D3.1 State of the art landscaping and initial platform requirements specification

Deliverable ID	D 3.1- State of the art landscaping and initial platform requirements specification
Work Package Reference	WP3
Issue	1.0
Due Date of Deliverable	28/02/2023
Submission Date	28/02/2023
Dissemination Level[1]	PU
Lead Partner	UPV
Contributors	CSIC, IISAS, INFN, KIT, PSNC, PRED
Grant Agreement No	**101058593**
Call ID	**HORIZON-INFRA-2021-EOSC-01**

Funded by
the European Union

[1] **PU** = Public, **PP** = Restricted to other programme participants (including the Commission Services),
RE = Restricted to a group specified by the consortium (including the Commission Services),
CO = Confidential, only for members of the consortium (including the Commission Services)

Prepared by	Reviewed by	Approved by
Germán Moltó, Amanda Calatrava, Giang Nguyen, Judith Sáinz-Pardo Díaz, Lisana Berberi, Valentin Kozlov, Mario Santa Cruz	Jesús Marco de Lucas, Marcin Plociennik, Jaime Céspedes Sisniega	PMB Members (27/02/2023)

Issue	Date	Description	Author(s)
1.0	30/09/2022	Initial structure of the deliverable by UPV	Germán Moltó Amanda Calatrava
1.1	05/10/2022	Initial input by IISAS	Giang Nguyen
1.2	17/10/2022	Initial input by CSIC	Judith Sáinz-Pardo Díaz
1.3	02/11/2022	Initial version of section 4 by UPV	Amanda Calatrava
1.4	17/11/2022	Initial input by KIT	Lisana Berberi Valentin Kozlov
1.5	11/12/2022	Input by Predictia	Mario Santa Cruz
1.6	11/01/2023	Finished section 4 and conclusions	Amanda Calatrava
1.7	12/01/2023	Internal review of the whole document	Germán Moltó
1.8	17/12/2023	Review by	Jesús Marco de Lucas
1.9	08/02/2023	Review by	Jaime Céspedes Sisniega
1.10	10/02/2023	Review by	Marcin Plociennik
2.0	16/02/2023	Final review	Amanda Calatrava

Table of contents

Funded by
the European Union

List of tables

List of figures

Funded by the European Union

List of abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BFV	Brakerski-Fan-Vercauteren scheme
BGV	Brakerski Gentry Vaikuntanathan Fully Homomorphic Encryption scheme
CCPA	California Consumer Privacy Act
CI/CD	Continuous Integration / Continuous Delivery
CI/CD/CT	Continuous Integration / Continuous Delivery / Continuous Training Testing
CD	Continuous Delivery
CL	Centralised Learning
CLI	Command-Line Interface
CI	Continuous Integration
CKKS	Cheon-Kim-Kim-Song Homomorphic Encryption scheme
CT	Continuous Training/Testing
CM	Code Management
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DAG	Directed Acyclic Graph
DCG	Dynamic Computational Graph
DevOps	Development and Operations
DL	Deep Learning
DNN	Deep Neural Network
DP	Differential Privacy
DSDM	Dynamic System Development Method
DT	Distributed Training
DVM	Data Versioning and Management
ETL	Extract-Transform-Load
ETMS	Experiment Tracking and Metadata Store
FL	Federated Learning
FM	Feature Management

FHE	Fully Homomorphic Encryption
FPGA	Field Programmable Gate Array
FS	Feature Store
GDPR	General Data Protection Regulation
GL	Gossip Learning
GPU	Graphics Processing Unit
gRPC	Google Remote Procedure Call
GS	Google Storage
GUI	Graphical User Interface
HE	Homomorphic Encryption
IT	Information Technology
JVM	Java Virtual Machine
MDP	Model Deployment
MDV	Model Development
MI	Model Inference
ML	Machine Learning
MLOps	Machine Learning Operations
MPC	Multi-Party Computation
MS	Model Store
MTV	Model Testing/Validation
MVM	Model Versioning and Management
NAS	Network-Attached Storage
NN	Neural Network
PETs	Privacy Enhancing Technologies
PHE	Partial Homomorphic Encryption
PII	Personally Identifiable Information
PPML	Privacy Preserving Machine Learning
REST	REpresentational State Transfer
RPC	Remote Procedure Call
S3	Amazon Simple Storage Service (Amazon S3)
SDK	Software Development Kit

