

# Towards Good Practices for Collaborative Development of ML-Based Systems

Cristiana Moroz-Dubenco<sup>a</sup>, Bogdan-Eduard-Mădălin Mursa<sup>b</sup> and Mátyás Kuti-Kreszác<sup>c</sup>

*Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj Napoca, Romania*

**Keywords:** Collaborative Development, Machine Learning Systems, Automatization.

**Abstract:** The field of Artificial Intelligence (AI) has rapidly transformed from a buzzword technology to a fundamental aspect of numerous industrial software applications. However, this quick transition has not allowed for the development of robust best practices for designing and implementing processes related to data engineering, machine learning (ML)-based model training, deployment, monitoring, and maintenance. Additionally, the shift from academic experiments to industrial applications has resulted in collaborative development between AI engineers and software engineers who have reduced expertise in established practices for creating highly scalable and easily maintainable processes related to ML models. In this paper, we propose a series of good practices that have been developed as the result of the collaboration between our team of academic researchers in AI and a company specializing in industrial software engineering. We outline the challenges faced and describe the solutions we designed and implemented by surveying the literature and deriving new practices based on our experience.

## 1 INTRODUCTION


Due to recent advances in Artificial Intelligence (AI), which used to be considered an academic topic, there is a surge in demand for integrating AI and, especially, machine learning (ML) capabilities into software products (Makridakis, 2017; Zhang and Lu, 2021). However, there is no standard between the development paradigms used in AI and Software Engineering (SE) (Lorenzoni et al., 2021; Cerqueira et al., 2022). If traditional software systems are built deductively, by translating the rules that control the system behaviour into code, ML techniques learn these rules in a training process, generating the requirements in an inductive manner (Khomh et al., 2018).


What is more, the process of integrating ML components into production-ready applications requires not only a high understanding of those components but also robust engineering mechanisms to guarantee their availability and scalability. Although the scientific literature highlights the necessity of standardized training and deployment processes, there is insufficient literature to guide ML practitioners (Serban


et al., 2020).

Our goal is to find a set of good practices for the deployment process based on our partnership with a well-established company that required academic expertise for integrating AI models into their software products, regardless of the type of ML model involved. As a means to this, we analyze the challenges our team of researchers has encountered and the processes and procedures employed to overcome each of these challenges.

The rest of the paper is structured as follows: in Section 2 we analyze the process of taking ML models from theory to practice and the state-of-the-art related to the problem; subsequently, in Section 3, we examine the challenges we faced and the solutions developed, and we also analyze the human aspects involved by presenting the results of a survey conducted within our industry partners; in Section 4 we propose a set of good practices derived from our experiences and in Section 5 we analyze possible threats to validity; and, finally, in Section 6 we present our conclusions and ideas for future work.

<sup>a</sup>  <https://orcid.org/0009-0008-7672-6453>

<sup>b</sup>  <https://orcid.org/0000-0002-4221-7297>

<sup>c</sup>  <https://orcid.org/0009-0004-4997-2000>

## 2 TRAINING MODELS - FROM THEORY TO PRACTICE

Training artificial intelligence models has become a demanding task in recent times, as there were no established norms or frameworks to facilitate the standardization of the training procedure in the past. This resulted in customized processes that often felt like academic procedures, which were difficult for the industry to replicate on a large scale. As a result, there are various logistical challenges that necessitated immediate collaboration between academic and industry teams.

In the upcoming subsections, we will examine the challenges encountered and the solutions developed by our team of researchers in partnership with a well-established company that required external expertise for a project centered around a series of AI models.

### 2.1 Problem Definition

The scientific literature has highlighted the recent necessity of standardizing the training and deployment process of AI models in new or existing software applications, employing robust mechanisms to ensure their high availability and scalability.

Breaking down the training process is a complex task that requires a meticulous analysis of each atomic component in the pipeline that collectively delivers the model. The pipeline can be broadly and simply described as consisting of the following stages (Ashmore et al., 2021): (1) data modeling (2) model training (3) model deployment (4) model maintenance.

The process of model training begins with data modeling, which encompasses data gathering, preprocessing, and engineering. Each of these subtasks is critical, as any issues encountered in this stage can compromise subsequent steps that rely on the data.

