STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVIII**, Number 2, 2023
DOI: 10.24193/subbi.2023.2.04

FACILITATING MODEL TRAINING WITH AUTOMATED TECHNIQUES

BOGDAN-EDUARD-MĂDĂLIN MURSA, MÁTYÁS KUTI-KRESZÁCS, CRISTIANA MOROZ-DUBENCO, AND FLORENTIN BOTA

Abstract. Automating artificial intelligence (AI) model training has emerged as a significant challenge in the field of automation. The complete pipeline from raw data to model deployment poses the need to define robust processes that ensure the efficiency of the services that expose the models. This paper introduces a generic architecture for automating data preparation, training of models, selection of models, and deployment of models as web services for third-party consumption using Microsoft Azure Machine Learning's (AzureML) CI/CD tools. We conducted a practical experiment utilizing AzureML pipelines with predefined and custom modules, demonstrating its readiness for integration into any production application. We also successfully integrated this architecture into a real-world product designed for industrial forecasting. This practical implementation demonstrates the effectiveness and adaptability of our approach, indicating its potential to address diverse training needs.

1. Introduction

Over the past decade, Artificial Intelligence (AI) has evolved from a buzzword to a state-of-the-art technology, providing reliable solutions in various domains such as fraud detection, healthcare, predictive maintenance, energy management, and retail. AI applications are now being used as stand-alone solutions in businesses' core processes. However, as the demand for AI grows exponentially, traditional processes for training AI models, which were once characterized as academic experiments, have now become a logistical problem. There is a need to migrate to reliable processes that can automate all

Received by the editors: 29.06.2023.

2010 *Mathematics Subject Classification.* 68T01.

1998 *CR Categories and Descriptors.* D.2.11 [**SOFTWARE ENGINEERING**]: Software Architectures – *Patterns (e.g., client/server, pipeline, blackboard)*; I.2.1 [**ARTIFICIAL INTELLIGENCE**]: Applications and Expert Systems – *Industrial automation* .

Key words and phrases. Artificial Intelligence, Automation, Optimization.

the steps, from cleaning a raw dataset to training the model and exposing it to other services that can consume it in real-world scenarios [12, 14, 26].

Moreover, there has been an emphasis on the necessity of creating user-friendly interfaces for software developers without an AI background, which can facilitate the maintenance of the AI processes encapsulated in modules, following the principles of gray-box. These requirements have already been addressed in the domain of software engineering, particularly in web programming, using the Continuous Integration/Continuous Deployment (CI/CD) mechanism and web services. In this paper, we present the latest advancements in automating AI model training using AzureML's CI/CD tools. Our approach allows for the automation of data preprocessing, training various AI models and selecting the best performing one, deploying the model as a web service for third-party consumption, while ensuring high availability and scalability of the service.

Our proposed solution was implemented and validated through an academic-industrial partnership aimed at developing a forecasting platform for industrial indicators. Subsequently, we present a comprehensive review of the logistics and challenges associated with AI model training (Section 2), followed by an overview of state-of-the-art techniques and successful applications of automating the AI model training process in various industries (Section 3). In Section 4, the final section of this paper, we describe our proposed generic architecture to automate the model training and deployment process using AzureML.

2. Problem definition

Developing a machine learning based solution in an industrial environment consists of two steps: building the model and deploying it into production. While the first part might be more interesting, it is the second part that takes the longest - all the tasks that come after a model is built and optimized. According to Gartner [1], one of the top five factors due to which 85% of Big Data projects fail is the complexity to deploy them.

Algorithmia's "2021 State of Enterprise ML" [3] reports that 64% of all organizations need at least one month to deploy a machine learning model and that the data scientists from 38% of organizations spend more than 50% of time deploying ML models to production.

According to [4], the development of a machine learning based solution in an industrial environment can be split into four stages. Each of these stages can be split into smaller steps, and [20] presents various issues and concerns associated to each step, as explained below.

The first stage concerns data management - preparing the data. Data is collected, gathered and understood. Problems can appear especially in large production environments, where, using the principle of ”single responsibility” [16], applications are usually built as multiple services communicating to each other, and this can easily lead to data being stored in different locations and forms by different services. After collection, data has to be cleaned and, according to [13], data cleanliness is the main reason causing expert to doubt the quality of their work. Insufficient, biased, noisy, irrelevant or imbalanced data can lead to either the under- or overfitting of the model. Moreover, if multiple data sources have to be integrated into a single one, they might differ in schema, convention or the procedure of storing and accessing the data. Once the data is preprocessed, it might be necessary to be augmented because in real life, more often than not, data is unlabeled and, since supervised learning techniques require labeled data for training, the process of assigning labels to large volumes of data can be a tedious task. And, finally, the labeled data has to be analysed in order to discover biases or unexpected distribution shifts. One area that is especially challenging in this step is visualization for data profiling, since there are very few tools available for efficiently executing this task.

