EAI Endorsed Transactions
on Smart Cities

Research Article EAI.EU

MLOPS and Microservices Frameworks in the Perspective of Smart Cities

I. B. Urias[1,*] and R. Rossi[2]

[1,2] University of São Paulo/Continuing Engineering Education Program (PECE), São Paulo, Brazil

Abstract

Information Technology involves solutions for many kinds of industries and organizations, offering conditions for solving problems of different types and complexities. Artificial Intelligence, and more specifically applications that considers Machine Learning (ML) and Software Technology are part of these solutions for solving problems, including solutions for solving problems that involve smart cities approach. In order to present frameworks that deal with the operationalization of Machine Learning and Software technology, this article is based on the study and evaluation of frameworks that involve Machine Learning Operations (MLOps) and microservices. Specifically, three frameworks that integrate ML algorithms with microservices are evaluated based on a bibliographical review in scientific journals of relevance to the area. From an exploratory analysis of these frameworks, it was possible to highlight their main objectives, their benefits, and their ability to offer solutions that favor the large-scale use of Machine Learning algorithms in problem solving. The main results are highlighted in the article through a qualitative analysis that considers six evaluation criteria, such as: capacity for sharing resources, scope of use by users, and use in a cloud environment. The results achieved are satisfactory since the work allows, through a qualitative view of the evaluated frameworks, a perspective of how the integration of MLOps and microservices has been carried out, its benefits and possible results achieved through this integration.

Keywords: Machine Learning, Machine Learning Operations, Microservices

Received on 01 August 2023, accepted on 31 October 2023, published on 13 November 2023

Copyright © 2023 I. B. Urias *et al.*, licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

doi: 10.4108/eetsc.3661

1. Introduction

Machine Learning has recently become a field of study and research highlighted by its proposals, either in a theoretical approach or even in a practical approach. Machine Learning (ML) is an area of knowledge focused on technology that aims to develop algorithms to solve machine learning problems. These algorithms represent the simulation of human intelligence, using concepts from neuroscience, probability and statistics, computation, psychology, control theory and philosophy [1].

Some of the main applications highlighted by [2] are: computer vision, semantic analysis, natural language processing, information retrieval, object recognition, object detection and processing, text and document classification, image analysis, diagnosis medical and prediction of network attacks. ML stands out in several areas of knowledge, which favours its use to solve the most diverse types of problems, being studied in a multidisciplinary way. These main applications can be used within the context of Smart Cities, solving problems and achieving goals. In the context of Smart Cities, it is possible to find works related to the topic encompass both a conceptual approach and practical aspects in [3], [4].

ML and Software Engineering complement each other, enabling an exploratory analysis to be conducted on the utilization and implementation of microservices in projects involving Data Engineering, Software Engineering, and ML; analysing and evaluating approaches for the development of automation solutions that utilize ML as services.

Microservices are programs that have unique and independent responsibilities, also known as units of work

*Corresponding author. Email: igor_bernardes_urias@hotmail.com

that include a connection to the external environment [5]. Microservices in general can generate advantages for technology and work teams. These are collaboration and communication actions between teams. Microservices have been observed in companies such as Amazon, Deutsche Telekom, LinkedIn, Netflix, SoundCloud, The Guardian, Uber, Verizon, among others that aim to adopt approaches based on microservices [6].

Applications involving microservices and ML have been explored through frameworks with the aim of making ML algorithms available as services. Frameworks, conceptually, can be considered models of a domain or an important aspect of it that provides a reusable design (modelling) and reusable implementations for the client [7].

Machine Learning Operations (MLOps), through frameworks, stands out as a means of providing ML services using microservices. MLOps uses some practices to operationalize ML algorithms as a service, and can be considered a paradigm for the development of ML algorithms. The main characteristics of this paradigm are based on the conceptualization, implementation, monitoring, deployment, and scalability of ML algorithms [8]. The proposals for integrating ML with microservices are grounded in certain principles that align with the adoption of microservices practices. These principles aim to ensure attributes like scalability and service independence.

Considering these introductory postulates of the article, its objective is highlighted, which refers to the presentation and qualitative evaluation of three frameworks that propose the automation of ML services and that corroborate for an investigative and exploratory analysis of the possibility of integrating MLOps together to microservices for providing ML services. Based on the objectives, it is possible to understand that these frameworks can potentially be used within the context of Smart Cities, considering the benefits of MLOps and Microservices.

The research contributes with the presentation of each of the three frameworks that can be used to automate services from ML algorithms, favouring the response for decision making. The frameworks correspond to significant structures for the application of microservices together with the ML algorithms, allowing an exploratory analysis on the approach referring to MLOps.

The three frameworks, despite having similarities, also there are particularities that differentiate them, for example, the target audience and their different ways of modularizing the components that belong to the ML application. However, it is possible to observe that the three frameworks have a common feature regarding the automation of algorithms with the use of microservices to provide ML services.

Some studies stand out regarding the presentation of studies on MLOps, such as: [9] which conceptualizes the MLOps and presents a proposal for the steps to carry out the operationalization of ML; [10] who also presents a concept about MLOps and contributes with the presentation of benefits and challenges of this paradigm, situating it as an approach that involves ML, DevOps, and Data Engineering.

The introductory elements of the article are emphasized, followed by a succinct overview of the remaining sections, corresponding to: section two, presentation of a literature review, specifically for microservices and machine learning; section three, presentation of the three frameworks that integrate microservice practices with ML; section four, discussion and qualitative results observed from the studies of the three frameworks; and, section five, conclusions and final considerations about the research.

