML life cycle, we identified a spectrum of potential issues that could contribute to the accumulation of code-related TD, as illustrated in Table 3.1. Our approach involved conducting two focus group sessions with nine ML experts to validate their occurrence and relevance. Out of the 34 candidate issues raised, the ML experts narrowed it down to 30 issues considered significant contributors to code-related TD, which were further assessed on their relevance.

The results of the four ML life cycle phases with issues presented to the practitioners are discussed hereafter. Each section presents a summary of the collected data in the focus group sessions formatted in tables. These tables are structured with three columns; the first column presents the names of the issues presented to the participants during the focus group session. These names are a combination of problem types (e.g., Missing, Incomplete/Insufficient, or Inappropriate/Wrong) and concise activity descriptions (e.g., consuming correct data), which were named *issue*. The second column contains a new name for the issue to enhance clarity and comprehension. The third column relates to the relevance of these problems, as discussed in each item. It is important to mention that the quotes are the transcripts of the comments from the ML experts extracted from the sessions.

5.2 Data Collection Phase

During the data collection phase, we observed that improper data integration and insufficient consumption could compromise the quality and reliability of resulting ML models. These challenges may arise due to inadequate planning or negligence in assessing the data quality. Furthermore, the lack of effective identification and treatment of missing values can distort model outcomes, leading to erroneous decisions.

There was a debate regarding the importance of proper data integration during the discussions on the data collection phase. Participant P5 emphasized this point by providing an illustrative example: "In the context of predictive models, a lack of data integration can lead to the absence of relevant information for the model, hampering the achievement of high accuracy. For example, in practice, a crucial explanatory column may be missing, undermining the model's ability to adequately capture relationships between the available variables."

Another significant discussion point was about insufficient data consumption. Participant P3 shared insights from their experience, stating, "I disagree with this statement because when data arrives incomplete, it is important to adopt an approach that allows us to handle this gap efficiently. This doesn't necessarily lead to TD. It's more a matter of recognizing the situation and establishing a policy to address it. It doesn't need to become a problem automatically."

Table 5.1 summarizes the final result of the Data Collection Phase.

Table 5.1: Final result for Data Collection

Focus groups issue name	**Final issue name**	**Relevance**
Inappropriate/Wrong consuming correctly data	Improper data consumption	High
Inappropriate/Wrong integrating correctly data	Improper data integration	High
Incomplete/Insufficient consuming correctly data	Insufficient data consumption	High
Incomplete/Insufficient integrating correctly data	Insufficient data integration	High
Missing integrating correctly data	Missing data integration	Low

5.3 Data Pre-Processing Phase

Many issues surfaced in the data pre-processing phase, ranging from feature selection to outlier detection, rebalancing, and data removal. Participants emphasized the critical importance of accuracy in these steps to ensure the quality of input data and, consequently, the effectiveness of ML models. Their insights shed light on the potential pitfalls and consequences of oversight in various aspects of data pre-processing.

There were discussions about the importance of identifying outliers to ensure accurate results, with Participant P3 emphasizing the risks of skipping this step. He mentioned, "It's an important step not to skip. Depending on the model you intend to use, it can significantly affect its outcome, potentially ruining your results". However, incomplete or inaccurate outlier identification was also discussed, with participant P7 highlighting the challenges of revisiting this stage after model development. He exemplified: "It's possible to add more columns even after that stage. For example, if you return to the data collection phase, you might forget to adapt the code to test the new columns."

Feature selection emerged as another focal point, with participants' insights highlighting this process's iterative nature. Participant P1 underscored the importance of revisiting feature selection to enhance model performance, while Participant P5 pondered the potential impacts of incomplete feature selection. He stated, "Most feature selection is already performed by most ML algorithms by design, so not doing it is not necessarily a problem."

These conversations underscored the complexity of data pre-processing in ML model development and the need for careful approaches to avoid potential code-related problems.

Table 5.2 summarizes the final result of the Data Pre-Processing Phase.

5.4 Model Creation and Training Phase

When creating and training ML models, we emphasized the importance of proper division of training, testing, and validation datasets and careful hyper-parameter tuning. Errors at these stages can lead to underfitted or overfitted models, undermining their ability to generalize to new data. Additionally, the lack of adequate testing of candidate algorithms can result in the selection of sub-optimal models.

Participants discussed various issues in this phase, including neglected testing of candidate algorithm possibilities. Participant P7 highlighted scenarios where outstanding initial results might lead to the neglect of exploring

Table 5.2: Final result for Data Pre-processing

Focus groups issue name	**Final issue name**	**Relevance**
Incomplete/Insufficient identifying missing values	Incomplete missing value identification	High
Incomplete/Insufficient pre-processing scaling	Incomplete pre-processing scaling	High
Incomplete/Insufficient removing inconsistent data	Incomplete data removal	High
Inappropriate/Wrong identifying missing values	Inaccurate missing value identification	High
Inappropriate/Wrong identifying outliers	Inaccurate outlier detection	High
Inappropriate/Wrong pre-processing scaling	Inaccurate pre-processing scaling	High
Inappropriate/Wrong rebalancing	Incorrect rebalancing	High
Inappropriate/Wrong removing inconsistent data	Removing inconsistent data inappropriately	High
Inappropriate/Wrong selecting features when needed	Inaccurate feature selection	High
Missing identifying missing values	Not identifying missing values	High
Missing identifying outliers	Missed identification of outliers	High
Missing pre-processing scaling	Missed pre-processing scaling	High
Missing removing inconsistent data	Not removing inconsistent data	High
Missing selecting features when needed	Missed features selection	High
Incomplete/Insufficient identifying outliers	Incomplete outlier identification	Low
Incomplete/Insufficient selecting features when needed	Incomplete feature selection	Low
Missing rebalancing	Missed rebalancing	Low

other algorithm options. Similarly, insufficient exploration of candidate algorithms was mentioned, with participant P5 emphasizing the need to prioritize the most relevant algorithms given the limitations of testing all possibilities. These discussions underscored the importance of thorough exploration to identify the most suitable algorithms for the given problem.

Furthermore, issues related to splitting training, testing, and validation data were addressed. Neglecting this division was deemed highly relevant, with Participant P3 stating, “It doesn’t make much sense to a data scientist. It is a basic premise.” Inappropriate data splitting, especially in time series analysis, was also highlighted as a significant concern. Participant P3 provided an example illustrating how random data splitting could compromise the essence of time series data analysis. These comments emphasized the need for careful consideration of data-splitting methods to preserve essential patterns and behaviors in the data.

Table 5.3 summarizes the final result of the Model Creation and Training Phase.

Table 5.3: Final result for Model Creation and Training

Focus groups issue name	**Final issue name**	**Relevance**
Inappropriate/Wrong hyperparameter tuning	Incorrect hyperparameter adjustment	High
Inappropriate/Wrong splitting training/test/validation data	Inappropriate splitting of training/test/validation data	High
Missing hyperparameter tuning	Neglected hyperparameter adjustment	High
Missing splitting training/test/-validation data	Neglected splitting training/test/-validation data	High
Missing testing of candidate algorithm possibilities	Neglected testing of candidate algorithm possibilities	High
Incomplete/Insufficient hyperparameter tuning	Suboptimal hyperparameter adjustment	Low
Incomplete/Insufficient testing of candidate algorithm possibilities	Insufficient exploration of candidate algorithm options	Low

5.5 Model Evaluation Phase

In the model evaluation phase, the need to select appropriate evaluation metrics that accurately capture the model's performance against project objectives was shown. Inappropriate metrics can lead to misconceptions about the model's effectiveness and erroneous or sub-optimal decisions.