**Challenge 1: Establishing a data delivery protocol** - The lack of standardization of the data delivery routine was crucial to our experiments in the model training phase. We encountered various issues, ranging from missing columns and improperly formatted values (e.g., integers represented as strings) to more complex problems such as excessively large files, corrupted files, and inadequate version control.

**Challenge 2: Training the models** - After preparing the dataset, we faced several challenges during the model training phase, more exactly training multiple AI models using the prepared dataset and selecting the one that demonstrated the best performance.

These challenges were primarily related to resource limitations and performance. Training the models necessitated an environment equipped with

physical resources, libraries, and tools. Additionally, we had to manage the orchestration of the training to ultimately identify the most accurate model for deployment in the subsequent stage.

**Challenge 3: Deploying the models** - While the first two challenges discussed thus far are commonplace in academic experiments, this last challenge presented our team with unfamiliar territory due to the need of using continuous integration and continuous deployment (CI/CD) frameworks suited for industrial architectures, continuous integration (CI) meaning that when data is updated the ML pipeline reruns generating new models, evaluation metrics and explanations while continuous deployment (CD) being the release of new models into a (pre-)production environment. Moreover, if the end-user lacks technical capabilities, it is essential to establish a mechanism that bridges the gap between the user's input and the model's prediction.

In real-world scenarios, it is crucial to have strict supervision over the deployment package to ensure smooth maintenance of the model. However, operational challenges arise as the responsibility of maintaining the model falls on a team that may lack expertise in AI. This adds to what we have called **Challenges related to human factor**, which are discussed in Section 3.4. In the event of a model failure in a live environment, the responsibility of maintenance falls on a team that typically has extensive expertise in DevOps but reduced knowledge in AI. This presents a significant issue due to the black-box nature of AI models, particularly as they grow in complexity, such as with neural networks. As a result, our team faced the problem of finding a solution to this issue and implementing protocols to facilitate maintenance in case of real-time failures.

### 2.2 State-of-the-Art

Serban A. et al (Serban et al., 2020) conducted an outstanding study that examines the extensive range of optimal practices suitable for teams and their structures. The study is a comprehensive case analysis that scrutinizes all the good practices available at each stage of AI model training. A quiz derived from the study was subsequently distributed to corporations actively involved in AI-based applications in the industry. The study findings indicate that there is no universally applicable set of optimal practices for AI model training; rather, their usage is context-dependent and influenced by factors such as the management framework (Agile, Waterfall, etc. (Beck, 2023; Fair, 2012)), team structure, market conditions, project size, and others. Statistical analyses carried out as part of the

study revealed that there is a direct correlation between team growth, resource availability, maturity, and the adoption of good practices and standardization. Specifically, as teams expand and mature, there is a corresponding increase in the rate of optimal practices employed and a greater degree of standardization observed.

With respect to the challenges associated with establishing standardized and reproducible processes for AI model training, deployment, and maintenance, a study of the challenges identified through discussions with several Microsoft teams (Amershi et al., 2019) describe the discovery, administration, and version control of data pertinent to ML applications as posing a greater challenge compared to other forms of software engineering as they can display intricate interdependence and exhibit non-monotonic error behavior.

Although discussions surrounding the establishment of optimal practices for the training, utilization, and maintenance of AI models typically focus on the procedural aspects of these processes, a separate category of pragmatic concerns also emerges through the level of standardization and safety checks incorporated into a process increases, there is often a corresponding increase in the amount of time required for the process to reach completion.

To address this issue, Zhang A. et al. (Zhang et al., 2017) conducted an experiment aimed at investigating the extent to which the training procedures associated with various AI models contribute to their relative time requirements. Based on their findings, the researchers proposed a novel framework called SLAQ, which is a scheduling system designed to prioritize quality in the context of large-scale ML training tasks executed within shared computing clusters.

In the upcoming sections, we will provide an overview of our research team's experience collaborating with a software company in the domain of AI. Leveraging the current state-of-the-art literature, we devised a pipeline of procedural and logistical steps aimed at proposing a scalable process for training AI models and deploying them in a production environment, while prioritizing ease of maintenance and monitoring for a team without previous expertise in the field of AI.