2. Literature Review

This section presents a conceptual review of microservices and their quality attributes; and a review of machine learning concepts and main models; and an overview of Machine Learning Operations (MLOps).

2.1. Microservices

Microservices can be conceptualized as: 1) small applications characterized by having unique responsibilities that can be deployed, scaled, and tested independently [11]; 2) an approach for distributed systems that promote the use of refined services with their own life cycles, which collaborate, being modelled mainly around the business domain [12]; 3) a programming paradigm made up of small services that communicate in applications. Small services are characterized by communication based on light mechanisms that execute their respective processes [13].

The concepts presented have characteristics in common. There is the characteristic of independence between the services, that is, each service can work independently. Another feature that stands out is connectivity, that is, despite the microservices being independent and with unique responsibilities, they are connected to each other, in order to form communications between them and with external environments.

The practice of isolating functions related to the business domain aims to optimize the autonomy and replacement of services. These characteristics are facilitators regarding autonomous management, that is, the issue of governance that is decentralized between services [14]. Characteristics such as isolation and autonomy of microservices make governance decentralized. Decentralization is part of what is also conceptualized as an expected quality attribute for microservices.

Microservices are a new trend in software design and development that are driven by the business domain. However, it should be noted that microservices are not necessarily the right solution for all cases [15]. The use of microservices requires business domain; and knowing what the need or problem is the goal to be met, however,

microservices are a new trend offering useful features in services.

The characteristics of microservices can be studied as quality attributes. Quality attributes will determine the quality of the architecture. It is essential to evaluate architectures based on the qualities it supports in order to guarantee that the built system satisfies the needs of stakeholders [16]. The quality of the microservices architecture can be evaluated through the application of quality attributes.

When performing a literature review on microservices, the main quality attributes are: scalability, independence, maintainability, deployment, diagnostic management, modularization, self-management, performance, reuse, heterogeneous technology, agility, security, balance, organizational alignment, open interface.

As a given architecture for microservices is elaborated, it must satisfy and improve quality attributes. If the quality attributes are acceptable and conform to stakeholder expectations, it means that the microservices architecture is meeting the quality standards.

2.2. Machine Learning

Statistics and Computer Science can be considered two of the main disciplines that make up Machine Learning (ML). ML is an area of study and multidisciplinary research that may belong to applications of different organizations that currently use data intelligence for their goals. A relevant feature that stems from the conceptualization of ML is the goal of simulating human learning activities through the computer. This feature plays a key role in pattern recognition, operational research, data analysis, optimization problem solutions and proposed algorithms for information extraction and knowledge generation.

Machine Learning refers to a set of studies in which the objective is the use of computers in simulations of activities like human learning, as well as the search to study methods of self-improvement of computers to obtain knowledge and skills, being possible identify existing knowledge, improving the objective and performance of ML algorithms [17]. ML aims to perform model training, focused on solving applied activities, constituting a segment of Artificial Intelligence (AI) with relevance in the evolution and improvement of technologies [18]. In general, solutions involving ML algorithms have data divided into subsets, a training subset aims to allow learning, through what is called training; and another test subset, which aims to verify how effective the learning steps were during training.

There are different types of algorithms for performing ML tasks that can be categorized according to their goals and purposes, as well as the way they are operated. These algorithms can sometimes rely on labelled data, where labelled data is data that has some description of what it represents. These representations could be, for example, age, name, weight, and height labels could be database table headers. Thus, given their respective specificities, ML algorithms are grouped as [19]:

(i) **Supervised Learning**: characterized by generating a function that performs the mapping of a set of inputs, later providing a desired output. Algorithms that use the model of supervised learning are generally used in classification problems. This algorithm is conditioned to learn in order to approximate the expected behaviour of a function.

(ii) **Unsupervised Learning:** given a set of inputs where there are no labelled examples, algorithms that use the unsupervised learning model are generally used, responsible for modelling the set of inputs.

(iii) **Semi-supervised Learning**: are characterized by using examples of labelled and unlabelled data in order to generate and provide an appropriate function or classification. Algorithms that use semi-supervised learning models also have some characteristics of supervised learning algorithms.

(iv) **Reinforcement Learning**: algorithms that use reinforcement learning models are characterized by learning from rules. These rules will define how the algorithm should act based on external information, such as observation. In each action, there is an external impact on the environment, where the external environment, on the other hand, provides the answers to the algorithm.

(v) **Transduction**: these models seek to predict new outputs, something very similar to what is done in supervised learning models, however, different from supervised learning. Transduction do not directly and explicitly build a function with an expected behaviour, this model to perform the prediction of the new outputs, uses the inputs that are trained, the training outputs, as well as new inputs.

2.3. Machine Learning Operations (MLOps)

MLOps is related to important areas of study that contribute to its conceptualization. The main areas of study are ML, DevOps and Data Engineering:

(i) **Machine learning**, which refers to computational models based on experience that aim to improve performance or even make accurate predictions. Experience as refers to a set of data that is used for the learning step. Generally, these data are also used for analysis purposes [20].

(ii) **DevOps**: set of collaborative practices that employ multidisciplinary with the objective of automating software development through continuous deliveries based on versioning, guarantee of corrections; making the developed software more reliable [21].

(iii) **Data Engineering**: practices aimed at structuring data aimed at modelling and later use. Data Engineering provides data characterized by having the appropriate quality for the individuals who will use them [22].