For instance, suboptimal evaluation metric selection can result in insufficient metrics that fail to provide adequate insights into model performance. Participant P1 highlighted this issue by presenting an example of an alarm system where using the wrong metric could lead to overlooking critical aspects. Similarly, inappropriate or wrong selection of evaluation metrics can further exacerbate this problem, potentially leading to unsuitable assessments of model performance. Participant P4 provided an example of using a regression metric to evaluate a classification problem, illustrating the misalignment between the chosen metric and the problem type.

Ultimately, the discussions underscored the importance of aligning evaluation metrics with project objectives to ensure accurate assessments of model performance and informed decision-making.

Table 5.4 summarizes the final result of the Model Evaluation Phase.

Table 5.4: Final result for Model Evaluation

Focus groups issue name	Final issue name	Relevance
Incomplete/Insufficient choosing evaluation metric	Suboptimal evaluation metric selection	High

In short, our research shows it's important to deal with code-related TD issues throughout the ML process. We must manage our data well, plan carefully, and test our models thoroughly to ensure they work properly in real situations. Addressing these issues early in the development cycle and implementing robust TD management strategies can strengthen the reliability and efficiency of ML applications, fostering greater trust and usability across various domains. Furthermore, the outcomes of this study lay the groundwork for further research into tailored TD mitigation approaches, ultimately advancing the reliability and effectiveness of ML applications in real-world settings.

5.6 RQ1. What are potential issues that could lead to ML code TD?

To address this question, Table 3.1 which maps fourteen key activities along the ML system life cycle phases. These activities are typical tasks that developers may execute during routine tasks throughout the life cycle of an ML system. Associating them with problem types (Missing, Incomplete/Insufficient, Inappropriate/Wrong). It resulted in 34 identified candidate *issues.* Some specific issues, such as emphIncomplete/Insufficient rebalancing, were excluded from the final list due to lack of practicality.

No ML code-related issues were identified in the Problem Understanding and Requirements, Model Deployment, and Model Monitoring phases.

5.7 RQ2. Do the identified issues occur in practice?

To assess whether the issues occur in practice, we conducted two focus group sessions with nine experts (4 in the first and 5 in the second). This was done to assess each issue's occurrence and evaluate its relevance. According to the ML experts, a total of 30 out of the 34 potential issues were considered to occur in practice and potentially lead to TD.

Find in 4.4 the synthesis of results.

5.8 RQ3. Are the issues perceived as relevant by practitioners in terms of leading to TD?

Overall, 24 out of the 30 issues that occur in practice and may lead to technical debt were perceived as highly relevant. Figure 5.1 illustrates the number of issues per ML life cycle phase, and Figure 5.2 summarizes these numbers.

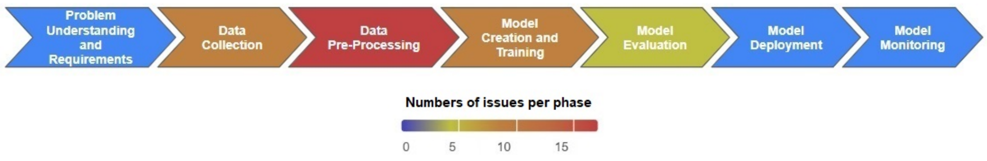

Problem Understanding and Requirements
Data Collection
Data Pre-Processing
Model Creation and Training
Model Evaluation
Model Deployment
Model Monitoring
Numbers of issues per phase
0
5
10
15

Figure 5.1: Heat map of numbers of issues per phase

Data collection	Data pre-processing	Model creation and training	Model evaluation
High: 4	High: 14	High: 5	High: 1
Low: 1	Low: 3	Low: 2	Low: -

Figure 5.2: Number of issues per phase

It is possible to observe that, during the data collection phase, four of the five issues that can lead to code-related TD in the focus group sessions were designated as high-relevance, and one was classified as low-relevance. In the data pre-processing phase, 14 of the 17 issues identified were established high relevant, while the remaining three were categorized as low relevant. In the model creation and training phase, seven issues were split into five high-relevant and two low-relevant categories. Lastly, the sole issue identified in the model evaluation phase was highly relevant.

5.9 Limitations and Threats to Validity

One limitation of our study is the possibility of unintentionally omitting critical ML code-related issues not addressed in our initial list (Table 3.1).

In this line, participants indicated the absence of certain issues during both focus group sessions. In the first session, they specifically mentioned topics such as model deployment and consumption, service orchestration, package versioning, and package dependencies, which could contribute to ML code TD. In the second session, they mentioned issues such as incorrect model finalization for production preparation (failure to train with all data) and failure to normalize required data during model usage in production.

A possible threat to validity is the potential for majority influence during discussions, wherein a participant may be swayed by the majority vote in moments of doubt. To mitigate the potential impacts of this threat, we conducted two independent sessions, selected qualified professionals, and moderated and recorded the sessions.

Another threat to validity concerns the number of participants and the number of focus group sessions. Notably, it is suggested to plan online focus groups with fewer participants than face-to-face focus groups (MENARY et al., 2021), with four considered appropriate (MATTHEWS; BAIRD; DUCHESNE, 2018). Considering this, we organized our two focus group sessions with four and five participants, respectively. Still, we know that more than two focus group sessions are needed to reach generalizable findings. Unfortunately, identifying teams with experience and willing to collaborate with academia is not a trivial task. Therefore, conducting new focus group sessions is part of future work.

5.10 Concluding Remarks

This chapter discussed the outcomes of our study, which focused on evaluating potential issues leading to code-related TD in ML-enabled systems. The validation process, conducted through two focus group sessions, provided valuable insights into the occurrence and relevance of these issues. We discussed the results for each ML life cycle phase. Additionally, we addressed the three research questions of this dissertation, shedding light on the systematic identification, practical occurrence, and perceived relevance of these issues by practitioners. Furthermore, we discussed the limitations and threats to validity, offering valuable considerations for future research endeavors.

6
Conclusion

6.1
Contributions

Sculley *et al.* (SCULLEY et al., 2015) shed light on the relationship between TD and ML-enabled systems. Bogner *et al.* (BOGNER; VERDECCHIA; GEROSTATHOPOULOS, 2021) identified four new types of TD that emerge in AI-based systems. Additionally, Tang *et al.* (TANG et al., 2021) introduced seven distinct TD categories relevant to ML, focusing on custom data types. However, substantial research gaps exist in identifying issues that lead to TD in ML code.

This study compiled and assessed a list of potential issues that can lead to code-related TD in each ML system life cycle phase. The methodology involved establishing a list of issues related to typical activities during the ML life cycle phases and assessing their occurrence and relevance through industry feedback in focus group sessions. This comprehensive approach provided a nuanced and insightful understanding of relevant issues leading to code-related TD in ML systems.

Throughout the study, a list of 34 potential issues that could lead to code-related TD in ML systems was compiled. These issues were refined into 30 consolidated issues through focus group sessions. It's worth noting that during the data pre-processing phase, 17 issues were identified, with 14 considered highly relevant. In the data collection phase, four issues were identified as highly relevant. The model creation and training phase revealed five highly relevant issues and one highly relevant in the model evaluation phase.

Hence, the data pre-processing phase was the most critical one, deserving particular attention to avoid ML code TD. We believe that the compiled list can help to raise awareness of issues to be addressed throughout the ML life cycle to minimize TD, helping to improve the maintainability of ML systems.

6.2
Future Work

While this study focuses exclusively on ML code-related TD, we acknowledge the existence of other forms of TD that are also relevant to ML systems. Therefore, additional studies could be conducted focusing on these other forms of TD. Furthermore, in this dissertation, we focused on quantitative and qualitative analyses. A new study focused on a more in-depth qualitative analysis could bring additional insights.

Furthermore, our investigations of the issues leading to ML code-related TD involved conducting two focus group sessions. Additional investigations using other empirical strategies (*e.g.*, experimental studies) could be conducted to refine the list further and strengthen our findings.

