Machine Learning Operations Landscape: Platforms and Tools

Lisana Berberi[1*], Valentin Kozlov[1], Giang Nguyen[3,5], Judith Sáinz-Pardo Díaz[2], Amanda Calatrava[4], Germán Moltó[4], Viet Tran[3] and Álvaro López García[2]

[1*]Scientific Computing Center (SCC), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany.
[2]Instituto de Física de Cantabria (IFCA), CSIC-UC, Avda. los Castros s/n, Santander, 39005, Cantabria, Spain.
[3]Institute of Informatics, Slovak Academy of Sciences (IISAS), Dúbravská cesta 9, Bratislava, 845 07, Slovakia.
[4]Instituto de Instrumentación para Imagen Molecular (I3M), Centro Mixto CSIC -Universitat Politècnica de València (UPV), Camino de Vera S/N, Valencia, 46022, Spain.
[5]Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava (FIIT STU), Ilkovičova 2, Bratislava, 84216, Slovakia.

*Corresponding author(s). E-mail(s): lisana.berberi@kit.edu;
Contributing authors: valentin.kozlov@kit.edu; giang.nguyen@stuba.sk; sainzpardo@ifca.unican.es; amcaar@i3m.upv.es; gmolto@dsic.upv.es; viet.tran@savba.sk; aloga@ifca.unican.es;

Abstract

As the field of machine learning advances, managing and monitoring intelligent models in production, also known as machine learning operations (MLOps), has become essential. Companies have begun to use artificial intelligence as a tool of change, fostering the need for highly reliable MLOps platforms and frameworks. However, every aspect of the machine learning lifecycle, from workflow orchestration to performance monitoring, presents both challenges and opportunities. This review explores the field of open source MLOps and examines the

platforms that serve as the backbone of machine learning systems. Recognizing how tracking experiments may improve the process of organizing and analyzing the results of machine learning experiments as well as team collaboration and knowledge sharing should be noted. This work also highlights how a robust infrastructure facilitates the seamless deployment of machine learning applications in production environments. In addition, it considers the monitoring of the performance of the machine learning model, particularly the detection of drifts. Drift, which refers to the model's learned concepts or data pattern vary over time, can negatively influence the model inference performance, thereby necessitate the retraining. This review includes an overview of existing frameworks dedicated to detecting change, concept, and data drift in both batch and streaming processes. This exploration will present a complete picture of MLOps introducing the continuously growing artificial intelligence world that defines the future of AI-driven systems.

Keywords: Machine learning operations, MLOps platforms, ML monitoring platforms, Drift detection

1 Introduction

Artificial Intelligence (AI)/ Machine Learning (ML) is progressively being included as a crucial solution in the design of new software systems across different industries. However, these ML-based systems bring new challenges of the software development process as compared to the ones we have been familiar with. For example, there are no model and/or data versions in the manual

ML pipeline, no model lineage available (that means no information about which training data, algorithms, hyperparameters, and so on were used to produce this model as output), no tracking of different experiment-runs (there is no automatic way to log the multiple ML executions to select the best model based on the evaluation metrics). Therefore, manual processes to create and control ML systems can cause additional costs, lack of reproducibility, and future problems in maintaining it (Ruf et al, 2021).

Furthermore, successful development requires collaboration between various specialists with different sets of skills and tools: software developers, data scientists, and ML engineers (Diaz-de Arcaya et al, 2023).

As ML faces these challenges, MLOps come up as a response, as a solution that automates the whole AI/ML infrastructure from development to deployment.

Moreover, it is vital to pay attention to the alteration in the behavior of the model or the data, in order to provide an identification of deviations at the right time(Hu et al, 2020). What is more important is that the deployed ML models, which are basically trained on historical data to predict the future, tend to suffer accuracy issues when the distribution of the real-world data changes (data drift). It may cause biased or inaccurate forecasting. Apart from that, the connection between the input data and the target variables may be turned upside down, which is also called a concept shift. Such drifts are detected by existing advanced drift detection techniques and tools. Once drift is detected, there is the need to retrain the already deployed model on newer data inputs or deploy a completely new model in production. CT (Continuous Training/Testing), as part of the MLOps set of practices, aims to retrain and deploy the model automatically as needed (Google, 2020).

The current landscape is seeing a growing offer and adoption of MLOps platforms, which serve as useful tools to automate and streamline tasks in building ML systems (AI-Infrastructure, 2023). Although these platforms are quite essential for proper coordination of various functions in a company, there is, however, a need to go for a better understanding and grouping of their capabilities.

To advise companies on the choice of appropriate MLOps platforms, the need to gain insight into the capabilities offered by modern popular open-source MLOps tools arises, which is a major foundation to making informed decisions. These include features such as orchestration, distributed training, code management, model development, and others. An additional critical dimension arises: the requirement to follow ML models in production closely with a view to detect and resolve possible drifts at any time. This dual focus, combining MLOps capabilities, as well as drift detection tools, makes it a good solution for companies that want a tailored solution which can be adapted to the current situation and help with maintaining accuracy of the models during unfolding of the real-world events.

The motivation behind this research is driven by two main challenges posed by the evolving landscape of MLOps platforms and the performance monitoring of the deployed ML models.

With a thorough examination of the MLOps platforms, we seek to present a better comprehension of what the platforms can do and to fill the knowledge gaps, if any. Also like that, the emphasis on the finding of the performance drifts of the ML models is one of the ways through which we sharpen the robustness of the deployed models. By carrying out these investigations, our research will be able to provide helpful insights and solutions, especially for the field of AI, with emphasis given to the ML applications and services.

The work presented in this paper makes key contributions to the field of ML operations, in particular to the monitoring of ML models with a focus on drift detection, including:

- A comprehensive landscape analysis assists practitioners and researchers in optimizing their choices for ML model deployment by providing detailed information on the supported features of reviewed solutions;
- Practical information is concisely detailed on the use, availability, and functionality of different MLOps products that make up the state-of-the-art in the research field;
- Providing a possible alternative approach to commercial MLOps platforms by utilizing a combination of open source tools, each supporting one or more features. This approach enables cost-effective automation of MLOps processes without relying on expensive commercial platforms.
- Providing a consolidated view of drift detection frameworks and acting as a valuable resource for diverse stakeholders, ranging from users and practitioners to developers and researchers.

In this context, the remainder of this work is structured as follows. Section 2 presents an overview of ML lifecycle management, highlighting the importance of efficient development in AI/ML applications and services; related work after reviewing the literature on the state-of-the-art development of MLOps, and Section 3 provides a concise overview of the methodology used in this study. Section 4 discusses the crucial role of MLOps in guiding the transition between data science and production systems; hence the article presents the numerous strategies and elaborate tools required in the automation, management, and monitoring of the whole life cycle of ML models. Section 4.1 investigates a notable list of open-source MLOps platforms, evaluating their support levels for key features such as orchestration, distributed training, code management, model development, and more. Section 4.2 describes in more detail each of the reviewed platforms/tools. Section 4.3 concludes by highlighting the accelerating growth of MLOps platforms within the wider AI landscape. Despite the common goal of supporting the ML lifecycle, these platforms vary in their objectives, creating space for diverse approaches. Section 5 explores the monitoring performance of ML models, specifically drift detection, in the context of

MLOps. Section 5.1 provides a comparative analysis of well-known drift detection frameworks based on the algorithms they support and their applicability to different data scenarios. The key findings of this comparison are summarized in Section 5.2. Finally, Section 6 provides a comprehensive overview of the key takeaways from this entire research work.

2 Context and scope

The increasing complexity of AI and ML applications in various industries requires a structured approach to development and deployment. The traditional ML lifecycle is an iterative process that includes tasks like: *create/update model* (e.g. models can be created from scratch using the prepared dataset as input or use an existing model as a pre-trained model), *train/test model* (train model and log various metrics and parameters), *evaluate model* (evaluating model accuracy), *store model* (in a model registry), *deploy model* (deploy and, for example, use it for inference queries via a model registry), and *monitor model performance* (track evaluation metrics derived from the model inference running on new data), as shown in a simplified diagram in Figure 1. We express this process model using the BPMN notation (OMG, 2011) as the actual de facto standard of workflow languages. There are two decision points to check during this sequential flow as designed in the diagram: (1) if the accuracy of the model is not satisfactory, we need to update the model otherwise it is deployed into the production environment; (2) if the performance degrades, we trigger the retraining of the model. To properly manage these tasks as steps, MLOps comes into play. These steps can be completed manually or automatically.

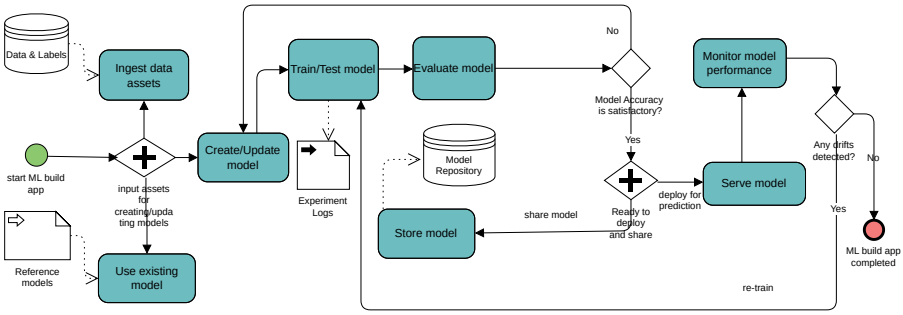

Fig. 1 Business Process Model and Notation (BPMN) diagram of simplified ML life cycle

In this paper, we refer to MLOps as an engineering practice that aims to automate and streamline the ML lifecycle (Kreuzberger et al, 2023). To achieve this, a robust MLOps toolchain is required, offering functionalities like version control, testing frameworks, deployment automation, continuous integration/delivery (CI/CD) and CT.

According to (Google, 2020), software systems, including ML ones, are similar in the continuous integration of source control, unit testing, integration

testing, and continuous delivery of the software module or package (Shahin et al, 2017). However, there are a few notable differences for ML systems (Symeonidis et al, 2022) as presented in Table 1: CI/CD takes into account data and models, and there is a new CT property for continuous training.

Table 1 MLOps with CI/CD/CT

Type	Description
CI (Continuous Integration)	CI is no longer only about testing and validating code and components, but also testing and validating data, data schemas and models.
CD (Continuous Delivery)	CD is no longer about a single software package or a service, but a system (ML training pipeline) that should automatically deploy another service (e.g. model prediction service).
CT (Continuous Training/Testing)	CT is a new property, unique to ML systems, that is concerned with automatically retraining and serving the models.

MLOps is based on the existing DevOps discipline and its key features are: (1) Automation, (2) Collaboration, (3) Integration, and (4) Configuration Management (GreatLearning, 2024), (Shahin et al, 2023).

In addition to MLOps, in recent years there has also been a rise in DataOps, ModelOps, and AIOps. A brief description of these life cycle management practices is given in Table 2.

However, as there is no MLOps standardized approach to manage the ML lifecycle, because each ML project has its own particularities, such as project goals and requirements, the MLOps maturity model (Microsoft, 2022), (Google, 2020) has been recently introduced as a metric to indicate to what extent these steps are automated. This maturity model has been proposed by commercial vendors such as Google and Microsoft. The main difference between the MLOps maturity models of them lies in the granularity and the specific focus. Microsoft's model offers a broader view, with five levels, from zero (no MLOps) to four (Full MLOps Automated Retraining). It emphasizes the level of automation across the entire ML lifecycle, from development to deployment and monitoring. Each stage highlights the most important functions and potential weaknesses of the ML lifecycle steps (Microsoft, 2022).

Google's model focuses on the level of automation in the training pipeline itself and comprises three stages, from zero (manual process) to two (CI/CD pipeline automation). It emphasizes the evolution from manual training to CT with automated pipelines (i.e., Data pipeline, ML pipeline) and CI/CD integration (Google, 2020).

Therefore, industries involved in AI and ML projects can use this maturity model to clarify what level of automation they need to achieve first and then gradually improve it, rather than being overwhelmed by the demands of a fully mature environment.

Table 2 Lifecycle Management practices

Type	Works	Description
DevOps	(Luz et al, 2019), (Qentelli, 2024)	DevOps combines software development (Dev) and IT operations (Ops) to shorten the development life cycle and provide CI/CD with high software quality assurance.
MLOps	(Kreuzberger et al, 2023), (Google, 2020)	ML Operations applies DevOps principles to developing, deploying, and managing ML models.
ModelOps	(Hummer et al, 2019), (Lefevre et al, 2022)	ModelOps is a subset of MLOps that enables organizations to operationalize ML models.
DataOps	(Munappy et al, 2020), (Garriga et al, 2021)	DataOps is the practice of applying DevOps principles to data engineering and management.
AIOps	(Li et al, 2020), (Notaro et al, 2021)	AI for IT operations use AI and ML techniques to automate IT issues such as detection, diagnosis, and resolution.