### 3 ADDRESSING CHALLENGES IN THE DEVELOPMENT OF ML-BASED SOLUTIONS

#### 3.1 Data Processing

In our collaboration with the software company team, we needed to establish the protocol for receiving datasets, which may seem straightforward. However, even this simple process presented significant challenges that greatly affected the productivity of our experiments.

**Challenge 1.1:** *Finding an all-in-one solution.* To meet the requirements for a data management platform with features such as warehouse storage for efficient dataset upload/download operations and file content validation, we sought a comprehensive solution that would address all of these issues. Our search led us to Microsoft AzureML, which proved to be an ideal candidate for our needs, as discussed in the previous section.

**Challenge 1.2:** *Integration of new dataset versions during ongoing experiments.* We utilized Microsoft AzureML blob storage as a repository for all the datasets provided by our collaborators. The storage was structured to allow for both horizontal and vertical flexibility, meaning new datasets could be added at any time, and new versions could be appended to existing datasets. This allowed our collaborators to effortlessly integrate new dataset versions at any point of the experiments without disrupting the process. Throughout the experiments, the latest stable version of the dataset was fetched until a new version was deemed stable.

**Challenge 1.3:** *Validation of new dataset versions.* Before a new version could be considered stable, it was subjected to a set of custom rules and validations that were implemented as scripts. The validation scripts aim to verify various aspects of the new version, such as computing the distributions of missing values by column and checking if the number of missing values exceeds a threshold. Additionally, the column names and types are checked against a predefined file definition to ensure that the file structure is as expected and that training processes will not fail due to unexpected data types or values. To automate this process, we set up an AzureML event trigger that automatically runs the validation scripts upon uploading a new version. The output of the validation script is then used to label the version as either "stable" or "not stable".

### 3.2 Model Training

Compared to data modeling, which is a common step in any AI model training process, the model training step is challenging when it comes to understanding why and how a model behaves during training, especially for more complex models that tend to behave like black boxes. Our project focused on two types of models - decision trees (Breiman et al., 1984) and neural networks (Hopfield, 1982) - for forecasting various indicators.

**Challenge 2.1:** *Understanding the models' behaviours.* Decision trees are mathematical models that can be reverse-engineered to understand their training accuracy or result. However, neural networks are much more complex, composed of thousands of operations between their neurons, making it close to impossible for a human mind to replicate their behavior, especially for individuals without expertise in artificial intelligence. Analyzing the weights, biases, and gradients of a neural network's layers to understand its behavior during training requires advanced skills in AI.

Our primary objective was to create an interface that would eliminate the need for advanced analysis and rely solely on a qualitative assessment of the results. We achieved this by implementing standard metrics for each trained model that can be numerically compared (Figure 1), thus shifting the focus from comprehending the intricate architecture of the black box model to a qualitative comparison of the trained models.

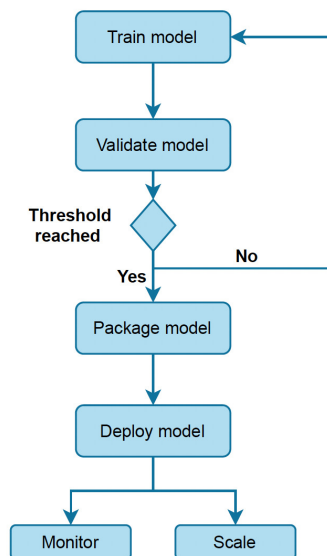


Figure 1: General description of the proposed pipeline for AI model training.

**Challenge 2.2:** *Setting up the training process.* We have established a training pipeline in AzureML by utilizing pre-defined architectures for the aforementioned models. The output of this pipeline is a report consisting of Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-Squared (R<sup>2</sup>) values (Kane, 2013), along with the identification of the model that performed the best based on at least two out of the three metrics. This approach is regarded as elegant by the team without any experience in AI model training since it requires minimal intervention. Whenever a new stable version of a dataset is associated with it, the AzureML pipeline automatically initiates and trains a set of models on a platform maintained by AzureML. Following the series of tests performed to ensure that the best model is selected, it returns the best model to be used.