SMA	Secure Model Aggregation
SMPC	Secure Multi-Party Computation
SL	Split Learning
SWHE	Somewhat Homomorphic Encryption
TEE	Trusted Execution Environments
TPU	Tensor Processing Unit
WMP	Workflow Management Platform
XAI	Explainable Artificial Intelligence

1.- Introduction

This document provides an overview of the vast landscape of AI/ML frameworks. It identifies the most relevant open-source developments in the area of ML model training and inference and, in particular, those that cover the lifecycle of AI model management. The focus is set on MLOps frameworks, which provides an extension of the DevOps methodology to develop, train, and deploy ML models via automated procedures that encompass both software, data, security and infrastructure. This study is key for the AI4EOSC project, as it provides a guide for future research and improves the decision-making regarding the design and implementation of the platform architecture. By gaining a deeper understanding of the current state of the field, WP3, WP4 and WP5, can make more conscious decisions. Moreover, the study can also help WP6 understand the current state of the field of MLOps frameworks and facilitate the task of implementing and tuning up their applications to take advantage of the AI/ML ecosystem. Thus, the aim of this document is not to select the proper AI/ML technologies to implement in the project's solution, but to provide an overview of the tools and frameworks available in the field.

The document also introduces the platform requirements specification, together with the methodology designed for requirements gathering, inspired by the Dynamic System Development Method (DSDM) Agile Project Framework, which is based on use cases definition and the identification of Personas, Epics, User Stories and Requirements. This is key for the actions made in WP6, which has performed the actual requirements collection following the methodology proposed and described in this document.

1.1.- Scope of the document

The aim of the AI4EOSC "Artificial Intelligence for the European Open Science Cloud" is to deliver an enhanced set services for the development of AI, ML and DL models and applications in the EOSC. The services will make use of advanced features such as distributed, federated and split learning; provenance metadata; event-driven data processing services or provisioning of AI/ML/DL services based on serverless computing. The vision of the AI4EOSC project is to increase the service offer in the EU landscape by expanding the European Open Science Cloud (EOSC) ecosystem to support the effective utilization of state of the art AI techniques by the research community [Alzubaidi2021, Dong2021, Dargan2020]. To this aim, this document summarises the results of a technology scouting procedure to identify the state of the art concerning AL/ML frameworks. It also includes an initial platform requirements specification, identifying the methodology employed for requirements gathering. The actual requirements are included in deliverable D6.1 "Analysis of user applications, collection of requirements" which is due on month 6, alongside this document.

Funded by
the European Union

1.2.- Target audience

The “State of the art landscaping and initial platform requirements specification” is to be publicly released. The target audience comprises both the technical partners involved in WP4 and WP5 together with use case owners in WP6. It also aims at external AI practitioners interested in a digested summary of the state-of-the-art on AI/ML frameworks.

1.3.- State of the art AI landscaping in the recent years

According to Matt Turck [MTu2022], the year 2020 was the year of resilience and vibrancy, especially in the context of the worldwide coronavirus pandemic [MTu2020]. The spectacular effect of the lock-down and online communication made the two years of digital transformation happen in two months. Cloud and data technologies such as data infrastructure, machine learning (ML) and Artificial Intelligence (AI) as well as data driven applications are at the heart of digital transformation. As a result, many companies in the data ecosystem have not just survived, but to the contrary they have thrived in an otherwise overall challenging political and economic context. Examples of such companies are Snowflake, a data warehouse provider, Palantir, a data analytics platform, Datadog and many more. This evolution is a trend that started ten years ago, and continues to bloom intensively in the next incoming decades. The evolution is formed based on the wide landscape and “*state of the union*” of the data and AI ecosystem.