The term MLOps appears as a proposal to unite ML and DevOps, however, it is important to highlight that, given that DevOps does not have an explicit definition, it presents many concepts to describe it. DevOps is considered an organizational approach that enables the existence of a collaborative team, promoting empathy between individuals, focusing on development and operationalization [23]. Consequently, MLOps also becomes a term that provides a range of conceptualization possibilities.

MLOps presents some other possible concepts: 1) according to [24] MLOps is analogous to DevOps in some characteristics, such as allowing software professionals to have greater efficiency for the development of AI algorithms through a more participatory collaboration. MLOps also provides greater efficiency for deploying, scaling, monitoring, and training AI algorithms; 2) for [25] MLOps can be considered a DevOps modality directed to the ML domain, that is, directed to the ML algorithms. In this regard, MLOps aims to unite teams that develop ML algorithms with teams focused on operationalization. For MLOps, some main steps are categorized, such as development, deployment, operation, and maintenance. These steps become useful to improve the use of ML algorithms, enhancing their growth in terms of the business domain; 3) for [26] MLOps aims to promote a framework of practices for the development of ML algorithms seeking to optimize the time of the steps contained therein with a reduction in the costs involved. For this, the appropriate tools, steps, and pipelines are used. These features make MLOps very similar to DevOps.

When MLOps practices are exercised by the teams that work in the development of ML algorithms, as well as the operationalization of these ML algorithms, there must be an emphasis on automation and monitoring that constitute the software steps. In this regard, MLOps also provides for greater collaboration and communication between teams to achieve objectives, for example, integration, testing, release, deployment, and infrastructure management.

MLOps emerges as a concept aimed at transforming ML algorithms into practical products or services, like software when DevOps is adopted as a development and operationalization approach.

2.4. Theoretical Use Cases

Theoretical use cases, such as the necessity to estimate the number and precise position of cars in a large car repair workshop, exemplify the development of microservices that utilize image processing techniques, video stream collectors, position classifiers, facility setup estimators, cloud collectors and data storage. Another theoretical use case considers the problem of weather prediction using ML, in this hypothetical use case, the necessary steps for using microservices in conjunction with ML algorithms are described, with a more details, offering a technical approach to the application [27]. This theoretical use case potentially correlates with Smart Cities problems, considering that the weather has an impact, for example, on car traffic in a city.

According to the example of the theoretical use case of weather prediction, the following file and directory structure can be created for building the ML algorithm microservices (Figure 1).

```
Learn_MLOps
├──08_API_Microservices
|    ├── Dockerfile
     ├── app
              └── variables.py
              └── weather_api.py
              └── requirements.txt
              └── artifacts
                        └── model-scaler.pkl
                        └── svc.onnx
```

Figure 1. Example of a repository with files and directories for the API [27]

In this same theoretical use case example, the ML algorithm is available as an API (Figure 2).

```
@app.post('/predict')
def predict_weather(data: WeatherVariables):
    data = data.dict()

    # fetch input data using data varaibles
    temp_c = data['temp_c']
    humidity = data['humidity']
    wind_speed_kmph = data['wind_speed_kmph']
    wind_bearing_degree = data['wind_bearing_degree']
    visibility_km = data['visibility_km']
    pressure_millibars = data['pressure_millibars']
    current_weather_condition = data['current_weather_
condition']
```

Figure 2. Example of endpoint used to the ML algorithm [27]

The presented use cases confirm, in principle, the integrated use of microservices (as an API) with ML algorithms. Moreover, the potential application of these use cases within the Smart Cities context.

3. Frameworks for MLOps and Microservices Technology

This section presents three frameworks that use microservices to provide ML algorithms as services. The characteristics of the architecture and the technical characteristics of the three frameworks are presented, as well as the respective objectives when used to meet a user's need, facilitating the adoption of the use. In this way, the three have the objective of integrating ML algorithms with microservices, each with their respective specificities.

3.1. Machine Learning as a Reusable Microservice (MLRM)

Ref. [28] proposes a framework nominated Machine Learning as a Reusable Microservice (MLRM) where one of the objectives is to encapsulate ML algorithms using

microservices. MLRM allows the separation of ML algorithm from the configurations, as well as allow a simple extension with several other algorithms, with communication performed by service modules using representational state transfer (REST).

MLRM also facilitates data analysis, allowing the reuse and sharing of executables and configurations. The possibility of reusing and sharing executable files and configurations is a feature related to the quality attributes of microservices, and one advantage proposed by this framework in this regard is the ability of reuse and share.

Initially, MLRM is not built to be used in cloud-based environments. One of the goals of the framework refers to the use of customization for different software, just by changing the configuration parameters. This customization is conceptually like the reuse in microservices, since the configurations can be applied agnostically to the software that use the services, just by changing the configurations.

The modules of MLRM framework are highlighted and described as follows:

(i) **ML Service**: module with the implementation of ML algorithms. ML microservices in executable files can be used by multiple microservices and customized via configuration.
(ii) **Configuration set**: module that contains the configurations that customize the microservices. Software professionals can customize the algorithm just by changing the configuration parameters.
(iii) **Set of microservices**: represent the customizable microservices through configurations. Having the configurations available, the microservices can easily invoke the ML services from an existing configuration, making available and sharing the configurations among the microservices.
(iv) **Training Service**: based on the service-oriented approach, there is a dedicated service for training the ML algorithms offered by the ML service.
(v) **Training Data**: The training data that is delivered to the Training Service.