**Challenge 2.3:** *Creating an environment for models' training.* Although there are differences in the required libraries and resources for each model, we have standardized these variations by setting up the models as AzureML pipeline modules. This one-time setup eliminates the need for external intervention.

### 3.3 Model Deployment

Our aim was to find a set of good practices for the deployment process for the trained models, irrespective of the type of model that performed the best during the training stage, such that the trained models can be deployed and consumed by third parties without the need for technical knowledge of the models used.

**Challenge 3.1:** *Encapsulation of the models for deployment.* We utilized state-of-the-art technologies commonly used for deploying software applications, such as Docker (Merkel, 2014). Docker is a framework that enables the encapsulation of an application and its dependencies, including operating systems, libraries, and other tools, as a container. These containers can be easily deployed using orchestration frameworks like Kubernetes.

**Challenge 3.2:** *Communication with the models.* Following a discussion with our collaborator's engineering team, we concluded that the optimal way of communicating with the trained model was through HTTPS requests of a REST API. AzureML provides a built-in feature to deploy any trained AI model as a REST web service that is automatically encapsulated as a container and deployed on AzureML infrastructure. Additionally, the web service endpoint interface created for prediction consumption is documented using Swagger (Swagger, 2022), a cutting-edge framework that offers a user-friendly interface to describe the structure of REST APIs and generates documenta-

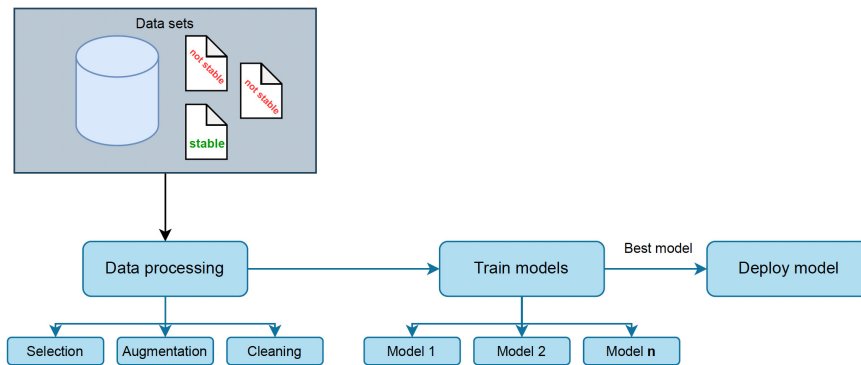


Figure 2: Architecture of the generic process implemented as an AzureML pipeline.

tion that is easily understandable for both developers and non-developers.

**Challenge 3.3:** *Ensuring that the deployed models are easy to use.* The approach we employed guarantees a high level of abstraction, which is reduced to a straightforward input/output protocol. As a user, you are aware of the expected input and output data structures when consuming the latest version of a trained model. Furthermore, the deployment is performed in versions, enabling quick rollback to any previous version in the event of unexpected behavior. Since we are dealing with a web service-based architecture, AzureML offers a built-in option for scaling the number of instances to meet production loads (Figure 2).

We have created an ecosystem that, despite being part of the project architecture, provides a continuous sense of autonomy and reliability as a third party.

### 3.4 Human Aspects in Model Development

Know-how transfer has been designed as part of the project unfolding. Based on the development team assessment, the academic team designed several activities aiming to increase the knowledge in the domain of Artificial Intelligence in general, respectively ML in particular, from both theoretical and applied perspectives. These activities include presentations, tutorials, collaborative working exercises, Q&A sessions, together with weekly technical meetings.

In order to evaluate the knowledge transfer process and to assess the acquired competencies of the development team, we investigated through a survey the apprehension of the development team. We collected 16 answers to this survey. The participants represent positions in the project ranging from junior programmers and programmers to business analysts, lead developers, and database administrators, with an average experience of 10.1 years (ranging from experience ranging from less than a year to more than 20

years).

