Titre: Title:	Machine Learning Engineering: An Exploratory Study of Challenges and Practices
Auteur: Author:	Alaleh Hamidi
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Hamidi, A. (2021). Machine Learning Engineering: An Exploratory Study of Challenges and Practices [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/6569/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6569/
Directeurs de recherche: Advisors:	Foutse Khomh, & Giuliano Antoniol
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Machine Learning Engineering: An Exploratory Study of Challenges and Practices

ALALEH HAMIDI
Département de génie informatique et génie logiciel

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

Mai 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Machine Learning Engineering: An Exploratory Study of Challenges and Practices

présenté par **Alaleh HAMIDI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Michel DESMARAIS, président
Foutse KHOMH, membre et directeur de recherche
Giuliano ANTONIOL, membre et codirecteur de recherche
Heng LI, membre

DEDICATION

Dedicated to
My dear fiance, Jaber
My beloved mother, my lovely siblings, and my lovely brother-in-law
My guardian angel niece forever, Avina
Thank you for your endless love,
sacrifices, and support . . .

ACKNOWLEDGEMENTS

I would like to acknowledge all who in one way or another contributed to the completion of this thesis. First and foremost, I wish to express my deep and sincere gratitude to my supervisor Prof. Foutse Khomh and my co-supervisor Prof. Giulio Antoniol for their solid support, motivation, caring and patience, as well as for providing me with an excellent atmosphere in which to conduct my research. Their vast knowledge and logical way of thinking helped me throughout this process. Their knowledgeable and warmth made the collaboration highly beneficial for me. It was a pleasure to work with them; the lessons they taught me go beyond what is written in this thesis and will help me in all aspects of my life.

I would like to acknowledge the members of my committee, Prof. Heng Li and Prof. Michel Desmarais for taking interest in my work, examining my thesis and providing insightful comments.

I would like to acknowledge the other professors and technical staff at the Computer Engineering Department at École polytechnique de Montréal for providing the best possible assistance with my project.

I would also like to extend my thanks to all my colleagues in Prof. Antoniol's and prof. Khomh's groups for sharing their knowledge and ideas, especially Dr. Md Saidur Rahman, and to all my colleagues who helped me at various points during the M.Sc. thesis process.

I am also thankful to Mr. Alexis Geslin for translating abstract to French.

And last but not least, from the bottom of my heart, I would like to express my endless gratitude to my adored fiance, Jaber, dear mother, lovely siblings, my brother–in–law, my lovely niece, Avina, and also all my friends for the absolute support they have provided me throughout my entire life, and without whose love, spiritual encouragement and altruism I would not be where I am now. Words are powerless to express what I feel in my heart for them.

RÉSUMÉ

Alors que la demande pour des systèmes et services intelligents augmentent rapidement, l'apprentissage automatique (ML) devient de plus en plus un élément indispensable des systèmes logiciels. L'intelligence artificielle (IA) et l'apprentissage automatique (ML) ont attiré l'attention des chercheurs en raison de leur application pour fournir des solutions basées sur les données à divers problèmes du monde réel. Bien que le ML prouve des potentiels prometteurs pour le génie logiciel (SE), il souffre encore de certains défis.

Malgré certains efforts de recherche récents, des défis demeurent. Nous menons la première partie de notre étude pour connaître les perceptions des praticiens du ML concernant ce sur quoi le chercheur devrait se concentrer pour innover et soutenir les développeurs afin d'assurer la qualité des applications de ML.

Dans la première étude, nous menons une enquête auprès de 80 praticiens du ML (avec des compétences, une expérience et des domaines d'application diversifiés) pour rapporter les commentaires sous forme de 17 résultats décrivant les défis et les meilleures pratiques du développement d'applications de ML.

Ce travail résume les meilleures pratiques des praticiens dans le développement de systèmes logiciels basés sur le ML pour améliorer la qualité de leurs systèmes. De plus, il rapporte que le ML est un défi que la communauté de recherche doit étudier.

La deuxième partie de cette recherche a abordé les défis ML plus spécifiquement pour comprendre dans quelle mesure les défis ML sont liés au langage de programmation (PL) utilisé dans le développement d'applications ML. La deuxième étude a analysé les publications de Stack Overflow (SO) liées au ML, afin d'étudier comment les questions sur le ML ont été modifiées au fil du temps et dans six langages de programmation différents. Nous avons analysé 43950 messages SO de 2008 à 2020 pour déterminer (i) comment le nombre de messages liés au ML évolue au fil du temps pour chaque langage de programmation, (ii) comment les messages sont répartis entre les différentes phases d'un pipeline ML, et (iii) si les messages appartenant à différentes langues ou phases sont plus ou moins difficiles à traiter.

Nous avons réalisé que certains PL s'estompent avec le temps tandis que d'autres deviennent de plus en plus populaires pour le développement d'applications ML. En outre, pour tous les PL sélectionnés, le nombre de messages liés à la phase de construction du modèle est supérieur à celui des autres phases. De manière générale, cette étude confirme la présence de tendances pour tout les languages.

ABSTRACT

As the demands for intelligent systems and services is growing fast, Machine Learning (ML) is increasingly becoming an indispensable element of current software systems. Artificial Intelligence (AI) and Machine Learning (ML) have grabbed researchers' attention because of their application for providing data-driven solutions to various real-world problems. Although ML proves some promising potentials for Software Engineering (SE), it still suffers from some challenges.

Despite some recent research efforts, it is still unclear where SE researchers should focus to support ML developers more effectively. We conduct the first part of our study to know the ML practitioners' perceptions regarding what the researcher should focus on to innovate for supporting developers and assure ML applications' quality.

In the first study, we conduct a survey on 80 ML practitioners (with diverse skills, experience, and application domains) to report the feedback as 17 findings outlining challenges and best practices of ML application development.

This work summarizes practitioners' best practices in the development of ML-based software systems to improve the quality of their systems. Moreover, it reports ML challenges that the research community needs to investigate.

The second part of this research has discussed the ML challenges more specifically to understand to what extent the ML challenges are related to the Programming Language (PL) used in ML application development. The second study analyzed Stack Overflow (SO) posts related to ML, to investigate how the questions about ML have been changed over time and across six different programming languages. We analyzed 43,950 SO posts from 2008 to 2020 to determine (i) how the number of ML-related posts changes over time for each programming language, (ii) how the posts are distributed across different phases of an ML pipeline, and (iii) whether posts belonging to different languages or phases are more or less challenging to address.

We realized that some PLs are fading over time while others are becoming more popular for ML application development. Also, for all selected PLs, the number of posts related to the Model-Building phase is more than the other phases'. However, the extent to which these questions are challenging to answer varied in different PLs. Generally, this study confirms the presence of trends for all languages.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

AI	Artificial Intelligence
CQA	Community Question and Answering
DSL	Domain-Specific language
KA	Knowledge Area
LDA	Latent Dirichlet Allocation
ML	Machine Learning
NLP	Natural Language Processing
PL	Programming Language
Q&A	Question and Answer
SE	Software Engineering
SME	Small and Medium-sized Enterprise
SO	Stack overflow
V&V	Validation and Verification

CHAPTER 1 INTRODUCTION

1.1 Context and Motivations

Since Artificial Intelligence (AI) and Machine Learning (ML) provide powerful remedies and tools as data-driven solutions for various academic and industrial problems, ML components are increasingly becoming vital in current software systems. Recent innovations in machine learning have significantly prompted the boosting adoption of AI capabilities to automate modern software and services by embedding ML components [1].

Nowadays, AI-based automated supports affect every aspect of human life: business, education, healthcare, research, communication, security, assistive technologies, and so on.

Application domains' variety leads to significantly different types of problems, data characteristics, and consequently, the ML algorithms. From an engineering viewpoint, various requirements should be met during the implementation of an algorithm; a well-built architecture, validation of data and model, proper application monitoring, dedicated release engineering methodologies, wise selection of design patterns and security controls, and proper adjustment based on recent user's experiences and needs. Even minor negligence to address these challenges can cause disastrous consequences. Classical software system construction is deductive or conducted by written rules that shape the system behaviors as program code. ML techniques can generate such rules inductively from training data. As the ML application development does not readily fit into the traditional software engineering process, we need a shift of software development paradigm that induces some unique challenges to ML application development [2, 3].

Recently, dominant software companies like Google [4], and Microsoft [1] have revealed their practitioners' experience during building ML-based applications to raise awareness on some of the ML application development challenges.

Sculley et al [4] listed and characterized some ML application development challenges by determining destructive design patterns that may impose high maintenance costs. They also suggest some ways to deal with those challenges.

Amershi et al [1] conducted a survey with Microsoft developers to illustrate AI application development through a nine-stage development workflow and identify fundamental differences between ML application development and traditional software development. According to this study, ML-based development requires coping with various workflow phases, such as model requirements, data collection, cleaning and labeling, feature engineering, model train-

ing, and evaluation, until deployment and monitoring. They also provided ML applications' software engineering with some best practices for data and model management, and the interfaces between ML elements and the whole system. This work provides the basis for both of our studies.

These studies (i.e., [1], [4]) have provided a list of the challenges of AI/ML application development, but in the context of large companies. We can see the lack of studies that provide proper insight into how small and medium-sized enterprises (SMEs) operate ML application development. It is essential to consider various domains and development settings to implement best practices building ML applications. The first part of our research is to investigate experiences and collect insights from ML practitioners worldwide with different levels of skills and experiences in diverse development domains.

Some studies identified question and answer (Q&A) forums such as Stack Overflow (SO) can provide researchers with a potential platform to study challenges developers face in a given field by analyzing the discussed posts [5, 6]. Since ML has recently captured the growing attention of software developers, especially after 2015 [7], the number of ML-related posts on SO is rapidly increasing [6, 8, 9].

However, developers may encounter various challenges when dealing with ML in different Programming Languages (PLs). Some conditions for the application under development may compel developers to use a language that may not be appropriate for ML-intensive development according to its characteristics or the provided libraries.

While previous work [8] has characterized the encountered ML-related challenges based on ML workflow, we don't have any knowledge about its evolution and variation across Programming Languages (PLs). Moreover, we didn't find any study regarding the characteristics of challenges developers face when using a specific PL for ML application development. This lack of research encouraged us to carry out our second study to analyze how the characteristics and evolution of SO questions about ML across six different programming languages.

1.2 Research Objectives

As the first study, we present the results of a survey about ML development practices and insights. We identified the participants based on their profile information on LinkedIn or their contribution to an ML project on GitHub. We emailed our questionnaire to over 700 AL/ML practitioners with diverse ML skills and knowledge levels and received 80 complete responses.

We analyzed the data from complete responses to derive ML practitioners' insights and

summarize them according to four main phases of the ML development workflow proposed in [1].

The contributions of this study are as follow:

- We conduct a comprehensive survey involving 80 ML practitioners with various technical or professional backgrounds to determine the characteristics of practices and challenges in ML application development.
- Our survey covers four key phases of ML application development life cycle, namely (1) data collection and processing, (2) feature engineering, (3) model building and testing, and (4) integration, deployment and monitoring, to identify challenges and practices from practitioners' viewpoint.
- We report our 17 key findings to present how those findings can benefit researchers and practitioners in developing ML applications of high quality.

In the second study, we systematically investigated the challenges developers face in each phase of the ML development life cycle when using a specific programming language. We analyzed 43,950 ML-related questions posted on SO from 2008 to 2020 to address the following research questions:

- **RQ1**: How does the number of ML-related posts related to different programming languages change over the years?
- **RQ2**: How are posts distributed across different phases of an ML pipeline, and how does this change over time?
- **RQ3**: To what extent are posts belonging to different languages and different phases answered, and after how long?

We first analyzed the distribution of posts over the years to understand how this distribution varies across Programming Languages (PLs). We totally considered six PLs, including popular PLs as well as the ones used in scientific/statistical computation; C/C++, C#, Java, Python, MATLAB, and R.

Then, two of the researchers manually inspected the high-rated posts to classify them based on the ML phases proposed by Amershi et al. [1], to understand the distribution of posts over ML phases and time.

Finally, to measure the difficulty level of a question, we adopted a method similar to previous works [8], [9], [10], [11], and [12].

We calculated the percentage of questions with no accepted answers or with no answer at all, as well as the median time a question takes to get an accepted answer.

According to the results, as one may expect, Python has the lion-share of the ML-related posts, which was not unexpected. PLs such as R are still very popular, while ML-related questions for other languages such as C, C++, or Matlab tend to increase at a slower rate or not increase at all. After considering the high-rated posts, we realized that model building, evaluation, and requirement phases involve the most frequent concerns.

1.3 Thesis Structure

This thesis is organized as follows. Chapter 2 overviews the related literature on Stack Overflow and Machine Learning challenges and practices. Chapter 3 presents our second study about ML challenges related to using some programming languages for ML application development. In chapter 4, we interpret and explain the findings of our studies. Finally, Chapter 5 includes the summary of our research and its limitations along with potential future researches. The results and findings of these studies have been collected as two papers and are ready to submit.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we describe the works in Machine learning challenges and categorize these challenges based on the last version of the guide to the Software Engineering Body of Knowledge (Swebok Version 3.0) as of February 2021. But as some of these works use Stack Overflow (SO) to find the challenges, we decided to present studies around SO first.

2.1 Stack Overflow (SO)

Community Question and Answering (CQA) platforms provide practitioners with a means to ask for help and solutions regarding their information needs and challenges. These platforms are increasingly gaining ground recently, and the number of posts submitted on them is growing dramatically.

Stack Exchange is a network of 173 CQA websites, including Stack Overflow (SO), as the most famous computer programming-related CQA.

SO is one of the 50 most prevalent websites globally as it serves 100 million developers per month. It has almost 14 million registered users and includes 21 M computer programming-related questions, and 31 M answers as of March 2021 . Developers can ask their questions on SO, and other SO users can answer the questions. Each SO question mainly contains “Title,” “Body,” and “Tags”. The title is the first part that grabs the attention to the question. It includes a representative short message and summarizes the entire question. The question body should explain the problem in detail. The user must follow special formatting to insert a code in this part. Users can also put relevant links to improve the understanding of the question. The tags are the relevant keywords or phrases representing the primary topic of that question [13].