In addition, certain open-source MLOps tools or a combination of them, which we have examined in this paper (see Section 4.2), can be used to achieve the different MLOps maturity levels, depending on their individual capabilities.

Commercial MLOps platforms such as Azure ML, AWS SageMaker, and GCP Vertex AI (López García et al, 2020) offer comprehensive ML capabilities, but can have higher ongoing costs due to the iterative nature of the ML lifecycle. While powerful, these platforms can be expensive and do not directly solve the core ML problem through a single dashboard. Moreover, some platforms may not offer free licenses suitable for use in embedded systems.

Due to the advantages that open source MLOps products offer, such as: Cost-effectiveness (e.g., they are a good alternative to commercial platforms, suitable for projects with low or limited budgets), accessibility (e.g., they are freely available and often with permissive licensing models) and flexibility (e.g., they offer more control and customization options compared to some commercial platforms) we have focused our work exclusively on open-source products.

Related Work

It is known that many research studies have contributed to the changing environment of MLOps through in-depth reviews and analysis. Insightful works, including Subramanya et al (2022); Mboweni et al (2022); Kreuzberger et al (2023); Diaz-de Arcaya et al (2023), explore theoretical concepts in depth,

while others (Zhou et al, 2020; Barrak et al, 2022) focus on practical applications. Additionally, investigations into MLOps tools presented by Ruf et al (2021); Hewage and Meedeniya (2022); Symeonidis et al (2022) together provide a multifaceted perspective on the MLOps framework.

Subramanya et al (2022) is mostly about reviewing the state-of-the-art in MLOps, introducing DevOps principles related to MLOps, and applying an MLOps framework to a specific time-series forecasting application. However, our research, based on prior work such as Hewage and Meedeniya (2022), goes further, covering a broader spectrum of MLOps tools by including open-source alternatives. Additionally, our investigation verifies a more diverse set of features than previously explored, recognizing the strengths and limits of MLOps tools.

Compared to Symeonidis et al (2022), which classifies tools by their primary use, such as data pre-processing (data versioning and data labeling), operationalization (model development, monitoring) or modeling ones (experiment tracking, hyperparameter tuning); our work challenges this exclusive assignment, recognizing the potential limitations. We emphasize that tools can cross specific categories through library imports or the development of new plugins (in our work, we mark it as partially supported), thus expanding their functionality within the MLOps workflow.

The systematic review conducted by Mboweni et al (2022) provides a scholarly exploration of ML-DevOps (MLOps) from 2015 to 2022. Our study instead explores not only the potential of MLOps, but we also examine the well-known open-source tools and frameworks so that the study is more practical. This approach contributes to a complete knowledge base aimed at researchers, practitioners, and industry experts.

Kreuzberger et al (2023) adopts a mixed method research approach, which covers literature reviews, tool evaluations, and expert interviews. Their work provides a holistic view of the principles, components, roles, architectures, and workflows necessary for MLOps. Our research complements this by offering a more detailed examination of various MLOps tools and their functionalities that can be useful for understanding how this can be applied in practice.

Other relevant works include Zhou et al (2020), analyzing time and resource consumption in the ML pipeline, Moreschini et al (2022) proposing a graphical representation for DevOps in ML-based applications, (Paleyes et al, 2022) exploring challenges in the deployment of ML, Barrak et al (2022) investigating trends in the use of serverless computing for ML deployment, and (Ruf et al, 2021) providing a recipe for tool selection, each contributing valuable insights to the MLOps domain.

In the following section, we give an overview of the methodology we have adopted.

3 Methodology

Our methodology is illustrated in Figure 2. It adopts a mixed approach, combining information from current academic publications (see Section 2), identified MLOps tools (see Section 4.1), reports to better understand technical components, and a comprehensive exploration of drift detection tools and their capabilities (see Section 5.1).

Our comprehensive analysis of the MLOps landscape (see Figure 2) focuses on two main areas: ML lifecycle management and drift detection.

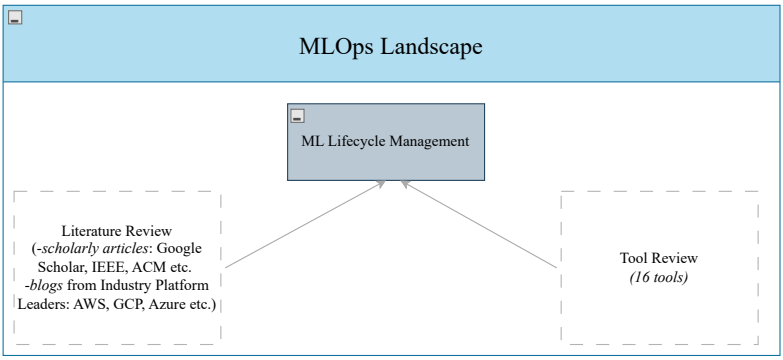

Fig. 2 Overview of the Methodology

To gain valuable insight on emerging trends and best practices, we thoroughly reviewed approximately 20 scholarly articles retrieved from Google Scholar, IEEE Xplore, ACM Library, Springer Link and Elsevier and around 10 industry-leading blogs retrieved from Google Cloud Platform, Amazon Web Services (AWS), Microsoft Azure, etc. for ML lifecycle management, along with a careful examination of 16 associated tools. Meanwhile, our analysis of drift detection includes a comprehensive evaluation of approximately 20 scholarly articles and 6 tools, highlighting the current state and progress within this critical aspect of MLOps.

This finding emphasizes a possible shortfall in the amount of open-source contributions being made towards drift detection tools, bringing attention to a specific area that could greatly benefit from more focus and joint efforts within the MLOps community.

4 MLOps Platforms

MLOps as the bridge between data science and the production system is indeed the backbone of ML systems. On the one hand, MLOps covers the practices and tools required to automate, manage, and monitor the entire lifecycle of ML models, from development to deployment, and beyond. This includes tasks such as data preparation, model training, version control, CI/CD/CT, as well as data curation and model curation in real-time (Kreuzberger et al, 2023). On the other hand, AI infrastructure refers to the underlying technology stack that supports the MLOps process. This includes hardware, software, and services that enable efficient data storage, processing, model development, and model deployment. It involves cloud computing platforms, data pipelines, distributed computing frameworks, and monitoring tools that work together to ensure smooth operation of ML workloads.

MLOps and AI infrastructures work together to bring AI/ML models from experimentation to production. They enable companies to build, deploy, and maintain AI-powered applications with greater efficiency, scalability, and reliability. Workflow and workload management are essential components of MLOps, enabling companies to efficiently develop, deploy, and maintain ML models on scale.

4.1 MLOps Platforms Capabilities Evaluation

In the following, we present a compact list of categories sorted by the importance of *capabilities* (also called *features* interchangeably in this work) in the management of AI infrastructure. Table 3 compares MLOps platforms based on their level of support for various capabilities. It assigns full and partial scores to each platform to evaluate their overall effectiveness.

- *Orchestration (O)* coordinates and manages the individual workflows that resulted from the disassembling of the end-to-end ML pipeline.
- *Distributed Training (DT)* when the workload to train a model is split up and shared among multiple processors, called worker nodes. These worker nodes work in parallel to speed up model training. Typically used to train DNNs models (Azure, 2022).
- *Code Management (CM)* is the process of handling changes to the application source code with control versioning.
- *Model Development (MDV)* involves data acquisition from multiple trusted sources, data processing to build the model, choosing an algorithm to build the model and eventually building the model (Yaram, 2021).
- *Model Testing/Validation (MTV)* is about accurately checking (e.g. unit tests) the expected behavior of the model and providing metrics/plots to summarize the performance on a validation/test dataset.
- *Model Inference (MI)* is the process of inferring results from input (i.e., live and unseen) data using a fully trained model.
- *Model Deployment* (MDP): an enterprise-grade MLOps system should allow organizations to deploy their models and generate consistent API access for

application teams on the other end, regardless of deployment environments and choice of cloud services and providers (DataRobot, 2023).
- *Experiment Tracking and Metadata Store* (ETMS) are two closely related concepts. Experiment tracking refers to keeping track of the details of each experiment, such as hyperparameters, training data, and evaluation metrics. The metadata store refers to keeping track of the context and dependencies of models and pipelines, such as data sources, code versions, and system configurations (DataCamp, 2023).
- *Data Versioning and Management* (DVM) is about tracking datasets by registering changes on a particular dataset. It also improves data compliance by reviewing data modifications in the audit process (lakeFS, 2023).
- *Model Performance Monitoring (MPM)* is the process of observing ML models for possible changes, such as drift detection. The goal is to curate the model to achieve a satisfactory level of performance over time.

In addition to the above one, the following MLOps platform features are also frequently mentioned: *Feature Management* (FM) and *Model Versioning and Management* (MVM). FM is also called a *Feature Store* (FS), and MVM is also called *Model Store* (MS). These features can stay as an MLOps product feature or can be seen as a part of the ETMS feature. The number of desired capabilities is increasing. There is a tendency to break DVM into FM and MVM and a tendency to break ETMS into FS, MS and the rest of metadata. Such similar tendencies can be temporary, which may reflect in platform documentation that evolves over time. The reason is the fast-changing and evolving state of MLOps products in the AI Infrastructure landscape.

In Table 3, we have marked the following *support levels*:

- empty=no support (means that the feature is not supported, not available, or not functional),
- 1x (✓=partially supported, means that some but not all aspects of the feature are supported, or that the feature has limited functionality),
- 2x (✓✓=supported, means that all aspects of the feature are supported, and it is expected to function as intended).

The most common features of the most notable open-source MLOps products presented in Table 3 are as follows.

- Most of the products are open-source under the Apache 2.0 license. The exception is Pachyderm and Seldon core, which have their specific open-source licenses, instead W&B (Wand) has a MIT license.
- All the above-presented MLOps products support Python, TensorFlow (and Keras as a part of TensorFlow 2) for DL, and Scikit-learn for ML. Most of them support the Jupyter Notebook (except for MLFlow and MLeap). PyTorch is also supported by almost all products (except Pachyderm and MLeap) (Oladele, 2024).
- The minority languages supported by some of these products are R, Java, Scala, Go, C++, shell, and Jsonnet.

Table 3 Notable open-source MLOps platforms

Product	**GitHub Stars**	**O** Orchestration	**DT** Distributed Training	**CM** Code Management	**MDV** Model Development	**MTV** Model Testing/-Validation	**MI** Model Inference	**MDP** Model Deployment	**ETMS** Experiment Tracking and Metadata Store	**DVM** Data Versioning and Management	**MPM** Model Performance Monitoring	**Full** Score	**Partial** Score
MLflow	17.1 K			✓			✓✓	✓✓	✓✓			30%	10%
Prefect	14.4 K	✓✓		✓						✓✓		20%	10%
Kubeflow	13.6 K	✓✓	✓✓		✓✓				✓✓			40%	0%
Dagster	10 K	✓✓						✓✓		✓✓		30%	0%
W&B (WB)	8.1 K	✓	✓✓	✓✓	✓✓	✓✓		✓	✓✓	✓✓	✓✓	70%	10%
MetaFlow	7.5 K	✓✓							✓✓	✓	✓	20%	20%
Mage	6.9 K						✓✓	✓	✓	✓✓	✓	20%	30%
Pachyderm	6.1 K	✓✓	✓	✓✓	✓✓	✓✓	✓✓	✓	✓	✓✓		60%	30%
ClearML	5.2 K	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	100%	0%
Flyte	4.7 K	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓	✓✓		80%	10%
Seldon core	4.2 K	✓✓				✓✓	✓✓	✓✓	✓		✓✓	50%	10%
ZenML	3.6 K	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	100%	0%
Polyaxon	3.5 K	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓	✓✓	90%	10%
TFX	2.1 K	✓✓	✓✓	✓✓	✓✓	✓	✓✓	✓✓	✓	✓✓		70%	20%
MLeap	1.5 K	✓✓						✓✓			✓✓	30%	0%
MLRun	1.2 K	✓✓	✓✓	✓	✓✓	✓	✓✓	✓✓	✓✓	✓✓	✓✓	80%	20%

The results of the analysis are visually represented in Fig. 3-4 that highlight the degree of support for different features in each of the 16 products using radar charts. Each figure offers a comprehensive overview of the capabilities and strengths of the respective MLOps platforms in terms of the features considered with more interactive details in (Berberi, 2024).