The following stage is about model learning. Firstly, the best-fitted model for the problem at hand needs to be chosen. As shown in [28], complexity is one of the most important factors when choosing a model to be used in an environment with resource constraints. In practice, simpler models, such as Decision Trees, Random Forests, Principal Component Analysis etc., are chosen instead of deep learning or reinforcement learning techniques because they require less resources and also lead to an decreased development and deployment time. Moreover, depending on the field of application, being able to interpret the output of a machine learning model can outbalance even its performance. Then, the model has to be trained. This step usually requires increased computational resources, which leads to increased costs. For instance, in [25], it is shown that the training of the BERT model [7] costs at least $50 K, which can be impossible to afford for many companies. Even more, challenges can arise while selecting hyper-parameters in order to find the optimal setting for the model. The size of the hyper-parameter optimization task can grow exponentially in the worst case, leading to tough computational challenges. Mainly when talking about deep learning techniques, this process can be extremely expensive and resource-heavy.

Once the model is trained, it must be verified. This stage involves defining requirements for the model, which should not solely prioritize increased model performance but also consider business-driven metrics for evaluating and monitoring the model in a production environment. Verification of all requirements is necessary, including adherence to business-defined regulatory frameworks, in addition to mathematical correctness or error bounds. Testing the model in a real-life setting is ideal for ensuring quality, but this can pose challenges regarding safety, security, and scaling.

And, finally, the last stage in the development of an ML-based solution in industrial environments is the deployment of the model. The model has to be implemented so that it can be consumed and an infrastructure for running the model needs to be built. Due to the fact that machine learning models can explicitly depend on external data, a lot of engineering anti-patterns, such as correction cascades, are widespread in software that uses ML [24]. Once the system is deployed to production, its maintenance intervenes. The system has to be monitored, but this process is still in the early stages within the ML community, and monitoring the overall performance of a machine learning model is still an open problem. Yet, the biggest challenge emerges when the existing models needs to be adapted to new data and the new model artifact delivered to production. While software engineering solves this problem using continuous delivery, things are more complicated with machine learning problems because, unlike regular software products that only have changes in code, ML solutions can be changed on three aspects: data, model, and code.

Therefore, we can state that the process of developing machine learning based solutions in production environments requires not only skills and time, but also new and emerging technologies that can ease the process.

3. Advances in Automated AI Training

Recently, AI has emerged as a cutting-edge technology with numerous use cases across diverse industries. AI models have been employed either as novel solutions to problems, as assisting systems, or as complete replacements for human intervention. However, this rise in demand has revealed certain challenges, such as the necessity for dependable model training, deployment, and real-time production consumption mechanisms. Large cloud service providers have promptly identified this demand and addressed it by offering a comprehensive suite of tools that facilitate large-scale AI model training and deployment. Furthermore, modern CI/CD mechanisms [15] have been integrated to ensure scalable solutions that meet the needs of high availability, performance, and logging.

AzureML [18], a cloud-based machine learning platform developed by Microsoft, provides a comprehensive suite of tools necessary for the implementation of artificial intelligence models for various use cases. With its ability to integrate with different AI model architectures, it has been utilized by several large companies as a trusted ecosystem for the entire pipeline from training to deployment. For instance, American Express uses AzureML to develop apps for fraud detection, Mediktor employs it for healthcare solutions to check symptoms, E-ON applies it to manage energy in solar panel farms and predict energy solutions, Belfius uses it to help detect fraud and money laundering, Cognizant and Claro personalize and improve users' learning experience with AzureML and Epiroc advances manufacturing innovation with its help [19].

On the other side, AzureML gains popularity in the scientific literature as well. [22] analyzes eight two-class and three multi-class machine learning algorithms for network intrusion detection using AzureML as a solution to the limitation of traditional models, which focus rather on the improvement of the attack detection rate and the reduction of false alarms rather than on time efficiency. The proposed algorithms are analyzed and evaluated not only on their performance related to the task, but also in terms of training and prediction time, concluding that the use of AzureML leads to saving computational resources and, thus, reducing excessive costs. Moreover, the study highlights the fact the AzureML is useful for large datasets handling, as it can successfully serve as an expedient Integrated Development Environment.