7
Bibliography

ALVES, A. P. S. et al. Status quo and problems of requirements engineering for machine learning: Results from an international survey. In: **Product-Focused Software Process Improvement - 24th International Conference, PROFES 2023, Dornbirn, Austria, December 10-13, 2023. Proceedings**. [S.l.: s.n.], 2023. p. 1–16.

ALVES, N. S. et al. Towards an ontology of terms on technical debt. In: **2014 Sixth International Workshop on Managing Technical Debt**. [S.l.: s.n.], 2014. p. 1–7.

AMERSHI, S. 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)**. [S.l.: s.n.], 2019. p. 291–300.

BASILI, V.; ROMBACH, H. The tame project: towards improvement-oriented software environments. **IEEE Transactions on Software Engineering**, v. 14, n. 6, p. 758–773, 1988.

BOGNER, J.; VERDECCHIA, R.; GEROSTATHOPOULOS, I. 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. Disponível em: <https://doi.org/10.1109%2Ftechdebt52882.2021.00016>.

CABRAL, R. et al. Investigating the impact of solid design principles on machine learning code understanding. In: **Proceedings of the 3rd International Conference on AI Engineering – Software Engineering for AI, CAIN 2024, Lisbon, Portugal, April 14-15**. [S.l.: s.n.], 2024. p. 1–11.

CHAPMAN, P. et al. Crisp-dm 1.0: Step-by-step data mining guide. In: . [S.l.: s.n.], 2000.

CUNNINGHAM, W. The wycash portfolio management system. **SIGPLAN OOPS Mess.**, Association for Computing Machinery, New York, NY, USA, v. 4, n. 2, p. 29–30, dec 1992. ISSN 1055-6400. Disponível em: <https://doi.org/10.1145/157710.157715>.

IEEE Standard Classification for Software Anomalies. **IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)**, p. 1–23, 2010.

KALINOWSKI, M. et al. **Engenharia de Software para Ciência de Dados: Um guia de boas práticas com ênfase na construção de sistemas de Machine Learning em Python**. [S.l.]: Casa do Código, 2023. 1-476 p.

KONTIO, J.; BRAGGE, J.; LEHTOLA, L. The focus group method as an empirical tool in software engineering. In: _____. **Guide to Advanced Empirical Software Engineering**. London: Springer London, 2008. p. 93–116. ISBN 978-1-84800-044-5. Disponível em: <https://doi.org/10.1007/978-1-84800-044-5_4>.

LI, Z.; AVGERIOU, P.; LIANG, P. A systematic mapping study on technical debt and its management. **Journal of Systems and Software**, v. 101, p. 193–220, 2015. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121214002854>.

MARTINEZ, I.; VILES, E.; OLAIZOLA, I. Data science methodologies: Current challenges and future approaches. **Big Data Research**, v. 24, p. 100183, 01 2021.

MATTHEWS, K. L.; BAIRD, M.; DUCHESNE, G. Using online meeting software to facilitate geographically dispersed focus groups for health workforce research. **Qualitative Health Research**, v. 28, n. 10, p. 1621–1628, 2018. PMID: 29911490. Disponível em: <https://doi.org/10.1177/1049732318782167>.

MENARY, J. et al. Going virtual: Adapting in-person interactive focus groups to the online environment. **Emerald Open Research**, v. 3, n. 6, 2021. Cited by: 13. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85111393492&partnerID=40&md5=8a45d831e50e0b8cc65bc8e7e3cc56cf>.

PIMENTEL, J. F. et al. A large-scale study about quality and reproducibility of jupyter notebooks. In: **2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)**. [S.l.: s.n.], 2019. p. 507–517.

SCULLEY, D. et al. Hidden technical debt in machine learning systems. In: **Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2**. Cambridge, MA, USA: MIT Press, 2015. (NIPS'15), p. 2503–2511.

TANG, Y. et al. An empirical study of refactorings and technical debt in machine learning systems. In: **2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)**. [S.l.: s.n.], 2021. p. 238–250.

TOM, E.; AURUM, A.; VIDGEN, R. An exploration of technical debt. **Journal of Systems and Software**, v. 86, n. 6, p. 1498–1516, 2013. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121213000022>.

ZIMELEWICZ, E. et al. Ml-enabled systems model deployment and monitoring: Status quo and problems. In: **Software Quality Days, 16th International Conference, SWQD 2024, Vienna, Austria, April 24-25**. [S.l.: s.n.], 2024. p. 1–20.

A
Appendix

The results of the data collection phase can be analyzed in session 4. Herein, the results of other phases can be found, as previously mentioned.

A.1
Issues of the Data Pre-processing ML Life Cycle Phase

– **Identification of outliers** - Example: An outlier is an observation that significantly deviates from the typical or expected values within a random sample from a population. These atypical observations can be either unusually high (positive outliers) or unusually low (negative outliers) compared to most data points in the sample. Outliers can distort statistical analyses and may indicate potential errors or interesting patterns in the data. Identifying and handling outliers is an important step in data analysis to ensure the accuracy and reliability of your results.

1. ***Missed identification of outliers:***

   In the first question, regarding the occurrence, all participants reported agreeing. Regarding relevance, two participants reported partial agreement in the second question, and two said they agreed.

   Participant P3, referring to one of the previous issues, *"Missing consuming data correctly"*, where it was emphasized that skipping is not advisable, mentioning that one might consider cutting it to save time, thinking it wouldn't significantly impact and, for instance, skipping the step of searching for outliers to build the model more quickly. However, this is quite serious because it can greatly influence the results. P3 said: *"It's not a good step to miss. Depending on the model you intend to use, it can significantly affect its outcome, potentially ruining your results. You deal with later stages, trying to improve when the error was in the step you skipped. I mentioned that it could be a serious issue"*.

   In the second session, in the first question, one participant reported partially agreeing, and four agreed. In the second question, two participants reported partially agreeing, and three said agreeing.

   Figure A.1 shows that despite not having unanimous agreement regarding its existence and relevance in both sessions, we can categorize this issue as highly relevant.

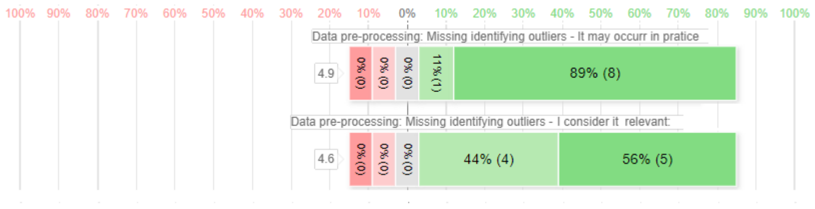


Figure A.1: Total - Missed identification of outliers

2. ***Incomplete outlier identification:***

   In the first session, all participants agreed on the question about occurrence. Regarding the question about relevance, all participants reported partially agreeing.

   In the second session, the first question regarding occurrence, one participant reported partially agreeing, and four agreed. In the second question, regarding relevance, two participants reported partially agreeing, and three said agreeing.

   Participant P7 exemplified: *"It's possible to add more columns even after that stage. For example, if you return to the data collection phase, you might forget to adapt the code to test the new columns"*.

   Figure A.2 summarizes the consensus between the two sessions, making this issue less relevant.

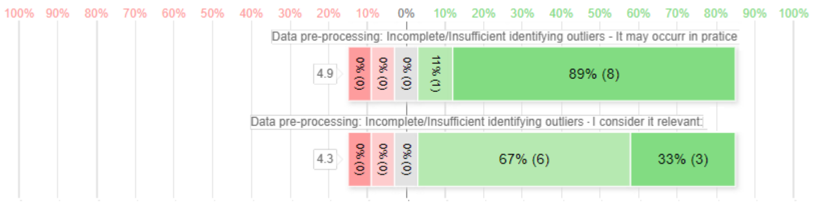


Figure A.2: Total - Incomplete outlier identification

3. ***Inaccurate outlier detection:***

   All participants agreed on the first question about occurrence in the first session. In the second question, regarding relevance, one participant reported partially agreeing, and three said agreeing.