SO also provides users with specific guidelines and voting mechanisms to Upvote, Downvote, Closevote, Reopenvote, and Deletevote the questions and answers. Based on the guidelines, active users can vote to close the off-topic, inferior, or duplicate questions to keep the platform’s quality. They can also vote to reopen a closed question after getting appropriately modified. But if the improvement of that closed question is not feasible, it would be deleted. To summarize, SO users can post questions, assign a tag(s) to their posts, answer questions, modify their submissions, vote, and comment on posts.

Moreover, users may register on SO to obtain reputation points. The reputation of a user on SO represents how much that user is trustable through the community. If a user actively

engages in posting high-quality questions, giving helpful answers, and voting posts, his reputation will increase. SO users with high reputation points have rights like modifying the other users' responses [13, 14].

Parnin and Treude [15] conducted web searches for jQuery API using Google and analyzed 1,730 results to find that at least one SO result emerges on the first page of results in 84.4% of these searches. It means that developers can refer to answered SO questions for their development problems.

SO has become one of the most visited sites to transfer expert knowledge for software development. The answer rate for SO questions about expert topics is over 92%, with a median time of 11 minutes, and these facts show software developers have significant mindshare via the SO site. According to the users' report, they refer SO first for programming problems instead of web searches and forums; others put their SO portfolio answers in their professional resumes as valuable experiment [14]. All of these points about SO encourage Mamykina et al [14] to consider factors that cause this success using a combined approach, including statistical analysis and interviews. This study demonstrated that the reasons for SO's s success are 1) tight involvement of the design team with the community 2) very responsive and repetitive attitude to design 3) a reward system to promote beneficial user behavior.

Some studies demonstrate that SO is a valuable source for research. Using a data mining technique, Chen and Xing [16] show that SO tags are worthwhile to get an overall vision about technologies trend. Meldrum et al [17] studied 266 papers about the SO platform that have been published from 2009 to 2017 to show the growing academic interest in SO. This study reveals that SO has a significant impact on software engineering research.

To explore SO posts, researchers have used different methods like manual labeling, topic modeling, text mining, etc. Treude et al [9] manually labeled 385 SO questions to classify them into ten general categories to determine the question types asked by developers. Barua et al [18] applied Latent Dirichlet Allocation (LDA) topic modeling on SO posts to extract raised topics and consider their transition over time. Joorabchi et al [6] analyzed 186,000 SO posts with a text mining approach to study the challenges software practitioners face in different topics and subjects.

Some works investigate SO posts based on a particular aspect. For example, Schenk et al [19] analyzed the transactions in SO according to users' geographical regions to understand how the users from each area contribute to knowledge transfer on the SO platform.

Some works propose some models or techniques to improve SO functionality. Stanley and Byrne [20] suggested a method based on the Bayesian probabilistic model to predict a SO

post's tags. Yang and Manandhar [21] presented a new probabilistic model to represent users' expertise according to their answers and descriptive capacity based on questions.

Since the quality of posts plays a significant role in SO's popularity, some researchers conduct studies on this topic. Baltadzhieva and Chrupała [22] proposed a technique to predict the quality of SO questions. Also, they investigated how the quality of a SO question is affected by features like question tags, length of the question title and body, presence of a code snippet, the user reputation, and terms used to formulate the question. This work considers the relation between question quality and the likelihood of getting an answer. It shows that high-quality questions are more likely to respond because the high-quality questions may attract more users' attention to answer the question [23].

While few would wonder about SO's usefulness to software developers, questions concerning the quality of the answers generated to questions are abundant. Jin et al [24] analyzed 101,291 members' posts to understand how a reward-driven approach can affect answer quality or the tendency to underrate the quality of answers provided by others. Anand and Ravichandran [25] understood the necessity to separate answer quality and its upvote points because the reward system offered by SO can sometimes conflict with the quality of answers. In other words, having a high number of upvotes doesn't necessarily mean a post has a high quality. These works recommend wariness when approaching SO for proposed solutions that address software engineering challenges.

The other issue about SO posts is unanswered questions. Even with developers' active participation in SO [26], there is a growing number of unanswered SO questions over time [27]. According to evidence, although the number of practitioners who answer SO questions increases, the number of unanswered questions is growing even faster in recent years [28].

Yang et al [29] did extensive experiments on 76,251 questions to study the relation between a user's reputation and the likelihood of getting an answer to the question he has asked. They realized that new users' questions are less likely to get answers than experienced users.

Asaduzzaman et al [27] conducted a qualitative study and realized that too short questions are less likely to get an answer because those questions may omit the critical information. Besides, this work shows that the most significant proportion of the unanswered questions can't fascinate any community expert to answer them.

Yang and Manandhar [21] identify that lack of relevant experts is the most significant factor that causes unanswered questions. Like this study, Alshangiti et al [8] also found an intense connection between lack of ML experts and unanswered ML-related questions in SO.

2.2 Machine Learning Challenges

Swebok Version 3.0 [30] is the most recent revised version of Guide to the Software Engineering Body of Knowledge as of February 2021. We can find categorized SE concepts in 15 knowledge areas (KAs) of Swebok 3.0: 1) Software Requirements 2) Software Design 3) Software Construction 4) Software Testing 5) Software Maintenance 6) Software Configuration Management 7) Software Engineering Management 8) Software Engineering Process 9) Software Engineering Models and Methods 10) Software Quality 11) Software Engineering Professional Practice 12) Software Engineering Economics 13) Computing Foundations 14) Mathematical Foundations 15) Engineering Foundations.

We only consider the KAs specific to software engineering (KA 1–12) to organize SE challenges. We ignore the KAs 13-15 because these KAs are also foundational for other engineering fields.

2.2.1 Software requirements

"*The Software Requirements knowledge area (KA) is concerned with the elicitation, analysis, specification, and validation of software requirements as well as the management of requirements during the whole life cycle of the software product.*" [30]

ML applications' software requirements imply ML-specific activities, namely, data and feasibility analysis, requirements elicitation, requirement specifications, and ML functions and performances validation. These are challenging because the requirements may frequently change in large-scale systems such as automated vehicles; they are also very complicated [31, 32]. Since the need for better apprehends requirement changes in the requirements, Khalajzadeh et al [33] determined a research direction in developing tools that capture the requirements, changes in those requirements, and adaptations of specified processes. This work aims to better support domain expert end-users in their requirements management for AI-based systems and provide approaches to capture their requirements (not so much about the software solution but the domain problem).

ML techniques are broadly used and integrated into mission-critical systems; accordingly, safety, security, and V&V (validation and verification) have become critical issues [34]. Several studies have discussed Various topics on software requirements [35–38]. Developing domain-specific languages and tools for ML applications is a line of research for these challenges [35]. In addition to safety, the interpretability of ML applications and the ways to realize it have become hot topics in artificial intelligence (AI) communities (e.g. [38]). As an element of software requirement, interpretation has come out with ML techniques and

applications' progress. Fairness is another emerging element [39]. Requirement practices in data-oriented works cause new challenges. Lwakatare et al [40] reported the difficulties developers face when trying to specify desirable datasets. Furthermore, the need to maintain sensitive datasets' privacy and safety and respect legal compliance with new regulations such as the European General Data Protection Regulation may affect research lines in requirements engineering [41].

2.2.2 Software design

"*A software design (the result) describes the software architecture – that is, how software is decomposed and organized into components – and the interfaces between those components. It should also describe the components at a level of detail that enables their construction*" [30].

Kumeno [34] divided the challenges related to software design KA into the following categories:

- Security, Safety, and V&V: Design challenges for security, safety, and V&V of ML applications [35–37].
- Software Structure: Anti-patterns in ML applications [4], the complicated software modules of ML algorithms [42], and various design challenges in ML models (model selection, customization, and reuse) [1].
- Data Design: Design challenges for data collection, pre-processing, cleaning, labeling, and augmentation, including big data challenges [4, 33].
- Visualization: Technical challenges on visualization techniques for the design of ML applications [33].
- Tools: Designing ML applications tools, like tools for non-expert ML designers, visualization for understanding the relations between data and algorithms' behavior, ensuring interoperability with other means, and domain-specific language (DSL) support [1, 33].
- User Interface: Challenges on user interface design such as the interaction between users and ML applications [43].
- Automated ML (Auto ML): The design and deployment of effective ML models is time-consuming and requires a notable amount of resources and highly skilled developers. Since these demands retard ML applications in the industry, automated machine learning (AutoML) has emerged as a new research direction [44].

- ML techniques (except AutoML): Brodley et al [45] showed that application-driven research causes novel ML techniques. Kumeno [34] revealed that the contrary is also true. The challenges and solutions on ML techniques (e.g., data processing) and specific ML functions (e.g., interpretability) can affect ML applications' structure and implementation.

2.2.3 Software construction

"*The term software construction refers to the detailed creation of working software through a combination of coding, verification, unit testing, integration testing, and debugging. The Software Construction knowledge area (KA) is linked to all the other KAs, but it is most strongly linked to Software Design and Software Testing because the software construction process involves significant software design and testing*" [30].

As mentioned above, the Software Construction KA is firmly associated with the Software Design and Software Testing KAs. But the papers discussed for Software Design KA are not related to the Software Construction KA [35–37] because these papers don't consider the challenges of constructing ML applications. However, their topics are potentially close to construction challenges.

2.2.4 Software testing

"*Software testing consists of the dynamic verification that a program provides expected behaviors on a finite set of test cases, suitably selected from the usually infinite execution domain*" [30].

ML testing has attracted significant attention from researchers and industrial communities because it is both crucial and challenging. Several studies focused on ML testing, but many problems are without any solution and still coming out. Kumeno [34] has categorized the studies on ML testing challenges as follow:

- Survey papers on ML testing [46].
- Research papers on ML testing which also discuss challenges on testing [47].
- Survey papers on the security, safety, or V&V for ML applications testing [35].
- Survey papers on the data or model management for ML applications testing [39].
- Papers discussing the SE testing challenges for ML applications [4], or the testing challenges in an application domain [32].

Kumeno [34] also lists the various challenge topics on ML testing:

- Oracle Problem: To conduct authentic test oracles on ML applications more automatically.
- Cost Reduction: Cost-cutting ML testing techniques, including the cost reduction of traditional methodologies like search-based test case generation, test prioritization, and test-case minimization.
- Testing ML techniques: Challenges on testing ML algorithm such as unsupervised learning, reinforce learning, transfer learning, and meta-learning.
- Testing Properties: To test ML-specific properties like overfitting, interpretability, and fairness.
- Benchmarks: The design and establishment of reusable testing properties for ML applications.
- Testing Data: Testing for data validation and data cleaning. (Designing secure ML applications, adversarial testing on the training and testing data to detect privacy violations).
- Mutation Testing: The design and insertion of mutants to better simulation of real-world ML bugs.

2.2.5 Software maintenance

"*In this Guide, software maintenance is defined as the totality of activities required to provide cost-effective support to software. Activities are performed during the pre-delivery stage as well as during the postdelivery stage. Pre-delivery activities include planning for post-delivery operations, maintainability, and logistics determination for transition activities. Postdelivery activities include software modification, training, and operating or interfacing to a help desk*" [30].

In the real world, unpredicted events might happen in the deployment phase of ML applications. The ML production framework may have a vastly different situation from the environment for training and evaluating the ML models. ML models may be retrained again and again with concept drifts in ML applications, and thus their behavior may change autonomously in unexpected ways. These conditions can cause different maintenance challenges of ML applications [34]:

– Troubleshooting: to determine problems, identify the root causes and effects of failures, and fix faults (debugging) in the deployment phase. Automatic failures recovery involves reconfigurations and code repairs [1, 4, 40].

– Runtime Monitoring: Metrics selection for monitoring. Live monitoring of system behavior allows automated responses without direct human intervention, and dynamic monitoring enables developers for runtime verification and certification [4, 41]

– Data Management: Tools for data dependencies, concept drift adaptation, and automatic runtime data validation and cleaning [4].

– Model Management: ML model management challenges in the deployment phase, including the decision on model validation, model retraining, adversarial settings, and backward compatibility for trained models [48].

– Operating Environment: The deployment phase of ML applications most likely have additional functional modules than the existing system. The potential differences in the platform of deployment/operation phases and the training/evaluation environment of the ML model may cause compatibility, scalability, and portability challenges [31, 41].

2.2.6 Software configuration management (SCM)

"*Software configuration management (SCM) is a supporting-software life cycle process that benefits project management, development and maintenance activities, quality assurance activities, as well as the customers and users of the end product*" [30].

Configuration management of both complex data and software is essential to operate real-world ML applications. Amershi et al [1] allude "*machine learning is all about data. The amount of effort and rigor it takes to discover the source, manage, and version data is inherently more complex and different than doing the same with software code*". A prominent part of ML application practices involves configurable objects such as the models and their options and the data practices like the pre or post-processing of data [4]. Configuration management tools for ML applications can somehow address the ML model management challenges (as mentioned in the previous section).

2.2.7 Software engineering management

"*Software engineering management can be defined as the application of management activities – planning, coordinating, measuring, monitoring, controlling, and reporting – to ensure that software products and software engineering services are delivered efficiently, effectively, and*

to the benefit of stakeholders" [30].

This KA is mainly associated with software engineering project management. Kumeno [34] identifies the topic of challenges related to this KA:

- Risk Management: Risk management of the development, deployment, and operation of ML applications is essential, but various uncertainties make it problematic [4, 41].
- Effort Estimation: It is challenging to know the extent to which the ML model has met its objectives and estimate the number of iterations to reach the acceptable performance level [49].
- Corporate Compliance: In a real-world ML application, the efforts to comply with the privacy policy of the organization and the legal framework may seriously influence the development, deployment, and operation. These cause challenges from both technical and management points of view [1].

2.2.8 Software engineering process

"*In this knowledge area (KA), software engineering processes are concerned with work activities accomplished by software engineers to develop, maintain, and operate software, such as requirements, design, construction, testing, configuration management, and other software engineering processes*" [30].

The general software development lifecycle can't satisfy ML applications because of additional data-oriented and model-oriented works for ML developments. Khomh et al [3] raised two critical questions about "*how to integrate the AI model lifecycle (training, testing, deploying, evolving, and so on) into the software process*" and "*what new roles, artifacts, and activities emerge and how these new cases accord existing agile or DevOps methodologies*".

Some studies propose various software processes for ML applications [1, 36, 50]. Amershi et al [1] argued the process maturity model to develop ML applications. Tool support for the development process can be further challenging. As Patel et al [51] argued, "non-expert tools need to support the entire exploratory and iterative process of applying statistical machine learning algorithms." Ishikawa et al [52] described the difficulties in making customers better understand ML applications' properties, such as imperfections. Trial-based processes can be a solution for these difficulties, but building a solid foundation for the engineering disciplines requires more researches.