We assess the features denoted by the abbreviations on a scale within two concentric circles. The inner circle corresponds to a rating of one ✓(0.5), indicating partial support, while the outer circle represents a rating of two ✓✓(1.0), denoting full support. This visual representation allows a clear distinction between features with different levels of support in MLOps products.

Even platforms with a similar scope have different concepts and strategies, making them difficult to compare directly. Table 3 consists of 16 open source MLOps products analyzed with 10 capabilities (O, DT, CM, MDV, MTV, MI, MDP, ETMS, DVM, MPM).

In the following, we summarize the overall product score:

- ClearML, ZenML, Polyaxon and MLRun support partly or fully all categories, scoring the highest degree of compliance.
- A good level of compliance (that is, >70%) is achieved for ML products such as the following: Pachyderm, Flyte, W&B and TFX.
- However, Seldon core, Kubeflow, Metaflow and Mage record approximately a medium score (i.e., between 40%-50%).
- Eventually, low-level compliance (that is, <39%) is observed for MLflow, MLeap, Prefect, and Dagster products.

Not all the MLOps features mentioned above (O, DT, CM, MDV, MTV, MI, MDP, ETMS, DVM, MPM) in Table 3 are fully supported by all presented MLOps products. A graph with the current status of each feature support level is given in Fig. 5. The partially supported state is frequent as well as currently no support or in the development plan.

- The feature O is the most supported among all products;
- The features MPM, MTV and CM are the least supported ones;
- The remaining categories are at a moderate level (>50%) supported by the selected products.

Commercial note: Of these 16 MLOps products, Pachyderm, ClearML, Polyaxon, W&B are more or less commercial with an open source free or community version license. The limitations of their open-source free versions are in the number of users, the scale of resources, and the supported levels of services.

Other products: The list of all existing MLOps platforms is flourishing, so we can talk about a rapid expansion of them. In addition to MLOps platforms, there is a long list of similar products oriented to ML lifecycle management such as Microsoft *FLAML* (Wang et al, 2021), Neural Network Intelligence *NNI* (Microsoft, 2021), Epistasis Lab *TPOT* (Fu et al, 2020) or Orchest (Orchest,

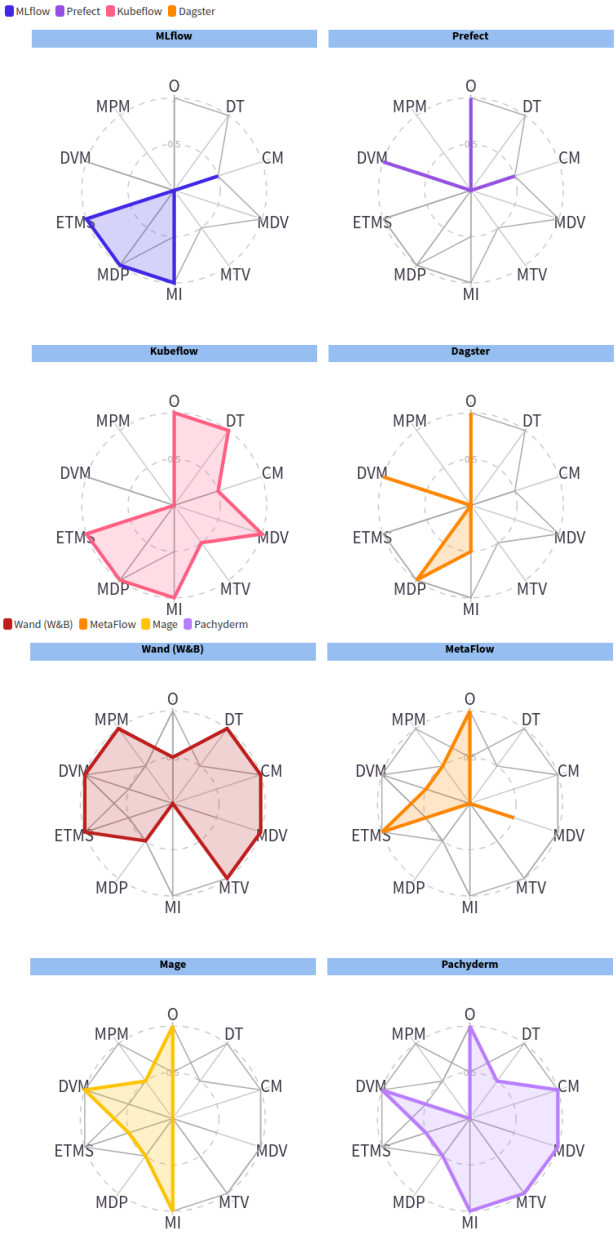

Fig. 3 Calculated degree of the category compliance for MLOps products (a)

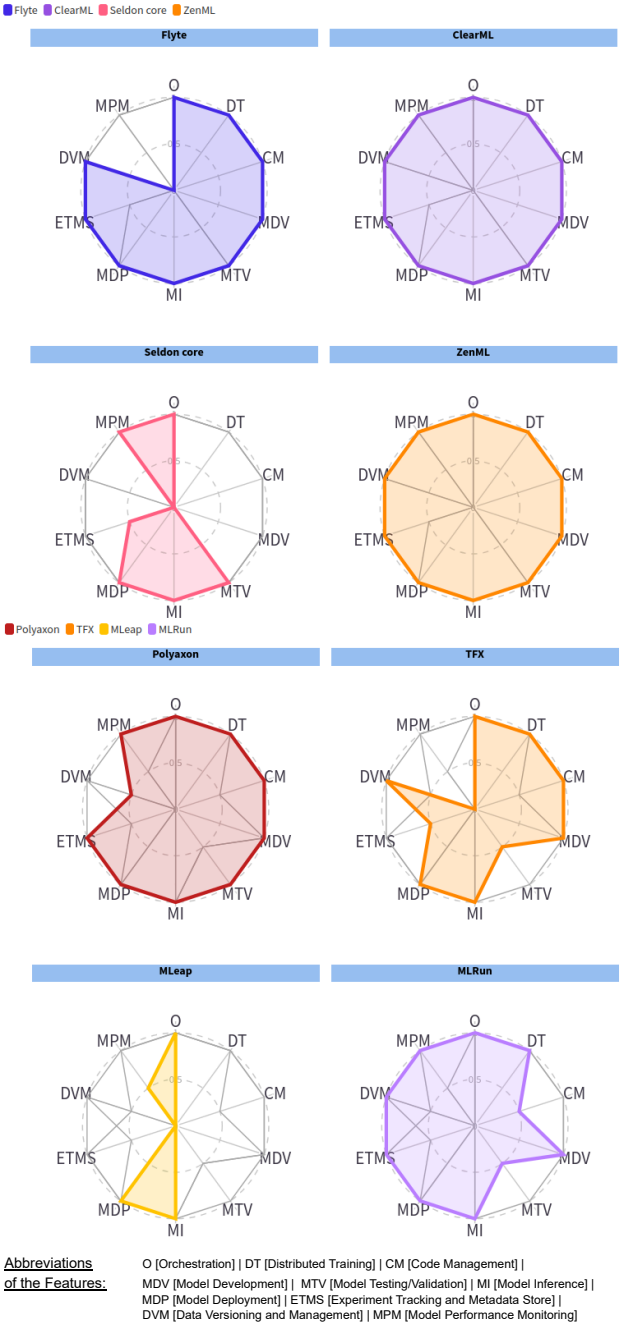

Fig. 4 Calculated degree of the category compliance for MLOps products (b)

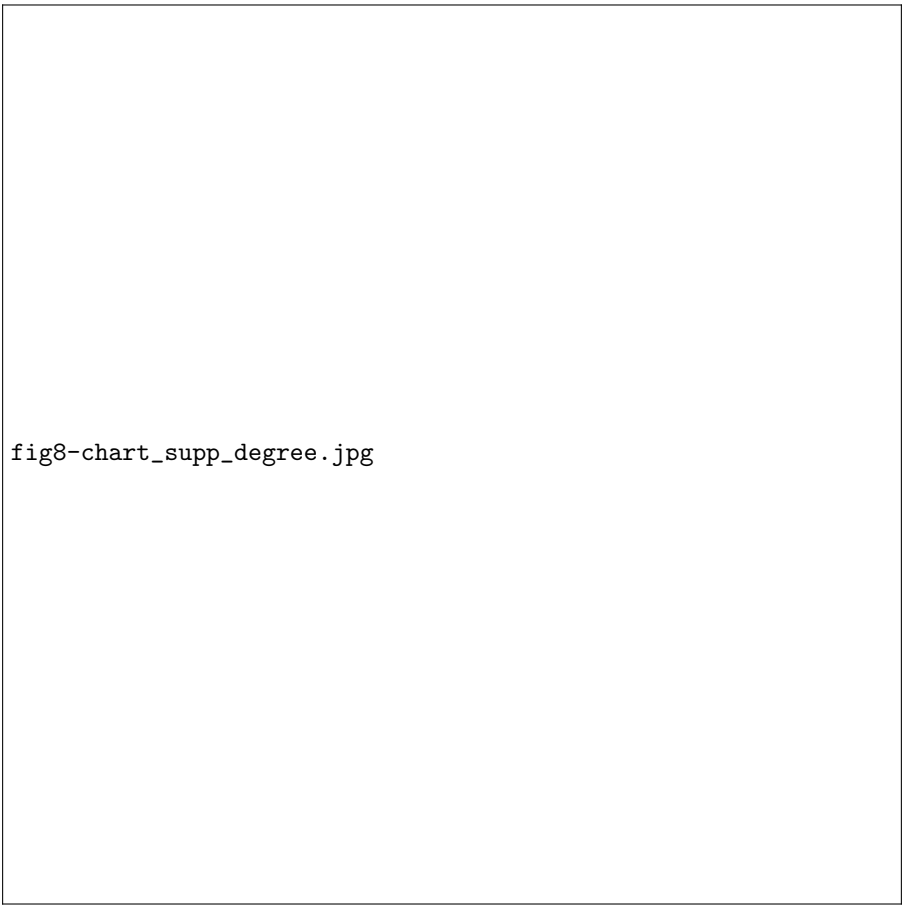

Fig. 5 Calculated percentage score of feature support levels

2021) (currently only beta version). The boundaries between MLOps platforms and ML lifecycle management products are not strict, while, in general, MLOps products are more closely related to ensuring the ML application's effectiveness and scalability in production environments.

In the next parts, we present the content of the in-depth reviewed MLOps products.

4.2 MLOps Platforms Description

4.2.1 MLflow

MLflow (MLflow-Org, Databricks) is a platform to streamline ML development, including tracking experiments, packaging code into reproducible runs, and sharing and deploying models. MLflow offers a set of lightweight APIs that can be used with any existing ML application or library (TensorFlow, PyTorch,

XGBoost, etc.), wherever (e.g., in notebooks, standalone applications, or the cloud). The current components of MLflow are:

- MLflow Tracking: tracking experiments to record and compare parameters and results;
- MLflow Projects: packaging ML code in a reusable, reproducible form in order to share with other data scientists or transfer to production;
- MLflow Models: managing and deploying models from a variety of ML libraries to a variety of model serving and inference platforms;
- MLflow Model Registry: providing a central model store to collaboratively manage the full lifecycle of an MLflow Model, including model versioning, stage transitions, and annotations.

Table 4 MLflow

Motto	An open source platform for the ML lifecycle.
URL	https://mlflow.org/
Open-Source	https://github.com/mlflow/mlflow/ (17.1K GitHub stars)
Peculiarity	MLflow Authentication is experimental and supports only basic HTTP auth

Insights: MLflow can be classified into the group of model metadata storage and management. It has the following main supported features (Table 3): Model Inference (MI) and Experiment Tracking and Metadata Store (ETMS). It is great as a basic ML lifecycle platform to manage the entire ML lifecycle that includes experimentation, reproducibility, deployment, and a central model registry. The tool is library-agnostic, which means it can be used with any ML library and in any programming language.

4.2.2 Prefect

Prefect (Prefect, 2023) is the second-generation dataflow coordination and orchestration platform of the Prefect company. Prefect 2 has been designed from the ground up to handle the dynamic and scalable workloads that modern data stack demands. It is powered by Prefect Orion, a brand new asynchronous rule engine.