In [23], a data-driven machine learning workflow is proposed for forecasting the outcome of Business to Business (B2B) sales. The workflow was implemented and deployed to a B2B consulting firm's sales pipeline using the AzureML platform. This cloud-based solution was chosen because it can be easily integrated into existing Customer Relationship Management (CRM) systems allowing for more scalability than traditional solutions. What is more, AzureML supports the creation of models' endpoints on Azure Kubernetes Service, which ensures high scalability and low latency for request-response service, therefore being suitable for production-level deployments. The authors conclude that the AzureML-based workflow is also highly sustainable due to relying on cloud computing power, rather than on on-premise resources.

As a solution for the task of designing the complex systems that an electrical machine consists of, [21] employ AzureML as a means of optimization and best candidate selection. That is, the platform was used to compare two searching algorithms, namely Boosted Decision Tree and Multiclass Neural Network, in order to predict the best configuration of an electric motor according to the maximum efficiency. The advantages presented by the usage of the platform as compared to the tools used on the developers' local computers consist in the 10-times decrease in work time and the simplification of the project creation, taking 15 minutes instead of 3 days.

In [27], one of the main capabilities of the AzureML platform is presented in-depth: automated machine learning. In order to help developers with little ML-related knowledge to build solutions that employ ML models, AzureML provides the possibility of using automated ML, such that developers do not need to fully understand the processes of selecting the learning algorithm or tuning the hyper parameters. Being given a dataset and only a few configuration parameters, AzureML can provide a high-quality, already trained, ML model that can be used for predictions, without further modifications.

The paper [8] once again demonstrates the advantages of the automated machine learning capability of the AzureML platform. Being applied in medical environment, specifically, for predicting antimicrobial resistance and selecting appropriate treatment, the benefit of automated ML presents itself. This paper presents a procedure that is easy applicable and, most importantly, can be explained and even used by non-technical experts, leading to the conclusion that AzureML can be used as a decision tool for physicians, the deduced models proving good performance.

In this paper, we describe our process of training a scalable number of AI models for a generic forecasting platform. Alike explained in [22], with our approach, we also intend to reduce the computational costs and use a solution that easily handles large datasets. Since we need high scalability and low latency and, most importantly, an architecture that is suitable for production-level deployments, our choice is based on the same reasons as presented in [23]. Similar to the work presented in [21, 27, 8], we need to train multiple AI models and choose the one that yields the best results. And, finally, the most notable characteristic of the AzureML platform that motivates the choice of AzureML for all of the presented solutions, including our own, is the possibility of being used even by non ML-specialized developers. For all of these reasons, our proposed procedure is inspired by these successful stories of AzureML's deployment, combining and adjusting the parts that suit our needs.

4. Automating Model Training Process

The present section provides an overview of an experiment that involves utilizing the most advanced state-of-the-art guidelines to define and construct a sturdy pipeline comprising all necessary steps required for training a model, using one or more datasets, and deploying it onto an infrastructure to enable real-time consumption by clients. Our objective is to streamline the conventional process of training AI models, enabling individuals without specialized expertise to train and deploy models through an automated procedure to the greatest extent possible.

The following sections will elaborate on an anonymous methodology that our university team employed while collaborating with a company operating in the IT industry that sought assistance in the field of artificial intelligence. Given that our deliverable was intended for utilization in a live production environment, our aim was to incorporate, right from the beginning, techniques used within the software development industry, which enable principles such as scalability, versioning, monitoring, and high availability. By implementing such techniques, we intended to ensure the functionality and effectiveness of our deliverable in a demanding production environment.

Furthermore, within the scope of our collaboration, we agreed to deliver a preliminary product consisting of a series of trained neural networks, and in parallel, we developed a protocol for knowledge transfer that enabled the company to take ownership of the pipeline process that we had designed. As the company's team of developers lacked expertise in the field of artificial intelligence, we developed a grey-box mechanism that allowed them to not only implement future model training but also edit and enhance modules defined within the pipeline. To accomplish this, we proposed a generic process with easily generalized steps that could be assembled into a framework with user-friendly features. We initiated this approach from the very beginning of the project to ensure that the process could be readily integrated into the existing framework, and easily manipulated like a puzzle.

After brainstorming with the architect of the company, we reached an agreement on a solution that was compatible with their current technology stack. This solution entailed using Microsoft AzureML as a framework to define modules, which were coded in Python, to facilitate data storage, data cleaning and augmentation, model training, and model deployment. Through this solution, we aimed to create a cohesive and efficient pipeline for the company, which would facilitate the utilization of the current technology stack, while also enhancing the process of developing and implementing AI models.