2.2.9 Software engineering models and methods

"*Software engineering models and methods impose structure on software engineering with the goal of making that activity systematic, repeatable, and ultimately more success-oriented. Using models provides an approach to problem-solving, a notation, and procedures for model construction and analysis. Methods provide an approach to the systematic specification, design, construction, test, and verification of the end item software and associated work products*" [30].

This KA concentrates more on official domain-specific languages (DSL) and methods. Portugal et al [53] didn't find any DSL for the expression of systems requirements after succinctly surveying DSLs for machine learning in Big Data. Hains et al [35] presented DSLs and tools for a formal specification, model-based testing tools, UML class diagrams to illustrate datasets, and theorem-proving techniques as research orientations. Some other works related to this KA also investigated the challenges of ML techniques' formal approach [36].

2.2.10 Software quality

Software quality is an umbrella term for multiple facets. It refers to whether the software products possess the desired characteristics, the extent to which a software product possesses those characteristics, and the processes, tools, and techniques by which the developer achieves those characteristics. Throughout its history, the term software quality has been differently defined by researchers and organizations. Kumeno [34] The Software Quality KA provides definitions and "*the practices, tools, and techniques for defining software quality and for appraising the state of software quality during development, maintenance, and deployment*" [30].

This KA widely covers software quality topics. One of the representative challenges of software quality in ML applications is software testing. But as we explained it in the "Software Testing" section, we consider other challenge topics in software quality as Kumeno [34] listed them:

- Software Quality Assurance: A series of activities that specify and evaluate the adequacy of software processes. It approves the software processes which complete the objective task and the software products which satisfy their purposes [30]. Some works discussed what adequate quality assurance is for ML applications and how to accomplish it [3, 32, 36, 50].

- Validation & Verification: Several studies propose solutions for technical challenges of

V&V for ML applications [32, 36].

- Fault Analysis: It involves fault detection, fault characterization, and fault fixing issues of ML applications (see also 2.5).
- Component Quality: Training data, ML models, and ML platforms (e.g., scikit-learn, Tensorflow, and Weka) are the critical constituents of ML applications; each has its particular quality challenges. Data quality challenges for ML applications involve data anomaly, data evaluation/cleaning, novelty detection, imbalanced or biased data, and encrypted data [54]. We discussed the quality of the ML model and ML platform in testing (see Section 2.5).
- Quality Measurement: Breck et al [55] suggested a test scoring technique for ML application development to assess the readiness of a given ML production. Quality measurement and system safety are conceptually close for ML applications. Varshney et al [56] considered safety definition from a risk and uncertainty reduction perspective. These risk and epistemic uncertainty are affiliated with undesirable consequences which are not negligible. The risk and uncertainty measures for ML applications can be considered as safety measures as well.
- Safety and Security: Safety and security are the critical requirements for Mission-critical systems in some domains. Industry standards such as DO-178 [57] address safety or security. Complying ML applications with these standards is demanding but important to perceive mission-critical ML applications in the industry [32, 37].
- Ethics and Regulations: ML applications' ethics is associated with issues on safety, privacy, and discrimination [31, 37, 41, 56]. Since the imposed regulations may influence ML applications' quality, ethical topics are a subset of quality evaluations of ML applications. Some studies investigated the effect of regulations on the development/deployment of ML applications [37, 41, 56].

2.2.11 Software engineering professional practice

"*The Software Engineering Professional Practice knowledge area (KA) is concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering in a professional, responsible, and ethical manner*" [30].

ML applications' lifecycle process comprises a wide range of practices like data analysis, data processing, data cleaning, ML model design and building, model deployment, and maintenance (e.g., monitoring, debugging, and retraining). But for coping with these practices,

practitioners need skills more than the ones required for traditional software engineering. The skillset supplement for ML applications is the main challenge of this KA [1, 31, 40, 48]. Kumeno [34] identified the other topics related to this KA as follows:

- Group Dynamics and Psychology: To expertly construct and deploy a well-functioning ML application, several collaborators with disparate knowledge, skills and cultures should engage in the project. This category's main challenges are cooperation and sufficient communication with customers to certify a successful project [3,4,31,48,49,52].
- Economic Impacts: As the business effect of real-world ML application is a critical factor, ML developers should utilize the methods and skills to analyze this factor (see also Section 2.12).

Ethics and Regulations: This topic is also related to ML applications engineers' professionalism (see Section 2.10).

2.2.12 Software engineering economics

"*This knowledge area (KA) provides an overview on software engineering economics. Economics is the study of value, costs, resources, and their relationship in a given context or situation. In the discipline of software engineering, activities have costs, but the resulting software itself has economic attributes as well. Software engineering economics provides a way to study the attributes of software and software processes in a systematic way that relates them to economic measures*" [30].

This KA involves risk and uncertainty management as well as economic topics. Although this KA is influential for read-world ML applications in industry, very few studies discussed the challenges associated with it. Kumeno [34] divided these challenges into two categories:

- Risk and Uncertainty management: Technical doubts may cause maintenance cost increase [4]. Project risk estimation [41] and challenges in estimating the attempts come about uncertainties in ML applications are costly [49, 52].
- Economic Impact: Lucas et al [54] identified the challenges of translating ML outcomes into real economic impacts. Unfamiliar customers with ML techniques can't understand most of the performance measures on ML techniques. Thus, such customers are unable to translate ML outcomes like accuracy into corresponding performance metrics like revenue. Dahlmeier [58] focused on the challenges of translating the ML outcomes into practical innovation in the field of natural language processing (NLP). They identified

the concept of "lack of value focus" in the NLP studies, but this concept may also occur in the other ML branches. The other priority is the optimization of AI's economic impact, as Russell et al [59] shed light on it. This topic involves market disruptions using AI methods, labor market prediction, and policy for handling detrimental consequences.

CHAPTER 3 ARTICLE 1 – TOWARDS UNDERSTANDING DEVELOPERS' MACHINE-LEARNING CHALLENGES: A MULTI-LANGUAGE STUDY ON STACK OVERFLOW

Alaleh Hamidi, Giuliano Antoniol, Foutse Khomh, Massimiliano Di Penta, and Mohammad Hamidi

"International Conference on Software Maintenance and Evolution (ICSME) 2021 2021-04-27"

Abstract Machine Learning (ML) is increasingly being used as an essential component of modern software systems, and it has rapidly attracted the attention of many software developers. Also, the maturity of the adopted techniques and the availability of frameworks has changed the way developers approach ML-related development problems. This paper aims at investigating, by analyzing Stack Overflow (SO) posts related to ML, how the questions about ML have been changed over the years, and across six different programming languages. Specifically, we analyzed 43,950 SO posts in the period 2008-2020, studying (i) how the number of ML-related posts changes over time for each programming language, (ii) how the posts are distributed across different phases of a ML pipeline, and (iii) whether posts belonging to different languages or phases are more or less challenging to address. We found that some programming languages are fading while the others are becoming more popular in ML development. Also, it is common for all selected programming languages that Model-Building questions are mostly discussed. However, the extent to which these questions are challenging to answer varied in different programming languages.

3.1 Introduction

Machine learning (ML) is increasingly becoming an essential component of modern software systems. The demand for industry adoption of ML has increased. The most recent NewVantage Partners Big Data and Artificial Intelligence (AI) Executive Summary shows that 92% of organizations are increasing their pace of investment and 62% of them have already seen measurable results from their investments in big data and ML [60].

Such growth in ML adoption produces new challenges for software developers. Some studies have suggested that ML can introduce special software development problems [55, 61, 62]. One consolidated way to study challenges developers face in a given field is to analyze posts on question and answer (Q&A) forums such as Stack Overflow (SO) [5, 6].

Some studies report an increasing number of ML-related posts on SO [6,8,9]. This indicates that, in recent years, ML has attracted growing interest among software developers, especially after 2015 [7].

However, developers may face different problems when coping with ML in different programming languages. That is, they may have to use a language given the application under development, but such a language may or may not be suitable for ML-intensive development, given its intrinsic characteristics and given the available libraries. Moreover, as Amershi et al [1] noted in their study of ML development practices at Microsoft, ML-based development requires coping with various phases of a workflow, such as model requirements, data collection, cleaning, and labeling, feature engineering, model training, and evaluation, until deployment and monitoring.

While previous work [8] has studied the increasing attention received by ML from developers, there is little knowledge about its evolution and variation across programming languages, and whether different ML phases receive more attention or create more challenges.

In this work, we systematically study what kind of problems developers encounter in each stage of the ML development life cycle when using a specific programming language. We analyze 43,950 posts of SO dated from 2008 to 2020, and related to ML, to answer the following research questions:

- **RQ1**: How does the number of ML-related posts related to different programming language change over the years?
- **RQ2**: How are posts distributed across different phases of a ML pipeline, and how does this change over time?
- **RQ3**: To what extent are posts belonging to different languages and different phases answered and after how long?

We first analyze how posts are distributed across the years, and how this varies among programming languages. We consider a total of six programming languages, including popular programming languages (eg C/C++, Java, and Python), as well as languages used in scientific/statistical computation (eg R). Then, two of the authors manually classified the high rated posts along the ML phases described by Amershi et al. [1], to understand the distribution of posts over phases, and how this varied over the years. Finally, to measure the level of difficulty of a question, we followed an approach similar to previous work [8], [9], [10], [11], and [12], ie we computed the percentage of questions with no accepted answers or with no answer at all, as well as the median time needed by a question to receive an accepted answer.

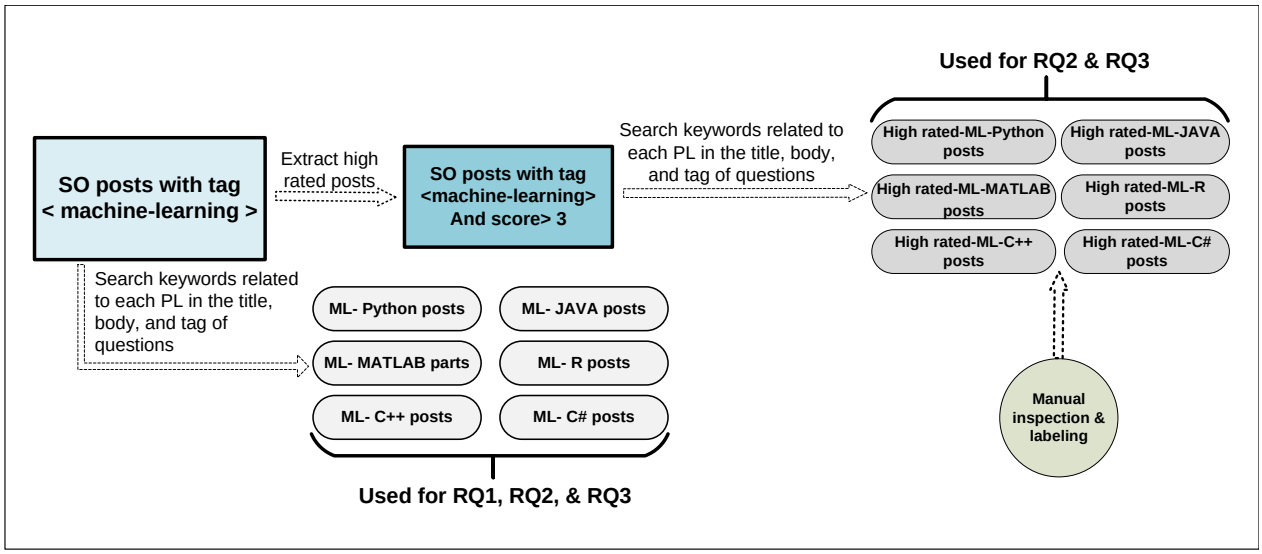

Figure Figure 3.1 Data analysis methodology

Results indicate that, as one may expect, Python has the lion-share as a programming language, while languages such as R are still very popular, while ML-related questions for other languages such as C, C++, or Matlab tend to increase less, or not increase at all. Concerning the ML phases to which questions refer, building, evaluation and requirement are the most frequently encountered concerns, when we consider posts with three or more stars. Model building alone accounts for about one third of posts. The building and evaluation phases are the source of most concerns, they account for more than 50% of concerns, but it is decreasing in recent years likely because of the availability of high-level framework (in Python) such as PyTorch or Keras.

3.2 Methodology

The *goal* of our study is to analyze SO posts related to ML, characterizing them from different perspectives. Specifically, the aim is to understand how the number of such posts evolves over the years, how this varies among different programming languages, how posts are distributed across the different ML phases, the extent to which posts receive a response, and the time taken to provide these responses. The *perspective* is that of educators interested in understanding ML topics that are challenging to ML developers. They can leverage this information to improve training materials introducing these topics. Researchers could also leverage the information, to develop better tools and analysis techniques to supports ML developers during challenging phases of the development life cycle of ML systems. The *context* consists of SO posts related to ML, dated between 2008 and 2020, and belonging to different

programming languages.

Fig. Figure 3.1 provides an overview of our methodology, described in the following.

3.2.1 Context and Data Collection

The popularity of SO and its large amount of ML-related questions, makes it a representative data source for our study. We consider 43,950 SO questions posted between July 2008 and December 2020. The first step in our study is to identify the subset of SO questions that represent the ML-related challenges. We followed a snowball sampling approach similar to Alshangiti et al [8]. We started with the "machine-learning" tag and extended the list of ML related tags based on relevance and co-occurrence. More specifically, for each tag expanding, we inspected the list of top 25 tags that occurred with that tag and chose the ones related to ML. We repeated this process until we obtained the top 50 most-used ML-related tags (reported in Table Table 3.1).

As Table Table 3.1 shows, the "machine-learning" tag includes the greatest number of ML-related questions. Note that this tag can cover the topic of other ML-related tags as well. We collected the questions with "machine-learning" tag, using Stack Exchange Data Explorer, which is an open-source tool to run queries against public data from the Stack Exchange network [63].

3.2.2 RQ1: Post classification across programming languages

Before automatically classifying posts across programming languages, we performed some textual preprocessing. Since the body of each question may contain source code, before starting to search for keywords, we preprocessed the content of the post using regular expressions matching and removed the content between the `<code>` and `</code>` tags. This preprocessing step is necessary because the code in the post can contain the searched keywords as the name of some variables like C and R, therefore introduce false positives.