For MLRM, the Training Service (Module 4) implements the functions to train the algorithms offered by the ML Service (Module 1). The training data (Module 5) is sent to the Training Service (Module 4) from the microservices (Module 5). Next, the Training Service (Module 4) sends the trained data to the ML Service (Module 1). This evaluates the output of the ML microservice and updates the configuration parameters in ML Settings (Module 2).

MLRM presents a way to build encapsulated ML microservices that offer configuration flexibility through parameterizations. According to [28], when providing ML microservices outside the cloud, a great advantage, in addition to the requirements of reducing latency and connectivity and greater bandwidth, is data privacy, since cloud-based approaches require data to be uploaded to cloud providers, which may conflict with privacy requirements.

3.2. Minerva

Minerva is another proposed framework proposed by [29] with the aim of integrating ML algorithms with microservices. It is a specific framework for companies that use SaaS.

Ref. [29] points out that for the development of ML algorithms there are some programming languages that can be used, for example, R language and Python. These languages use several libraries characterized by being open source. In software development, ML algorithms developed are different when compared to traditional software in SaaS environments.

Minerva framework aims to propose the implementation of microservices considering ML algorithms in a SaaS environment. This proposal is used in corporate domains and some technical requirements become relevant for the implementation of ML algorithms in SaaS environments:

- There is a need to reuse subsystems that use ML algorithms in order to serve a variety of possible services;
- There is a need for data governance, emphasizing that the governance of these data is decentralized. Also, the existence of a preprocessor to contribute with the engineering of features that are necessary for ML algorithms;
- There is a need to guarantee some characteristics, such as scalability. This scalability must be horizontal, that is, with the ability to add more machines. Vertical scalability, on the other hand, aims to add components such as processors and memories;
- There is a need to ensure real-time or online performance for forecasts. This performance provides serving the results in the systems;
- There is a need for training to be carried out offline, and it is also necessary for these training to be in batches;
- There is a need to protect and exchange data, these data being generally confidential. Data exchange takes place between a resource processing system and a system in charge of processing ML algorithms;
- There is a need to build microservices to support ML algorithms by configuring different libraries. This construction of microservices with a variety of different libraries must be technology independent of the systems, where these systems are known as legacy systems.

In addition to technical requirements, [29] also highlights some business domain requirements for innovative solutions other than typical ML solutions in a cloud environment:

- There is a need to realize smart service delivery. These intelligent services are provided for a variety of traditional SaaS applications. Such services are made available with the characteristic of being lightweight, reducing the impact on related systems or infrastructures;

- There is a need for ML algorithms to be executed "next to the data," which contradicts the idea of having to "move the data to the algorithm." This condition is evident, since for some states or countries there are legal rules that may eventually make the task of migrating data to an external environment outside the datacentres difficult;
- There is a need for ML solutions classified as "interim." The concept of "interim" solutions arises from the fact that such solutions are compatible with so-called traditional SaaS systems, but which, in turn, are used for a defined period, due to difficulties such as delays or obstacles in the adoption of emerging technologies. or modern, like cloud-computing (cloud computing).

In the framework construction, [29] proposes a set of microservices described as ML-oriented subsystems. This set of microservices responsible for making ML algorithms available as services represents the Minerva framework. This framework integrates the traditional SaaS ecosystem. Regarding the subsystems, it is possible to highlight the user interface (UI), database (DB), the core subsystem, the platform, among other subsystems, where all belong to the datacentre.

Minerva framework has another relevant characteristic: the transactional solution useful for companies that are in the process of migrating from an on-premises environment to a cloud environment.

For Minerva, ML microservices are a docker container, where this container has three layers, as highlighted below:

- **Central layer**: responsible for interaction and communication with other systems, as well as managing processes that are contained in a container. Other features such as concurrency control and security mechanisms and settings also belong to the central layer;
- **Abstraction layer**: the abstraction layer belonging to the Minerva framework is responsible for carrying out the dynamic load of the ML algorithms, as well as dealing with versioning, exception management and call control of functions known as callbacks;
- **Application layer**: the application layer belonging to the Minerva framework aims to deal with abstractions and support the coding of ML algorithms using specific libraries for the context of developing ML algorithms.

Ref. [29] highlights that Minerva interacts with legacy subsystems. Such interaction occurs, for example, with the UI or the Main Subsystem, responsible for making predictions or classifications in ML. In this framework, requests for training can be orchestrated as soon as there is data prepared through the data processing unit (Data Pre-Processing Subsystem).

Ref. [29] describes that the Orchestration Subsystem may belong to the set of other subsystems called inherited subsystems. However, the Orchestration Subsystem can also be used in an external environment, that is, outside the set of subsystems, if there is a need to use it this way. The data used and processed by the ML algorithms are extracted, performing a pre-processing in the unit responsible for data processing.

In conclusion, Minerva is a framework that integrates ML with microservices, dedicated to SaaS environments, with a focus on legacy systems. One of the main goals is to encapsulate the ML algorithms and provide flexibility for using different ML libraries. This flexibility is related to the concept of Bring Your Own Model or Algorithm (BYOMOA). The BYOMOA concept allows, for example, flexibility and ease in choosing the use of libraries that will be used for the ML algorithms. Also noteworthy is the development of ML algorithms, which have an abstract interface to perform predictions and training using microservices.

3.3. Machine Learning in Microservices Architecture (MLMA)

Machine Learning in Microservices Architecture (MLMA) is a framework whose objective is to promote some specific design patterns for building microservices with unique responsibilities, that is, segregated from a monolithic architecture [30].