   During the second session, all participants unanimously agreed with both questions.

   Participant P6 provided feedback on your interpretation of this matter: *"An example of this situation would involve considering an extended timeline. At a certain point, there was a shift in the behavior of the data, causing it to deviate from the established standard range. Consequently, we excluded this data, even though* *it merely represented a change in data behavior and should not have been classified as outliers".*

Figure A.3 shows the consensus between the two sessions in affirming the significance of this issue as highly relevant.

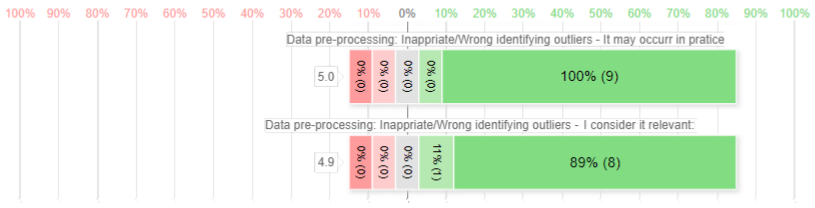


Figure A.3: Total - Inaccurate outlier detection

– **Selecting features when needed** - Example: Feature selection is a crucial technique that carefully chooses a subset of input variables for your model while retaining only the most relevant information and eliminating noisy or less important data. This process enhances model performance and clarity.

1. ***Missed features selection:***

   In the first session, all participants reported to agree in the first question. In the second question, all participants reported agreeing.

   In the second session, the question about occurrence, two participants reported partially disagreeing, one not sure, and two agreeing. Regarding the question about relevance, two participants reported agreeing, and three were not sure.

   Participant P5 stated: *"Most feature selection is already performed by most ML algorithms by design, so not doing it is not necessarily a problem. On the other hand, there are performance concerns. In requirements engineering, unmet non-functional requirements can be more problematic than those. For instance, if the application becomes too slow, it may not be used, and in this case, it can be a significant issue".*

   Despite a divided vote, with some participants marking "not sure" for both questions, the prevailing sentiment is that this issue can be classified as highly relevant. This outcome is further illustrated in Figure A.4.

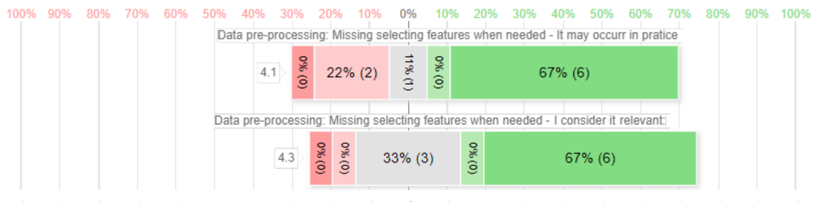


Figure A.4: Total - Missed features selection

2. ***Incomplete feature selection:***

   In the first session, all participants reported agreeing partially on the question regarding occurrence. Regarding the question about relevance, one participant reported partially disagreeing, and three reported partially agreeing.

   Participant P1 mentioned that *"Incomplete, to me, means the following: I have various approaches to feature selection. In the project I am working on, it's common for us to quickly create an initial version of the model to deliver it. Later, we refined and adjusted this model as we gained more knowledge about the problem. This can happen through data analysis or discussions with experts in the problem domain. There are several stages involved, and this constitutes a cycle.*

   Sometimes, we backtrack and discover that we have more data that wasn't provided to us previously. Within these stages of feature selection, one of them is especially important. We start with a specific selection and build a solid model. Then, we evaluate another selection that makes sense, analyze the importance of variables, and remove some of them to improve the models. This is done to expedite the initial delivery and then evolve from there.

   The key point is that, in certain cases, people may not realize that the initial approach was insufficient, and later on, it may be necessary to revisit this stage to enhance the model".

   Participant P3 said: *"I think 'incomplete' is a business decision, depending on whether it's worth investing more time in improvements or if the result is satisfactory enough. Not executing is a problem. The results of doing incomplete could probably improve later. Doing something wrong is always an issue, such as, for example, excluding an important variable."*

   During the second session, in both questions, two participants reported partial agreement, one agreed, and two reported being unsure.

Based on the findings from the two focus group sessions, shown in Figure A.5, this issue can be categorized as low relevant as most voters cast partially agree votes.

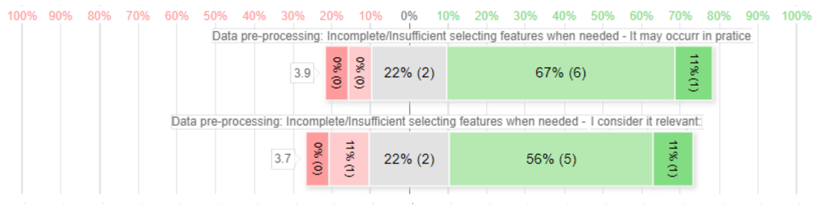


Figure A.5: Total - Incomplete feature selection

3. ***Inaccurate feature selection:***

   In both sessions and questions, all participants reported agreeing.

   Participant P1 mentioned that *"Visualizing and identifying the problem can be challenging; it often requires time for debugging and comprehension."* Participant P3 affirmed the previous comment: *"This happened to me when the variables had similar names. One ended in '31,' and I mistakenly used the one ending in '33.' I spent quite some time trying to figure out why my model wasn't producing coherent results. It was frustrating"*

   In the second session, Participant P6 shared an experience: *"This happened to me while working on a model. We had a variable for which we were missing data for an extended period, resulting in the loss of half of the information. We decided to remove this variable for that reason. However, when we presented the model to the stakeholders, we discovered that this variable was the most crucial"*.

   Participant P5 agreed and added another example: *" I've seen this happen before. We had around 150 columns, and we would ask the stakeholders to determine which ones were important. Still, we performed a preliminary cleanup to ensure they got all the information. However, during this cleanup, we removed about 100 columns. Out of those 100 pulled, approximately 30 of them should have been kept. We were trying to help but ended up causing more harm than good"*.

   Figure A.6 depicts unanimous agreement between the two sessions regarding the occurrence and relevance of this issue as highly relevant.

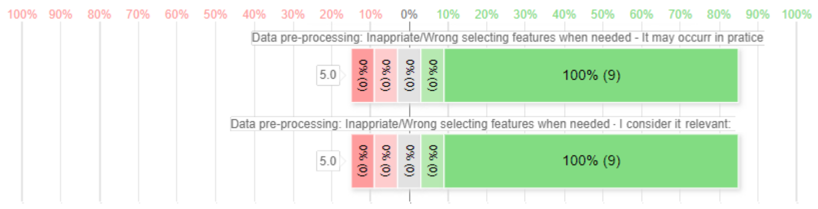


Figure A.6: Total - Inaccurate feature selection

– **Identifying missing values** - Example: Missing data refers to the absence of values or information for certain variables within a given dataset.

  Participant P1 provided an illustrative example of a typical approach when dealing with this topic before the participants of the first session started to discuss the first issue: *"For instance, in the context of a regression problem, it's common practice to perform a process known as form filling, if the project' schedule permits. This often arises when there are numerous gaps in the dataset. However, in certain situations where we are confident that a specific variable remains relatively stable over a given time frame, we can propagate the preceding value as a solution"*.

  1. ***Not identifying missing values:***

     In the first session, regarding the question about occurrence, three participants reported partially disagreeing, and one wrote partially agreeing. Regarding the question about relevance, one participant reported disagreeing, and three reported partially disagreeing.