To determine the ML posts related to each programming language, we used the language name as a keyword to search in the title, question body, and tag of each post from our initial dataset of ML-related questions. Since some posts could contain questions related to more than one programming language, we count such posts in the datasets of each relevant programming language separately. After removing source code, we pruned out HTML tags.

We validated the reliability of such a categorisation manually inspecting a subset of them, and, in particular, all posts having more than three stars, because these posts will be used for the analyses of RQ2 and RQ3. Results of this manual analysis are reported in Table Table

Table Table 3.1 Top 50 most commonly used Machine Learning Tags

Tag	Occur.	Tag	Occur.
machine-learning	43,953	neural-network	17,391
supervised-learning	460	conv-neural-network	6709
unsupervised-learning	541	recurrent-neural-network	2300
reinforcement-learning	1805	rnn	668
prediction	2304	deep-learning	19,804
regression	7522	image-recognition	1287
linear-regression	4966	object-detection	3230
non-linear-regression	570	sentiment-analysis	1629
nls	545	cluster-analysis	5391
classification	6850	hierarchical-clustering	1013
classifier	545	pca	2282
multilabel-classification	670	autoencoder	1155
multiclass-classification	560	word2vec	1865
document-classification	217	word-embedding	811
text-classification	1405	tf-idf	1157
logistic-regression	2995	rfe	113
svm	4270	feature-engineering	296
svmlight	98	feature-selection	1213
decision-tree	2139	feature-extraction	1458
random-forest	2902	cross-validation	2144
naivebayes	950	confusion-matrix	791
perceptron	436	precision-recall	352
dbscan	496	scikit-learn	22015
knn	1409	tensorflow	65,213
k-means	2991	r-caret	1864

3.2, showing how many posts were misclassified and how many reassigned. When a post was reassigned, it was reassigned to a different language, including the *Other* language category (ie not one of the analyzed languages), thus the column *Total* is the results of manual reassignment and cannot be obtained by adding the number on the same row.

In our sample of 4186 posts, we found a total of 107 miss-classifications (about 3%), out of which 62 were manually re-assigned. The remaining 47 posts were discussing other programming languages and though not strictly relevant we kept and are reported as *Other*.

Table Table 3.2 2008 - 2020 SO ML posts with more than three stars per language manually verified

	Manually-Verified Posts					Posts Per
Lang.	Nbr.	Correct	Misclassified	Reassigned	Lost	Language
C#	47	42	5	4	1	42
C/C++	128	111	17	14	3	114
Matlab	169	153	16	14	2	157
Java	133	119	14	4	10	126
R	320	285	35	26	9	289
Python	3389	3367	22	0	22	3411
Other						47
Overall	4186	4077	109	62	47	4186

We then computed the proportion of questions asked for each programming language, from 2008 to 2020. We also computed the proportion of accepted questions for each programming language during the same period. Moreover, to determine whether there is a trend in the questions for each programming language, we apply the non-parametric Mann-Kendall test for trend detection [64, 65] and the Sen's slope statistic [66]. Specifically, we first apply the Mann-Kendall test to reject the null hypothesis H_0 : *There is no significant trend in the time series.* The Mann-Kendall test is a measure of association between an increasing time series, automatically generated, and the observed time series. It computes the S statistic, which is closely related to the τ correlation, ie it is based on the ranks of the samples and indicates the trend direction ie if positive, the trend increases (conversely, it decreases). Then, we compute the Sen's statistic [66] to gain an insight of the slope size [66]: the higher the value, the greater the slope. In a nutshell, the slope is the median of the deltas between consecutive points. Being based on the median, Sen's slope (also known as Theil-Sen estimator [66]) is robust to outliers.

Finally, we use the Lanzante statistic [67] which, given a time series, determines whether it has points in which the slope significantly changes, and if yes indicates where.

To address RQ1, we report and discuss the post trends over the years among the different programming languages, using the aforementioned statistical procedures to identify the presence of trends.

3.2.3 RQ2: Classification of questions into phases

In this research question, we classify posts according to the ML phases defined by Amershi et al [1]. The analysis has been conducted manually (see Table Table 3.2) and, to consider more representative and high-quality posts, we consider those having a score greater than three stars (ie four or five stars). This selection also makes the manual inspection more feasible. While we are aware that excluding some (low-rated) posts might somewhat bias our dataset, our intent is to look at very meaningful (highly-rated) posts that can be considered as good representatives of the various phases. Low-rated posts may be useless, trivial, or even wrong, and may produce a misleading representation of the post distribution along phases.

Note that such a filtering was not performed to address RQ1, because, in that case, we were simply interested to gather an overview of the posts across programming languages and over the year. We manually inspected 4186 high rated ML posts to categorize them and consider the accuracy of the programming language classification.

Two ML researchers independently classified and labeled 100 randomly-selected questions. They were asked to assign only one label to each question. We computed the Cohen's k [68] inter-rater agreement, which resulted equal to 0.88, which is an almost perfect agreement.

This substantial level of agreement means that the two researchers shared a common understanding of the posts, therefore, for the remaining dataset, we divided it into two groups and tasked each of the researchers to manually classify one group.

During the manual inspection, each rater performed two tasks (1) determined the correctness of programming language assignment to post (which for RQ1 was only done on a sample, and (2) by reading the post content, assigned the post to one or more ML phases as they were defined by Amershi et al [1].

Amershi et al [1] suggest a nine-phase ML workflow informed by prior experiences developing AI applications at Microsoft. Some phases are data-oriented like collection, cleaning, and labeling. But other stages are model-oriented (eg model requirements, feature engineering, training, evaluation, deployment, and monitoring). We determined our ML categories based on this workflow to classify and label the questions. Next, we describe these categories as they were interpreted by the raters when manually classifying SO posts:

Model Requirement (MR): The questions under this category are related to considering the implementation feasibility of features with ML, looking for appropriate libraries, choosing the most appropriate models for a given problem, identifying relevant and representative data, and installing required infrastructure, eg packages.

Data Collection & Preprocessing (DCP): All the questions related to data collection

and preprocessing are placed in this category; integrating datasets, removing inaccurate, noisy or duplicate records and outliers from the dataset, data adoption into suitable data format required, data transformations (like normalization, min-max scaling, and data format conversion), and balancing the classes.

Feature Processing (FP): This category includes questions related to data labeling and feature engineering; assigning ground truth labels to each record, encoding data, extracting and selecting informative features to reduce data dimensionality.

Model Building (MB): The questions under this category are related to identifying the training and test dataset, creating the ML model using the APIs, training the model, tuning the model parameters, storing models to disk, and loading them to use later.

Model Evaluation (ME): These questions are about evaluation method selection, evaluating the performance of a model, visualizing the behavior of the model, and interpreting the output of a model.

Model Deployment (MD): The questions related to deploying the application on suitable devices or platforms, model reuse (like conversion of a model trained using one library and then using the trained model for prediction in an environment using another library), getting different results from one ML model implemented by different platforms (when using similar setting), and robustness are placed in this category.

Model Monitoring (MM): This category includes the questions related to monitoring the functionality of ML applications for performance and potential errors, during real-world executions.

Not Related to ML Challenges (NO): This category includes questions that cannot be placed under any of the above categories.

All Phases (all): Includes posts related to all ML development phases: model requirement, data collection, data preprocessing, feature processing, model building, model evaluation, model deployment, and monitoring.

To address RQ2, we report the distribution of the manually-classified posts along phases, and how this varies between programming languages.

3.2.4 RQ3: Analysis of how posts were answered

To address RQ3 we analyzed, for different programming languages and for different ML phases, the percentage of posts that received an accepted answer, and, if this happens, the distribution of response time (in hours). That is, we replicate the analysis conducted

by Alshangiti et al [8], but (i) considering the impact on different programming languages separately, and (ii) by using the phase classification taxonomy of Amershi et al [1]. As also done for RQ2, and since posts could not receive an accepted answer because they are trivial/uninteresting/misleading, we only consider posts having a score greater than three.

In summary, we report (i) the number and percentage of posts without confirmed answers (ii) for posts that have been answered, boxplots depicting their distribution along programming languages and along ML phases. Also, to statistically compare different programming languages and different phases we use the (non-parametric) Dunn's test [69] using the Benjamini-Hochberg correction [70] because of multiple comparisons being performed.

3.3 Study Results

This section reports the study results, aimed at addressing the research questions formulated in Section 3.2.

3.3.1 RQ1: How does the number of ML-related posts related to different programming language change over the years?

Table Table 3.3 reports the overall number of posts, along with the number and ratio of accepted ones. As the table shows, languages such as Python and R play the lion-share role while others such as C# and C/C++ have a lower number of posts.

It is not surprising at all, to observe the role played by Python, not only for the availability of many data-science related libraries (eg pandas, numpy, PyTorch, or Scikit-learn) but, more importantly, the hype of deep learning [71] techniques, and the adoption of related libraries. In that case, Python represents an almost compulsory choice as the support in other programming languages is relatively limited.

Table Table 3.3 Overall number of posts per languages

	With accepted answers	Without accepted answers	Total	Without accepted answers %
C#	150	215	365	0.59
C/C++	272	294	566	0.52
Matlab	589	636	1225	0.52
Java	622	823	1445	0.57
R	1074	1616	2690	0.60
Python	9591	12,663	22,254	0.57

Table Table 3.4 Overall number of posts over the years per language with and without accepted answers

	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	Total
C#	0	5	9	9	22	11	14	22	33	42	65	70	63	365
C/C++	0	3	14	33	41	48	49	85	88	70	48	40	47	566
Matlab	2	11	24	39	108	157	161	168	183	149	120	54	49	1225
Java	0	6	28	60	112	115	123	171	177	191	136	135	191	1445
R	0	2	12	28	61	119	153	225	438	404	374	360	514	2690
Python	0	4	46	83	170	314	441	819	1855	3330	4366	4423	6403	22,254

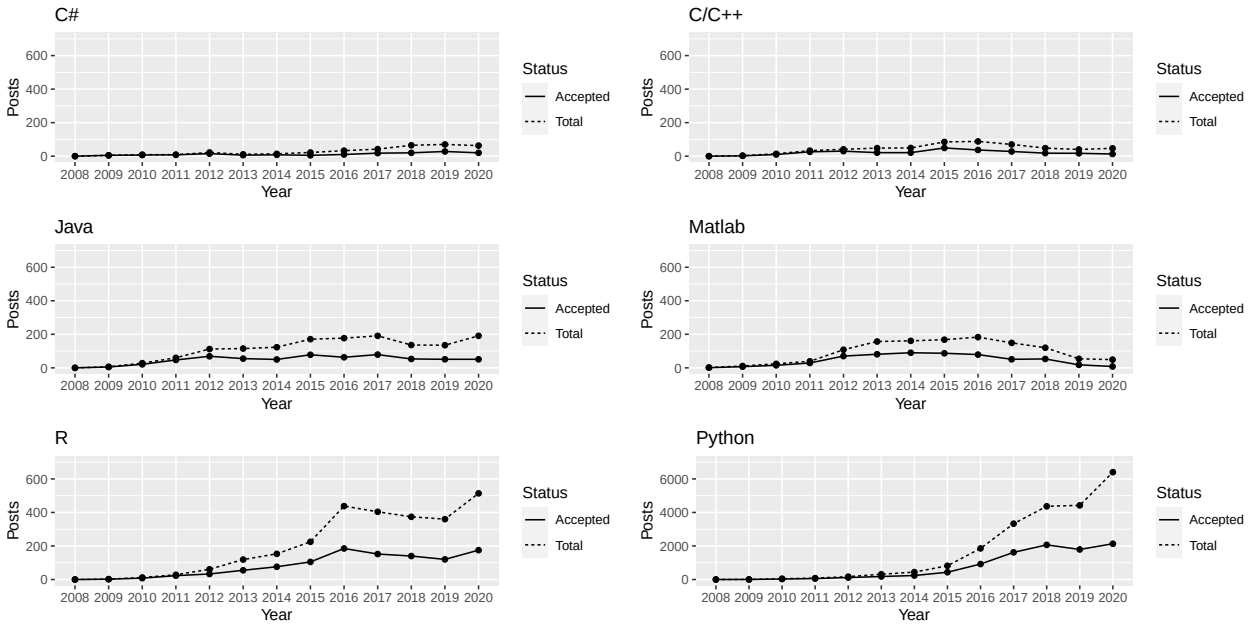

Figure Figure 3.2 Evolution of total and accepted posts over the years. Note: Python has a different scale.

The case of R also interesting to discuss. R has a long-term tradition in terms of usage by the scientific community coupled with a large community of developers and a consolidated infrastructure (ie the CRAN archive). While nowadays Python libraries such as the ones mentioned above make available almost any statistical procedure, R remains popular for researchers who got use to it (especially working in areas beyond computer science), and is a valid choice for those having to use very specific statistical procedures that may not be available in Python yet.

Table Table 3.4 and Fig. Figure 3.2 show the overall evolution of posts over the years, and its trend per programming language, respectively. As it is possible to notice from the plots in Fig. Figure 3.2, there is a consistent trend within two groups of programming languages. On the one hand, we have Java, R, and Python where we can clearly notice an increasing trend. On the other hand C#, C/C++ and Matlab have either a lower increasing trend (if any) or

even a decrease (Matlab). It is unclear why this happens, although the only clear difference between Matlab and the other languages is that the former is part of a commercial product (by MathWorks), although Matlab-like (eg GNU Octave) open source solutions exist. However, it may also be the case that Matlab targets a different category of developers, eg including industrial developers working in automation control and especially in the automotive domain, where the adoption of ML solutions is rapidly increasing especially to support self-driving or driving assistance features. Such developers, especially when working on industrial projects, may target internal sources of knowledge rather than SO (however, this conjecture needs to be further investigated).

Consistently to what we observed in terms of the number of posts, Python and R have a consistent and steeper increase, plus they do not exhibit a plateau (nor a decrease) in the last two/three years.

We confirmed this observation by applying the Mann-Kendall Test to determine whether our time series dataset have monotonic upward or downward trends. We also applied the Sen's slope statistic [64–66] to estimate the slope of the trends. The results of these analysis are reported in Table Table 3.5 and Table Table 3.6, respectively. As for the Mann-Kendall test, Table Table 3.5 reports the test p-value, the S statistics (the higher S, the steeper the trend is expected), the S variance (VarS), and the τ coefficient. Table Table 3.6 reports the Sen's trend analysis p-value and slope.