Despite the initial level of AI knowledge in the team, being close to non-existing, the survey results presented in Figure 3 indicate promising outcomes. The perspectives of each team member on how each component works, starting from data processing steps such as parameter selection to model understanding/implementation, training, and deployment using the proposed CI/CD pipelines, were assessed. The results suggest that each team member’s understanding of these components and their workings beyond their black box implementation is more than promising in most categories. Based on the survey results, the level of understanding of the proposed AI model training process was mostly evaluated as medium. This level of knowledge is deemed sufficient for the maintenance and minor improvement of each step in the process.

As anticipated, the evaluation results demonstrate that neural networks, as a family of AI models, are more challenging to comprehend compared to decision trees across all categories. Nonetheless, the encouraging aspect is that the ratio does not exclusively label neural networks as models that cannot be accommodated in the proposed pipeline. Instead, the results illustrate that the proposed pipeline is well abstracted and can support models with diverse complexities in terms of architectures, ranging from simpler to more complex.

The survey results highlight an interesting finding: despite the team’s limited level of understanding, they exhibit high levels of trust in the deployed models. Interestingly, the trust increases with the complexity of the models, as neural networks were found to have higher trust than decision trees. This observation suggests that the knowledge-sharing process has been effective, as the developers seem to have understood that the more complex architectures yield higher accuracy in the models.

Based on the survey results overviewed in the pre-

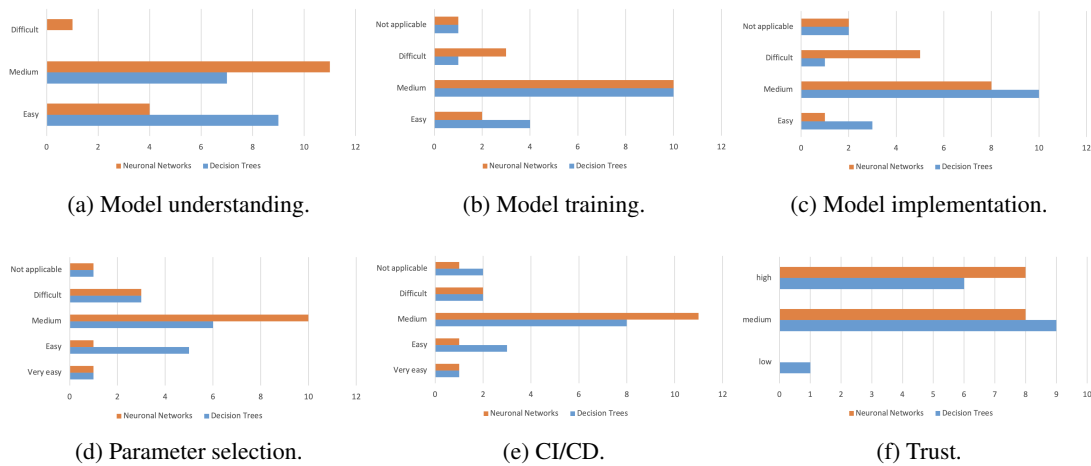


Figure 3: People’s perceptions over aspects related to decision trees and neural networks.

vious paragraphs, it can be concluded that the proposed generic framework for the AI model training process is successful in achieving its intended goal. The survey showed that team members with limited knowledge in the field of AI were able to understand and trust the proposed pipeline, indicating that the framework is well abstracted and easily applicable to a real-world scenario. The pipeline was able to accommodate both simple and complex AI models, including neural networks, demonstrating its versatility and scalability. These findings validate the effectiveness of the proposed framework in simplifying the process of AI model training, enabling software engineering teams to adopt AI technology in a simple and smooth manner.

## 4 TOWARDS PROPOSING GOOD PRACTICES

The insights gained from this collaboration were shaped by the disparity between an experimental undertaking and an industrial enterprise. Notably, numerous industrial applications in the realm of computer science have their roots in academia and have evolved into implementations that adhere to principles that impact the economic aspects of the business. Marketing a product to a customer entails a set of obligations that become a compulsory part of the logistics and maintenance of the said product.