The need for data quality solutions for AI and ML leads to the modern data stack (building all sorts of data pipelines) becoming mainstream as one of the key trends in data and AI infrastructure. The technology transition shift was from the Hadoop ecosystem to Cloud technology services like Kubernetes with data governance, cataloguing, and lineage. The big shift in this direction has been the enormous scalability and elasticity of Cloud computing and services as well as Cloud data warehouses (for example, Amazon Redshift, Snowflake, Google BigQuery, Microsoft Synapse companies) with the rise of an AI-specific infrastructure data stack such as DataOps, MLOps and AIOps [AI-Infra2022]. Overall, data governance continues to be a key requirement for enterprises, whether across the modern data stacks or ML pipelines. Orchestration engines are evolving. Beyond early entrants like Airflow and Luigi, a second generation of engines has emerged such as Prefect and Metaflow as well as other ML lifecycle management engines. Those products are open source workflow management systems, using high level languages (Python) and designed for modern infrastructures that create abstractions to enable automated data processing such as job scheduling and visualise data flows through directed acyclic graphs (DAGs). Pipeline complexity (as well as other considerations, such as bias mitigation in ML or even drift detection) also creates a huge need for MLOps and DataOps solutions, in particular around data lineage (metadata search and discovery), to understand the flow of data and monitor failure points. The main difference between DataOps and MLOps is that DataOps improves the availability, accessibility, and integration of data, while MLOps aims to facilitate the deployment of ML models in production environments.

The year 2021 and the year 2022 were busy years for MLOps and DataOps as well as the rise of ModelOps as a subset of MLOps [MTu2021]. The fundamental trend is that every company is becoming not just a software company, but also a data company. While Big Data warehouse and data lake vendors are pushing their customers towards centralizing all things on top of their platforms [GoogleUnifying2022], here is the emerging push in data mesh with the decentralization general concept, i.e., different teams are responsible for their own data products. Recently, ML models are getting more deployed and embedded. Four years ago, it was unusual to find a company already deploying models in production, and two years ago, it was unusual to find a company with 50 models in production. Today, two independent surveys estimate that 1/3 of companies that use AI already have more than 50 models in production: Algorithmia in 2021 said about 40%, and Arize in 2022 said 36% [VentureBeat2022]. On the one hand, it is time for real time MLOps accompanied with data and model curation, together with drift detection, defined as the process of detecting a significant change in the concept previously learned by a model or a change related to data distributions [Cespedes2022]. During the development of ML/DL models, it is imperative that metrics are continually stored and monitored. The reverse Extract-Transform-Load (ETL) is more emphasised in the deployment context of AI/ML models.

The art landscaping reflects the recent transition from focusing on modelling to the underlying data used to train and evaluate models with the support from specialised hardware for speeding up general purpose computation (GPU/TPU/FPGA). The evolution is dynamic and involving in the context of the responsible development of human-centric and trustworthy AI systems with the most notable document *“European Union Guideline on Ethics in Artificial Intelligence: Context and Implementation”* as well as other worldwide data regulation and protection laws [EUguidelines] [GDPR], [CCPA], [APPI]. After 4 years of GDPR since 2018, there are two high contrasts: 1) the rise of distributed data analytics using AI/ML with the need for cross-organization data sharing and 2) the rise of data privacy protection and data security. First, in order to address the problem of enabling collaboration between different organizations without the need to share their raw data, several distributed ML architectures emerge. The most prominent is federated learning (FL), which we will focus on in Section 2.2.2, but many others are also on the rise, such as gossip or split learning. In terms of data privacy protection, there are also numerous techniques that are increasingly used in real use cases. Among them, in addition to the classic techniques focused on anonymizing sensitive data (e.g. k-anonymity, l-diversity or t-closeness among many others), differential privacy (DP) has experienced a great boom in recent years, and is present in numerous applications as exposed in [OpenMinedDP]. These transitions are more important and visible in the art landscaping more than ever, and this will continue to be an innovation area over the next incoming years.