Table Table 3.5 Overall posts Mann-Kendall Trend Test - H_0 : slope is equal to zero

Language	p-value	S	VarS	τ
C#	<e-04	68	266	0.88
C/C++	0.02	39	267	0.50
Matlab	0.07	30	268	0.38
Java	0.0001	63	267	0.81
R	<e-4	66	268	0.85
Python	<e-05	78	268	1

As one could expect by observing Fig. Figure 3.2, the Mann-Kendall test confirms the presence of trends for all languages but Matlab. Sen's slopes vary evenly between the minimum of about five (for C# and C/C++) and the maximum of about 450 for Python.

Finally, we applied the Lanzante statistic [67] to further confirm the presence of a trend and divide the likely point of change. Lanzante statistic identified two possible change points in 2012 and 2015. Also in this case, languages can be broadly divided into two groups. On the one hand, C/C++ and Matlab have a change point in 2012, while the other languages

Table Table 3.6 Overall posts Trend analysis: Sen's slope

Language	p-value	Slope
C#	< e-04	5.50
C/C++	0.02	5.41
Matlab	0.07	10.0
Java	0.0001	16.28
R	< e-04	42.77
Python	< e-05	455.52

exhibit a change point in 2015. The latter has a very clear explanation, ie the introduction of very popular framework for deep learning, such as Tensorflow and Keras (both released in 2015) or PyTorch, released in 2016. Indeed, if we consider two sub-series up to 2014 and from 2014 on, we obtain the trend analysis results shown in Table Table 3.7, where only Python is increasing.

Table Table 3.7 2014 on: trend analysis: Sen's slope

Language	p-value	Slope
C#	0.02	9.5
C/C++	0.13	-7.6
Matlab	0.03	-23.8
Java	0.44	3.5
R	0.2	49.66
Python	0.002	1018.5

Overall, results show an increasing trend for Java, R, and Python, with Python having the steepest increase. The number of posts related to Python exploded between 2015–2016, and its largest delta was observed in 2019–2020. We link this phenomenon to the surge of Python frameworks that occurred during 2015 - 2016, and the increasing adoption of ML in research and practice.

3.3.2 RQ2: How are posts distributed across different phases of a ML pipeline, and how does this change over time?

Table Table 3.9 reports the distribution of posts (among the manually analyzed ones) for the ML various phases defined by Amershi et al [1], and Table Table 3.8 the percentage of SO

posts per phases and language. Note that since a post may belong to multiple categories, the sum of percentages does not add up to 100.

Looking at Table Table 3.9, the difference in terms of relative importance of phases is evident. The majority of questions in our dataset are related to model building (MB), model evaluation (ME), and model deployment (MD) phases. Some languages exhibit a specificity. For example, model requirement (MR) seems to be a concern for C# and C/C++ (because many C# and C/C++ developers are looking for the libraries correspond to the ML python packages), while data collection (DC) and feature processing (FP) posts are frequent in C#. We can conclude that MB, ME, and MD are the main sources of concerns.

Table Table 3.8 Percentage of ML posts (with more than three stars) per phase and languages

	MR	DCP	FP	MB	ME	MD	MM	NO	ALL	Unclear
C#	26	29	3	43	-	3	-	3	15	-
C/C++	33	10	3	38	7	24	-	8	-	-
Matlab	13	11	6	66	10	3	-	6	2	-
Java	1	12	7	48	3	27	2	4	6	2
R	10	15	8	46	22	13	2	4	1	-
Python	19	13	18	47	33	12	2	5	1	5
Other	5	7	3	19	-	12	-	10	-	55

Table Table 3.9 Distribution of phases of ML posts with more than three stars for different languages

	MR	DCP	FP	MB	ME	MD	MM	No	all	Unclear	Total
C#	9	10	1	15	0	1	0	1	5	0	42
C/C++	31	9	2	36	6	23	0	7	0	0	114
Matlab	17	15	8	90	13	4	0	8	2	0	157
Java	1	13	8	56	3	31	2	4	6	2	126
R	23	35	19	114	54	30	4	8	2	0	289
Python	429	291	384	1046	734	270	39	108	1	109	3411
Other	2	3	1	8	0	5	0	4	0	24	47
Overall	512	376	423	1365	810	364	45	140	16	135	4186

Table Table 3.10 reports the Python distribution of posts per phase and years. Similar tables for other languages are omitted due to space reasons. Given the conclusion of RQ1, Python is indeed the most interesting language to consider for an in-depth discussion. Table Table 3.10 shows an almost steady increase for MB, MD, and ME posts. Moreover, MB, ME, and MR alone account for about 50% and more of concerns, and this throughout the years. For the

other programming languages, overall, we observed similar trends. Table Table 3.11 reports links to examples belonging to different phases.

Table Table 3.10 Python number of posts (with more than three stars) per phase and over the years

	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	Total
MR	0	3	17	30	26	40	28	44	74	65	55	28	19	429
DCP	0	0	7	4	6	15	18	32	44	65	60	26	14	291
FP	0	0	3	4	8	26	28	40	67	88	72	33	15	384
MB	0	0	12	15	34	50	42	86	186	242	231	105	43	1046
ME	0	0	6	9	27	36	37	56	156	171	144	72	20	734
MD	0	0	1	3	9	13	11	25	46	52	65	36	9	270
MM	0	1	1	1	1	4	3	1	5	8	4	7	3	39
NO	0	0	2	1	6	4	2	6	24	28	18	14	3	108
ALL	0	0	0	0	0	1	0	0	0	0	0	0	0	1
Unclear	0	0	1	5	4	3	8	8	20	30	13	14	3	109
Overall	0	4	50	72	121	192	177	298	622	749	662	335	129	3411

Table Table 3.11 Examples of SO posts for different phases

Phase	Language	Link
MR	Python	P1: https://stackoverflow.com/questions/2524608/machine-learning-algorithm-for-predicting-order-of-events
		P2: https://stackoverflow.com/questions/40158232/updating-an-old-system-to-q-learning-with-neural-networks
	C#	P3: https://stackoverflow.com/questions/56365587/backpropagation-algorithm-giving-bad-results
DCP	Python	P5: https://stackoverflow.com/questions/52184142/keras-sequential-dense-input-layer-and-mnist-why-do-images-need-to-be-reshape
	R	P6: https://stackoverflow.com/questions/9316794/am-i-using-the-wrong-data-type-with-predict-nnet-in-r
FP	Python	P7: https://stackoverflow.com/questions/28254824/feature-space-reduction-for-tag-prediction
	Matlab	P8: https://stackoverflow.com/questions/40560139/matlab-feature-selection
MB	Python	P9: https://stackoverflow.com/questions/36986815/what-is-the-parameter-max-q-size-used-for-in-model-fit-generator
		P10: https://stackoverflow.com/questions/50744565/how-to-handle-non-determinism-when-training-on-a-gpu
	Matlab	P11: https://stackoverflow.com/questions/38853370/matlab-regularized-logistic-regression-how-to-compute-gradient
	Java	P12:https://stackoverflow.com/questions/38910471/tensorflow-retrained-inception-v3-model-crashes-on-android
ME	Python	P13: https://stackoverflow.com/questions/37796595/tensorflow-lstm-rnn-output-activation-function
		P14: https://stackoverflow.com/questions/35956902/how-to-evaluate-cost-function-for-scikit-learn-logisticregression
	Java	P15: https://stackoverflow.com/questions/21674522/get-prediction-percentage-in-weka-using-own-java-code-and-a-model
MD	Python	P16: https://stackoverflow.com/questions/42945509/keras-input-shape-valueerror
		P17: https://stackoverflow.com/questions/34175174/extract-features-using-pre-trained-tensorflow-cnn

MR posts address a variety of specific topics related to model requirements. Post P1 asks: *There is an infinite stream of 4 possible events. What ML algorithm is the best to predict what the next event will be, based on the order that events have come in in the past? And how long of a history that the predictor should maintain? And if more than one event can happen simultaneously on each round, does that change the solution? Python or C libraries are nice, but anything will do.* The developer was not sure what ML tool fits such a prediction problem regardless of the language. Also, adapting to more complex learning and improving it performance could be an issue, eg the P2 issue *Recently I've been reading a lot about Q-learning with Neural Networks and thought about updating an existing old optimization system in a power plant boiler composed of a simple feed-forward neural network approximating an output from many sensory inputs* . As another example for C#, the post P3 is asking *I'm*

trying to tackle the classic handwritten digit recognition problem with a feed forward neural network and backpropagation, using the MNIST dataset...I finished writing it some time ago and been debugging since, because the results are quite bad. At its best the network can recognize 4000/10,000 samples after 1 epoch and that number only drops on the following epochs, which lead me to believe there's some issue with the backpropagation algorithm. Here, in the end,the problem was not of a better algorithm, but rather, an input rescaling problem, thus more a matter of pre-processing.

Data collection, pre-processing and feature processing are a concern too. Consider post P6, where the problem was a missing rescaling of the inputs, causing the Deep Neural Network to produce inconsistent results. Similarly, P5, P7 and P8 cope with re-shaping, rescaling, and feature space reduction.

In summary, developers asking questions are concerned about ML algorithm and their implementation, given the problem requirements at hand. Since languages such as C, C++, or C# where MR questions occur more (in percentage) are not languages specifically targeting ML or statistics (as instead R, Matlab, and also Python thanks to its frameworks), such questions likely originate from developers working on a domain-specific application, which need to use ML but do not have specific ML-related expertise. Indeed, languages like C, C++ are used, for example, in the development of embedded systems or Internet-of-Things (IoT) applications, where ML may find a meaningful usage. Other areas of application include video games, where C++ and C# are used as languages.

MB questions vary from very specific questions, for example, P9: *What is the parameter "max_q_size" used for in "model.fit_generator"*, related to the evaluation of deep learning models in Python, to more implementation-oriented issues, eg P10 *While tuning the hyper-parameters to get my model to perform better, I noticed that the score I get (and hence the model that is created) is different every time I run the code despite fixing all the seeds for random operations. This problem does not happen if I run on CPU* and eg P11 for Matlab: *I am using gradient descent to minimize the cost function to implement Logistic Regression. I wrote a function in Matlab that returns both the cost and the gradient of each parameter evaluated at the current set of parameters. My cost function works, but the gradient function does not. What's wrong?* Finally, P12 concerns with language-specific issues occurring when porting Deep Neural Networks on Android. We conjecture that MB questions like the ones discussed exhibit an increase also considering the growing adoption of deep learning algorithms.

ME is also an often encountered concern. Consider the Python posts P13 and P14: *... I have a working network that is training, but the minibatch loss is at about 425 right now*

and the accuracy at 0.0, and for the LSTM MNIST example code (linked) the minibatch loss was about 0.1 and the accuracy about 1.0. My hope is that if I can change the activation function to use the SoftMax function, I can improve results... and *After using "sklearn.linear_model.LogisticRegression" to fit a training data set, I would like to obtain the value of the cost function for the training data set and a cross validation data set." How can I evaluate cost function for scikit learn LogisticRegression?* Similar doubts about evaluation are raised with other languages, eg a Java developer asked *How can I get prediction percentage in Weka using my own Java code and a model?* (P15).

We observed that model deployment (MD) touches a large variety of issues from operating systems to the availability of pre-trained models. In P16 the concern is an error due to the use of 32 versus 64 bits software *I made an image classifier with Keras using Theano as a Backend and a Sequential model. When I run my script on Windows 7 32 Bit, I get an error but If I run it on Elementary OS 64 Bit, it runs without any problems. Is there such a big difference between different Operating Systems?* , while in P17 the concern is the availability of a model ready to be deployed *when one does not have enough data to train the CNN, I may expect this to outperform a pipeline where the CNN was trained on few samples. I couldn't find a pickle file (or similar) with a pre-configured CNN feature extractor. Do such pre-trained networks exist and where can I find them. Where could I find a CNN+weights.*

Different post distributions and trends are visible among phases. Python still plays the lion-share. We observe that model requirement, building, evaluation, requirement, and deployment are the most frequently encountered concerns. While general questions related to model requirement become less important over time, model building (eg for a specific deep learning network) and deployment on certain architectures are becoming increasing challenges for developers.

3.3.3 RQ3: To what extent are posts belonging to different languages and different phases answered and after how long?

As shown in the last column of Table Table 3.3, the ratio of posts without accepted answers for all languages is between 52% and 60%. If we consider the sample of posts with more than three stars, as shown in the last column of Table Table 3.12, such a percentage lowers between 21% and 41%, with a noticeable exception (Java) of only 4%. It must be noted that the number of such posts considered for Java is relatively small (126, as from Table Table 3.9), all highly-rated posts may get accepted. Confirming the findings of RQ1 (where we have observed an increase of Python and R posts), R and Python have higher percentages of not

accepted posts than other languages. This is also the case for C# which, however, similarly to Java, has a very small number of high-rated posts (42 as from Table Table 3.9). While there could be many reasons for such observation, it is possible that the greater adoption of Python and R for ML generates more questions, but also more controversial answers. Languages such as Matlab and C/C++, which are more specialized and tied to specific usages and needs in the context of ML (eg speed, see for instance the Darknet project[1]) or the need to integrate into simulation engines (eg Matlab).

Table Table 3.12 Percentage of ML posts (with more than three stars) with no accepted answer per phase and languages

	MR	DCP	FP	MB	ME	MD	MM	NO	ALL	Unclear	Overall
C#	45	40	100	27	-	100	-	0	0	-	34
C/C++	42	34	0	42	17	44	-	43	-	-	40
Matlab	30	27	13	16	24	26	-	38	50	-	21
Java	0	16	0	2	0	4	0	0	0	0	4
R	40	46	37	44	47	30	0	13	50	-	41
Python	37	23	26	30	31	36	16	38	100	41	31
Other	100	34	100	50	-	40	-	26	-	34	41

When looking at the distribution across phases, RQ2 findings are confirmed also in terms of “controversial" posts that received no answers. First, we can see a relatively high percentage of posts with no accepted answers for MR, but also (confirming results of Alshangiti et al [8]) for DCP which is however lower for Python than for C#, C/C++, and R. This can be explained because Python has, with respect to those languages (including also R), much more data parsing/cleaning facilities thanks to the wide availability of libraries. Interestingly, this also happens for FP, MB, and ME, despite R has a quite variety of facilities for feature processing, and model building, and to some extent for model evaluation. However, the availability in Python of recent frameworks, including PyTorch, Scikit-learn, or Keras makes the life of ML developers easier.