     Participant P2 stated: *"When confronted with missing data mistakes, you typically have two scenarios: leave it unaddressed (missing) or handle it incorrectly. Rarely do you manage it incompletely"*.

     Participant P3 stated: *"It has happened to me that we needed to create a model, and I tried to do it, skipping this step. I had to backtrack because I couldn't progress, and I stopped. Therefore, in this sense, no TD was generated because it was not completed; hence, it does not fall into the category of TD"*.

     In the case of this proposed issue, all participants agreed unanimously on both questions in the second session.

     Despite the diversity of the votes, the median leads us to conclude that this issue can be considered highly relevant, as shown in Figure A.7.

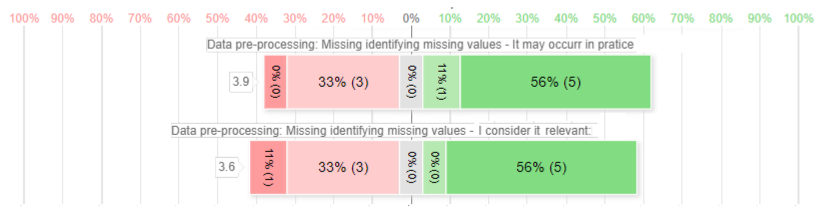


Figure A.7: Total - Not identifying missing values

2. ***Incomplete missing value identification:***

   In the first session, the first question, two participants reported disagreeing, and two reported partially agreeing. In the second question, two participants disagreed, and two reported partially agreeing.

   In the second session, all participants agreed on both questions.

   Participant P5 emphasized: *"Overall, a problem resulting from a step performed incompletely is challenging to identify because it can give the impression that it was both done and not done. In this case, perhaps pair programming can be helpful, as your partner may be able to spot this issue more easily. Having a checklist of steps is also very helpful"*.

   Participant P7 exemplified: *"Another example would be when values of 0 are considered empty, but in some cells, the value is -1, which has also been treated as empty. Addressing only the 0 values and not handling the -1 would be an incomplete approach"*.

   In this focus group session, based on the data presented in Figure A.8, it is reasonable to categorize this issue as highly relevant.

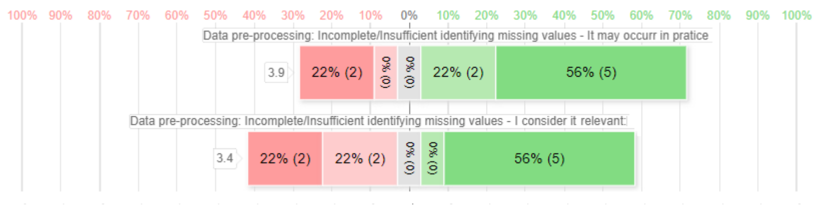


Figure A.8: Total - Incomplete missing value identification

3. ***Inaccurate missing value identification:***

   In the first session, in both questions, all participants reported agreeing.

   Participant P1 expressed: *"Usually when you encounter problems, you start looking at the end of the whole process. You question whether the model is performing well, if the model's hyper-parameters are properly tuned, and if the chosen technique is the* *best option. Having to go back to the pre-processing stage can be quite challenging. So, any incorrect step in the pre-processing phase that needs correction should be taken seriously, as it can have serious repercussions"*.

Participant P2 affirmed what P1 said: *"The issue here is that processes are functional, which means things are working; however, the outcomes achieved are not optimal, and it's challenging to identify the root causes precisely"*.

Participant P3 said: *"Imagine a form which primarily accepts text inputs. For instance, in the address field, someone entered "none." While there is no missing data, the provided value seems irrelevant. It would have been important to recognize and address this inconsistency, which was disregarded"*.

In the second session, in both questions, all participants reported agreeing.

All possible votes were in complete agreement regarding the occurrence and its relevance, and this issue can be categorized as highly relevant, as shown in Figure A.9.

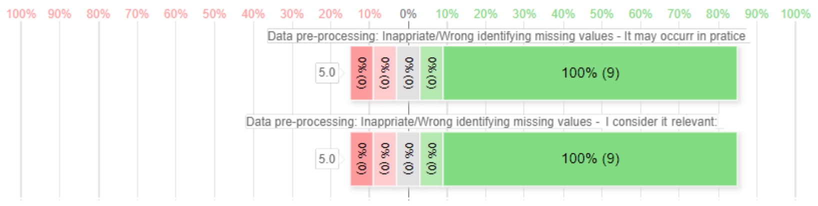


Figure A.9: Total - Inaccurate missing value identification

– **Rebalancing (typically by resampling)** - Example: Resampling is a technique that entails iteratively drawing samples from the training dataset, which can be done to address issues like imbalanced data or to improve the robustness of a machine learning model.

1. ***Missed rebalancing (typically by resampling):***

   In the first session, all participants reported partially agreeing with both questions.

   In the second session, four participants reported agreeing, and one was unsure about the first question. Three participants reported partially agreeing, and two agreed with the second question.

   Participant P7 argued: *"Depending on the dataset, this may not be such a relevant issue"*.

Participant P5 completed saying: *"If you don't perform resampling but use appropriate evaluation metrics, it may not be considered as a TD."*.

Although there is divergence regarding the severity of occurrence and relevance, to some degree, all voters agreed that this issue is low relevant, as shown in Figure A.10.

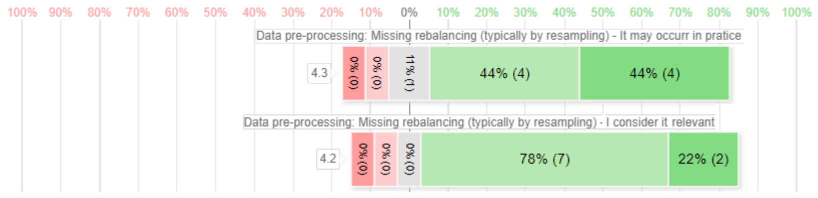


Figure A.10: Total- Missed rebalancing (typically by resampling)

2. ***Incorrect rebalancing:***

   One participant reported partially agreeing in the first session, and three agreed with the first question. One participant reported partially disagreeing, two reported partially agreeing, and one agreed with the second question.

   Participant P1 remarked: *"Upon recognizing the error, it becomes necessary to revisit the data distribution and conduct an exploratory analysis. It's crucial to understand why no results are being found, especially when limited data of this type is available for training. In other words, there's an investigation needed to discover the error. Debugging is always a bit frustrating."*.

   In the second session, all participants agreed on both questions.

   Participant P5 concluded: *"It can significantly result in generating incorrect data for training"*.

   The final result can be analyzed in Figure A.11, and as a result, in these focus group sessions, this issue can be classified as highly relevant.

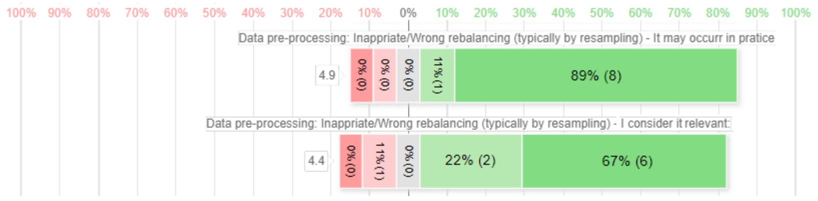


Figure A.11: Total - Incorrect rebalancing

– **Removing inconsistent data (remove the inconsistency)** - Example: Initial text pre-processing includes tasks such as eliminating white spaces and standardizing capitalization to address inconsistencies in the text data.

1. ***Not removing inconsistent data:***

   In the first session, in the first question, all participants reported partially agreeing. Three participants reported partially agreeing with the second question, and one said agreeing.