Insights: Prefect has two major concepts, Workflow and Task as the next data orchestration tool for Python. These concepts are quite similar to Airflow DAGs and nodes inside a DAG. However, Prefect claims that it is much better than Airflow (Prefect-Docs, 2022) thanks to the result of years of experience working on Airflow and related projects with a lightweight and user-friendly API backed by a powerful set of abstractions that fit most data-related use cases. Prefect treats workflows as standalone objects that can be run at any

Table 5 Prefect

Motto	Prefect is a workflow orchestration tool that empowers developers to build, observe, and react to data pipelines.
URL	https://www.prefect.io/opensource
Open-Source	https://github.com/PrefectHQ/prefect (14.4K GitHub stars)
Peculiarity	Prefect Core provides only basic workflow orchestration features, whereas the Cloud version extends functionality with advanced features like Single Sign-on (SSO) for user management and Role-based Access Controls (RBAC)

time, with any concurrency, for any reason. A schedule is a predefined set of start times with a flow parameter to define the workflow time dependency.

4.2.3 Kubeflow

The Kubeflow project (Kubeflow, 2023) is dedicated to making deployments of ML workflows on Kubernetes simple, portable and scalable. Its goal is not to recreate other services, but to provide a straightforward way to deploy best-of-breed open-source systems for ML to diverse infrastructures.

Table 6 Kubeflow

Motto	The ML toolkit for Kubernetes. Kubeflow is the cloud-native platform for ML operations: pipelines, training and deployment.
URL	https://kubeflow.org/
Open-Source	https://github.com/kubeflow/kubeflow/ (13.6K GitHub stars)
Peculiarity	requires Kubernetes

Insights: Kubeflow has the following supported features (Table 3): Orchestration (O), Distributed Training (DT), Model Development (MDV), and Experiment Tracking and Metadata Store (ETMS). Kubeflow can also be classified into the group of orchestration and workflow pipelines MLOps products. It is the ML toolkit for Kubernetes to maintain ML systems by packaging and managing Docker containers.

4.2.4 Dagster

Dagster (DagsterLabs, 2018) is an orchestrator that is designed to develop and maintain data assets, such as tables, data sets, ML models, and reports. It is built to be used at every stage of the data development lifecycle: local development, unit tests, integration tests, staging environments, all the way up to production. Dagster can be used as the orchestration engine for ML pipelines (feature pipelines, training pipelines, and batch inference pipelines).

Table 7 Dagster

Motto	An orchestration platform for the development, production, and observation of data assets.
URL	https://dagster.io/
Open-Source	https://github.com/dagster-io/dagster (10K GitHub stars)
Peculiarity	*dagster dev* does not include authentication or web security.

Insights: Dagster and Prefect (both launched in 2018) are often compared with each other in the context of Airflow next generation (Navid, 2022). Prefect adheres to a philosophy of negative engineering, built on the assumption that the user knows how to code, and makes it as simple as possible to take that code and build it into a distributed pipeline, backed by its scheduling and orchestration engine. Dagster takes a first-principles approach to data engineering. It is built with the full development lifecycle in mind, from development to deployment to monitoring and observability.

4.2.5 Weights & Biases

Weights & Biases, W&B, or WandB (WeightsAndBiases, 2023) is a comprehensive tool designed for tracking ML experiments and visualization of the model. It offers a variety of features to enhance the development and management of ML projects, including experiment logging, visualization of model metrics, and collaboration tools for teams.

Table 8 W&B (Wand)

Motto	Building the best tools for ML practitioners.
URL	https://wandb.ai/site
Open-Source	https://github.com/wandb/ (8.1K GitHub stars)
Peculiarity	Production-grade features for W&B Server are available for enterprise-tier only (paid). Free teams are available for academics.

Insights: W&B fully manages Experiment Tracking, capturing key parameters and metrics for a centralized view of ML development. It enhances collaboration through seamless sharing, integrates with popular frameworks, optimizes hyperparameters, prioritizes reproducibility, and efficiently automates and scales ML workflows. W&B offers different deployment options to accommodate various needs, including the possibility of setting up a server on your on-premise infrastructure. However, when it comes to the availability of features, it is essential to consider the terms of service and licensing agreements. To set up a W&B Server in a production environment, there are the following three ways:

- *Production Cloud*: Set up a production deployment on a private cloud in just a few steps using Terraform scripts provided by W&B.
- *Dedicated Cloud*: A managed, dedicated deployment on W&B's single-tenant infrastructure in your choice of cloud region.
- *On-Prem/Bare Metal*: W&B supports setting up a production server on most bare metal servers in on-premise data centers by quickly starting by running the wandb server to easily start hosting W&B on a local infrastructure.

4.2.6 Metaflow

Metaflow (Netflix, 2023) is a human-friendly Python/R library that helps scientists and engineers build and manage real-life data science projects. Metaflow was originally developed at Netflix to boost the productivity of data scientists who work on a wide variety of projects, from classical statistics to state-of-the-art DL. It is a Python library that provides a framework to structure Python code as a DAGs describing a number of steps of work (e.g., data loading, model training, and evaluation) that will be executed, as well as dependencies between them. These steps can be run locally or distributed using AWS Batch.

Table 9 Metaflow

Motto	A framework for real-life data science.
URL	https://metaflow.org/
Open-Source	https://github.com/Netflix/metaflow (7.5K GitHub stars)
Peculiarity	Metaflow Sandbox allows you to experiment with modern ML and the infrastructure behind it, without having to install anything locally.

Insights: Netflix released as open-source Metaflow in 2019. Metaflow helps design a workflow, run at scale, and deploy it to production. It versions and tracks experiments and data automatically with the ability to easily inspect results in notebooks. Today, Metaflow powers thousands of ML and data science applications in companies such as Netflix, CNN, SAP, 23andMe, realtor.com, REA, Coveo, and Latana. Commercial support for Metaflow is provided by Outerbounds company.

4.2.7 Mage AI

Mage AI (Mage-AI, 2023) is a versatile platform that facilitates the seamless integration and synchronization of data from various third-party sources. Immediately see results from code output with an interactive notebook UI. Data is a first-class citizen: Each block of code in the pipeline produces data that can be versioned, partitioned, and cataloged for future use. ML pipelines

can be built, and they consist of modular code blocks that are reproducible in any environment.

Table 10 Mage AI

Motto	The modern replacement for Airflow. Build, run and manage data pipelines for integrating and transforming data.
URL	https://mage.ai/
Open-Source	https://github.com/mage-ai/mage-ai (6.9K GitHub stars)
Peculiarity	-can be deployed on AWS, GCP, Azure, or Digital Ocean infrastructures -utilize the tool directly within mage.ai

Insights: The main features supported by Mage are: *Orchestration*: Schedule and manage data pipelines with observability; *Notebook*: Interactive Python, SQL, & R editor for coding data pipelines; *Data integrations*: Synchronize data from third-party sources to your internal destinations; *Streaming pipelines*: Ingest and transform real-time data and data build tool (DBT).

4.2.8 Pachyderm

Pachyderm (Pachyderm, 2023) is cost-effective on the scale, enabling data engineering teams to automate complex pipelines with sophisticated data transformations across any type of data. It provides parallelized processing of multi-stage, language-agnostic pipelines with data versioning and data lineage tracking (CI/CD engine for data). Its main features are: 1) Data-driven pipelines automatically trigger based on detecting data changes. 2) Immutable data lineage with data versioning of any data type. 3) Autoscaling and parallel processing built on Kubernetes for resource orchestration. 4) Uses standard object stores for data storage with automatic deduplication. 5) Runs across all major cloud providers and on-premises installations.

Table 11 Pachyderm

Motto	Data-Centric Pipelines and Data Versioning.
URL	https://www.pachyderm.com/
Open-Source	https://github.com/pachyderm/pachyderm (6.2K GitHub stars)
Peculiarity	requires Kubernetes

Insights: Pachyderm combines the data lineage with end-to-end pipelines on Kubernetes. It is available in three versions, Community Edition (open source), Enterprise Edition, and Hub Edition (still a beta version). Pachyderm has moved some components of the Pachyderm Platform to a source-available limited license. Pachyderm serves, above all, for data processing and orchestration. It can be classified into data and pipeline versioning MLOps product group with data versioning principles:

- Repository – a Pachyderm repository is the highest level data object. Typically, each dataset in Pachyderm is its own repository.
- Commit – an immutable snapshot of a repository at a particular point in time.
- Branch – an alias to a specific commit, or a pointer, that automatically moves as new data is submitted.
- File – files and directories are actual data in your repository. Pachyderm supports any type, size, and number of files.
- Provenance - to track the dependencies and relationships among datasets.

4.2.9 ClearML

ClearML (ClearML, AllegroAI) declares to turn an ML experiment into MLOps with only two lines of code, and it can easily develop, orchestrate, and automate ML workflows at scale. The hosted ClearML version comes with free

(up to 3 users), Pro, Scale, and Enterprise price plans. ClearML can be self-hosted, offering a 100% free solution with limits dictated by your own system's storage capacity.

Table 12 ClearML

Motto	ClearML - Auto-Magical CI/CD to streamline ML workflow. Experiment Manager, MLOps and Data-Management.
URL	https://clear.ml/
Open-Source	https://github.com/allegroai/clearml/ (5.2K GitHub stars)
Peculiarity	The hosted by clear.ml version is free for up to 3 users, a self-hosted system is 100% free

Insights: ClearML is a ML/DL development and production suite containing 4 main modules:

- Experiment Manager - experiment tracking, environments and results;
- MLOps - Orchestration, Automation & Pipelines solution for ML/DL jobs (K8s / Cloud / bare-metal);
- Data-Management - Fully differentiable data management & version control solution on top of object-storage like Amazon Simple Storage Service (S3), Google Storage (GS), Azure storage or Network-Attached Storage (NAS);
- Model-Serving - deploy new model endpoints in under 5 minutes, includes optimized GPU serving support backed by NVIDIA-Triton, with out-of-the-box Model Monitoring.

ClearML enables many MLOps features (Table. 3), for example, to track and upload metrics and models, reproduce experiments, create bots that send Slack messages based on experiment behavior, manage data (store, track, and version control), remotely execute experiments on any compute resource available with ClearML Agent, automatically scale cloud instances according to resource needs with ClearML's GPU Compute, AWS Autoscaler, and GCP Autoscaler GUI applications, run hyperparameter optimization and build pipelines from code.

4.2.10 Flyte

Flyte (Flyte, 2023) is an open-source Kubernetes-native workflow automation platform for complex mission-critical data and ML processes at scale. It has been tested on Lyft, Spotify, Freenome, and others. Flyte's main features are as follows: Kubernetes-native workflow automation platform Ergonomic SDKs in Python, Java, and Scala to develop Flyte workflows versioned, auditable, and reproducible pipelines that are data-aware and strongly typed resource-aware and deployments at scale.

Table 13 Flyte

Motto	Kubernetes-native workflow automation platform for complex, mission-critical data and ML processes at scale.
URL	https://flyte.org/
Open-Source	https://github.com/flyteorg/flyte (4.7K GitHub stars)
Peculiarity	requires Kubernetes

Insights: Flyte is community-driven and community-owned software. Flyte is more than a workflow engine; it uses workflow as a core concept, and task (a single unit of execution) as a top-level concept. Multiple tasks arranged in a data producer-consumer order create a workflow. Flyte is a platform for ML project maintenance released by Lyft with Large-scale project support; Improved reproducibility; Multi-language support such as Python, R, Java, Scala and Julia. Flyte has been tested out by Lyft internally before they released it to the public. It has a proven record of managing more than 7,000 unique workflows that total 100,000 executions every month. Flyte can also be classified into the group of MLOps testing and maintenance products.

4.2.11 Seldon Core

Seldon Core (SeldonCore, 2023) declares itself as the standard open-source platform to rapidly deploy ML models on Kubernetes on a massive scale. It converts ML models (TensorFlow, PyTorch, H2O, etc.) or language wrappers (Python, Java, etc.) into production REST/gRPC (Google Remote Procedure Call) microservices. Seldon handles scaling to thousands of production ML models and provides advanced ML capabilities out of the box including Advanced Metrics, Request Logging, Explainers, Outlier Detectors, A/B Tests, Canaries and more.

Table 14 Seldon Core

Motto	An open-source platform to deploy ML models on Kubernetes at a massive scale
URL	https://www.seldon.io/solutions/open-source-projects/core
Open-Source	https://github.com/SeldonIO/seldon-core (4.2K GitHub stars)
Peculiarity	- requires Kubernetes - it is part of the Seldon commercial platform.

Insights: Seldon Core is the open-source framework for easily and quickly deploying models and experiments at scale with the following summary:

- Simplify model deployment with various options like canary deployment;

- Monitor models in production with the alerting system;
- Use model explainers to understand why certain predictions were made.