MD seems to be less of a concern for Matlab and R. For the former, as previously discussed the language is quite mature in fields related to automation, and therefore deployment mechanisms may be clear. Also, often Matlab source code is translated into C before being deployed, therefore challenges arise (as it can be seen in the table) for that language instead. R is more used for experimentation than for production, and this could explain a relatively lower percentage of unaccepted questions there. Finally, MM does not seem to be an issue. There could be a variety of explanations, including the fact that ML is not applied yet in contexts where monitoring is a crucial concern.

[1] `https://pjreddie.com/darknet/`

Fig. Figure 3.3 depicts boxplots (positive outliers truncated to ease readability) showing the distribution of response times (in hours) for ML posts related to different programming languages. A Dunn's test with Benjamini-Hochberg correction indicates that Python has the shortest response time than other languages, with a statistically significant difference (p-value< 0.05) with all other languages but C# and Matlab. This, unsurprisingly, confirms the high popularity of Python for ML, and consequently the availability of developers able and willing to quickly provide answers on SO. C# and Matlab also have relatively short response times, however as explained in RQ2 C# has a high percentage of MR questions, most of which very basic ones. Also, it can be noticed a vary long interquartile difference, indicating, therefore, a high variability.

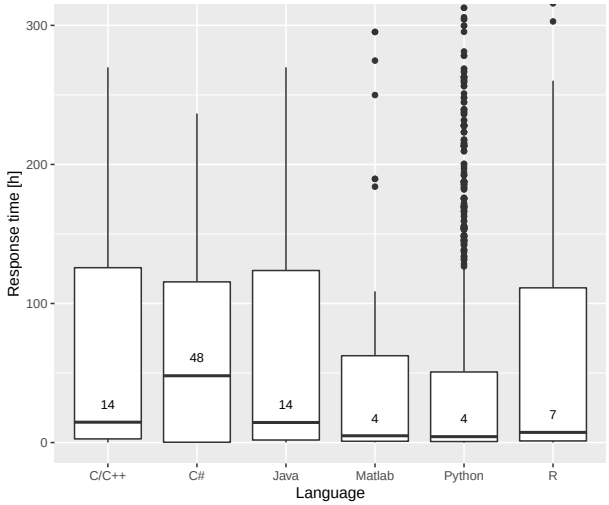

Figure Figure 3.3 Distribution of post response time by language.

Fig. Figure 3.4 shows, instead, boxplots related to response times related to different phases. Results indicate that posts related to data collection and processing (DCP), feature processing (FP), and those related to all phases (ALL) have significantly shorter response time (Dunn's test p-value< 0.05) than others. DCP and FP are less specific to the technicality of ML algorithms, therefore they are likely to receive attention also from non-experts, but just people knowledgeable about how to extract and clean data from given sources, as well as from domain experts. Model Monitoring (MM) is, for example, a very specific feature which not all ML developers may take care of, therefore the reaction time appears to be longer.

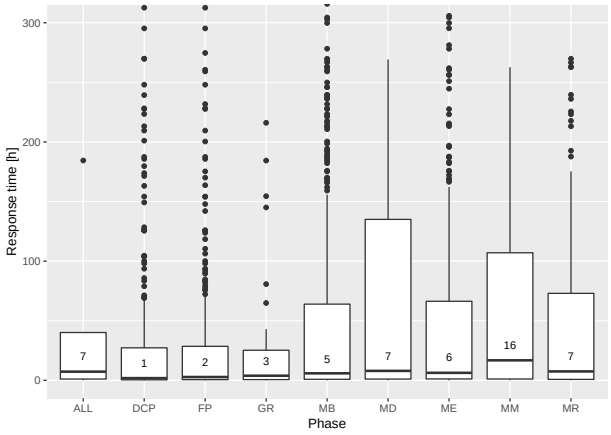

Figure Figure 3.4 Distribution of post response time by phase.

Model requirements, data cleaning/processing, and model building are the phases generating more questions without accepted answers than others. Python has, in general, fewer questions without accepted answers than R. Generic questions, and those related to data collection and processing or feature processing are answered faster than others. Questions related to ML development using Python are answered faster than those related to ML development using other programming languages.

3.4 Threats to Validity

Threats to *construct validity* concern the relationship between theory and observation. These are mainly related to imprecision in our measurements. First, querying SO for “machine learning" posts may miss relevant posts. Also, the programming language classification can be error-prone. This threat has been mitigated in RQ1 by manually analyzing a sample of posts having a high rate, as well as manually checking all data for RQ2. Also, the classification for RQ2 can be subjective and error-prone. We mitigated this threat by having two raters separately assessing a sample of 100 posts before completing the task independently, and computing their Cohen k inter-rater agreement. Finally, the number of SO posts is only a proxy of the developer base, but it may rather represent a technology’s popularity among developers.

Threats to *internal validity* concern factors internal to our study that can influence our results. While we discuss possible reasons for the trends of ML questions among programming languages (RQ1) and phases (RQ2), these are only possible interpretations, and there might be other reasons why such trends occur.

As for RQ3, while previous work has used the extent to which questions were answered and the time needed to answer a question as indicators of "challenging" questions [8], we cannot really claim a causal-effect relationship between them. This is because there might be many other reasons why a question receives (or does not get) answers, and why a question is answered immediately or after a long time. To partially (mitigate) this threat in RQ3, we consider posts having a high-score, which at least are less likely to receive low attention because they are not considered particularly interesting nor relevant by developers.

Threats to *external validity* concern the generalizability of our findings. Since our study focuses only on SO posts, results are limited only to the challenges that some developers state on SO. Nevertheless, developers may use other specific forums, as well as internal mailing lists of their projects/organizations. To generalize the results of our study, additional studies on other sources should be conducted. Also, while we analyzed popular programming languages, including the most widely used language for ML (Python) and languages used for scientific software (Matlab and R), we may have excluded interesting discussion related to other languages.

3.5 Related Work

In the following, we discuss studies aimed at understanding developers' challenges when dealing with ML.

Amershi et al. [1] studied ML development activities in Microsoft projects, classifying, as we have discussed in section 3.2, ML development into nine phases. Based on the analysis conducted, Amershi et al. also distilled some best practices for ML development. We take inspiration from Amershi et al. work, as we study how SO posts are distributed along ML phases, and how this varies over the years and across programming languages.

Islam et al. [5] conducted a manual inspection on 3,243 highly-rated SO posts related to ten ML libraries and classify these questions into seven typical categories of an ML pipeline to determine the correlation between the library and the phase. By considering the questions and performing statistical analysis, this work explores the answer to find the most difficult stages, understand the nature of problems, the used libraries, and study whether the difficulties stayed consistent over time. However, Islam et al. only study posts related to some ML libraries, therefore their results cannot be generalized to the whole body of ML challenges. In our study, we look at ML posts related to six different programming languages.

Bangash et al [72] studied SO posts about ML to understand which ML topics are significantly more discussed than others. To this aim, they applied Latent Dirichlet Allocation (LDA) [73]

on the textual content of SO posts to categorize them. Our work differs from Bangash et al, because we fully rely on a manual analysis to classify posts into ML phases. While in principle it could be possible to achieve this goal automatically in the future, for this first study we preferred to have a reliable, manually-curated dataset.

Alshangiti et al [8] studied the extent to which ML posts on SO are challenging to be answered, by considering the proportion of unanswered posts among six phases of a ML process. Complementary to Alshangiti et al, we look at the ML-related posts on SO also from the perspective of considering different programming languages, trends over the years, and using a phase classification proposed by Amershi et al [1].

Patel et al [51], [74] conducted an early study on the difficulties faced by software developers in the adoption of ML techniques. They interviewed ML researchers and users to identify three key difficulties: (1) following the iterative and exploratory process of ML (related to understanding different ML phases), (2) understanding data and output (understanding the relationship between data and model), and (3) evaluating the model's performance in the context of specific applications.

Yang et al [75] interviewed non-expert ML developers to understand the challenges they encounter when developing ML applications. This study shows that the most important challenges are related to modeling performance issues, unexpected errors, and understanding the ML algorithm's internal mechanism.

With respect to the aforementioned interview-based studies, our analysis focuses on a different data source, ie SO posts. Also, our analysis specifically deals with (i) the comparison of different programming languages, (ii) the classification of posts into phases, and (iii) the extent to which posts belonging to different phases and languages are answered or not, as well as the time required to receive an answer. That being said, the results of our study indirectly highlight challenges that developers are facing with ML for different languages and different phases. Also, the measurements of RQ3 are, similarly to Alshangiti et al [8], a proxy of the extent to which posts are challenging to be answered.

3.6 Conclusion

In this paper, we report the results of an empirical study that examined the characteristics of questions asked by developers about machine learning (ML) development. Through a quantitative and qualitative analysis of Stack Overflow posts from 2008 to 2020, and related to ML, we observe that the number of ML-related posts has changed over time for different programming languages. More specifically, there is an increasing number of posts related to machine

learning development, especially for Java, R, and Python. The majority of highly-rated ML-related questions concern issues related to model requirement, model building, and evaluation. Model building alone accounts for about one-third of all the studied posts. Questions about ML development in Python are usually answered much faster than questions related to ML development using other programming languages. This is not surprising given the current popularity of Python language in the ML community. However, given the important amount of questions asked by developers about ML development using other programming languages, educators and tool creators should ensure that enough training resources and tool support are available to support ML development using languages other than Python.

CHAPTER 4 GENERAL DISCUSSION

This section interprets and describes the findings of our two studies, the survey study and the Stackoverflow one.

4.1 Machine Learning Application Development:Practitioners' Insights

In this section, we deliver our insights into the survey responses from the expert practitioners concerning the challenges and best practices in ML application development. We describe the findings in the following subsections:

4.1.1 Trends in ML Application Development

Based on the practitioners' responses regarding the trends in ML application development:

- Business Intelligence (BI) systems are at the critical focus of ML application development. Other ML application types are healthcare, security, document processing, entertainment, broad areas for human life and business, etc.
- The agile software development methodology is the widely used method by practitioners to develop ML applications.
- Practitioners broadly use TensorFlow, followed by PyTorch and Keras. However, ML developers choose the development processes and frameworks according to the specific ML development context.

4.1.2 Data Precessing

As reported by practitioners, the main quality attributes of ML data are feature representation, adequacy, diversity, labeling accuracy, completeness, consistency, reliability, noise level, relevance, class balance, data distribution, performance impact, and bias. Noise removal, replacement of missing values, dimensionality reduction, class-balancing, and normalization are the data transformation operations that practitioners apply for quality assurance of ML data. They also utilize various tools and techniques to analyze and visualize ML data for better understanding. However, 76.6

The main challenges of ML data cleaning approaches are generalization, scalability, automation, data quality, lack of standard, required efforts and costs, lack of tools and tool fea-

tures, domain knowledge requirement. According to the questionnaire responses, we understand that ML developers are sometimes unaware of the importance of data cleaning and its domain-specific challenges.

The critical challenges of data labeling that practitioners faced are large data volume, cost, domain expertise requirements, automation, domain dependency, biases, data quality, and ensuring labeling reliability. Practitioners also feel a need for comprehensive feature labeling guidelines.

Although some practitioners apply tools and automated scripts, manual investigation is common to validate feature labeling. Domain knowledge is essential for both automatic and manual validation of feature labels.

4.1.3 Feature Processing

According to practitioners' responses, they test class balancing of labeled data using statistical analysis, data visualization, analyzing randomly sampled data, manual verification, and model performance. Performing data re-sampling (up or downsampling) and stratification of distribution for class balancing are the common approaches to ensure class balance.

For feature extraction, practitioners commonly apply manual analysis, customize solutions using existing libraries and frameworks, and utilize existing feature engineering tools. However, deep learning applications may don't need explicit feature extraction.

The typical limitations of the present feature engineering tools and techniques are generalization, scalability, automation, domain knowledge requirement, and adaptability. Besides, the quality of features and the performance of the feature processing tools and techniques require a challenging evaluation.

The common approaches practitioners utilize for assessing and validating features are statistical analysis, visualization, evaluating the resulting model performance, and incremental feature selection. Feature evaluation also depends on knowledge and expertise in the domain.

Three fourth of the practitioners need domain knowledge for feature selection. In addition, statistical analysis and visualization, automated tools, and incremental selection of features are the most used approaches for feature selection.

4.1.4 Model Building

ML developers in more than 93

To test the model implementation, practitioners observe performance, visualize model struc-

ture or the outcomes, test or debug code using existing tools. Domain knowledge is a critical factor for testing ML implementation.

Based on practitioners' perspective, the common signs of defects in ML code are extreme model performance, inconsistency, generalizability, and bias in model outcomes. In addition, poor model convergence, unusual training and inference time, and output values and their distribution can be the main symptoms of defects.

The key ML testing challenges include the black-box nature of the ML models, robustness to errors, and the characteristics of the data (adequacy, correctness, bias, labeling accuracy, distribution, and divergence of the data). The other ML testing challenges are diverse performance impacting factors and the absence of definite testing techniques. Besides, the shortage of explainability of ML models and some requirements such as domain expertise, time, and efforts hamper ML testing.

4.1.5 Model Management

The critical testing challenges in ML model development are related to the data quality (availability, format, and labeling accuracy) and the model performance evaluation (functional accuracy, generalizability, performance tracking, and metrics). Besides, the challenges in post-deployment assessment and testing of ML applications model are associated with complexity, resource requirements, target platform diversity, adaptability, and overall user acceptance.

As practitioners reported, they primarily monitor deployed models according to the performance (accuracy, latency, robustness, and resource consumption), business parameters (customer conversion or retention rate, click-through rates), and overall user acceptance of the ML applications.

4.2 Towards Understanding Developers' Machine-Learning Challenges: A Multi-Language Study on Stack Overflow

Investigating Stack overflow posts shows an increasing trend for Java, R, and Python, while Python has the steepest increase. The growing adoption of ML in research and practice and the outpouring of Python frameworks may be respectively the reasons for the rapid increase of Python posts in 2015–2016 and the delta in 2019-2020.

The most frequently encountered challenges are associated with the model requirement, building, evaluation, and deployment phases. While the importance of general questions related

to model requirements decreases over time, model building and deployment on specific architectures are becoming increasing challenges encountered by developers. Model building and evaluation concerns for Python started declining after 2018 that may be related to appearing new frameworks like Keras that facilitate ML development.