   Participant P1 mentioned a previous experience: *"In one of the projects, a situation arose where, following the technical manual, values were identified that didn't make sense when the variables were analyzed together. It was necessary to perform a manual analysis to address this inconsistency, as it couldn't be resolved through code. An expert in the domain had to alter the data directly"*.

   In the second session, all participants unanimously agreed on both questions.

   Participant P5 provided an example related to image recognition: *"This issue won't lead to coding rework but will necessitate redoing all the training steps. Consider, for example, if you're working with images, and a single training cycle takes five days. If you overlook this step, intentionally or unintentionally, you'll have to undergo the entire training process again, which could pose a substantial problem"*.

   The outcome can be analyzed using Figure A.12, providing consistency to assert that this issue can be classified as highly relevant.

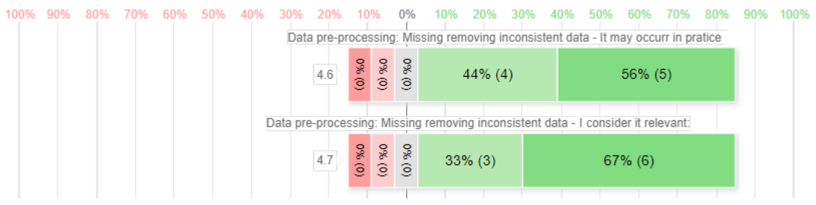


Figure A.12: Total - Not removing inconsistent data

2. ***Incomplete data removal:***

   In the first session, three participants reported partially agreeing, and one said agreeing in the first question. All participants reported partially agreeing with the second question.

   During the second session, all participants voted in agreement on both questions.

   Participant P5 exemplified this item: *"I have an interesting example of a diabetes dataset I've worked with. In this dataset, when values* *are missing, or columns are blank, they are filled with zeros. However, if you're unfamiliar with what each column represents, this can lead to misinterpretations. For instance, consider a column representing blood pressure; it's clear that a blood pressure of zero doesn't make sense. Imagine someone unfamiliar with the medical specifics, assuming zero is a valid value for blood pressure. This underscores the importance of deeply understanding the domain of the data you are working with".*

Total outcome refers to Figure A.13 for a detailed analysis. Based on this, we can categorize it as highly relevant.

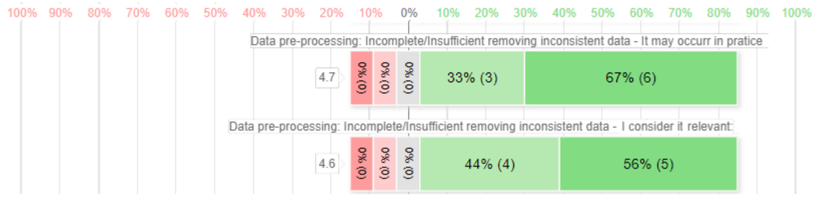


Figure A.13: Total - Incomplete data removal

3. ***Removing inconsistent data inappropriately:***

   In the first session, regarding the question about occurrence, two participants reported partially agreeing, and two said agreeing. Regarding the question about relevance, all participants reported agreeing.

   In the second session, all participants agreed on both questions.

   Participant P5 made a brief comment: *"This scenario is quite problematic, as the improper removal of inconsistent data can lead to the loss of critical information for training".*

   This issue can be categorized as highly relevant since all participants agreed on its relevance. For a more comprehensive analysis of the overall results, follow Figure A.14.

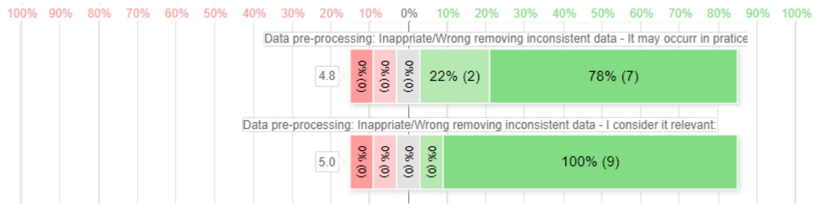


Figure A.14: Total - Removing inconsistent data inappropriately

– **Pre-processing scaling** - Example: For instance, employing normalization, standardization, and logarithmic transformation.

1. ***Missed pre-processing scaling (e.g., normalization, standardization, logarithmic transformation):***

   In the first session, in the first question, three participants reported partially agreeing, and one said agreeing. In the second question, two participants reported partially disagreeing, one partially agreeing, and one agreeing.

   At this point, Participant P2 argued: *"I realize that identifying TD in experimentation activities is quite abstract, mainly because there's always room for improvement. Even reaching an optimal solution is possible, but it's legitimate to question why we tested only two possibilities. Why didn't we test three, four, or as many possibilities? It's more about an opportunity for improvement"*.

   Participant P1 continued saying *"Exactly, it's an opportunity for improvement, and there is a limit to how much you should keep trying to enhance it. What's important is to consider the cost-benefit of continuing to invest in it"*.

   In the second session, all participants unanimously agreed on both questions.

   A more in-depth analysis of the final result in Figure A.15 highlights that, although there may be differing opinions regarding its relevance, there is no question about its occurrence. Based on that, we can categorize this issue as highly relevant.

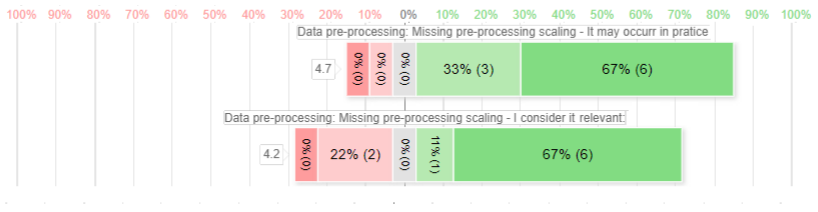


Figure A.15: Total - Missed pre-processing scaling

2. ***Incomplete pre-processing scaling (e.g., normalization, standardization, logarithmic transformation):***

   All participants reported partially agreeing with the first question in the first session. Two participants reported partially disagreeing, and two partially agreeing with the second question.

   In the second session, all participants unanimously agreed on both questions.

   Participant P5 said: *"You don't necessarily need to normalize all numeric columns. You can choose to normalize just a few. For ex-*

ample, you might normalize a salary column with a wide range of values while leaving others unnormalized. However, if you accidentally overlook the column that should be normalized, it can lead to poor results, and you might mistakenly believe that you have the best possible model".

A more detailed examination of the outcome in Figure A.16 reveals that, despite differing opinions on its relevance, there is no doubt about its occurrence. We can categorize it as highly relevant using the median.

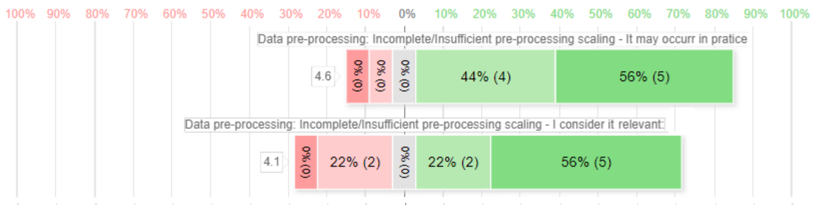


Figure A.16: Total - Incomplete pre-processing scaling

3. ***Inaccurate pre-processing scaling (e.g., normalization, standardization, logarithmic transformation):***

   In the first session, in both questions, all participants reported agreeing.

   Participant P1 gave an example: *"A simple example of incorrect scaling is to apply it to the training set, and then apply it incorrectly to the test set. This is quite wrong and a common example. Another example is applying one approach to the training set and a different one to the test set or swapping columns and rows".*

   One participant reported partial agreement in the second session, while four fully agreed in the first question. Similarly, one participant reported partial agreement for the second question, and four fully agreed.