Seldon Core is part of the commercial Seldon platform https://www.seldon.io/. Seldon also provides other MLOPs features, for example, drift detection by the Alibi Detect open-source library (Section 5.1.3).

4.2.12 ZenML

ZenML (ZenML, 2023) is an extensible open-source MLOps framework for creating portable, production-ready MLOps pipelines. It is built for Data Scientists, ML Engineers, and MLOps Developers to collaborate as they develop to production. ZenML offers a simple and flexible syntax, is cloud- and tool-agnostic, and has interfaces/abstractions catered toward ML workflows. The aim is to have all selected tools in one place to tailor a workflow that caters to specific needs.

Table 15 ZenML

Motto	Build portable, production-ready MLOps pipelines.
URL	https://docs.zenml.io
Open-Source	https://github.com/zenml-io/zenml (3.6K GitHub stars))
Peculiarity	supports other MLOps tools for implementing MLOps *capabilities* (e.g. Kubeflow for orchestration, MLFlow for the model registry, etc)

Insights: In ZenML, a *stack* represents a set of configurations for the MLOps tools and infrastructure selected by users. For instance:

- Kubeflow for ML workflow orchestration;,
- Amazon S3 bucket as a storage to save ML artifacts;
- Weights & Biases for experiment tracking;
- Seldon or KServe for model deployment on Kubernetes.

Apart from the infrastructure required to run ZenML itself, ZenML also boasts a ton of integrations into popular MLOps tools. The ZenML Stack concept ensures that these tools work nicely together, therefore bringing structure and standardization into the MLOps workflow. However, ZenML assumes that the stack infrastructure for these tools is already provisioned. Currently, there is no native mechanism to distribute large-scale workloads to computing workloads for ZenML users.

4.2.13 Polyaxon

Polyaxon (Polyaxon, 2023) is a platform for building, training, and monitoring large-scale DL applications with the aim of solving the reproducibility,

automation, and scalability of ML applications. Polyaxon is deployed to any data center or cloud provider. It supports all major DL frameworks such as TensorFlow, MXNet, Caffe, Scikit-learn, and Torch. Polyaxon makes it faster, easier, and more efficient to develop DL applications by managing workloads with smart containers and node management (also with GPU).

Table 16 Polyaxon

Motto	A platform for reproducible and scalable ML and DL.
URL	https://polyaxon.com/
Open-Source	https://github.com/polyaxon/polyaxon (3.5K GitHub stars)
Peculiarity	requires Kubernetes

Insights: Polyaxon can also be classified into the group of run orchestration and workflow pipelines MLOps products. The tool can be deployed into any data center or cloud provider and can be hosted and managed by Polyaxon. When it comes to orchestration, Polyaxon can maximize the use of the cluster by scheduling jobs and experiments through its command-line interface (CLI), dashboard, software development kits (SDKs), or REST application programming interface (API). Polyaxon is, in general, a commercial product, but it has an open-source version. It comes in three versions: Community Edition (open source free tool), Hybrid Cloud and Enterprise Edition. It is a well-documented platform, with technical reference docs, getting started guides, learning resources, guides, tutorials, change-logs, etc.

4.2.14 TensorFlow Extended

TensorFlow Extended or TFX (Tensorflow-Extended, Google) is a Google production-scale ML platform based on TensorFlow. It provides a configuration framework to express ML pipelines consisting of TFX components. TFX pipelines can be orchestrated using Apache Airflow and Kubeflow pipelines. TFX components interact with a ML Metadata backend that keeps a record of component runs, input and output artifacts, and runtime configuration. This metadata back-end enables advanced functionality like experiment tracking or warm-starting/resuming ML models from previous runs. TFX libraries include 1) TensorFlow Data Validation, 2) TensorFlow Transform, 3) TensorFlow Model Analysis, 4) TensorFlow Metadata, 5) ML Metadata. TFX requires Apache Beam, an open-source unified model for defining both batch and streaming data-parallel processing pipelines to implement data-parallel pipelines executed on, e.g., Apache Flink, Apache Spark, Google Cloud Dataflow, and others.

Insights: The current version of TFX 1.0.0 provides stable public APIs and artifacts along with nightly built packages. The popularity of TFX is still not at the top, but is growing thanks to the popularity of the TensorFlow family.

Table 17 TensorFlow Extended

Motto	TensorFlow Extended (TFX) is an end-to-end platform to deploy production ML pipelines.
URL	https://www.tensorflow.org/tfx
Open-Source	https://github.com/tensorflow/tfx (2.1K GitHub stars)
Peculiarity	- requires Tensorflow - TFX pipelines can be orchestrated with Airflow and Kubeflow

4.2.15 MLeap

MLeap (MLeap-Docs, CombustML) aims to deploy ML data pipelines and algorithms to avoid being a time-consuming or difficult task. MLeap allows data scientists and engineers to deploy ML pipelines from Spark and Scikit-learn to a portable format and execution engine. Its main goals are as follows. Allow to build data pipelines and train algorithms with Spark and Scikit-Learn Extend Spark/Scikit/TensorFlow by providing ML pipelines serialization/deserialization to/from a common framework (Bundle.ML) Use MLeap Runtime to execute pipeline and algorithm without dependencies on Spark or Scikit (numpy, pandas, etc.).

Table 18 MLeap

Motto	Deploy ML Pipelines to Production.
URL	https://combust.github.io/mleap-docs/
Open-Source	https://github.com/combust/mleap (1.5K GitHub stars)
Peculiarity	no support for PyTorch

Insights: MLeap is a common serialization format and execution engine for ML pipelines. It supports Spark, Scikit-learn, and TensorFlow to train pipelines and export them to an MLeap bundle. For portability, MLeap is built on the JVM and uses only serialization formats that are widely adopted.

4.2.16 MLRun

MLRun (MLRun, Iguazio) is an open MLOps platform to quickly build and manage continuous ML applications throughout their lifecycle. MLRun integrates into a development and CI/CD environment and automates the delivery of production data, ML pipelines, and online applications. It aims to reduce engineering efforts, time to production, and computational resources and break the silos between data, ML, software, and DevOps/MLOps teams and to enable collaboration and fast continuous improvements.

Table 19 MLRun

Motto	The Open Source MLOps Orchestration Framework.
URL	https://www.mlrun.org/
Open-Source	https://github.com/mlrun/mlrun (1.2K GitHub stars)
Peculiarity	recommended self-hosted installation requires Kubernetes with elastic scaling

Insights: In MLRun the assets, metadata, and services (data, functions, jobs, artifacts, models, secrets, etc.) are organized into projects. Projects can be imported/exported as a whole, mapped to git repositories or IDE projects (in PyCharm, VSCode, etc.), which enables versioning, collaboration, and CI/CD. Project access can be restricted to a set of users and roles. MLRun supports the following MLOps tasks: 1) Project management and CI/CD automation, 2) Ingest and process data, 3) Develop and train models, 4) Deploy models and applications, 5) Monitor and alert. MLRun ecosystem supports the latest data stores, development tools, services, and platforms such as:

- Data stores: object (S3, GS, Azure Blob), files, NFS, Pandas/Spark DF, BigQuery, Snowflake, Redis, Iguazio V3IO object/key-value;
- Event sources: HTTP, cron, Kafka, Iguazio V3IO streams;
- Execution frameworks: Nuclio, Spark, Dask, Horovod/MPI, K8s Jobs;
- Dev environments: PyCharm, VSCode, Jupyter, Colab, AzureML, SageMaker, Codespaces;
- ML frameworks: scikit-learn, XGBoost, LGBM, TensorFlow/Keras, PyTorch;
- Platforms: Kubernetes, AWS EKS, Azure AKS, GKE, VMWar, Local (Kubernetes on Docker Desktop), Docker, Linux/KVM, NVIDIA DGX;
- CI/CD: Jenkins, Github Actions, Gitlab CI/CD, KFP.

However, currently the number of GitHub stars for the product is still not high despite promising its MLOps options and features.

4.3 MLOps Landscape Key Finding

The MLOps landscape features an extensive array of products that support various ML lifecycle tasks, with over 250 listed software products (RocketScience, 2022). There are also approaches to classify MLOps products, e.g., into subgroups with narrowed purposes as: a) Model metadata storage and management, b) Data and pipeline versioning, c) Hyperparameter tuning, d) Run orchestration and workflow pipelines, e) Model deployment and serving, f) Production model monitoring

The boundaries between the subgroups of the MLOps product are not strict. One MLOps product can belong to more groups or get lost/evolved with time going and feature requirements evolving. It is difficult to decide on the best criteria for an MLOps platform, and it can be tempting to look

for popularity, such as the score of GitHub stars or the completeness of the platform capabilities (Dawson, 2021) such as those listed in (Table 3).

Newly appearing products with more supporting features may have fewer stars just because they have a shorter life on the market, but this does not mean that they are not attractive or competent enough. There is a risk of preferring popularity or number of desired supporting features (or vice versa) to dominate considerations because: a) although product popularity is a good indicator to filter out low-ranked products, which are new or not of such good quality, popularity is not in line with the number of supported features of MLOps products; b) all products are evolving and adding new features over time. The higher range of features can come at a cost of inflexibility and a higher cost of maintenance. Many use cases do not require a wide range of features. If required features are missing, it is often sufficient to add supplementary tools. In this review, we focus on a subset of MLOps solutions, specifically 16 open-source MLOps platforms. Despite the apparent common goal of supporting the ML lifecycle, these platforms exhibit distinct objectives, allowing for diverse approaches.

The best balance between popularity and the number of supported features belongs to Pachyderm, ClearML, and Polyaxon (Table 3). Polyaxon has the most partially supported features (in progress evolving). However, it is aimed at commercial users with an open-source free version. Similar commercial plans with free open-source versions are with ClearML and Polyxagon, which has the highest number of supported features.

The next interesting ones are Flyte, TFX, and MLRun. Their popularity is still not very high, and the number of features is promising. It is clear that there is no perfect solution 'one size fits all' when it comes to AI/ML and the same is true for MLOps products and a competent MLOps platform should have (at least) the following capabilities:

- *Orchestration*: capability to spin up a service (compute, network, and storage management) based on the model artefact to train them and consequently to deploy them at the end of the process.
- *Model Development*: capability to build model artefacts that contain all the information needed to preprocess the data and generate a result.
- *Model CI/CD/CT* including Model deployment and Model Inference: capability to mark models as ready for staging and production and run them through a CI/CD/CT process.
- *Experiment Tracking and Metadata Store*: capability to track the details of each experiment, such as hyperparameters, evaluation metrics, and the data on which they were trained and tested.
- *Data Versioning and Management*: ability to keep track of how the models, their code, and their data are related, e.g., for drift monitoring.

The purpose of an MLOps platform is to automate tasks involved in building ML-enabled systems and to make it easier to get value from ML pipelines. There are many steps involved in building ML models and getting value from

them, such as exploring and cleaning the data, executing a long-running, iterative, or incremental training process, and deploying and monitoring a model. An MLOps platform can be thought of as a collection of tools to perform the tasks involved in getting value from ML. A good platform is not only a collection of tools, but it has to fit together to provide consistency in how activities are handled and also consistency across the organization as a single platform for different use cases.

5 Monitoring Platforms

As the landscape of AI infrastructure and MLOps platforms continues to evolve, the importance of certain features becomes increasingly evident. One such crucial aspect that is gaining attention is Drift Detection (DD) within the ML lifecycle, positioned under the broader umbrella of Model Performance Monitoring (MPM). In this context, we present the reasons why drift detection has emerged as a pivotal capability and compare the recent frameworks developed through different drift techniques in Section 5.1.

In a production environment, the efficacy of ML models is crucial to long-term success. The accuracy of these models should not decrease over time; instead, they should maintain their integrity of performance. For companies providing AI solutions to end-users, the assurance of continuous accuracy is not just a technical concern, but a highly important aspect of their product offering. Therefore, it becomes essential for these companies to ensure the trust and satisfaction of their customers while keeping their business ahead in the competitive world of AI.