4.1. **Pipelines architecture.** During the architecture brainstorming process, one of the most important considerations was the business requirement of receiving various datasets from clients and subsequently triggering the pipeline. These datasets were sourced from diverse warehouses and in different formats, including SQL databases, NoSQL databases, and files. Consequently, a layer was required to abstract the source of the training data and to allow for pipeline interaction, providing an interface that exposed simple data downloading operations, but also the uploading of the results. This approach enabled us to provide a seamless and flexible pipeline that could process a wide range of data sources, while also facilitating the integration of multiple data sources into the pipeline. In this particular case we can consider that these modules will receive the dataset as input, and output it with the data processing operations applied.

Figure 1 represents an overview of the proposed pipeline that represents a state-of-the-art mix between software engineering principles in CI/CD deployments and procedures of AI model training. After defining the data interface,

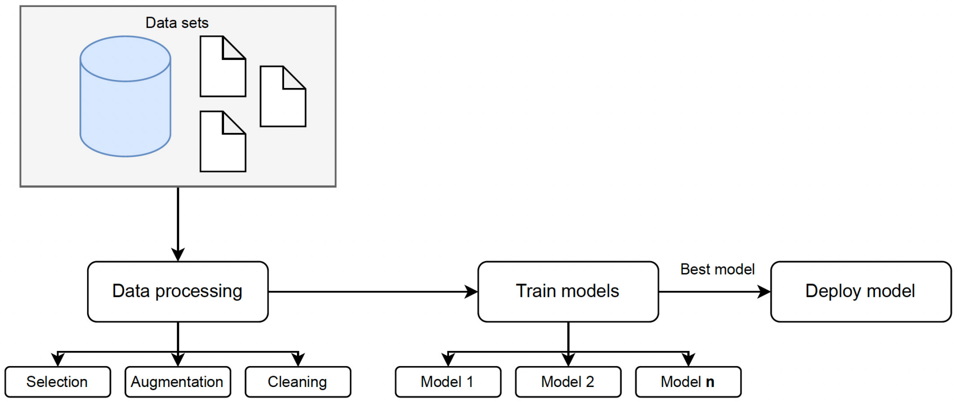

FIGURE 1. Generalized architecture of a model training pipeline

we proceeded to develop a series of modules that would code the state-of-the-art operations required in the process of training an AI model. Each module was defined through an interface, and each model had an input and output, enabling seamless assembly within the desired pipeline. To initiate the pipeline, we first defined modules for the most well-known operations of data processing, beginning with data cleaning (i.e., removing empty values, deduplication, and filling incomplete data), data augmentation (i.e., one-hot encoding of categorical columns or data normalization), and column selection (i.e., selecting only specific columns if needed). Through this process, we aimed to establish a flexible and scalable pipeline capable of performing various data processing operations to facilitate the effective and efficient training of AI models.

After the data cleaning process, the next step in our pipeline involved model training. This stage was comprised of an umbrella module that encapsulated various sub-models, each representing a different type of model such as neural networks, logistic regressions, decision trees, and many others. The module initiated the training of all these models and subsequently chose the model with the highest metric, which was determined based on its accuracy. The output of this pipeline module was a flexible binary file that could be deployed in the subsequent stage of model deployment.

To expose the trained binary model to other components of the company's software stack, we implemented a mechanism that allowed for the input of new data and retrieval of inferred results. Following best practices in web development, we decided to utilize microservices due to their capabilities in high availability, versioning, and scalability.

Following the architecture described, we propose a hands-on application of Microsoft AzureML pipelines and its features following the state-of-the-art CI/CD principles that will enable the desired automatization of the AI tasks [9, 11, 5].

4.2. **AzureML Pipelines.** The Microsoft AzureML platform represents a significant advancement in cloud-based technology, providing users with a range of functionalities for data processing, model training and versioning, and large-scale deployment. Given our architectural requirements and project needs, AzureML proved to be a highly suitable solution for our work.

AzureML's pipeline designer functionality is a low-code/no-code solution for developing machine learning pipelines both for model training and prediction. The visual components enable the configuration of the pipelines, which can be performed with limited knowledge of the underlying transformations and ML algorithms. Also, its pre-built infrastructure with curated environments allows us to run these pipelines with minimal maintenance overhead, while its versatile suite of features enabled us to develop the necessary modules for our project. The visual pipeline designer and the managed infrastructure with curated environments in the AzureML platform provide an innovative and efficient solution for organizations seeking to streamline their AI workflow and improve the efficiency of their operations.