   Participant P5 said: *"In an example where, instead of standardization, normalization was applied, machine learning algorithms have mechanisms to identify associations, which can mitigate problems. Although this is not the most appropriate approach (normalizing instead of standardizing), it is at least addressed in some way. Documenting the decisions you make is always a good practice, as it can facilitate the process if, by any chance, you need to return to this stage".*

   There is nearly unanimous consensus regarding its existence and relevance. For a more detailed examination of the final results, please consult Figure A.17. We can categorize this issue as highly relevant based on these facts.

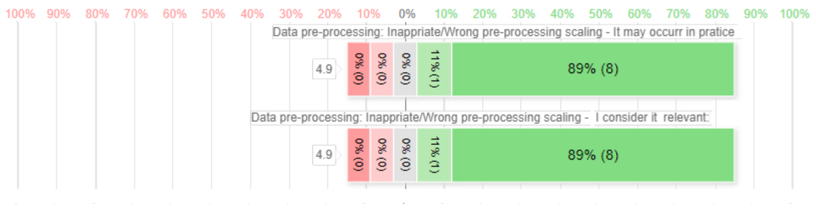


Figure A.17: Total - Inaccurate pre-processing scaling

A.2 Issues of the Model Creation and Training ML Life Cycle Phase

– **Testing candidate algorithm possibilities** - Example: For Classification, regression, clustering, and association algorithms.

1. ***Neglected testing of candidate algorithm possibilities:***

   In the first session, in the first question, three participants reported partially agreeing, and one said agreeing. In the relevance question, one participant reported partially agreeing, and three reported agreeing.

   In the second session, all participants agreed on both questions.

   Participant P7 said: *"An example that comes to mind in this regard is when someone achieves such an outstanding result right from the start that they don't explore other possibilities"*.

   The final results depicted in Figure A.18 unmistakably establish that this issue, in alignment with these focus group sessions, can be classified as highly relevant.

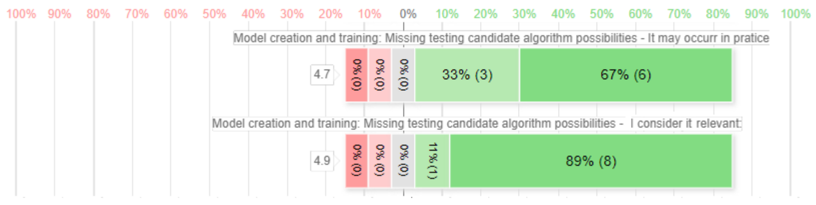


Figure A.18: Total - Neglected testing of candidate algorithm possibilities

2. ***Insufficient exploration of candidate algorithm options:***

   In the first session, the first question, two participants reported partially agreeing, and two said agreeing. In the second question, two participants reported partially disagreeing, and two reported partially agreeing.

In the second session, three participants reported partially agreeing, and two agreed with both questions.

Participant P5 explained your perspective: *"It's important to remember that the incomplete approach should be used cautiously because it's impossible to test all existing algorithms, many of which may not even be known. In this context, incompleteness is an inherent limitation, and it's essential to prioritize the most relevant and suitable algorithms for the given problem. The choice of algorithms to be tested depends on various factors, such as the available time and your familiarity with the algorithms"*.

In terms of occurrence, there is consensus. However, regarding relevance, it is clear that this issue has low relevance. This low priority may be explained by the comment made by participant P5, who mentioned that testing candidate algorithms will always be insufficient. Figure A.19 illustrates the result.

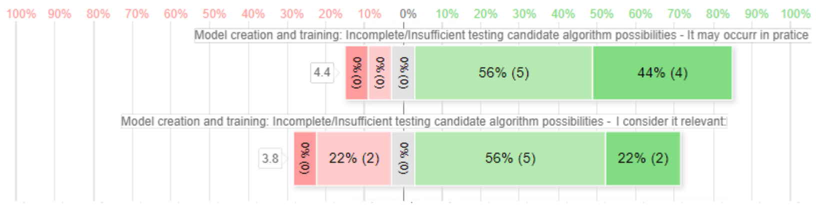


Figure A.19: Total - Insufficient exploration of candidate algorithm options

3. ***Inappropriate/Wrong exploration of candidate algorithm options:***

   In the first session, the first question, one participant reported disagreeing, and three reported being unsure. In the second question, regarding relevance, all participants reported being unsure.

   In the second session, both questions yielded identical results: two participants reported partial agreement, while three fully agreed.

   Participant P5 stated: *"Handling it inappropriately can lead to a more complex code, but it's likely that the final result won't be significantly compromised"*.

   Many participants marked "not sure" regarding the occurrence and relevance in the first session, suggesting we can rule out this issue. Furthermore, the partial agreement of two voters in the second session, both in terms of occurrence and relevance, reinforces this conclusion.

– **Splitting training/test/validation data** - Example: Training data selection, holdout

1. ***Neglected splitting training/test/validation data***

   In the first session, in the question regarding occurrence, two participants reported partially disagreeing, one reported partially agreeing, and one said agreeing. In the relevance question, all reported agreeing.

   Participant P3 said: *"It doesn't make much sense to a data scientist (missing case). It is a basic premise. I don't see this happening. People usually don't forget to perform the data split, and if there is an error in the split, such as data leakage from the training set to the test set, that would be a case of inappropriateness"*.

   Both questions yielded identical results in the second session: all participants reported agreement.

   Figure A.20 suggests that, based on both sessions, this issue can be categorized as highly relevant despite some partial disagreements related to its occurrence.

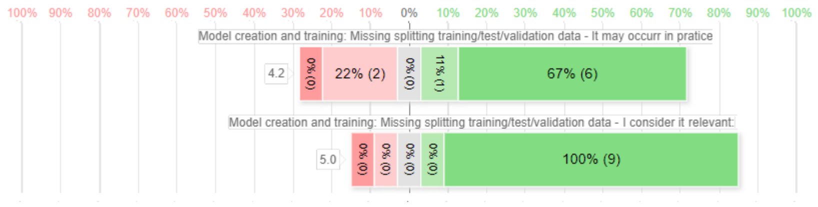


Figure A.20: Total - Neglected splitting training/test/validation data

2. ***Inappropriate splitting of training/test/validation data***

   In the first session, in the question regarding occurrence, two participants reported partially agreeing, and two said agreeing. In the second question, all reported agreeing.

   Participant P3 gave an example: *"This is an illustrative example of this in cases involving time series. When you choose an approach that randomly separates the training and testing data, you are compromising the essence of this time series; this happens because, in time series, we are interested in capturing and preserving the patterns and behaviors that evolve, such as seasonality and trends."*.

   Regarding the second session, all participants agreed on both questions.

   The results can be examined in Figure A.21, and it can be concluded that this issue can be categorized as highly relevant.

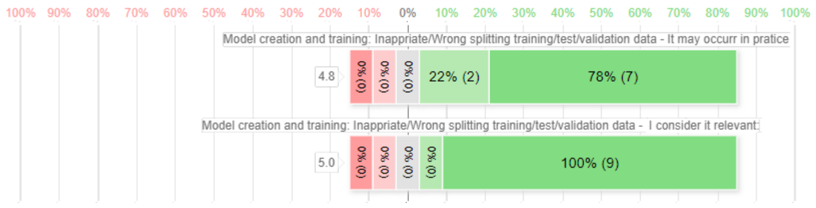


Figure A.21: Total - Inappropriate splitting of training/test/validation data

– **Hyperparameter tuning** - Example: Adjust parameters such as C (SVM), K and distance metrics (KNN), Information gain, and Gini index (Decision Tree).