ML models in real-world applications often assume that the same concepts learned during the training phase will still be valid at inference time, which is not always verified. In this sense, the concept of drift arises and can be defined as the process of trying to detect a significant change in the concept previously learned by a model or a change related to the data distributions. Specifically, three variants of this concept are generally analyzed (see (Moreno-Torres et al, 2012) and (Gama et al, 2004)). To define these three approaches, suppose that there are two time points t_i, t_j, with $t_i \neq t_j$, $X \in \mathbb{R}^{nxm}$ which is the feature space (assuming m features) and $y \in \mathbb{R}^n$ are the labels (Céspedes-Sisniega and Álvaro López-García, 2024):

- *Data drift:* in this case we only focus on data, assuming that there are no available y labels, so it can be defined as $\mathbb{P}_{t_i}(X) \neq \mathbb{P}_{t_j}(X)$ (a change in the probability $\mathbb{P}(X)$), being $P_{t_i}(X)$ the probability of X at the time point t_i.
- *Concept drift:* is produced when there is a change in the joint probability $\mathbb{P}(X, y) = \mathbb{P}(y \mid X)\mathbb{P}(X)$, so it can be defined as $\mathbb{P}_{t_i}(X, y) \neq \mathbb{P}_{t_j}(X, y)$.
- *Virtual drift:* is defined as $\mathbb{P}_{t_i}(X) \neq \mathbb{P}_{t_j}(X)$ and $\mathbb{P}_{t_i}(y \mid X) = \mathbb{P}_{t_j}(y \mid X)$, as it refers to the case in which there is a change in $\mathbb{P}(X)$ that does not influence $\mathbb{P}(y \mid X)$.

Drift detection methods can be classified according to the type of drift they can detect and also by the form of detection: supervised, semi-supervised, or unsupervised.

ML models derive their predictive power from historical data, but once deployed in real-world applications, they face challenges in maintaining accuracy and relevance. The occurrence of data drift, where the distribution of the incoming data in production deviates from the training data, can lead to biased predictions. Consider, for instance, an e-commerce recommendation system trained on past user behavior. As user preferences evolve over time or during special events like holiday seasons, the model needs continuous monitoring to detect data drift. By automating retraining processes or incorporating manual interventions, the system ensures that its recommendations remain in line with current user trends, highlighting the importance of ongoing model adaptability in dynamic environments (Aguiar and Cano, 2023),(Ali, 2023). Therefore, monitoring these behavioral shifts in the model becomes crucial to address data drift, allowing for timely identification of deviations and facilitating automated processes for model retraining with updated data or necessary manual interventions. This ensures the model's ongoing relevance in production, delivering fair and unbiased predictions over an extended period.

Once data drift is detected, the next step is mitigation. A common approach is to annotate the new data and retrain the model. You may want to compare the performance of the models before deploying them to production.

A well-structured MLOps pipeline can help automate these tasks, minimizing the manual effort required to retain models, and ensuring faster response times by triggering workflows when data drift is detected. For instance, when encountering anomalies like shifts in image quality due to factors like lens degradation, lighting changes, or sensor malfunctions, the system can prompt appropriate actions, such as maintenance tasks, rather than immediately triggering model retraining. At a minimum, ML monitoring should be configured to send an alert when data drift occurs so that action can be taken (Rabanser et al, 2019), (Elliott, 2023).

5.1 Drift Detection Frameworks

We review the most recent developments in the drift detection area and have identified six noteworthy frameworks: River, Evidently, Alibi Detect, TorchDrift, Frouros, and Menelaus. These frameworks serve as essential tools for monitoring and managing changes in data patterns. Some of the most commonly implemented detectors are the following:

- *Change detectors* monitor single variables in the streaming context and alarm when that variable starts taking on values outside of a predefined range.
- *Concept drift detectors* monitor the performance characteristics of a given model, trying to identify shifts in the joint distribution of the data's feature

values and their labels. Note that change detectors can also be applied in this context.
- *Data drift detectors* monitor the distribution of the features; in that sense, they are model-agnostic. Such changes in distribution might be for single variables or to the joint distribution of all features.
- *Ensembles* are groups of detectors, where each watches the same data, and the drift is determined by combining their output. Menelaus implements a framework for wrapping detectors in this way Nicholl et al (2022).

In addition, as will be remarked when explaining the different frameworks, it is important to note that regarding drift detection the detectors may be applied in two settings, as displayed in Table 27:

- *Streaming*, in which each new observation that arrives is processed separately, as it arrives.
- *Batch*, in which the data have no meaningful ordering with respect to time, and the goal is to compare two data sets as a whole.

In this review, we dive into various drift detection techniques and algorithms employed by the frameworks under study, assessing their efficacy on both streaming and batch data. Table 27 presents an overview of these frameworks, highlighting their ability to detect different types of drift and their compatibility with batch or streaming processing. This exploration aims to provide insight into the evolving landscape of drift detection, helping practitioners choose the most suitable framework for their specific ML lifecycle needs. The list of these drift detection frameworks/libraries is presented in Table 20 and Table 27. Table 20 provides an overview of various frameworks and tools for drift detection, including their ML/DL frameworks, GitHub stars, and licenses. In contrast, the second table, Table 27 presents a detailed breakdown of specific drift types, corresponding detector algorithms, abbreviations, and support for streaming and batch processing within the reviewed frameworks. Together, these tables offer a comprehensive exploration of drift detection tools and their capabilities.

5.1.1 River

River (Montiel et al, 2021) is a Python library for online ML. It aims to be the most user-friendly library to perform ML on streaming data. River is the result of a merger between creme and scikit-multiflow.

Insights: River can be used in different ML problems, such as regression, classification, and unsupervised learning. It is mainly focused on online ML, not only drift detection. It includes a set of tools for online ML such as anomaly detection, time series forecasting, feature extraction and selection, online statistics and metrics, etc. The architecture is designed for flexibility and ease of use to facilitate the deployment of stream learning in quite different fields.

Table 20 Frameworks and tools for drift detection.

ML/DL Framework	GitHub Stars	Licence and notes
River	4.8K	BSD-3-Clause Features for online ML
Evidently	4.6K	Apache 2.0 Evaluate and monitor ML models from validation to production
Alibi Detect	2.1K	Apache 2.0 Algorithms for drift, outlier and adversarial detection by SELDON
TorchDrift	307	Apache 2.0 Drift detection for PyTorch
Frouros	159	BSD-3-Clause Data (streaming and batch) drift detection and concept (streaming) drift detection in ML systems
Menelaus	60	Apache 2.0 Online and batch-based concept and data drift detection

Table 21 River

Motto	Online ML in Python
URL	https://riverml.xyz/
Open-Source	https://github.com/online-ml/river (4.8K GitHub stars)
Peculiarity	River is mainly online ML library that includes drift detection.

5.1.2 Evidently

Evidently is an open-source framework to evaluate, test, and monitor ML models in production with the aim of preventing incidents, building trust, and making continuous improvements. Evidently has three components: (1) Evidently reports, (2) Evidently test suites, and (3) Evidently ML monitoring. Evidently can be used with 100+ metrics to evaluate by the mean of understanding, visualizing and tracking for:

- Data quality: track data quality and integrity.
- Data drift: explore statistical distribution shifts.
- Model performance: evaluate and compare the quality of the ML model.
- NLP and LLM: monitor text-based models.

Insights: Evidently is an open-source Python library under the Apache 2.0 license. The open-source version is designed for individual data scientists

Table 22 Evidently

Motto	Evaluate and monitor ML models from validation to production.
URL	https://www.evidentlyai.com/
Open-Source	https://github.com/evidentlyai/evidently (4.6K GitHub stars)
Peculiarity	Evidently is a part of Evidently Cloud commercial product.

and small ML teams to run a proof-of-concept without additional approvals. Evidently is a part of *Evidently Cloud*, which is a commercial product built upon the functionality of open-source Evidently libraries. It is designed for growing teams and larger enterprises that operate ML on a scale and need advanced analytics, reliability, and support to reliably run ML systems in production.

5.1.3 Alibi Detect

Alibidetect (Van Looveren et al, 2019) is a Python library for outlier, adversarial, and drift detection.

Table 23 Alibi Detect

Motto	Algorithms for outlier, adversarial and drift detection
URL	https://docs.seldon.io/projects/alibi-detect/en/latest/
Open-Source	https://github.com/SeldonIO/alibi-detect (2.1K GitHub stars)
Peculiarity	Alibi Detect is a part of the Seldon commercial platform (https://www.seldon.io/) together with Seldon Core (Section 4.2.11).

Insights: For drift detection, it supports both the Tensorflow and PyTorch backends. In addition, it provides different detectors according to the kind of data analyzed: images, tabular, text, time series, categorical features, etc. Alibi Detect presents several examples of the use of the different methods, which are available in the GitHub repository, together with different datasets, which make it easy for the user and user-friendly. As previously stated, it is not only focused on drift detection, but it also implements tools for outlier detection, such as isolation forests and autoencoders, among others.

5.1.4 TorchDrift

TorchDrift (TorchDrift, 2024a) is a Python library for the detection of data and concepts of drift for PyTorch. TorchDrift is mainly composed of a set of detectors that have 2d-Tensors as expected input. This framework is still under rapid development.

Table 24 TorchDrift

Motto	Drift Detection for your PyTorch Models
URL	https://torchdrift.org/
Open-Source	https://github.com/TorchDrift/TorchDrift (305 GitHub stars)
Peculiarity	TorchDrift is only for PyTorch community.

Insights: It includes a comprehensive set of drift detection tools that makes it useful for both time series and tabular data. It provides many examples in Jupyter Notebooks that make it very easy to start using it (TorchDrift, 2024b). It contains the capability to apply dimensionality reduction using Principal Components Analysis (PCA), which can be useful in multiple scenarios before performing drift detection with statistical tests, as shown in an example of TorchDrift documentation when comparing drift detectors.

5.1.5 Frouros

Frouros (Céspedes-Sisniega and Álvaro López-García, 2024) is an open-source Python library dedicated to concept drift and data drift detection. It can be used for drift detection in ML pipelines with the most widely used methods and algorithms.

Table 25 Frouros

Motto	Frouros is an open-source Python library for drift detection in ML systems
URL	https://frouros.readthedocs.io
Open-Source	https://github.com/IFCA-Advanced-Computing/frouros (159 GitHub stars)
Peculiarity	Frouros lacks a user-friendly interface, such as a dashboard, for clear visualization of drift patterns

Insights: Regarding concept drift, it implements three families of methods for streaming: change detection, statistical process control, and window-based. Concerning data drift, it implements two kinds of method both for streaming and batch: distance-based and statistical tests. It is compatible with any ML framework and easily adaptable to real-world use cases.

5.1.6 Menelaus

Menelaus (Nicholl et al, 2022) is a Python library that employs algorithms designed for drift detection, a ML aspect dedicated to identifying unexpected changes in data. Relationships between variables in a data set are rarely static

and can be influenced by changes in internal and external factors, such as alterations in data collection methods, external protocols, or population demographics. Undetected changes in data and unnoticed model under-performance both pose risks to users. The objective of this package is to facilitate the monitoring of both the data and the performance of the model.

Table 26 Menelaus

Motto	Online and batch-based concept and data drift detection algorithms to monitor and maintain ML performance
URL	https://menelaus.readthedocs.io
Open-Source	https://github.com/mitre/menelaus (60 GitHub stars)
Peculiarity	Menelaus lacks a user-friendly interface, such as a dashboard, for clear visualization of drift patterns.

Insights: It implements the different type of detectors previously exposed: *change detectors*, *concept drift*, *data drift* and *ensemble*. For more details, including references to the original papers, we provide in the column *Abbreviation and References* of Table 27. Furthermore, *Menelaus* presents practical examples of each drift-type technique in their Git repository. Finally, the list of drift detection techniques implemented in Menelaus is available on GitHub (Nicholl et al, 2022).

Other drift detection libraries

Even though most of the methods used for drift detection are not a fairly recent field of study (Gama et al, 2004), this technique applied to ML pipelines is on the rise. This is reflected in the low (but increasing) number of stars that emerging frameworks have on GitHub (see Table 20).

For example, MOA (Bifet et al, 2016) is an open source framework for mining Big Data streams (568 stars on GitHub). It includes a collection of ML algorithms (classification, regression, clustering, outlier detection, concept drift detection, and recommender systems) and tools for evaluation, written in the Java programming language. It is related to WEKA, which is a Java ML software.

Comparison of drift detection solutions

Table 27 compares various drift detection frameworks, presenting information on different aspects of each framework. The frameworks included in the comparison are Alibi-detect, Menelaus, River, Torchdrift, and Frouros. The table provides details on the type of drift detection, the algorithm used, both its abbreviations and its reference sources, and the applicability to streaming or batch data for each framework. Additionally, the Gitsource column includes links to the respective GitHub repositories for further reference.

In Table 27, we use a ✓to indicate the framework support for the algorithm; otherwise, the cell is left empty. This assessment is performed for both batch and streaming data.