1.4.- Structure of the document

After the introduction, this document includes four main sections:

- The “AI/ML Landscape of Frameworks and Tools” describes the approaches from centralised learning to distributed learning, together with the evolution of large-scale AI/ML frameworks and tools, including a summary of AI/ML privacy-preserving tools.
- The “Landscape of AI/ML Lifecycle Management” introduces the MLOps approach and its relationship with Cloud computing. It identifies the most relevant MLOps frameworks and provides a tabular summary of the main capabilities supported by these frameworks.
- The “Platform Requirements Specification” describes the methodology employed for requirements gathering, inspired by the Agile Project Framework, which is used as the basis for actual requirements gathered and described in D6.1 “Analysis of user applications, collection of requirements”.

2.- AI/ML Landscape of Frameworks and Tools

Recently, digitization and globalization have led companies and institutions of many different kinds to collect a vast amount of data. This can range from industrial and banking data to medical data. The large volume of information collected on a daily basis requires thorough analysis in order to infer new knowledge or make predictions, which has led to many advances in ML and DL techniques. However, the great value of such information also implies an imperative need to ensure the security and privacy of the analyzed data [Sainzpardo2023].

2.1 - Centralised Learning

The main characteristic of the federation is cooperation between parties and it is also applied to FL with AI/ML approaches. Classically, the ML process in many life areas follows the Cross-Industry Standard Process for Data Mining cycle (CRISP-DM) [Shearer2000], which consists of six steps:

1. business understanding,
2. data understanding,
3. data preparation,
4. modelling,
5. evaluation and
6. deployment.

The whole CRISP-DM cycle is repetitive. It can be divided into two groups as follows.

- The development phase is the group of the first five phases. It can be repeated with different settings according to the evaluation results.
- The deployment phase is critical for real production under repetitive requisites; it implies online evaluation, monitoring, model maintenance, diagnosis, and retraining.

Currently, data science is an interdisciplinary field to extract knowledge and insights from data to solve analytically complex business problems using AI/ML. The data science process goes through exploratory, data preparation, model planning, model building, communicating results, and operationalizing steps [Ngu2019]. The change in data generation and data collection lead to changes in data processing, data modelling as well as DevOps and MLOps operations.

The classic approach of applying ML to decentralised data (i.e. with several data owners) in a centralised way consists of sending the data from each and every one of the clients or data owners to a central server (Fig. 01), training the model that returns the prediction, and then sending the prediction back to the corresponding client in each case [Sainzpardo2023].

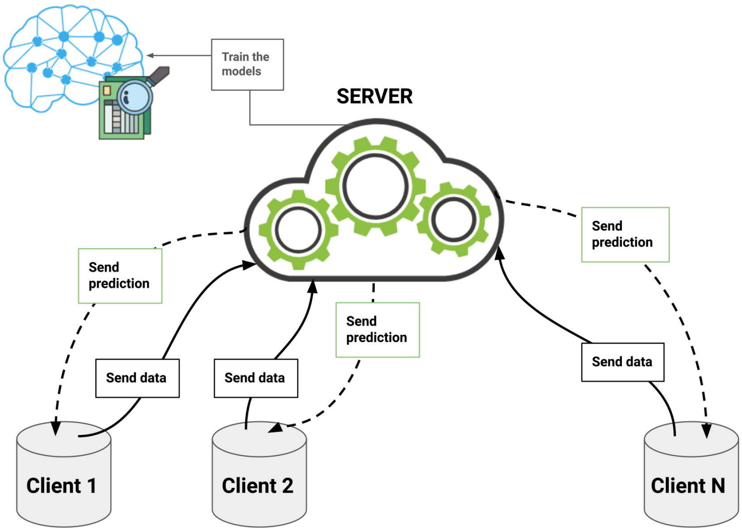


Fig. 01 Diagram of the ML centralised approach [Sainzpardo2023]

2.2 - Distributed Learning, Federated Learning and Split Learning

From the data protection viewpoint, especially in the context of the European Union (EU) General Data Protection Regulation (GDPR), businesses (and companies) should collect the data they need but no more and companies have to promote transparency and guard privacy for sensitive data protection and have the ability to react quickly and professionally to data breaches [Ngu2022]. Here it is suitable to note that sensitive (and usually distributed) data is not only personal data, which is protected under various privacy and compliance regulators. Security information and business data are also sensitive data, which have very similar protection requirements like personal data.