Microsoft AzureML provides another crucial interface for managing the various datasets utilized by clients, through the implementation of Azure Datasets. This functionality allows for the loading of datasets in tabular format from diverse sources, including both internal and external data repositories. Moreover, an Azure Dataset can be versioned, and its creation can trigger the execution of associated pipelines. To illustrate the practical application of these concepts, Figure 2 presents a diagram of a simulated AzureML pipeline. While we are unable to provide detailed insights into the proprietary work conducted with external collaborators, we can attest that the principles guiding the construction of this pipeline align with the ones from the original experiments.

The demo pipeline depicted in Figure 2 begins with an *Azure Dataset* that serves as the primary input for subsequent data processing steps. Specifically, the pipeline executes a sequence of *data processing modules*, including column selection, missing data removal, and data splitting. The *data splitting module* divides the input dataset into separate training and testing sets, which are subsequently fed into a training module for model development. Each module contains customizable properties that can be adjusted according to specific user needs. For instance, the percentage of data allocated to the training and testing sets in the data splitting module can be configured as needed.

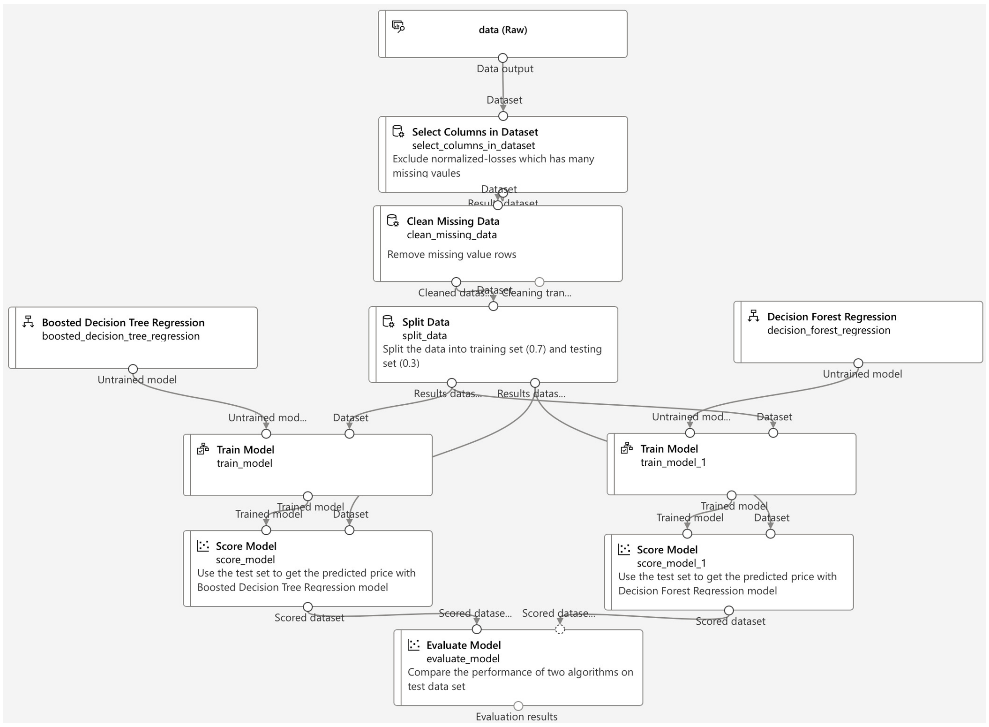
data (Raw)
Data output
Dataset
Select Columns in Dataset
select_columns_in_dataset
Exclude normalized-losses which has many missing vaules
Dataset
Result dataset
Clean Missing Data
clean_missing_data
Remove missing value rows
Cleaned datas...
Dataset
Cleaning tran...
Boosted Decision Tree Regression
boosted_decision_tree_regression
Untrained model
Split Data
split_data
Split the data into training set (0.7) and testing set (0.3)
Results datas...
Results datas...
Decision Forest Regression
decision_forest_regression
Untrained model
Untrained mod...
Dataset
Train Model
train_model
Untrained mod...
Dataset
Train Model
train_model_1
Trained model
Trained model
Dataset
Score Model
score_model
Use the test set to get the predicted price with Boosted Decision Tree Regression model
Trained model
Trained model
Dataset
Score Model
score_model_1
Use the test set to get the predicted price with Decision Forest Regression model
Scored dataset
Scored datase...
Scored datase...
Scored dataset
Evaluate Model
evaluate_model
Compare the performance of two algorithms on test data set
Evaluation results