Based on the findings presented in Table 27, it is evident that methods such as Kolmogorov-Smirnov Windowing Detection (Raab et al, 2020), Cram'er-von Mises test Method (Cramér, 1928), Maximum Mean Discrepancy Detection (Gretton et al, 2012), Kullback-Leibler divergence detection (Kullback and Leibler, 1951) and PCA-based changed detection (Cramér, 1928) exhibit the highest level of support.

In contrast, when considering tools that provide extensive support for a wide array of methods, Frouros, River stand out. On the other hand, Menelaus and Alibi-detect offer support at a moderate level, while Torchdrift demonstrates the most limited coverage for these functionalities.

In the upcoming section, a succinct summary refines the core findings from the comprehensive review, providing a streamlined overview of the research on drift detection frameworks and tools.

Table 27 Drift Detection Frameworks Comparison

Drift Type	Drift Detector Algorithm	Abbrev. | Ref.	River		Alibi Detect		Torchdrift		Frouros		Menelaus		Evidently	
			Streaming	*Batch*	*Streaming*	*Batch*	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch
Change Detection	Bayesian Online Change Detection	BOCD, (Adams and MacKay, 2007)	✓		✓				✓					
	Cumulative Sum Test	CUSUM, (Page, 1954)	✓						✓		✓		✓	
	Geometric Moving Average Detection	-, (Roberts, 1959)	✓		✓				✓					
	Page-Hinkley	PH, (Page, 1954)	✓						✓		✓		✓	
	ADaptive WINdowing	ADWIN, (Bifet and Gavaldà, 2007)	✓						✓		✓			
	Kolmogorov-Smirnov Windowing Detection	KSWIN, (Raab et al, 2020)	✓		✓		✓		✓					✓
Concept Drift	Drift Detection Method	DDM, (Gama et al, 2004)	✓						✓		✓			
	Early Drift Detection Method	EDDM, (Baena-Garcıa et al, 2006)	✓						✓		✓			
	EWMA Concept Drift Detection Warning	ECDDWT, (Ross et al, 2012)	✓						✓					
	Statistical Test of Equal Proportions to Detect concept drift	STEPD, (Nishida and Yamauchi, 2007)							✓		✓			
	Hoeffding's drift detection Method	HDDM, (Frías-Blanco et al, 2015)	✓						✓					
	Fast Hoeffding drift detection	FDDM, (Pesaranghader and Viktor, 2016)							✓					
	Reactive Drift detection Method	RDDM, (Barros et al, 2017)							✓					
Data Drift	Cramér-von Mises test Method	CVMTest, (Cramér, 1928)		✓		✓				✓		✓		✓
	Hellinger Distance Drift Detection Method	HDDDM, (Hdlinger, 1909)		✓						✓		✓		✓
	Kullback-Leibler divergence Detection	KL, (Kullback and Leibler, 1951)				✓				✓	✓	✓		✓
	PCA-Based Change Detection	PCA-CD, (Cramér, 1928)	✓			✓		✓			✓			✓
	Earth Mover's Distance Detection Method	EMD, (Rubner et al, 2000)								✓				✓
	Maximum Mean Discrepancy Detection	MMD, (Gretton et al, 2012)	✓		✓		✓		✓	✓				✓
	Incremental Kolmogorov-Smirnov	IncrementalKSTest, (dos Reis et al, 2016)	✓		✓				✓					✓
Ensemble	Streaming Ensemble	-, (Maciel et al, 2015)	✓		✓				✓		✓		✓	
	Batch Ensemble	-		✓		✓						✓		✓
		Gitsource	https://github.com/online-ml/river		https://github.com/SeldonIO/alibi-detect		https://github.com/torchdrift/torchdrift		https://github.com/IFCA-Advanced-Computing/frouros		https://github.com/mitre/menelaus		https://github.com/evidentlyai/evidently	

5.2 Drift Detection Key Finding

This part of the article briefly describes the role of drift detection troubleshooting ML models. We explored various frameworks designed to identify three main types of drift: *data* drift, *concept* drift, and *domain* drift. These frameworks offer a diverse set of algorithms for detecting drift in both real-time streaming data and static batch data.

The drift detection concerning ML models investigation was conducted, which showed several important results. To begin with, drifts that may rely on the transformations of the fundamental assumptions are able to almost entirely disrupt the performance of the models. It is essential to diagnose the ways of drift at the early stage in order to overcome this problem and to allow the models to work effectively in the real-world applications.

Secondly, the consideration of different drift detection frameworks sheds light on which tools like Frouros and River are effective in solving the issue. These architectures usually provide a set of various algorithms which can allow for detecting drift in both continuously flowing data streams (streaming) and static datasets (batch). If there is no flexibility, this would be an impossible task to perform. Performance degradation is one of the initial clue signals, which may result in an appropriate investigation being carried out.

Finally, the analysis may indicate a possible trend toward the industry's commonness of drifting detection methods. Certain algorithms like Kolmogorov-Smirnov windowing detection, Cramér-von Mises test method, and maximum mean discrepancy detection received the maximum percentage of support across all the different frameworks. It shows how these systems are taking over the place of traditional methods as standard methods used by the ML group.

6 Conclusion and Future Work

While the MLOps system is going through a radical transformation, an increasing number of products are being launched across the ML lifecycle in various dimensions. Although MLOps platforms limit themselves to supporting ML processes only, they differ in their scope and methodology according to the destructive analysis.

The wide range of MLOps products presents us with the opportunity and dilemma of choosing the ideal software to be used in certain demands.

The criteria for choosing MLOps platforms can be very complex, keeping in mind things like GitHub stars as a measure of popularity or more powerful features that a specific platform has. It is very important to allow for a trade-off between popularity and the number of features supported, and platforms such as Pachyderm, ClearML, and Polyaxon achieve this. While Polyaxon is a platform with many features and provides commercial support, there is also a free and open source version.

In addition to the above features, a good MLOps ecosystem must have capabilities for orchestration, support for model development, robust

CI/CD/CT processes, experiment tracking, and effective data versioning and management.

A properly designed MLOps pipeline can help automate these processes, which in turn can minimize the manual efforts required to sustain the models and ensure faster response times by triggering workflows when the data drift is noticed. In the end, ML monitoring should be configured to start a warning message when data drift is discovered, to prompt an immediate response and appropriate actions.

Although the study offers a comprehensive assessment of 16 free MLOps tools that gives an understanding of some specific capabilities and popularity of them, recognizing some *limitations* is also significant. Primarily, our study is restricted to the free open-source MLOps tools. This is so because it can be considered a limitation and also a strength: in order to support open-source tool development and use is what we want to achieve. Nevertheless, due to the fact that our findings cover only open source tools, enterprises that need proprietary solutions could have to do more research to get insight about the products beyond the suggestions in this review.

Several platforms, such as Polyaxon, ClearML, Weights and Biases (W & B), and Pachyderm, offer commercial plans alongside open-source versions. The influence of these commercial orientations on the overall effectiveness and suitability for diverse use cases needs to be further explored. Second, it is important to note that the MLOps landscape is highly dynamic and products evolve rapidly. Then, new features may be added and existing ones may be modified, impacting the relevance and completeness of our analysis. Finally, the evaluation of MLOps platforms encompasses critical features but may not cover all types of their capabilities. Some platforms might offer unique functionalities that were not captured in our analysis.

When analyzing the different frameworks for drift detection, we found that algorithms such as Kolmogorov-Smirnov window detection, the Cramér-von Mises test method, and Maximum Mean Discrepancy detection receive the highest percentage of support across all different frameworks. This shows that these systems are indeed taking the place of traditional methods as the standard methods of the ML group.

Future work may involve a more tangible connection with end users and companies that have already adopted MLOps platforms. These goals may be achieved through receiving user feedback (via surveys), exploring the real-life implementation of the solution, and including user-based perspective, which can lead to more practical MLOps platform evaluations.

Furthermore, future research can focus on ethical considerations within MLOps platforms as ethical AI becomes vital. This includes evaluating how platforms are facilitating safe AI usage, data privacy, and transparency in model development and deployment.

Thus, the creation of standardized benchmarks and performance metrics for MLOps platforms will make the comparison fair more objectively. In the future, such work should be focused on identifying benchmarks that allow

companies to evaluate and select the platforms based on clear criteria and standards in the industry.

Funding

This work is funded by the European Union through the Horizon Europe AI4EOSC project under grant number 101058593.

Declaration

During the preparation of this work, the authors used the Writefull extension integrated in the Chrome browser with Overleaf to check grammar and improve the English language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

Adams RP, MacKay DJC (2007) Bayesian online changepoint detection. https://doi.org/https://doi.org/10.48550/arXiv.0710.3742

Aguiar GJ, Cano A (2023) Enhancing concept drift detection in drifting and imbalanced data streams through meta-learning. In: 2023 IEEE International Conference on Big Data (BigData), IEEE Computer Society, pp 2648–2657, https://doi.org/10.1109/BigData59044.2023.10386364

AI-Infrastructure (2023) AI Infrastructure Landscape (2023). URL https://ai-infrastructure.org/ai-infrastructure-landscape/, accessed on 30.11.2023

Ali M (2023) Understanding data drift and model drift: Drift detection in python. URL https://www.datacamp.com/tutorial/understanding-data-drift-model-drift, accessed on 08.01.2024

Diaz-de Arcaya J, Torre-Bastida AI, Zárate G, et al (2023) A joint study of the challenges, opportunities, and roadmap of mlops and aiops: A systematic survey. ACM Computing Surveys 56(4):1–30. https://doi.org/10.1145/3625289

Azure M (2022) Distributed training with Azure Machine Learning. URL https://learn.microsoft.com/en-us/azure/machine-learning/concept-distributed-training?view=azureml-api-2, accessed on 07.12.2022

Baena-Garcıa M, del Campo-Ávila J, Fidalgo R, et al (2006) Early drift detection method. In: Fourth international workshop on knowledge discovery from data streams, Citeseer, pp 77–86, URL https://api.semanticscholar.org/CorpusID:15672006

Barrak A, Petrillo F, Jaafar F (2022) Serverless on machine learning: A systematic mapping study. IEEE Access 10:99,337–99,352. https://doi.org/10.1109/ACCESS.2022.3206366

Barros RS, Cabral DR, Gonçalves PM, et al (2017) Rddm: Reactive drift detection method. Expert Systems with Applications 90:344–355. https://doi.org/https://doi.org/10.1016/j.eswa.2017.08.023

Berberi L (2024) Interactive radar charts on analysis results of mlops products. URL https://public.flourish.studio/visualisation/16000803/, published on 11.12.2024

Bifet A, Gavaldà R (2007) Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining (SDM). Society for Industrial and Applied Mathematics, pp 443–448, https://doi.org/10.1137/1.9781611972771.42, https://epubs.siam.org/doi/pdf/10.1137/1.9781611972771.42

Bifet A, Holmes G, Kirkby R, et al (2016) Moa: Massive online analysis. URL https://github.com/Waikato/moa

ClearML(AllegroAI) (2023) Auto-magical ci/cd to streamline your ml workflow. experiment manager, mlops and data-management. URL https://github.com/allegroai/clearml/, accessed on 10.12.2024

Cramér H (1928) On the composition of elementary errors: First paper: Mathematical deductions. Scandinavian Actuarial Journal 1928(1):13–74. https://doi.org/10.1080/03461238.1928.10416862

Céspedes-Sisniega J, Álvaro López-García (2024) Frouros: An open-source python library for drift detection in machine learning systems. https://doi.org/10.1016/j.softx.2024.101733

DagsterLabs (2018) An orchestration platform for the development, production, and observation of data assets. URL https://github.com/dagster-io/dagster, accessed on 10.12.2024

DataCamp (2023) 17 top mlops tools you need to know. URL https://www.datacamp.com/blog/top-mlops-tools, accessed on 04.12.2023

DataRobot (2023) MLOps 101: The Foundation for Your AI Strategy. URL https://www.datarobot.com/mlops-101/, accessed on 29.12.2023

Dawson R (2021) Guide to evaluating mlops platforms. URL https://www.thoughtworks.com/what-we-do/data-and-ai/cd4ml/guide-to-evaluating-mlops-platforms, accessed on 14.12.2023

Elliott S (2023) Data drift monitoring and its importance in mlops. URL https://whylabs.ai/blog/posts/data-drift-monitoring-and-its-importance-in-mlops, accessed on 08.01.2024

Flyte (2023) Kubernetes-native workflow automation platform for complex, mission-critical data and ml processes at scale. URL https://github.com/flyteorg/flyte, accessed on 10.12.2024