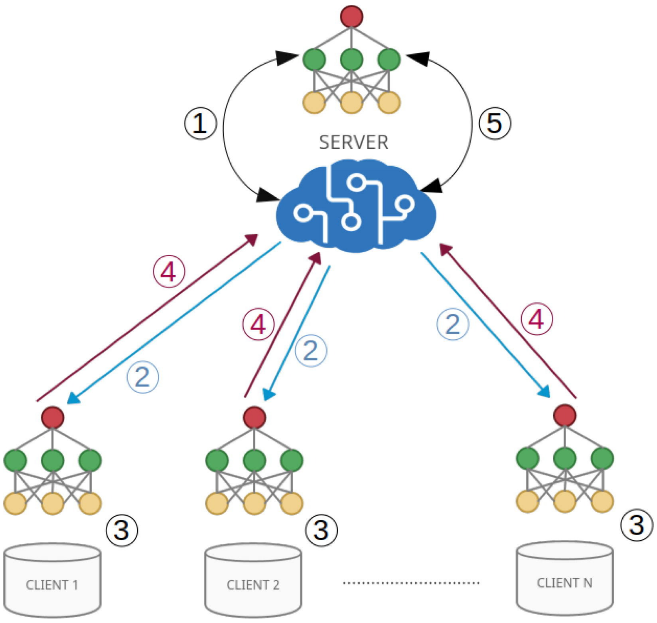


Fig. 02 Scheme of a federated learning (FL) approach [Sainzpardo2023]

Privacy-preserving machine learning (PPML) techniques, such as distributed learning, federated learning (FL), split learning (SL) or gossip learning (GL), aim to address siloed and unstructured data including privacy and regulation of data sharing and incentive models for data transparent ecosystems (Fig. 02).

2.2.1 Distributed Learning

The demand for AI has grown significantly over the past decade, and this growth has been fueled by advances in ML techniques and the ability to leverage hardware acceleration. However, to increase the quality of predictions and render ML solutions feasible for more complex applications, a substantial amount of training data is required. Due to the demand for processing training data, especially in the PPML context, there is a need for distributing the ML workload across multiple sites, and turning the centralised learning into a distributed learning. These distributed manners present new challenges: the efficient parallelization of the training process across (private) network(s) of machines and the creation of a coherent model [Verbraeken2020].

In general, in order to produce robust data-driven learning systems, large amounts of labelled data are required. This means that the data must be processed and labelled with prior knowledge by domain experts before it is centralized in a single place for use to build models with an acceptable level of accuracy. While in some cases this is not difficult to achieve (e.g., datasets that are owned by a single entity) in other cases it is difficult or impossible to obtain a good quality and sufficiently large dataset. This is the case for distributed datasets that cannot be aggregated or merged in a single location, due to a number of different reasons such as privacy concerns or even legal or ethical issues. In this context, the concept of distributed learning (other words: distributed ML) arises, and can be defined as ML/DL algorithms and systems designed to improve performance, preserve privacy, and scale to more training data and larger models [Mah2017], [Heg2019], [SuW2022], [Sainzpardo2023]. This approach is especially useful when the nature of the data itself implies a decentralization thereof, such as in the case of collaboration between different organizations, or in the case of data distributed in different locations belonging to an ecosystem with limited connectivity so that they can be centralized, for example in the case of IoT sensors.

2.2.2 Federated Learning

Federated learning (FL) is a collaborative and decentralised approach to ML, based on the idea of data decentralisation. It was firstly introduced by McMahan in 2017 [Mah2017] and refers to a technique that allows to build data driven models exploiting distributed data without the need to centrally store it (Fig. 02). The main idea in FL is closely related to PPML [Dua2016], especially in the context of deep learning [Shokri2015], [Verbraeken2020].

There are scenarios in which it is beneficial or even mandatory to isolate different subsets of the training data from each other. The furthest extent of this is when a model needs to be trained on datasets that each live on different machines or clusters and may under no circumstance be co-located or even moved. Four different architectures can be proposed in addition to FL as an alternative to distributed ML performed over different clients, without data communication between them, as presented in [SuW2022] as follows.