FIGURE 2. Example of Microsoft AzureML pipeline implementing the generalized architecture

Following the initial data processing steps, the next group of modules in the pipeline pertains to the training of multiple AI models, specifically regression models from the family of decision trees. Each model is trained using the designated training set via the Train Model module, and subsequently tested using the testing set through the *Score Model module.* At the conclusion of this phase, the *Evaluate Model* module is employed to compare and assess the performance of each trained model, with the ultimate goal of identifying the optimal model. An established threshold can then be used to determine whether the best-performing model meets the requirements of the business case. If the model surpasses the predefined threshold, it may be deployed for use in a production environment. Thus, through the use of these modules, AzureML enables the efficient development, testing, and evaluation of multiple AI models, with the ultimate aim of identifying the best solution for a given task.

The flexibility and extensibility of AzureML is further exemplified through its open library of built-in components, maintained by a vibrant and supportive community. For many use cases, the built-in components satisfy the necessary requirements for developing machine learning models. However, in cases where specific and proprietary steps are required, developers may leverage the Python programming language and the *azureml* library to implement custom-built components. In the context of our collaboration with external clients, we designed and developed a series of neural networks (specifically, LSTM [10] and GRU models [6]) that were customized and optimized to suit the needs of our collaborators. These models were then deployed as custom components within the AzureML pipelines we developed. While the modules and components we implemented are tailored to our specific use case, they still follow the overarching principle of letting the models compete to determine a winner, based on their respective levels of accuracy. By providing the ability to use both pre-defined components and implement custom-built ones, AzureML affords significant gains in both performance and maintainability for AI development, even for non-specialized developers.

4.3. **Automating models training and deployment.** To achieve a robust ecosystem for model training and deployment, it is essential to have a well-structured and automated pipeline that can respond to a series of events, validate models against a predefined threshold, and initialize model deployment as a new version only when the threshold is exceeded. To ensure a smooth and efficient process, the proposed pipeline must adhere to the principles of CI/CD, which provide a set of protocols for training machine learning models in real-time and deploying them with high availability and scalability using the infrastructure provided by AzureML. By following the principles of CI/CD, the pipeline can be structured to streamline the model development process, reduce overhead costs, and maximize the return on investment for the AI development process. In general, the CI / CD principles are essential to ensure that the model training and deployment pipeline is robust and scalable, facilitating the rapid deployment of models with minimal interruption to the overall workflow.

Figure 3 depicts the proposed automated mechanism of model training using CI/CD in the Microsoft AzureML ecosystem. The CI/CD system is triggered whenever a change is made to the Azure Dataset associated with the pipeline, indicating the availability of a new dataset for training. The system then pulls the latest version of the dataset and initiates the pipeline, as described in the previous section. Notably, the components in the pipeline will be parallelized if they are on the same level in the hierarchy, allowing all model training to

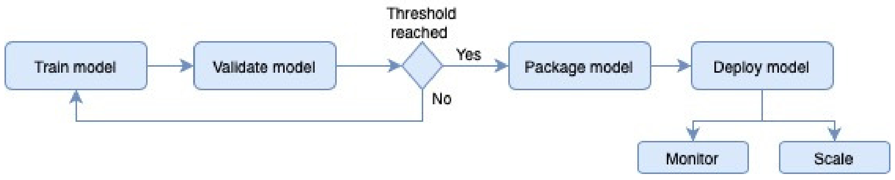

FIGURE 3. CI/CD proposed for the automatized model training pipeline.

be executed simultaneously, thus saving time and resources and, therefore, leading to cost savings.

In the post-training phase of the proposed CI/CD pipeline, the best model's accuracy will be compared to the threshold, which, if not reached, will trigger a notification to the administrators to perform additional fine-tuning on the training set or modify the components (e.g., by adding more cleaning methods). On the other hand, if the threshold is exceeded, the best model will be packaged as a binary model encapsulated into a Dockerized web microservice [17]. This microservice will be deployed on the AzureML infrastructure as a service, which will expose the model through an endpoint. This process ensures that the model is readily available to be consumed by other services.