Specifically, this framework demonstrates two use cases designed for Smart Cities: Tourism Recommendation based on Social Media Photos, which aims to identify the types of environments where the photos were taken and build a preference profile using algorithms in the recommendation system. Predictive Policing, which aims to make spatial predictions related to criminal incidence values for a future time interval [30]. Quantitative runtime data were collected in the Tourism Recommendation based on Social Media Photos use case (Table 1), comparing the execution time in a monolithic architecture and MLMA.

Table 1. Processing times for recommendation application [30]

Step	Monolithic	MLMA
Scene Classifier	29.863 s	9.848 s
Fuzzyfication	0.241 s	0.247 s
Recommendation	0.079 s	0.071 s

MLMA is a generic architectural proposal. This architecture enables the implementation of pipelines for ML algorithms, where the framework has the separation of common steps in order to build more adequate services.

Ref. [30] points out that microservices bring benefits to the maintenance and performance of the framework, where the design makes it possible to work with codes in pipelines of ML algorithms, as well as being an approach that uses microservices. The framework also enjoys other benefits such as, for example, the independence between each of the microservices and, consequently, favouring reuse for a set of different tasks that are present within a specific workflow.

One of the primary advantages or benefits that has been suggested is the capacity to migrate from a monolithic architecture to an architecture based on microservices, enabling the utilization of ML algorithms in a reusable manner across various tasks within the same workflow. [30] detail the components of MLMA as follows:

(i) **Flow Controller Service and Post Processing Service**: generates the information that comes from the result of data classification processing. If the Flow Controller Service is not applied, the steps related to communication can be performed directly between the client and the Data Collection Service.

(ii) **Data Collector Service**: performs the extraction of information, data, and content from a specific source to make available for a later stage of analysis. This service does not perform any type of analysis or application of data analysis techniques. The Service supports both structured and unstructured data;

(iii) **Data Orchestrator Service**: presents intermediary responsibilities between three different services. This service communicates with the Data Collection, Feature Extraction, and Classification and Prediction Services through an orchestration. After obtaining the data, they are sent through a structure that makes the data available for use. Features are extracted and processed in order to be used by other services, with a diversity of structures and categories of possible classifiers. Once the preparation process is complete, the data is sent to the new data structure through a data analysis service and, at the end, the results of the previously performed processing are received.

(iv) **Data Handler Service**: in some circumstances, it is necessary to pre-process the data before extracting the features. This service handles and processes data, where some information is not necessarily extracted from the data. On the other hand, a filter is performed on the data and, therefore, it can generate some complexities for the classification, or the service can perform some modification to adapt the data considering some pattern.

(v) **Features Service**: extracts feature that are used by ML algorithms from the generic architecture. In this service are located the raw or pre-processed data that are later used in classification or prediction. The concept of feature extraction considers the action of obtaining information or performing a data processing step to provide learning in a simplified way through ML algorithms and training techniques. The MLMA framework is characterized by using microservices in its approach and separating the responsibilities of extracting features from classification or prediction.

(vi) **Predict/Classification Service**: responsible for generating the most significant and important information for the user. The result is a classification or prediction, as well as any other possible process that uses other ML algorithms, these ML algorithms being implementable in this service. Unlike other proposals, where the classifier or predictor structures are kept together, the generic architecture has the difference by separating the data and the classifier structure from the classification algorithms, providing reuse, and avoiding duplication or redundancy. The result is the respective classifications or predictions, when using such ML algorithms.

(vii) **Volume**: is where the data is stored. The volume also stores the files of the ML algorithms that can be used. These files are handled by the Classification and Prediction Service. Data can be stored in files or in a database. The generic architecture makes decision-making more flexible regarding the use of files or databases, varying according to the user's needs.

The MLMA framework contains a specific component where ML algorithms and data are stored. In this way, the services of this architecture can use such data and ML algorithms according to the need of the problem. This generic architecture for ML pipelines can be useful for problems involving reuse and for using ML algorithms as services.

4. Results & Discussion

This section presents the results of the analysis of the three frameworks presented in section III. The proposed frameworks are analysed with a focus on the operationalization of ML algorithms. The described frameworks use microservices to build ML algorithms as services, but the MLOps approach is not explicit in all of them. However, MLOps as an organizational approach is important for these frameworks.

MLOps is the main approach for the operationalization of ML algorithms, even if the concept of MLOps is still abstract and does not have enough maturity. The purpose of this analysis is not to delve into technical issues of implementation, integration, or implantation, but to present a qualitative view of the frameworks.

Even though certain concepts remain abstract regarding the integration of microservices with MLOps, in the integration proposal, part of benefits derives from the contribution of MLOps [10].

- Easy implementation of high-precision ML algorithms;
- Reduction in data collection and preparation time;
- Delivering of value to customers;
- ML algorithm deployment on a large scale;
- Efficient management of the full ML lifecycle.

Table 2 presents the three frameworks described in section III that are evaluated, and analysed in more details in this section.