Model requirements, data collection/processing, and model building include more questions without accepted answers than others. Python generally has fewer questions without accepted answers than R. Generic questions, as well as data collection/processing or feature processing, get the answer faster than others. ML development questions related to Python obtain answers more quickly than those associated with ML development using other programming languages.

CHAPTER 5 CONCLUSION AND RECOMMENDATIONS

5.1 Summary of Works

The first part of our research investigates the challenges practitioners face and the practices they consequently perform during four main phases of the machine learning (ML) application development life cycle: (1) data collection and preprocessing, (2) feature engineering, (3) model building and testing, and (4) model deployment and maintenance. This work conducted a survey of 80 ML practitioners with different ML knowledge and skills and summarized that shared knowledge in 17 key findings. We believe that our findings can represent ML practitioners' practical experience, which helps understand the diverse challenges ML practitioners of all experience levels face in academia and industry during ML application development. Our findings can also provide valuable guidelines and examples of best practices to adopt in their ML workflow in a context-specific way.

As the second part of the research, we conducted an empirical study to analyze more specific challenges which are related to the PLs used for ML application development. We collected developers' SO questions about ML development posted during 2008-2020 to identify PL-related challenges' characteristics through a quantitative and qualitative analysis. We observe that the number of ML-related posts has changed over time for different programming languages; For example, the number of ML-related posts increases for PLs such as Java, R, and Python. The majority of highly-rated questions about ML development are associated with the model requirement, model building, and evaluation phases. The model building phase alone comprises about one-third of all analyzed posts. Answering time for questions about ML development in Python is significantly less than ML development questions using other programming languages. It was not unexpected for Python to become the most popular PL in the ML community. However, due to the significant amount of ML-related questions using other programming languages, ML educators and tool developers should ensure that adequate ML tutorials and tools for languages other than Python support ML developers.

5.2 Limitations

The limitations for the first part of our research are as follow:

- Since we chose survey as the methodology of our first study to ask participants about their ML domain experiences, the potential biases in responses from the participants may threaten the validity of results.

- We didn't find any tool or method to ensure the quality of our questionnaire. Therefore we decided to conduct a pilot survey to get feedback from participants on the questionnaire. We applied their comments to the questionnaire and refined the questions to design the final survey questionnaire.

- As we aimed to have the participants' insight in detail, most of the questions were open-ended in our questionnaire. However, this type of question provides more practical information; it may cause a low response rate because of its time-consuming nature.

- We tried to carefully select the participants based on their professional profiles in LinkedIn and their contribution to machine learning projects in GitHub. But this group of participants may not represent ML practitioners' general population as our survey didn't get a high number of responses.

We can list the limitations of the second study as below:

- Although manual analysis of posts can leverage the posts' classification accuracy and help determine the irrelevant selected posts, it is time-consuming and exhausting.

- We only analyzed the SO posts related to popular PLs, including the extensively used PLs for ML development, such as Python, and languages used for scientific software (Matlab and R). We excluded some other PLs because of their inefficient number of high-rated posts.

- Since some developers may use other platforms than SO, such as specific forums or internal mailing systems, to raise their ML challenges, identifying the challenges according to SO posts may not be generalizable.

5.3 Future Research

To categorize ML challenges, we can consider additional aspects like users' geographical regions to understand whether there is a relation between the type of challenges and region or not. Besides, we can identify the areas which contribute more to address ML challenges. As Schenk et al [19] investigated geo-locating the knowledge transfer in Stack Overflow, we can replicate this work but for a specified topic such as ML challenges. A survey can be conducted in different regions of the world as a supplementary study to generalize the results.

REFERENCES

[1] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Montreal, Canada, 25-31 May 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8804457

[2] F. Khomh and G. Antoniol. (2018) Bringing ai and machine learning data science into operation. [Online]. Available: https://www.redhat.com/en/blog/bringing-ai-and-machine-learning-data-science-operation

[3] F. Khomh, B. Adams, J. Cheng, M. Fokaefs, and G. Antoniol, "Software Engineering for Machine-Learning Applications: The Road Ahead," *IEEE Software*, vol. 35, pp. 81–84, 2018.

[4] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," In Proc NIPS'15 NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems, December 2015, pp. 2503–2511. [Online]. Available: https://dl.acm.org/doi/10.5555/2969442.2969519

[5] M. J. Islam, H. A. Nguyen, R. Pan, and H. Rajan, "What do developers ask about ml libraries? a large-scale study using stack overflow," in *arXiv:1906.11940v1*, 2019, pp. 1–13. [Online]. Available: https://arxiv.org/abs/1906.11940

[6] A. Joorabchi, M. English, and A. E. Mahdi, "Text mining stackoverflow: Towards an Insight into Challenges and Subject-Related Difficulties Faced by Computer Science Learners," *Journal of Enterprise Information Management*, vol. 29, no. 2, pp. 255–275, 2016.

[7] A. McIntosh, S. Hassan, and A. Hindle, "What can android mobile app developers do about the energy consumption of machine learning?" *Empirical Software Engineering*, vol. 24, pp. 562–601, 2018.

[8] M. Alshangiti, H. Sapkota, P. K. Murukannaiah, X. Liu, and Q. Yu, "Why is developing machine learning applications challenging? a study on stack overflow posts," *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–11, 19-20 September 2019.

[9] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?" in *ICSE '11: Proceedings of the 33rd International Conference on Software Engineering*, May 2011, pp. 804–807. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1985793.1985907

[10] S. Ahmed and M. Bagherzadeh, "What do concurrency developers ask about? a large-scale study using stack overflow," in *ESEM '18: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, October 2018, pp. 1–10. [Online]. Available: https://dl.acm.org/doi/10.1145/3239235.3239524

[11] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, pp. 1192–1223, 2016.

[12] X.-L. Yang, D. Lo, X. Xia, Z. Wan, and J.-L. Sun, "What Security Questions Do Developers Ask? A Large-Scale Study of Stack Overflow Posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016.

[13] R. Gupta and K. Reddy, "Learning from gurus: Analysis and modeling of reopened questions on stack overflow," in *CODS '16: Proceedings of the 3rd IKDD Conference on Data Science*, March 2016. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2888451.2888460

[14] L. Mamykina, B. Manoim, M. Mittal, G. H. Hripcsak, and B. Hartmann, "Design lessons from the fastest q&a site in the west," in *CHI '11: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, May 2011. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1978942.1979366

[15] C. Parnin and C. Treude, "Measuring api documentation on the web," in *Web2SE '11: Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering*, May 2011. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1984701.1984706

[16] C. Chen and Z. Xing, "Mining technology landscape from stack overflow," in *ESEM '16: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, September 2016. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2961111.2962588

[17] S. Meldrum, S. A. Licorish, and B. T. R. Savarimuthu, "Crowdsourced knowledge on stack overflow: A systematic mapping study," in *EASE'17: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, June 2017. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3084226.3084267

[18] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, pp. 619–654, 2014. [Online]. Available: https://link.springer.com/article/10.1007/s10664-012-9231-y

[19] D. Schenk and M. Lungu, "Geo-locating the knowledge transfer in stackoverflow," in *SSE 2013: Proceedings of the 2013 International Workshop on Social Software Engineering*, August 2013. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2501535.2501540

[20] C. Stanley and M. D. Byrne, "Predicting tags for stack overflow posts," in *Proceedings of ICCM*, 2013. [Online]. Available: http://creu-research.ysu.edu/docs/stanley.pdf

[21] B. Yang and S. Manandhar, "Exploring user expertise and descriptive ability in community question answering," in *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*. Beijing, China: IEEE, 17-20 August 2014. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6921604

[22] A. Baltadzhieva and G. Chrupała, "Predicting the quality of questions on stackoverflow," in *Proceedings of the International Conference Recent Advances in Natural Language Processing*, 2015, pp. 32–40. [Online]. Available: https://www.aclweb.org/anthology/R15-1005.pdf

[23] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak, "Analyzing and predicting question quality in community question answering services," in *WWW '12 Companion: Proceedings of the 21st International Conference on World Wide Web*, April 2012. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/8082

[24] Y. Jin, X. Yang, R. G. Kula, E. Choi, K. Inoue, and H. Iida, "Quick trigger on stack overflow: A study of gamification-influenced member tendencies," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. Florence, Italy: IEEE, 16-17 May 2015. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7180111/authors#authors

[25] D. Anand and S. Ravichandran, "Investigations into the goodness of posts in Q&A forums—popularity versus quality," *Information Systems Design and Intelligent Applications*, vol. 340, pp. 639–647, 2015. [Online]. Available: https://link.springer.com/chapter/10.1007/978-81-322-2247-7_65

[26] S. Wang, D. Lo, and L. Jiang, "An empirical study on developer interactions in stackoverflow," in *SAC '13: Proceedings of the 28th Annual ACM Symposium on Applied Computing*, March 2013. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2480362.2480557

[27] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider, "Answering questions about unanswered questions of stack overflow," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. San Francisco, USA: IEEE, 18-19 May 2013. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6624015

[28] B. Shao and J. Yan, "Recommending answerers for stack overflow with lda model," in *ChineseCSCW '17: Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*, September 2017. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3127404.3127426

[29] L. Yang, S. Bao, Q. Lin, S. Jiao, X. Wu, D. Han, Z. Su, and Y. Yu, "Analyzing and predicting not-answered questions in community-based question answering services," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. AAAI, August 2011. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/8082

[30] P. Bourque and R. E. Fairley, Eds., *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*, 3rd ed. Washington, United States: IEEE Computer Society, 2014. [Online]. Available: https://dl.acm.org/doi/book/10.5555/2616205

[31] M. S. Rahman, E. Rivera, F. Khomh, Y.-G. Gueheneuc, and B. Lehnert, "Machine learning software engineering in practice: An industrial case study," in *arXiv:1906.07154*, June 2019. [Online]. Available: https://arxiv.org/abs/1906.07154

[32] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, pp. 15–24, 2016. [Online]. Available: https://www.jstor.org/stable/26167741

[33] H. Khalajzadeh, M. Abdelrazek, J. Grundy, J. Hosking, and Q. He, "A survey of current end-user data analytics tool support," in *2018 IEEE International Congress on Big Data (BigData Congress)*. San Francisco, USA: IEEE, 2-7 July 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8457729

[34] F. Kumeno, "Sofware engineering challenges for machine learning applications: A literature review," *Intelligent Decision Technologies*, vol. 14, pp. 463–476, 2019. [On-

line]. Available: https://content.iospress.com/articles/intelligent-decision-technologies/idt190160

[35] G. Hains, A. Jakobsson, and Y. Khmelevsky, "Towards formal methods and software engineering for deep learning: Security, safety and productivity for dl systems development," in *2018 Annual IEEE International Systems Conference (SysCon)*. Vancouver, Canada: IEEE, 23-26 April 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8369576

[36] A. Senyard, E. Kazmierczak, and L. Sterling, "Software engineering methods for neural networks," in *Tenth Asia-Pacific Software Engineering Conference*, Chiang Mai, Thailand, 12 December 2003. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1254402/authors#authors

[37] M. Borg, C. Englund, K. Wnuk, B. Duran, C. Levandowski, S. Gao, Y. Tan, H. Kaijser, H. Lonn, and J. Tornqvist, "Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry," in *arXiv:1812.05389*, 2018. [Online]. Available: https://arxiv.org/abs/1812.05389

[38] C. Otte, "Safe and interpretable machine learning: A methodological review," in *Computational Intelligence in Intelligent Data Analysis, Studies in Computational Intelligence*, vol. 445. Springer, 2013, pp. 111–122. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-32378-2_8

[39] N. Polyzotis, S. Roy, S. E. Whang, and M. A. Zinkevich, "Data lifecycle challenges in production machine learning: A survey," in *ACM SIGMOD Record*, December 2018. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3299887.3299891

[40] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, and I. Crnkovic, "A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation," in *International Conference on Agile Software Development*, vol. 335. Springer, 2019, pp. 227–243. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-030-19034-7_14

[41] I. Flaounas, "Beyond the technical challenges for deploying machine learning solutions in a software company," in *arXiv:1708.02363*, 2017. [Online]. Available: https://arxiv.org/abs/1708.02363

[42] P. Kriens and T. Verbelen, "Software engineering practices for machine learning," in *eprint arXiv:1906.10366*, June 2019, pp. 1–5. [Online]. Available: https://arxiv.org/abs/1906.10366

[43] J. Tullio, A. K. Dey, J. Chalecki, and J. Anthony, "How it works: a field study of non-technical users interacting with an intelligent system," in *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Springer, April 2007, pp. 31–40. [Online]. Available: https://dl.acm.org/doi/10.1145/1240624.1240630

[44] D. J.-L. Lee, S. Macke, D. Xin, A. Lee, S. Huang, and A. Parameswaran, "A Human-in-the-loop Perspective on AutoML: Milestones and the Road Ahead," in *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2019. [Online]. Available: http://dorisjunglinlee.com/files/MILE.pdf

[45] C. E. Brodley, U. Rebbapragada, K. Small, and B. Wallace, "Challenges and opportunities in applied machine learning," *AI Magazine*, vol. 33, pp. 11–24, 2012. [Online]. Available: https://ojs.aaai.org//index.php/aimagazine/article/view/2367

[46] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9000651

[47] J. Sekhon and C. Fleming, "Towards improved testing for deep learning," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. Montreal, Canada: IEEE, 25-31 May 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8805668

[48] S. Schelter, F. Biessmann, T. Januschowski, D. Salinas, S. Seufert, and G. Szarvas. (2018) On challenges in machine learning model management. [Online]. Available: https://assets.amazon.science/7d/38/968b82c745bd9859a79dab0aade8/on-challenges-in-machine-learning-model-management.pdf

[49] A. Arpteg, B. Brinne, L. Crnkovic-Friis, and J. Bosch, "Software engineering challenges of deep learning," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Prague, Czech Republic: IEEE, 29-31 August 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8498185

[50] L. Ma, F. Juefei-Xu, M. Xue, Q. Hu, S. Chen, B. Li, Y. Liu, J. Zhao, J. Yin, and S. See, "Secure Deep Learning Engineering: A Software Quality Assurance Perspective," in *arXiv:1810.04538*, 2018. [Online]. Available: https://arxiv.org/abs/1810.04538