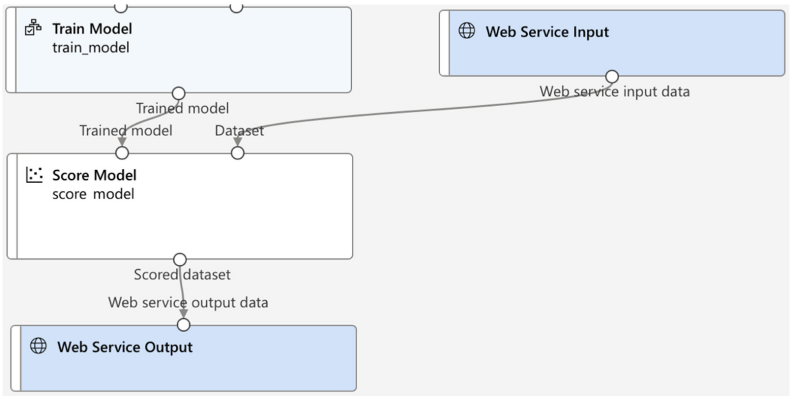

FIGURE 4. AzureML pipeline adjustment to package model as a webservice.

In the context of AzureML, the mechanism of packaging a trained model can be achieved by using two key components, namely the *Web Service Input* module and the *Web Service Output* module, as depicted in Figure 4. These components enable AzureML to recognize that the output of the Train Model section is intended to be encapsulated in a web service. The *Web Service*

Input component is responsible for receiving a set of input parameters, which are then passed on to the component responsible for handling the input and feeding it to the model for inference. The model output is then passed to the *Web Service Output* component, which in turn returns the result to the service that initiated the request.

Upon the completion of the CI/CD pipeline, the resulting web service can be located in the AzureML Endpoints section, which provides all the necessary details to interact with the model. In addition to supporting the deployment of microservices, AzureML incorporates a range of features that are particularly useful during the post-deployment phase, ensuring the high availability and performance of the model. We consider two of these built-in features to be extremely helpful: the fault tolerance - AzureML guarantees that at least one instance of the web service will be operational at all times - and the automated scaling - in situations where there is a high demand, the web service can be scaled to multiple instances managed through a load-balancer.

The underlying infrastructure used for model training, validation, and deployment can be configured to operate on either a CPU or a GPU. Typically, GPUs are essential during the training phase. While GPU virtual machines are considerably more expensive than those equipped with a CPU, weighing the time required for model training against the hourly cost may result in a cost-effective solution involving a GPU instance.

The proposed solution of using an AzureML pipeline for model training and deployment is an automated mechanism that starts with the upload of a new dataset version to the associated Azure Dataset and continues until the last step of model deployment. This approach provides the benefits of an autonomous and intuitive maintenance process, as every step of the pipeline is executed automatically based on the defined logic. The pipeline and its deployment mechanism are suitable for web applications used in live environments, fulfilling requirements such as availability, ease of maintenance and debugging, and fast execution. The monitoring system also enables quick identification and resolution of any issues. Thanks to all of these, the proposed solution offers a comprehensive and reliable solution for automating the model training and deployment process.

4.4. **Consuming trained models.** This section describes the deployment of a model in AzureML as a microservice and the standardization of web services or APIs using Swagger [2]. After a model is deployed, it can be accessed through HTTP requests, and all relevant information regarding the web service, such as the REST endpoint, authentication type, and monitoring logs, can be found in the AzureML Endpoints section. Additionally, the web service created via AzureML Pipelines includes its own documentation that provides details on existing web service endpoints, their expected input structure, and the expected output structure. Swagger is a popular standard for normalizing web services or APIs, which is implemented in AzureML to enable language-agnostic solutions. Therefore, any programming language that has an HTTP framework can make requests to the model and expect output following the structure described by its Swagger.

5. Conclusions

This paper provides an overview of the latest advancements in automating the training of AI models and presents a scalable CI/CD architecture for industrial forecasting utilizing AzureML. Our aim is to develop a user-friendly solution that is easily maintained by individuals with minimal experience in AI. To this end, we conducted a practical experiment utilizing AzureML pipelines with both pre-defined and custom modules, demonstrating its readiness for integration into any production application. While presenting a quantitative validation of our proposed solution may not be feasible, we deem the successful integration of the architecture into the products of our collaborators, along with their teams utilizing and maintaining the proposed automated pipelines for various training purposes, as a valuable qualitative validation. This practical implementation demonstrates the effectiveness and adaptability of our approach, indicating its potential to address various training needs.

Moving forward, we plan to enhance our solution by creating a separate pipeline architecture dedicated to deploying external custom modules. This will allow contractors with AI expertise to easily maintain and improve pipelines. When a new module is created, it will be automatically deployed via the dedicated CI/CD pipeline for custom modules and added to the list of pipelines for training models enabling the module for further use.