1. ***Neglected hyperparameter adjustment:***

   In the first session, in the question regarding occurrence, one participant reported partially agreeing, and three said agreeing. In the relevance question, one reported partially disagreeing, two reported partially agreeing, and one said agreeing.

   One participant reported partially agreeing with both questions in the second session, while four expressed full agreement.

   Figure A.22 shows that we can categorize this issue as highly relevant, according to the discussions during the focus group sessions.

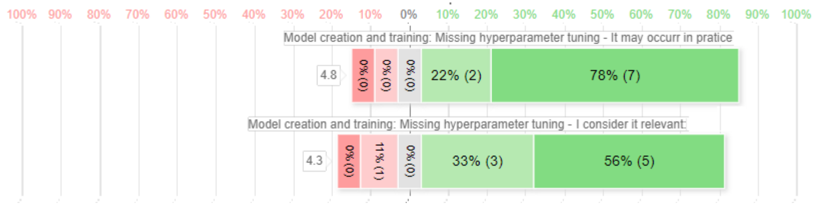


Figure A.22: Total - Neglected hyperparameter adjustment

2. ***Suboptimal hyperparameter adjustment:***

   In the first session, in the first question, all participants reported partially agreeing. In the second question, two said partially disagreeing, and two reported partially agreeing.

   Participant P1 made an addendum: *"It is challenging to establish an absolute criterion for determining what is considered insufficient, as the choice of approach can vary depending on the context and complexities of the problem. Determining whether something was incomplete or insufficient is difficult, as it may not necessarily result in significant TD that requires revisiting the issue. Sometimes, it may not be necessary to go back and address it"*.

Participant P3 said: *"If you ask a data scientist, it will usually be considered insufficient because they are always looking for opportunities to continue testing and improving models"*.

Four participants reported partially agreeing in the second session, and one agreed with both questions.

Based on discussions from the two focus group sessions, we can categorize this issue as low relevant. The final result is shown in Figure A.23.

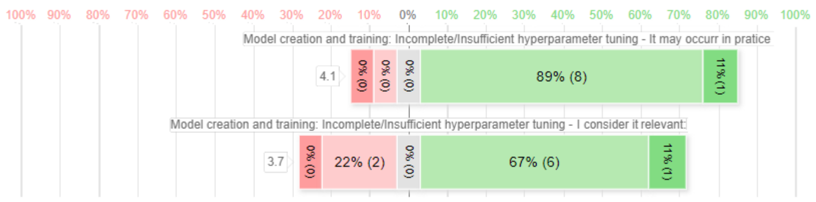


Figure A.23: Total - Suboptimal hyperparameter adjustment

3. ***Incorrect hyperparameter adjustment:***

   In the first session, in the question regarding occurrence, two participants reported partially agreeing, one reported agreeing, and one reported being not sure. In the relevance question, two reported agreeing, and two reported being not sure.

   Participant P1 gave an uncommon example that could happen: *"It is possible to create a script to test various hyperparameters, but it can be challenging to avoid human errors, such as accidentally changing the input data or forgetting to adjust the hyperparameters for a specific algorithm. In a scenario where you have developed code to test different algorithms and hyperparameters, it can be easy to make the mistake of not consistently applying the appropriate hyperparameters to each algorithm. These situations are uncommon"*.

   In the second session, all participants agreed with both questions.

   In this case, it is possible to observe a difference in voting patterns between the first and second sessions. In the first session, votes were categorized as 'not sure'; in the second session, there was unanimous consensus on both questions that this issue is relevant. However, this difference allows us to categorize this issue as highly relevant according to the discussions in the focus group sessions. Figure A.24 reveals the final result.

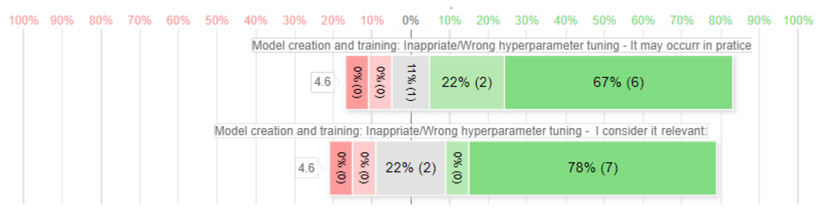


Figure A.24: Total - Incorrect hyperparameter adjustment

A.3 Issues of the Model Evaluation ML Life Cycle Phase

– **Choosing evaluation metric** - Example: Accuracy, Confusion Matrix, AUC, Precision, Recall, F-Measure (Classification); MRSE, R-Squared (Regression).

1. ***Suboptimal evaluation metric selection:***

   In the first session, the first question, three participants reported partially agreeing, and one said agreeing. In the second question, all reported agreeing.

   Participant P1 stated: *"For example, consider an alarm in a secret room with millions of dollars. The alarm should go off when there is no real threat and vice versa. You may not notice this aspect if you don't use the right metric. In this case, it's not about doing something wrong, but rather using an insufficient metric that doesn't provide adequate clues, making it challenging to interpret"*.

   In the second session, the first question, three participants reported agreeing, and two said they were not sure. In the second question, two reported agreeing, and three were not sure.

   Participant P5 made a statement about this case: *"The use of 'incomplete' does not necessarily generate TD. It may be the case of needing to fully meet the client's needs optimally. For example, when creating a model with excellent accuracy but without considering execution time, it is possible that the client is not being served in the best way. In this scenario, there may not be a problem in the code itself but in satisfying the client's needs"*.

   Figure A.25 illustrates the conclusive outcome of the voting process for this particular issue. Based on the discussions, it can be concluded that this issue can be categorized as highly relevant. The votes on its occurrence and relevance exhibited variability, leading to this determination.

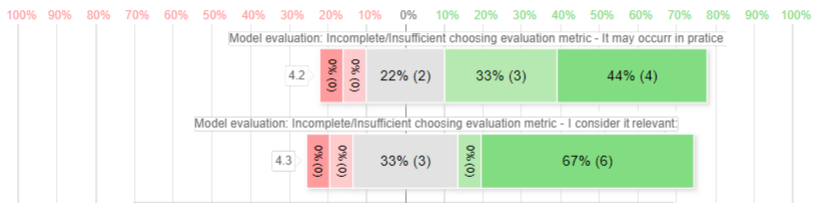


Figure A.25: Total - Suboptimal evaluation metric selection

2. ***Inappropriate/Wrong selection of evaluation metric:***

   In the first question during the first session, three participants reported partially disagreeing, and one said agreeing. In the second question, all reported partially agreeing.

   Participant P4 provided an example to illustrate their understanding of this concept: *"Utilizing a metric designed for a different type of problem, like employing a regression metric to assess a classification problem, is unsuitable"*.

   The outcome of the first session is intriguing. Although this issue is not considered common in practice, it is still relevant.

   In the second session, both questions received equal responses from participants, with two indicating agreement and three voting being not sure. This suggests a mixed reaction to the questions, with some participants leaning towards understanding and another segment needing to be more certain.

   Participant P8 said: *"In my opinion, choosing the wrong metric doesn't necessarily result in a TD, but it can cause harm to the business"*.

   The outcome is that this issue can be discarded.

– **Properly using methods for evaluating a model's performance** - Example: Holdout and Cross-validation

1. ***Inappropriate/ Wrong properly using methods for evaluating a model's performance:***

   In the first session, in the first question, three participants reported partially agreeing, and one said agreeing. In the second question, one reported partially agreeing, and three said agreeing.

   In the second session, the first question, two participants reported agreeing, and three said they were not sure. In the second question, one reported partially agreeing, one reported agreeing, and three said not sure.

Participant P5 comment: *"It doesn't seem as bad as those items before"*.

There was only one vote fully agreeing in the first session on the first question and a relatively high number of abstentions during the second session on the same question; we can conclude that this issue can be discarded.