One of the demanding experiences encountered by our academic team was to conform to these principles and transform our experimental methodologies into ones that could be integrated into scalable processes, which could be effectively maintained within state-of-the-art software architectures. The most valu-

able insights acquired during this collaboration are those that we think could serve as a foundation for establishing a set of good practices. The hurdles encountered, ranging from data modeling to model deployment, were overcome through the utilization of state-of-the-art practices, ultimately resulting in the successful training of accurate models and proposing an interface for interacting with said models:

- **Data Processing**

- Utilizing data warehouses that support versioning to streamline the process of uploading datasets. It is critical to ensure that the correct dataset is not lost due to accidental overwriting by a corrupted one.
- Implementing unit testing principles on datasets to validate new versions against a set of rules for each field, such as validating field types. This autonomous and automated step facilitates model training.

- **Model Training**

- Creating an interface that eliminates the need for advanced analysis and shifts the focus from understanding the models’ architecture to a qualitative comparison. Standard metrics that can be numerically compared are much easier to understand and instruct the personnel on than understanding a black box model’s behavior.
- Implementing AI model architecture as modules that can be orchestrated based on a common interface. Platforms like AzureML offer parallel training capabilities for multiple models, enabling the selection of the best model based on exhaustive analysis (Microsoft, 2023).

- **Model Deployment**

- Deploying models as web services to align with modern software architectures characterized by interactions between micro-services over the internet. Our experience suggests that deploying models as web services is a seamless operation from the perspectives of infrastructure and integration. Utilizing containerization principles to create contained models using technologies like Docker to facilitate DevOps activities.

- **Collaborative Development**

- Conducting a series of theoretical sessions, focused on the applications of AI models and their architectures. These sessions should be followed by practical workshops, featuring interactive coding sessions aimed at transforming the traditional black-box approach into a more flexible grey-box approach. The primary objective of this practice is to enable maintenance activities that involve lightweight improvements to existing processes, by providing a better understanding of the internal workings of AI models.

Commencing from the data processing procedures, through to model training and deployment, we underwent a learning curve of contemporary technologies and good practices to determine the optimal combination of tools and frameworks that could accommodate the requirements of an industrial application infrastructure. Given that this domain is nascent, there is no standardization with regard to proposals. Therefore, through our collaboration, we developed bespoke solutions that were tailored to meet our specific needs.

The principal takeaway from our experience was navigating the learning curve of these technologies, wherein we recognized a significant potential for integrating them into our scientific research activities to augment overall productivity. Platforms such as AzureML offer an excellent means of developing experimental pipelines that can execute experiments at scale, curate and archive datasets, and conveniently share outcomes across a team of researchers (Microsoft, 2022).

## 5 THREATS TO VALIDITY

*Internal validity* refers to factors that might influence the obtained results. In our study, they refer to project processes, respectively to the survey conducted as an investigation method for human aspects in model development. While it may be difficult to quantify the validation of our defined practices, we believe that our in-depth research of the existing literature (Sub-

section 2.2) provided a reliable foundation for the practices that we designed and implemented throughout the data processing, model training, and model deployment processes discussed in the previous sections.

The survey was constructed and then validated by members of the research team, not part of the target population. The biggest threat is represented by the small number of participants, so the findings are formulated as lessons learned from this experiential study, with no proposed generalization. As mitigation strategies, we strive to include all members of the project benefiting from the knowledge transfer, with different roles and experiences.

*External validity* refers to the generalization of the findings. The findings are not generalized, but rather formulated as remarks and lessons learned, as the study is an experience report and does not represent a general view of the domain. In our assessment, the proposed good practices represent a suitable level of formalization comparable to other experiments that have been studied in the literature. We believe they can enable more effective collaboration between AI engineers and software engineers, ultimately leading to more efficient and scalable development processes.

## 6 CONCLUSIONS

This paper presents a study based on a collaborative development project of an ML-based system. The project involved a team of researchers with expertise in the field of Artificial Intelligence (AI) and a software engineering company specialized in creating large-scale industrial applications. Through this collaboration, the study examines a series of experiences that were gained during the development process. The study aims to highlight the challenges encountered and the strategies that were employed to address them, to provide insights that can inform future efforts to develop ML-based systems in industrial settings.