Table 2. Frameworks and its proposals

Framework	Description of Proposals
Machine Learning as a Reusable Microservice (MLRM) [28]	• Separate the ML algorithm implementation from the configurations; • Improve ML-based data analysis and enable reuse and sharing of ML algorithms and configurations; • Encapsulate ML algorithms as REST services with a unified interface that can be used without the resources of the cloud; • Customizing services by-configuration-only useful for beginners or novices in using ML; • Reduce network latency when compared to services offered in cloud environments.
Minerva [29]	• Modularize and deploy microservices in SaaS environments, especially in the corporate domain; • Deploy ML microservices in software; • Accelerate the delivery of ML algorithms in software; • Separating the traditional SaaS application from the ML microservices using REST for communication.
Machine Learning in Microservices Architecture (MLMA) [30]	• Development of a generic architecture for providing ML services; • Migrate monolithic ML architectures to ML microservices with separate responsibilities; • Separate steps of similar processes into small services; • Provide classification and prediction results; • Communicate the different services of the generic ML architecture through REST communication; • Separation of feature processing services in relation to classification and prediction.

The proposal of the Machine Learning as a Reusable Microservice (MLRM) framework, described by [28], presents a relevant characteristic, the construction of the framework without the need for cloud-based resources, that is, the proposal is suitable to be built for example, locally. MLRM not depending on a cloud environment, allows some degrees of customization freedom, something that is not necessarily possible with cloud-native approaches. MLRM also proposes a modularization for better performance in the implementation of ML algorithms as services when compared to other services offered by companies. This performance arises from the fact that, as it is a framework that is independent of a cloud environment, it means a possible reduction in network problems such as latency and connectivity. MLRM is still in the vision of being independent of a cloud environment, favours the issue of data security and privacy, in a way that prevents external information traffic to cloud services. Potentially, some applications for Smart Cities can utilize the features of this framework when there is a need for privacy and high performance, such as in the public healthcare sector, like the allocation of emergency resources, where patient data is required, as well as real-time responses using the public healthcare services infrastructure.

The second framework presented, entitled Minerva and proposed by [29], aims to implement ML microservices in SaaS environments, often these environments are typically legacy, or even called traditional, as described by the author. Minerva is a framework proposed specifically for companies, in this way, it starts from the premise that companies seek, given their business domain and the use of SaaS environments, to offer software as a service, where microservices can be suitable for this objective, given the quality attributes. Minerva has an architecture where the traditional SaaS layer and the microservices layer are defined, the latter being responsible for making ML algorithms available as services. It is a proposal that requires little adaptation between the SaaS environment and the ML microservices, given that communication can be carried out through REST-type communication, as presented in the framework. The features of this framework can be used in the context of Smart Cities when there is a need to perform an intermediate migration from a monolithic architecture to a Cloud-based architecture, being efficient during this transitional phase.

Potentially, Minerva can be useful for companies offering software as a service and looking to avoid a major change in the company's existing environment, avoiding major code changes. Another feature of Minerva is that it can be used in cloud environments. Minerva seeks to meet a need of the software industry and companies that carry out activities in the ML context, which is to adapt a traditional SaaS environment to be integrated with ML microservices modules. Another feature is the presence of logs used to support monitoring and operations, with the sharing of these logs. This becomes interesting in terms of traceability and monitoring of the framework, as well as bringing insights regarding the available logs.

The third framework, entitled Machine Learning in Microservices Architecture (MLMA) and proposed by [30], presents a generic architecture for implementing ML algorithms. Specifically, this architecture details with greater granularity the necessary services as well as the operation of the pipeline present in the framework. MLMA is proposed for classification and prediction problems and offers a generic architecture for ML pipelines in microservices, reinforces the concept that the topology of the architecture is preserved regardless of the classification problem and prediction of interest by the user. MLMA is also characterized, like other services, by modelling its architecture in order to segregate responsibilities. MLMA explicitly separates the feature processing step from the processing of classification and prediction ML algorithms. This conceptually corroborates the idea of microservices applied in the context of ML, because the concept of topology, in the context of the framework, reinforces that there is flexibility to adapt the architecture according to the problem to be solved using ML.

The presented frameworks focus on a common objective, which is to provide the results of ML algorithms as services, that is, the construction of ML services, specifically using microservices to achieve the goal. However, each of the presented frameworks contains their respective specificities. It is possible to consider, for example, the applicability and intended audience. It is argued in favour of this statement, when analysing the Minerva framework, which, unlike other frameworks, is directed to a specific public, where the public is companies that use SaaS environments and aims to integrate these traditional SaaS environments with the microservices of ML.

The analysed frameworks do not necessarily need a cloud environment to be used. The benefits obtained correspond to better data control and greater guarantee of customization with a greater degree of freedom, since they do not depend on an infrastructure and external services offered by a cloud environment. However, not all frameworks are necessarily used exclusively without a cloud environment, such as the Minerva framework, which in its proposal also seeks to offer connectivity to cloud environments. Microservices are suitable for environments that use a cloud-native approach, so their use in such an environment brings benefits such as, for example, better management and orchestration of services, scalability and elasticity, automation of deployments and deliveries. It can become very difficult to manage microservices outside a cloud environment, since potentially all issues of support, maintenance and management of the infrastructure are, for example, the responsibility of the company, and the cloud environment could contribute to these issues of infrastructure, architecture and platforms used, taking responsibility for them.

Other beneficial points can be found in the frameworks: processing a large volume of data. Generally, it is more advantageous to process a large volume of data without using a cloud environment, as it ensures better performance in terms of network latency and data traffic. This local processing is suitable for a large volume of data, avoiding network traffic when compared to a cloud environment. However, this same benefit can generate some challenges, such as, how scalable is the local infrastructure for using the frameworks? That is, what is the machine resource limit that the infrastructure must be able to handle the processing that use large volume of data? This problem is reduced in a cloud environment, given that many services offer scalability and elasticity, that is, they grow or shrink organically according to the need for use. Therefore, they become a favourable point for the use of cloud-integrated frameworks, as well as a possible cost reduction in maintaining the cloud environment when compared to the use of local infrastructures.