Frías-Blanco I, Campo-Ávila Jd, Ramos-Jiménez G, et al (2015) Online and non-parametric drift detection methods based on hoeffding's bounds. IEEE Transactions on Knowledge and Data Engineering 27(3):810–823. https://doi.org/10.1109/TKDE.2014.2345382

Fu W, Olson R, Nathan, et al (2020) Epistasislab/tpot: v0.11.5, v0.11.5. https://doi.org/10.5281/zenodo.3872281

Gama J, Medas P, Castillo G, et al (2004) Learning with drift detection. In: Bazzan ALC, Labidi S (eds) Advances in Artificial Intelligence – SBIA 2004. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 286–295, https://doi.org/https://doi.org/10.1007/978-3-540-28645-5_29

Garriga M, Aarns K, Tsigkanos C, et al (2021) Dataops for cyber-physical systems governance: The airport passenger flow case. ACM Transactions on Internet Technology (TOIT) 21(2):1–25. https://doi.org/10.1145/3432247

Google (2020) Cloud Architecture Center - MLOps: Continuous delivery and automation pipelines in machine learning. URL https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning, accessed on 12.12.2023

GreatLearning (2024) DevOps Architecture Features. URL https://www.mygreatlearning.com/devops/tutorials/devops-architecture-features, accessed on 16.02.2024

Gretton A, Borgwardt KM, Rasch MJ, et al (2012) A kernel two-sample test. The Journal of Machine Learning Research 13(1):723–773. URL http://jmlr.org/papers/v13/gretton12a.html

Hdlinger VHE (1909) Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. Journal für die reine und angewandte Mathematik 1909:210 – 271. URL https://api.semanticscholar.org/CorpusID:121150138

Hewage N, Meedeniya D (2022) Machine learning operations: A survey on mlops tool support. https://doi.org/10.48550/ARXIV.2202.10169, URL https://arxiv.org/abs/2202.10169

Hu H, Kantardzic M, Sethi TS (2020) No free lunch theorem for concept drift detection in streaming data classification: A review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 10(2):e1327. https://doi.org/10.1002/widm.1327

Hummer W, Muthusamy V, Rausch T, et al (2019) Modelops: Cloud-based lifecycle management for reliable and trusted ai. In: 2019 IEEE International Conference on Cloud Engineering (IC2E), IEEE, pp 113–120, https://doi.org/10.1109/IC2E.2019.00025

Kreuzberger D, Kühl N, Hirschl S (2023) Machine learning operations (mlops): Overview, definition, and architecture. IEEE Access https://doi.org/10.1109/ACCESS.2023.3262138

Kubeflow (2023) Machine learning toolkit for kubernetes. URL https://github.com/kubeflow/kubeflow, accessed on 10.12.2024

Kullback S, Leibler RA (1951) On information and sufficiency. The annals of mathematical statistics 22(1):79–86. URL https://api.semanticscholar.org/CorpusID:120349231

lakeFS (2023) Scalable Data Version Control. URL https://lakefs.io/, accessed on 29.12.2023

Lefevre K, Arora C, Lee K, et al (2022) Modelops for enhanced decision-making and governance in emergency control rooms. Environment Systems and Decisions 42(3):402–416. https://doi.org/10.1007/s10669-022-09855-1

Li Y, Jiang ZM, Li H, et al (2020) Predicting node failures in an ultra-large-scale cloud computing platform: an aiops solution. ACM Transactions on Software Engineering and Methodology (TOSEM) 29(2):1–24. https://doi.org/10.1145/3385187

López García Á, De Lucas JM, Antonacci M, et al (2020) A cloud-based framework for machine learning workloads and applications. IEEE Access 8:18,681–18,692. https://doi.org/10.1109/ACCESS.2020.2964386

Luz WP, Pinto G, Bonifácio R (2019) Adopting devops in the real world: A theory, a model, and a case study. Journal of Systems and Software 157:110,384. https://doi.org/10.1016/j.jss.2019.07.083

Maciel BIF, Santos SGTC, Barros RSM (2015) A lightweight concept drift detection ensemble. In: 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), pp 1061–1068, https://doi.org/10.1109/ICTAI.2015.151

Mage-AI (2023) Mage is an open-source, hybrid framework for transforming and integrating data. URL https://github.com/mage-ai/mage-ai, accessed on 10.12.2024

Mboweni T, Masombuka T, Dongmo C (2022) A systematic review of machine learning devops. In: 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET), pp 1–6, https://doi.org/10.1109/ICECET55527.2022.9872968

Microsoft (2021) Neural network intelligence. URL https://github.com/microsoft/nni, accessed on 29.07.2023

Microsoft (2022) Azure Architecture Center - Machine Learning operations maturity model. URL https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/mlops-maturity-model, accessed on 12.12.2023

MLeap-Docs(CombustML) (2023) Deploy ml pipelines to production. URL https://github.com/combust/mleap, accessed on 10.12.2024

MLflow-Org(Databricks) (2023) Mlflow: A machine learning lifecycle platform. URL https://github.com/mlflow/mlflow/, accessed on 10.12.2024

MLRun(Iguazio) (2023) Machine learning automation and tracking. URL https://github.com/mlrun/mlrun, accessed on 10.12.2024

Montiel J, Halford M, Mastelini SM, et al (2021) River: machine learning for streaming data in python. URL https://github.com/online-ml/river/

Moreno-Torres JG, Raeder T, Alaiz-Rodríguez R, et al (2012) A unifying view on dataset shift in classification. Pattern recognition 45(1):521–530. https://doi.org/10.1016/j.patcog.2011.06.019

Moreschini S, Lomio F, Hästbacka D, et al (2022) Mlops for evolvable ai intensive software systems. In: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp 1293–1294, https://doi.org/10.1109/SANER53432.2022.00155

Munappy AR, Mattos DI, Bosch J, et al (2020) From ad-hoc data analytics to dataops. In: Proceedings of the International Conference on Software and System Processes, pp 165–174, https://doi.org/10.1145/3379177.3388909

Navid P (2022) Navid p.: Airflow, prefect, and dagster: An inside look. 2022. URL https://towardsdatascience.com/airflow-prefect-and-dagster-an-inside-look-6074781c9b77, accessed on 10.12.2024

Netflix (2023) Build and manage real-life data science projects with ease! URL https://github.com/Netflix/metaflow, accessed on 10.12.2024

Nicholl L, Schill T, Lindsay I, et al (2022) Alibi detect: Algorithms for outlier, adversarial and drift detection. URL https://github.com/mitre/menelaus/

Nishida K, Yamauchi K (2007) Detecting concept drift using statistical testing. In: Corruble V, Takeda M, Suzuki E (eds) Discovery Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 264–269, https://doi.org/https://doi.org/10.1007/978-3-540-75488-6_27

Notaro P, Cardoso J, Gerndt M (2021) A survey of aiops methods for failure management. ACM Transactions on Intelligent Systems and Technology (TIST) 12(6):1–45. https://doi.org/10.1145/3483424

Oladele S (2024) Mlops landscape in 2024: Top tools and platforms. URL https://neptune.ai/blog/mlops-tools-platforms-landscape, accessed on 16.02.2024

OMG (2011) Business process model and notation. URL https://www.bpmn.org/, accessed on 08.01.2024

Orchest (2021) Build data pipelines, the easy way. URL https://github.com/orchest/orchest, accessed on 28.07.2023

Pachyderm (2023) Data-centric pipelines and data versioning. URL https://github.com/pachyderm/pachyderm, accessed on 10.12.2024

Page ES (1954) Continuous inspection schemes. Biometrika 41(1/2):100–115. https://doi.org/10.2307/2333009

Paleyes A, Urma RG, Lawrence ND (2022) Challenges in deploying machine learning: A survey of case studies. ACM Comput Surv https://doi.org/10.1145/3533378, URL https://doi.org/10.1145/3533378, just Accepted

Pesaranghader A, Viktor HL (2016) Fast hoeffding drift detection method for evolving data streams. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16, Springer, pp 96–111, URL https://api.semanticscholar.org/CorpusID:31066439

Polyaxon (2023) Mlops tools for managing & orchestrating the machine learning lifecycle. URL https://github.com/polyaxon/polyaxon, accessed on 10.12.2024

Prefect (2023) Prefect is a workflow orchestration tool empowering developers to build, observe, and react to data pipelines. URL https://github.com/PrefectHQ/prefect, accessed on 10.12.2024

Prefect-Docs (2022) Why not airflow? URL https://docs-v1.prefect.io/core/about_prefect/why-not-airflow.html#overview, accessed on 10.12.2024

Qentelli (2024) Comprehensive List of DevOps Tools 2024. URL https://www.qentelli.com/thought-leadership/insights/devops-tools, accessed on 16.02.2024

Raab C, Heusinger M, Schleif FM (2020) Reactive soft prototype computing for concept drift streams. Neurocomputing 416:340–351. https://doi.org/https://doi.org/10.1016/j.neucom.2019.11.111

Rabanser S, Günnemann S, Lipton ZC (2019) Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift, Curran Associates Inc., Red Hook, NY, USA

dos Reis DM, Flach P, Matwin S, et al (2016) Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, New York, NY, USA, KDD '16, p 1545–1554, https://doi.org/10.1145/2939672.2939836

Roberts SW (1959) Control chart tests based on geometric moving averages. Technometrics 1(3):239–250. https://doi.org/10.1080/00401706.1959.10489860

RocketScience (2022) A review of +250 mlops tools and solutions. URL https://rocketscience.one/mlops-software-review/, accessed on 01.06.2023

Ross GJ, Adams NM, Tasoulis DK, et al (2012) Exponentially weighted moving average charts for detecting concept drift. Pattern Recognition Letters 33(2):191–198. https://doi.org/https://doi.org/10.1016/j.patrec.2011.08.019

Rubner Y, Tomasi C, Guibas LJ (2000) The earth mover's distance as a metric for image retrieval. Int J Comput Vision 40(2):99–121. https://doi.org/10.1023/A:1026543900054

Ruf P, Madan M, Reich C, et al (2021) Demystifying mlops and presenting a recipe for the selection of open-source tools. Applied Sciences 11(19). https://doi.org/10.3390/app11198861, URL https://www.mdpi.com/2076-3417/11/19/8861

SeldonCore (2023) Seldon core: Blazing fast, industry-ready ml. URL https://github.com/SeldonIO/seldon-core, accessed on 10.12.2024

Shahin M, Babar MA, Zhu L (2017) Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. IEEE access 5:3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629

Shahin M, Rezaei Nasab A, Ali Babar M (2023) A qualitative study of architectural design issues in devops. Journal of Software: Evolution and Process 35(5):e2379. https://doi.org/10.1002/smr.2379

Subramanya R, Sierla S, Vyatkin V (2022) From devops to mlops: Overview and application to electricity market forecasting. Applied Sciences 12(19). https://doi.org/10.3390/app12199851, URL https://www.mdpi.com/2076-3417/12/19/9851

Symeonidis G, Nerantzis E, Kazakis A, et al (2022) Mlops - definitions, tools and challenges. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp 0453–0460, https://doi.org/10.1109/CCWC54503.2022.9720902

Tensorflow-Extended(Google) (2023) Tensorflow extended (tfx) - end-to-end platform for deploying production ml pipelines. URL https://github.com/tensorflow/tfx, accessed on 10.12.2024

TorchDrift (2024a) Torchdrift: drift detection for pytorch. URL https://torchdrift.org/, accessed on 17.01.2024

TorchDrift (2024b) Torchdrift (github repository). URL https://github.com/TorchDrift/TorchDrift, accessed on 18.01.2024

Van Looveren A, Klaise J, Vacanti G, et al (2019) Alibi detect: Algorithms for outlier, adversarial and drift detection. URL https://github.com/SeldonIO/alibi-detect

Wang C, Wu Q, Weimer M, et al (2021) FLAML: A Fast and Lightweight AutoML Library. URL https://github.com/microsoft/FLAML, accessed on 28.07.2023

WeightsAndBiases (2023) 17 top mlops tools you need to know. URL https://www.datacamp.com/blog/top-mlops-tools, accessed on 12.12.2023

Yaram S (2021) Machine Learning Model Development and Operations: Principles and Practice. URL https://www.kdnuggets.com/2021/10/machine-learning-model-development-operations-principles-practice.html, accessed on 07.12.2022

ZenML (2023) Build portable, production-ready mlops pipelines. URL https://zenml.io., accessed on 10.12.2024

Zhou Y, Yu Y, Ding B (2020) Towards mlops: A case study of ml pipeline platform. In: 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE), pp 494–500, https://doi.org/10.1109/ICAICE51518.2020.00102