Through our collaborative work, we came to realize a gap in the existing literature related to best practices for defining interfaces of communication and technical designs for creating autonomous pipelines. These pipelines are capable of automating the experimental processes involved in data engineering, model training, model deployment, and their maintenance in a real-world infrastructure. Our study highlights the challenges encountered due to this void in the literature and the strategies that were employed to address them. Our work aims to contribute to the development of more robust and efficient processes for the

collaborative development of ML-based systems. We propose a set of good practices for defining interfaces of communication and technical designs for creating autonomous pipelines that can help to streamline the development process and improve the scalability and maintainability of ML-based systems.

In order to address the challenges encountered during the collaborative development of our ML-based system, we structured our difficulties under a series of challenges. We then designed solutions to these challenges based on other state-of-the-art experiments and practices that exist in the literature. Our aim was to combine the knowledge gained from a survey of proposed practices with our own experiences, to design more general and standardized practices that could be applied successfully in future projects. The solutions we proposed were based on careful consideration of the unique needs and constraints of our project, as well as the broader context of industrial ML development. We believe that our approach can help to improve the efficiency and effectiveness of collaborative ML development efforts, while also contributing to the development of more robust and scalable ML-based systems.

## ACKNOWLEDGEMENTS

This research was partially supported by DataSEER project, financed through POC 2014-2020, Action 1.2.1, by European Commission and National Government of Romania (Project ID: 121004).

The authors express their gratitude to the industrial partner, OPTIMA GROUP SRL, for their collaboration and valuable information exchange.

## REFERENCES

- Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., and Zimmermann, T. (2019). Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300.
- Ashmore, R., Calinescu, R., and Paterson, C. (2021). Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *ACM Computing Surveys (CSUR)*, 54(5):1–39.
- Beck, K. (2023). The agile manifesto. agile alliance. <http://agilemanifesto.org/>. Accessed: Apr. 10, 2023.
- Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). Classification and regression trees (wadsworth, belmont, ca). *ISBN-13*, pages 978–0412048418.
- Cerqueira, M., Silva, P., and Fernandes, S. (2022). Systematic literature review on the machine learning approach in software engineering. *American Academic Scientific Research Journal for Engineering, Technology, and Sciences*, 85(1):370–396.
- Fair, J. (2012). Agile versus waterfall: approach is right for my erp project? In *Proceedings of Global Congress 2012—EMEA, Marsailles, France*. Newtown Square, PA: Project Management Institute.
- Hopfield, J. (1982). Neural networks and physical systems with emergent collective properties like those of two-state neurons. *Proc. Natl. Acad. Sci.(USA)*, 79:2554–2558.
- Kane, M. T. (2013). Validating the interpretations and uses of test scores. *Journal of Educational Measurement*, 50(1):1–73.
- Khomh, F., Adams, B., Cheng, J., Fokaefs, M., and Antoniol, G. (2018). Software engineering for machine-learning applications: The road ahead. *IEEE Software*, 35(5):81–84.
- Lorenzoni, G., Alencar, P., Nascimento, N., and Cowan, D. (2021). Machine learning model development from a software engineering perspective: A systematic literature review. *arXiv*.
- Makridakis, S. (2017). The forthcoming artificial intelligence (ai) revolution: Its impact on society and firms. *Futures*, 90:46–60.
- Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2.
- Microsoft (2022). Microsoft customer stories. *Microsoft Azure Blog*.
- Microsoft (2023). Azure machine learning. <https://azure.microsoft.com/en-us/services/machine-learning/>. Accessed: Apr. 10, 2023.
- Serban, A., van der Blom, K., Hoos, H., and Visser, J. (2020). Adoption and effects of software engineering best practices in machine learning. In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. ACM.
- Swagger (2022). Swagger: The world’s most popular framework for apis. <https://swagger.io>. Accessed: Apr. 10, 2023.
- Zhang, C. and Lu, Y. (2021). Study on artificial intelligence: The state of the art and future prospects. *Journal of Industrial Information Integration*, 23:100224.
- Zhang, H., Stafman, L., Or, A., and Freedman, M. J. (2017). SLaq: Quality-driven scheduling for distributed machine learning. In *Proceedings of the 2017 Symposium on Cloud Computing, SoCC '17*, page 390–404, New York, NY, USA. Association for Computing Machinery.