Another point to highlight, regarding the Minerva framework, is the abstraction of ML algorithms, known as black boxes. Regarding this abstraction, it is possible to consider how advantageous it is to have the models abstracted for the users of the frameworks. This point, in fact, leads to the question that users of a framework do not necessarily need to have specific knowledge about ML. This, in fact, can become a problem, given that users who are using frameworks with ML algorithms available as services do not necessarily need to have knowledge or experience in ML, being just an operational user of the service. Here, it becomes relevant to question the threshold of knowledge that a user of this type of framework needs to perform their duties as a professional in this area.

According to the analysis carried out on the frameworks, it appears that they seek to automate the ML algorithms and make them available as microservices, either in a local environment or in a cloud environment. The automation of ML algorithms does not necessarily mean using the MLOps approach. This is an important point to be highlighted, given that, in the analysed frameworks, the observed maturity level converges towards the objective of automating ML algorithms and not explicitly applying MLOps practices. This fact makes it possible to conclude that there are still conceptual obstacles to effectively MLOps becoming an effective approach for real and practical use of ML algorithms made available in microservices. However, such frameworks, eventually, may suggest new practical studies that allow the use of MLOps practices, given that they are initial proposals and studies related to the subject can still be investigated.

Table 3 presents a qualitative analysis based on the analysed frameworks. Six evaluative criteria are used in relation to their characteristics. The square (green colour) represents that the framework fully attends the criteria, triangle (yellow colour) represents partial attendship, and the circle (red colour) represents that the framework do not attend the criteria.

Table 3. Qualitative analysis of frameworks

Framework	Problem generalization	Different programming languages	Resource sharing	User coverage	Quantitative results	Suitable for cloud utilization
Machine Learning as a Reusable Microservice	■	●	■	■	■	▲
Minerva	■	■	▲	●	●	▲
Machine Learning in Microservices	▲	▲	▲	■	■	■

- **Problem generalization**: quantity and variety of different ML problems that the framework proposes to be used;
- **Different programming languages**: number of programming languages supported by the framework;
- **Resource sharing**: flexibility that the framework has in providing modules, resources, and components;
- **User coverage**: audience that the framework proposes to serve. More specific or more generic;
- **Quantitative results**: results from using and experimenting with the framework are available;
- **Suitable for cloud utilization**: the framework can be used, in addition to the local environment, in a cloud environment.

For example, the Minerva framework stands out in terms of a specific audience that proposes to provide a degree of sharing and meets part of the evaluated criteria. However, this evaluation does not define which is the best framework, given that the use of the framework depends on the context of use, the need, and the problem to be solved, where a set of factors allows those involved to decide which is the best framework to be used.

Effectively in the short or long term, the analyzed frameworks favor MLOps practices since they correspond to approaches that can guarantee the operationalization of ML algorithms as services, taking advantage of the previously presented benefits.

Considering the characteristics previously presented of the frameworks, potentially these frameworks can be applied to different problems and use cases of Smart Cities, such as, for example: Intelligent traffic management, smart parking, public security services, energy efficiency, smart tourism, public health services and other possibilities.

5. Conclusion

The research aimed to address questions regarding the integration of MLOps with microservices to provide ML services. The main analysis is conducted based on the frameworks proposing this integration.

Regarding the frameworks, it is important to highlight that they do not explicitly address the integration of MLOps with microservices. This statement is mainly corroborated due to two reasons. The first reason is that the frameworks aim to automate ML services using microservices, not to directly integrate with MLOps. The second reason is the fact that MLOps is not explicitly conceptualized, and therefore, there is no conclusive and consensus concept. There is a degree of freedom for different interpretations about MLOps.

The benefits as well as the challenges are rather abstract when it comes to integrating MLOps with microservices. Certain challenges arise from the inherent difficulty in conceptualizing, such as defining the role and responsibilities of a professional tasked with operationalizing an ML algorithm. This can lead to a misconception that the professional's goal is merely the deployment or utilization of ML algorithms without possessing a deep understanding of their workings. Consequently, they become professionals responsible for operationalizing or deploying ML algorithms to provide services and consume results without essential comprehension of the algorithms and the related concepts and technologies.

With the objective of addressing the issues related to the integration of MLOps with microservices, it is concluded that there are several gaps and challenges that need to be resolved, particularly concerning the conceptualization of terms and a practical study of integrating MLOps with microservices. There is potential for advancements in this area, leading to an evolution of frameworks that automate ML algorithms using microservices, making them more sophisticated and encompassing both technical and organizational aspects, ultimately enabling integration with MLOps. The frameworks highlighted in this article could be potential candidates for deployment and integrated use with MLOps, providing an opportunity to aggregate knowledge on the subject, enhance existing frameworks, and develop new proposals adopting MLOps.

It was also presented that there is a potential synergy between the concepts related to Smart Cities context. The use case examples presented also provide inputs to be used in frameworks that integrate microservices with ML algorithms.

The studies on MLOps integration with microservices can be extensive in the future, offering substantial potential to overcome barriers and meet technological needs.

References

[1] El Naqa, I., Murphy, M. J. What is machine learning? In: Machine Learning in Radiation Oncology. Springer International Publishing. 2015. 3-11.