[51] K. Patel, J. Anthony, J. A. Landay, and B. L. Harrison, "Investigating statistical machine learning as a tool for software development," in *CHI '08: Proceedings of the*

SIGCHI Conference on Human Factors in Computing Systems, April 2008, p. 667–676. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1357054.1357160

[52] F. Ishikawa and N. Yoshioka, "How do engineers perceive difficulties in engineering of machine-learning systems?: questionnaire survey," in *CESSER-IP '19: Proceedings of the Joint 7th International Workshop on Conducting Empirical Studies in Industry and 6th International Workshop on Software Engineering Research and Industrial Practice*, May 2019, pp. 2–9. [Online]. Available: https://dl.acm.org/doi/10.1109/CESSER-IP.2019.00009

[53] I. Portugal, P. Alencar, and D. Cowan, "A Preliminary Survey on Domain-Specific Languages for Machine Learning in Big Data," in *2016 IEEE International Conference on Software Science, Technology and Engineering (SWSTE)*. Beer Sheva, Israel: IEEE, 23-24 June 2016. [Online]. Available: https://arxiv.org/abs/1810.04538

[54] L. Baier, F. Johren, and S. Seebacher, "Challenges in the deployment and operation of machine learning in practice," in *27th European Conference on Information Systems (ECIS)*, Stockholm & Uppsala, Sweden, 8-14 June 2019. [Online]. Available: https://aisel.aisnet.org/ecis2019_rp/163

[55] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "What's your ML Test Score? a rubric for ML production systems," in *Reliable Machine Learning in the Wild - 30th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 2016, pp. 1–5. [Online]. Available: https://research.google/pubs/pub45742/

[56] K. R. Varshney and H. Alemzadeh, "On the Safety of Machine Learning: Cyber-Physical Systems, Decision Sciences, and Data Products," *Big Data*, vol. 5, pp. 246–255, 2017. [Online]. Available: https://www.liebertpub.com/doi/abs/10.1089/big.2016.0051

[57] L. A. Johnson. (2019) DO-178B, Software Considerations in Airborne Systems and Equipment Certification. [Online]. Available: http://www.dcs.gla.ac.uk/~johnson/teaching/safety/reports/schad.html

[58] D. Dahlmeier, "On the Challenges of Translating NLP Research into Commercial Products," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*. Vancouver, Canada: Association for Computational Linguistics, 30 July- 4 August 2017, pp. 92–96. [Online]. Available: https://www.aclweb.org/anthology/P17-2015.pdf

[59] S. Russell, D. Dewey, and M. Tegmark, "Research priorities for robust and beneficial artificial intelligence," *AI Magazine*, vol. 36, pp. 105–114, 2015. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/2577

[60] New Vantage Partners (NVP), *Big data and AI executive survey 2019: How big data and AI are accelerating business transformation*, 2019. [Online]. Available: http://newvantage.com/wp-content/uploads/2018/12/Big-Data-Executive-Survey-2019-Findings-122718.pdf

[61] D. Sculley, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, *Machine learning: The high-interest credit card of technical debt*, 2014.

[62] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, ""hidden technical debt in machine learning systems," in *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems*, December 2015, pp. 2503–2511. [Online]. Available: https://dl.acm.org/doi/10.5555/2969442.2969519

[63] J. Atwood, *Introducing Stack Exchange Data Explorer*, 2010. [Online]. Available: https://stackoverflow.blog/2010/06/13/introducing-stack-exchange-data-explorer/

[64] K. W. Hipel, *Time series modelling of water resources and environmental systems*. Elsevier Amsterdam; New York, 1994. [Online]. Available: https://www.elsevier.com/books/time-series-modelling-of-water-resources-and-environmental-systems/hipel/978-0-444-89270-6

[65] C. Libiseller and A. Grimvall, "Performance of partial mann–kendall tests for trend detection in the presence of covariates," *Environmetrics*, vol. 13, no. 1, pp. 71–84. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/env.507

[66] P. K. Sen, "Estimates of the regression coefficient based on kendall's tau," *Journal of the American Statistical Association*, vol. 63, p. 1379–1389, 2012.

[67] J. Lanzante, "Resistant, robust and non-parametric techniques for the analysis of climate data: Theory and examples, including applications to historical radiosonde station data," *International Journal of Climatology*, vol. 16, pp. 1197–1226, 1996.

[68] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[69] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[70] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: A practical and powerful approach to multiple testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[71] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[72] A. A. Bangash, H. Sahar, S. Chowdhury, A. W. Wong, A. Hindle, and K. Ali, "What do developers know about machine learning:a study of ml discussions on stackoverflow," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, Montreal, Canada, 25-31 May 2019, pp. 260–264. [Online]. Available: https://ieeexplore.ieee.org/document/8816808

[73] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[74] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison, "Investigating statistical machine learning as a tool for software development," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, July 2008, p. 1563–1566. [Online]. Available: https://dl.acm.org/doi/10.5555/1620270.1620333

[75] Q. Yang, J. Suh, N. Chen, and G. A. Ramos, "Grounding interactive machine learning tool design in how non-experts actually build models," in *DIS '18: Proceedings of the 2018 Designing Interactive Systems Conference*, June 2018, pp. 573–584. [Online]. Available: https://dl.acm.org/doi/10.1145/3196709.3196729

APPENDICES

This chapter includes the questionnaire designed for our survey study. This questionnaire consists of three main parts as follows.

Development/Testing practices for Machine Learning (ML) Applications (Part 1)

We hereby invite you to participate in a research project aimed at considering the testing methods which Machine Learning (ML) engineers use. You have been selected for this project because of your experiences in developing Machine Learning applications.

We invite you to ask any question you might have to the researcher in charge of the project or the other members of the research team and to ask them to explain to you any word or information that is not clear.

This survey has three parts:
Part 1 (this form): Personal and organizational information
Part 2: Data processing and feature engineering
Part 3: Training, evaluation, and deployment of ML models

We would very much appreciate if you could fill all the parts.

Please note that if some questions do not apply to your development activities, please write "Not Applicable".
* Required

1. Your response ID : *

Organizational Information

2. Q1. What is type of your organization ? *

Mark only one oval.

- ◯ Academic
- ◯ Industry
- ◯ Other: _____

3. Q2. What are the common application types you typically develop in your organisation? *

Check all that apply.

- [ ] Games
- [ ] Communication Software
- [ ] E-commerce
- [ ] Healthcare
- [ ] Business Intelligence
- [ ] Environment data analysis and forecasting
- [ ] Social Network analytics
- [ ] Multimedia and entertainment
- [ ] Educational Software
- [ ] Security
- [ ] Document Processing

Other: ☐ _____

4. Q3. What ML domains are associated with the applications developed in your organization? *

Check all that apply.

- [ ] Natural Language Processing
- [ ] Image Processing
- [ ] Video Processing
- [ ] Speech Processing
- [ ] Predictive analytics
- [ ] Clustering

Other: ☐ _____

5. Q4. What software development process (e.g., agile methodology SCRUM) is used to develop ML applications in your organization? *

Personal Information and Experience

6. Q1. How long have you been working in software development? *

Mark only one oval.

- ◯ Less than 1 year
- ◯ 2 Years
- ◯ 3 Years
- ◯ 4 years
- ◯ 5 Years or more

7. Q2. How long have you been working on ML application development? *

Mark only one oval.

- ◯ Less than 1 year
- ◯ 2 Years
- ◯ 3 Years
- ◯ 4 years
- ◯ 5 Years or more

8. Q3. What is your highest level of education or training?

Mark only one oval.

- ◯ Diploma or certification
- ◯ Bachelor
- ◯ Masters
- ◯ PhD or above
- ◯ Other: _____

9. Q4. What programming language(s) do you use for machine learning ? *

Check all that apply.

- ☐ Java
- ☐ Python
- ☐ R
- ☐ MATLAB
- ☐ C++

Other: ☐ _____

10. Q5. What framework(s) do you use for machine learning? *

Check all that apply.

- ☐ TensorFlow
- ☐ PyTorch
- ☐ Keras
- ☐ Theano
- ☐ Deeplearning4j
- ☐ MXNet
- ☐ Microsoft Cognitive Toolkit
- ☐ Caffe
- ☐ Chainer

Other: ☐ _____

11. Q6. What is your job title? *

Mark only one oval.

- ◯ Data Analyst
- ◯ Data Scientist
- ◯ Data Engineer
- ◯ AI/ML Engineer
- ◯ AI/ML Developer
- ◯ AI/ML Researcher/Scientist
- ◯ ML Software Architect
- ◯ Full Stack Developer
- ◯ Other: _____

12. Q7. Please briefly list your key job responsibilities. *

_____

_____

_____

_____

_____

Development/Testing practices for Machine Learning (ML) Applications (Part 2)

We hereby invite you to participate in a research project aimed at understanding the testing methods used by Machine Learning (ML) engineers. You have been selected for this project because of your experiences in developing Machine Learning applications.

Should you have any questions about this survey please contact the researcher in charge of the project or the other members of the research team.

If a question does not apply to your development activities, please write "Not Applicable" as response.

* Required

1. Your response ID: *

Data for Machine Learning

In this section, you will be sharing your data analysis experience in the context of ML application development.

2. Q1. What are the primary source(s) of data for Machine Learning in your organization? Please check all applicable. *

Check all that apply.

- [ ] Open datasets
- [ ] Internal company data
- [ ] Data from third-party clients

Other: [ ] _____

3. Q2. Once you have raw data, how do you test the quality of the data? *

4. Q3. Do you use any specific tool to test the quality of data?

   Mark only one oval.

   - ◯ Yes
   - ◯ No
   - ◯ Other: _____

5. Q4. What kind of tools do you use for data cleaning? *

6. Q5. In your opinion, what are the key limitations/challenges of the current data cleaning tools/methods? *

7. Q6. Once you have defined/implemented functions for data cleaning and transformation, how do you test the correctness of your code? *

Check all that apply.

- [ ] Manually checking the results for test samples
- [ ] Debugging the code
- [ ] Validating results by automated scripts
- [ ] Visualization of output data
- [ ] Descriptive statistics or data distribution analysis

Other: [ ] _____

8. Q7. What are the quality attributes that you consider important for your machine learning dataset(s)? *

_____

_____

_____

_____

_____

9. Q8. For a large volume of training data, how do you test your data quality in terms of class balancing in the dataset? *

_____

_____

_____

_____

_____

10. Q9. How do you test the accuracy of your data labeling? *

Check all that apply.

- [ ] Manual Investigation
- [ ] Automated script/tools
- [ ] Other: _____

11. Q10. In your opinion, what are the key challenges in labeling data for ML algorithms? *

_____

_____

_____

_____

_____

Feature Engineering

This section is about 'Feature Engineering' for Machine learning. By feature engineering we refer to the activities related to the extraction, transformation, and selection of features for machine learning.

12. Q1. What kind of tools do you use for feature engineering? *

_____

_____

_____

_____

_____

13. Q2. In your opinion, what are the key limitations in existing feature engineering methods/tools? *

14. Q3. In your opinion, what are the key challenges in feature engineering for machine learning applications? *

15. Q4. How do you select features for your model from the list of fields in the data? *

Check all that apply.

- [ ] By incrementally adding features
- [ ] Feature selection is done based on the analysis of data and the data correlation
- [ ] Features are selected based on the domain knowledge
- [ ] Using automated feature selection tools or techniques
- [ ] Other: _____

16. Q5. How do you validate whether your created features represent the characteristics of the dataset? *

Development/Testing practices for Machine Learning (ML) Applications (Part 3)

We hereby invite you to participate in a research project aimed at understanding the testing methods/practices of Machine Learning (ML) engineers. You have been selected for this project because of your experience developing Machine Learning applications.

Should you have any questions about this survey please contact the researcher in charge of the project or the other members of the research team.

In case a question does not apply to your development activities, please provide "Not Applicable" as answer.

* Required

1. Your response ID: *

_____

ML Model Training

This section is related to the testing of the implementation of ML models in the training phase.

2. Q1. Do you write your ML training code from scratch or use machine-learning libraries and--or frameworks? *

Check all that apply.

- [ ] From the scratch
- [ ] Using ML libraries or frameworks

Other: [ ] _____

3. Q2. If you happen to implement your ML training code from scratch, how do you test your code?

4. Q3. If you use ML libraries to write custom functions for ML training, how do you test your implementation?

5. Q5. What software testing techniques or frameworks (if any) do you use for machine learning application testing? *

6. Q6. Which symptoms do you believe indicates the presence of defects in a ML code implementation? *

7. Q7. Which tools (if any) do you use to verify the presence of defects? *

8. Q8. Do you review the code of ML applications in your organization? If yes, which approach do you follow and which tools do you use when performing these code reviews? *

9. Q9. In your opinion, what are the key challenges in reviewing the code of ML applications? *

ML Model Evaluation

This section is related to the testing of the ML models regarding their learned behavior and performance.

10. Q1. Do you conduct testing of ML models in your organization? if yes, please briefly describe your testing strategies. *

11. Q2. What metrics do you use to assess the performance of ML models? *

12. Q3. How often do you rely on black box (without knowing the internal structure or implementation) testing of ML models using test data sets? *

Mark only one oval.

	1	2	3	4	5	
Never	◯	◯	◯	◯	◯	Always

13. Q4. How often do you use white box (knowing the internal structure and implementation) testing techniques for your ML models? *

Mark only one oval.

	1	2	3	4	5	
Never	◯	◯	◯	◯	◯	Always

14. Q5. Do you leverage generative models or other techniques to generate synthetic data and identify adversarial examples for testing? *

Mark only one oval.

	1	2	3	4	5	
Never	◯	◯	◯	◯	◯	Always

15. Q6. What are the key challenges you have experienced when ensuring the correctness of your ML models? *

ML Model Deployment

This section concerns the deployment of ML models. This involves the integration of models in the target environment, performance analysis and monitoring.

16. Q1. Which deployment strategies are adopted in your organization for ML models? *

17. Q2. How do you test the integration of your ML code into your software ecosystem? *

18. Q3. Do you perform post deployment testing of ML applications? *

19. Q4. How often do you monitor model performance of your ML model after deployment?

Mark only one oval.

	1	2	3	4	5	
Never	◯	◯	◯	◯	◯	Always

20. Q5. What are the key performance parameters that you monitor at the post-deployment phase of your ML models? *

21. Q6. In your opinion, what are the key challenges in assessing the correctness of the deployment of ML models? *