6. Acknowledgement

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

[1] Gartner. `https://www.gartner.com/en`. Accessed: June 16, 2023.
[2] Swagger: The world's most popular framework for apis. `https://swagger.io`, 2022. Accessed: Feb. 7, 2022.

[3] Algorithmia. 2021 state pf enterprise ml. https://info.algorithmia.com/hubfs/2020/Reports/2021-Trends-in-ML/Algorithmia_2021_enterprise_ML_trends.pdf. Accessed: June 9, 2023.

[4] Ashmore, R., Calinescu, R., and Paterson, C. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *ACM Computing Surveys (CSUR) 54*, 5 (2021), 1–39.

[5] Boer, A., Koolen, M., van den Berg, J., and van der Werf, J. Continuous delivery pipelines: Best practices in safe. In *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)* (2018), IEEE, pp. 217–224.

[6] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[7] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Feretzakis, G., Sakagianni, A., Loupelis, E., Kalles, D., Skarmoutsou, N., Martsoukou, M., Christopoulos, C., Lada, M., Petropoulou, S., Velentza, A., et al. Machine learning for antibiotic resistance prediction: A prototype using off-the-shelf techniques and entry-level data to guide empiric antimicrobial therapy. *Healthcare informatics research 27*, 3 (2021), 214–221.

[9] Fischer, D., Jung, M., Kasparick, M., and Momm, C. Continuous integration and deployment for software of things using jenkins and docker: A case study. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)* (2019), IEEE, pp. 9–16.

[10] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[11] Huang, W. Best practices for continuous integration and delivery. In *2016 11th International Conference on Computer Science & Education (ICCSE)* (2016), IEEE, pp. 422–426.

[12] Hutter, F., Kotthoff, L., and Vanschoren, J. Automated machine learning: Methods, systems, challenges. *arXiv preprint arXiv:1908.02259* (2019).

[13] Kim, M., Zimmermann, T., DeLine, R., and Begel, A. Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering 44*, 11 (2017), 1024–1038.

[14] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Automating neural architecture search using bayesian optimization and hyperband. In *International Conference on Learning Representations* (2018).

[15] Li, W., Chen, J., and Huang, W. A survey on continuous integration, delivery and deployment tools. *Journal of Systems and Software 147* (2018), 1–15.

[16] Martin, R. C. The single responsibility principle. *The principles, patterns, and practices of Agile Software Development* (2002), 149–154.

[17] Merkel, D. Docker: lightweight linux containers for consistent development and deployment. *Linux journal 2014*, 239 (2014), 2.

[18] Microsoft. Azure machine learning. https://azure.microsoft.com/en-us/services/machine-learning/, 2021. Accessed: February 7, 2023.

[19] Microsoft. Microsoft customer stories. *Microsoft Azure Blog* (January 2022).

[20] Paleyes, A., Urma, R.-G., and Lawrence, N. D. Challenges in deploying machine learning: a survey of case studies. *ACM Computing Surveys 55*, 6 (2022), 1–29.

[21] Pliuhin, V., Pan, M., Yesina, V., and Sukhonos, M. Using azure maching learning cloud technology for electric machines optimization. In *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)* (2018), IEEE, pp. 55–58.

[22] Rajagopal, S., Hareesha, K. S., and Kundapur, P. P. Performance analysis of binary and multiclass models using azure machine learning. *International Journal of Electrical & Computer Engineering (2088-8708) 10*, 1 (2020).

[23] Rezazadeh, A. A generalized flow for b2b sales predictive modeling: An azure machine-learning approach. *Forecasting 2*, 3 (2020), 267–283.

[24] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. Hidden technical debt in machine learning systems. *Advances in neural information processing systems 28* (2015).

[25] Sharir, O., Peleg, B., and Shoham, Y. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900* (2020).

[26] Tan, M., and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning* (2020), PMLR, pp. 6105–6114.

[27] Umamahesan, A., and Babu, D. M. I. From zero to ai hero with automated machine learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), pp. 3495–3495.

[28] Wagstaff, K. L., Doran, G., Davies, A., Anwar, S., Chakraborty, S., Cameron, M., Daubar, I., and Phillips, C. Enabling onboard detection of events of scientific interest for the europa clipper spacecraft. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (2019), pp. 2191–2201.

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

Email address: bogdan.mursa@ubbcluj.ro

Email address: matyas.kuti @ubbcluj.ro

Email address: cristiana.moroz@ubbcluj.ro

Email address: florentin.bota@ubbcluj.ro