[2] Shinde, P. P., Shah, S. A review of machine learning and deep learning applications. In: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) IEEE. 2018. 1-6. DOI: 10.1109/ICCUBEA.2018.8697857.

[3] Silva, R., Silva, M., Caldas, G., Portela, F., Santos, H. Intelligent Dashboards to Monitor the Occurrences in Smart Cities–A Portuguese Case Study. EAI Endorsed Transactions on Smart Cities. 2022; 6(4):1-8. DOI: 10.4108/eetsc.v6i4.2796

[4] Adamuscin, A., Golej, J., Panik, M. The challenge for the development of Smart City Concept in Bratislava based on examples of smart cities of Vienna and Amsterdam. EAI Endorsed Transactions on Smart Cities. 2016; 1(1):1-13.

[5] Yousif, M., Microservices. IEEE Cloud Computing. 2016; 3(5):4-5. DOI: 10.1109/MCC.2016.101.

[6] Larrucea, X. *et al*. Microservices. IEEE Software. 2018; 35(3), 96-100. DOI: 10.1109/MS.2018.2141030.

[7] Riehle, D. Framework Design: A role modeling approach. 2000. (Doctoral dissertation). ETH Zurich.

[8] Kreuzberger, D., Kuhl, N., Hirschl, S. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. arXiv preprint arXiv:2205.02302, 2022.

[9] Symeonidis, G., *et al*. MLOps-Definitions, Tools, and Challenges. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) IEEE. 2022. 453-460. DOI: 10.1109/CCWC54503.2022.9720902.

[10] Goyal, A. Machine Learning Operations. International Journal of Information Technology Insights & Transformations. 2020; 4(2).

[11] Thones, J. Microservices. IEEE Software. 2015; 32(1): 116-116. DOI: 10.1109/MS.2015.11.

[12] Newman, S. Building Microservices. O'Reilly Media Inc., 2021.

[13] De Lauretis, L. From monolithic architecture to microservices architecture. In: 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE. 2019. 93-96. DOI: 10.1109/ISSREW.2019.00050.

[14] Hassan, S., Bahsoon, R. Microservices and their design trade-offs: A self-adaptive roadmap. In: 2016 IEEE

International Conference on Services Computing (SCC). IEEE. 2016. 813-818. DOI: 10.1109/SCC.2016.113.
[15] Jamshidi, P. *et al*. Microservices: The journey so far and challenges ahead. IEEE Software. 2018; 35(3): 24-35. DOI: 10.1109/MS.2018.2141039.
[16] Bass, L., Clements, P., Kazman, R. Software Architecture in practice. Boston: Addison-Wesley Professional Publishing, 2003.
[17] Wang, H., Ma, C., Zhou, L. A brief review of Machine Learning and its application. In: 2009 International Conference on Information Engineering and Computer science. IEEE. 2009. 1-4. DOI: 10.1109/ICIECS.2009.5362936.
[18] Prudius, A. A., Karpunin, A. A., Vlasov, A. I. Analysis of Machine Learning Methods to improve efficiency of Big Data processing in Industry 4.0. Journal of Physics: Conference Series. 2019; 1333(3):1-6. DOI: 10.1088/1742-6596/1333/3/032065.
[19] Ayodele, T. O. Types of Machine Learning Algorithms. New Advances in Machine Learning. 2010; 3(1):19-48.
[20] Mohri, M., Rostamizadeh, A., Talwalkar, A. Foundations of Machine Learning. MIT Press, 2018.
[21] Leite, L. *et al*. A survey of DevOps concepts and challenges. ACM Computing Surveys (CSUR). 2019; 52(6): 1-35.
[22] Wang, R. Y., Kon, H. B., Madnick, S. E. Data quality requirements analysis and modeling. In: Proceedings of IEEE 9th International Conference on Data Engineering. IEEE. 1993. 670-677. DOI: 10.1109/ICDE.1993.344012.
[23] Dyck, A., Penners, R., Lichter, H. Towards definitions for release engineering and DevOps. In: 2015 IEEE/ACM 3rd International Workshop on Release Engineering. IEEE. 2015. 3-3. DOI: 10.1109/RELENG.2015.10.
[24] Garg, S. *et al*. On continuous integration/continuous delivery for automated deployment of machine learning models using MLOps. In: 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). IEEE. 2021. 25-28. DOI: 10.1109/AIKE52691.2021.00010.
[25] Mei, S. *et al*. Model Provenance Management in MLOps Pipeline. In: 2022 The 8th International Conference on Computing and Data Engineering. 2022. 45-50. DOI: https://doi.org/10.1145/3512850.3512861.
[26] Liu, Y. *et al*. Building A Platform for Machine Learning Operations from OpenSource Frameworks. IFAC-PapersOnLine. 2020; 53(5): 704-709. DOI: https://doi.org/10.1016/j.ifacol.2021.04.161.
[27] Raj, E. Engineering MLOps. Packt Publishing, 2021.
[28] Pahl, M., Loipfinger, M. Machine Learning as a reusable microservice. In: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. IEEE. 2018. 1-7. DOI: 10.1109/NOMS.2018.8406165.
[29] Duvvuri, V. Minerva: A portable Machine Learning Microservice Framework for Traditional Enterprise SaaS applications. arXiv preprint arXiv:2005.00866, 2020.
[30] Ribeiro, J. L. *et al*. A microservice based architecture topology for machine learning deployment. In: 2019 IEEE International Smart Cities Conference (ISC2). IEEE. 2019. 426-431. DOI: 10.1109/ISC246665.2019.9071708.