- All-reduce architecture: it simply consists of removing the dependency on the central server of the FL schema.
- Ring-all-reduce architecture: proposed as an alternative to the all-reduce architecture in order to reduce bandwidth consumption, consists of setting up the clients on a ring structure.
- Gossip Learning (GL) or gossip architecture is presented in [Heg2019]. This configuration is a variation of FL in which no central server is required. Instead, clients directly share their updates among them, and they can conditionally or randomly choose which clients they will communicate with. In this approach the aggregation takes place in a distributed manner.
- Neighbour architecture: similar to the gossip architecture, in this case each client only communicates with its neighbours.

Another approach to training models in a privacy-sensitive context is the use of a distributed ensemble model. This allows perfect separation of the training data subsets, with the drawback that a method needs to be found that properly balances each trained model's output for an unbiased result. Parameter server–based systems can be useful in the context of privacy, as the training of a model can be separated from the training result. FL systems can be deployed where multiple parties jointly learn an accurate DNN while keeping the data itself local and confidential.

2.2.3 Split Learning

Split Learning (SL) is distributed DL and inference without sharing raw data as proposed by the MIT Media Lab's [SplitLearningMIT]. The intuitive idea of the simplest SL configuration: each client trains a neural network up to a specific layer (cut layer). The output at that layer is sent to the server (or another client, depending on the configuration) which completes the training without seeing the raw data. This completes a round of forward propagation. Gradients are back propagated from the last layer to the cut layer. The gradients in the cut layer are sent back to the initial client. The rest of the backpropagation is completed by the initial client.

2.2.4 Other Machine Learning paradigms and Composite AI

ML is an application in which algorithms can learn from experience without being explicitly programmed. Like humans, machines can have different approaches to learn the material. Those different approaches are called ML paradigms, and they present a way how a computer learns from data. The three most well-known learning paradigms are: supervised learning, unsupervised learning and reinforcement learning. The boundaries between learning paradigms are also blurred in many shades. We can briefly list more examples of ML paradigms: hybrid learning (semi-supervised learning, weak-supervision learning, self-supervised learning, multi-instance learning, transfer

Funded by
the European Union

learning, active learning), statistical learning/inference (inductive learning, deductive learning, transductive learning), and learning technique paradigms (multi-label learning, one-shot learning, few-shot learning, multi-task learning, online learning, adaptive learning, incremental learning, ensemble learning), and many more [Lheureux2017] [Emmert2022]. The list is not closed and it gets longer day by day along with research involvement.

In the context of the distributed and federated learning presented in the previous section, it is suitable to mention other machine learning paradigms that also emerge from the classic ML concept in the similar context as follows:

- **Incremental learning**: is a ML approach in which the learning process takes place each time new data are available, adjusting the learning process learned based on the new samples [Belouadah2021]. This type of technique does not require a sufficiently large training set before starting to train the ML models, as new samples will be incorporated gradually over time [Geng2009].

- **Ensemble learning:** The idea is to use different ML models in order to improve the predictions obtained individually with each one of them [Dong2020]. Some classic examples are *bagging* methods (such as Random Forest, which combines different decision trees) *boosting* models, such as AdaBoost or Gradient Tree Boosting. However, it is also possible to combine different models, using for example a voting ensemble, or the stacked generalization method, which allows to reduce the bias of the models. These methods are implemented, among others, in the *ensemble* package of the Python *sklearn* library [EnsembleSKL].

- **Reinforcement Learning:** is an area of ML in which new effective strategies to be followed are proposed based on experimentation with data. That is, the AI learns based on its own experience and interacts with the environment in order to find the optimal behaviour. For this purpose, positive or negative rewards are available, and based on its own experience the machine will reinforce the actions according to whether the reward it obtains is positive or negative [Botvinick2019]. In short, the aim is for the machine itself to learn how to optimize the decision process [RLOptim].

It is particularly interesting to mention a concept that is especially in the state of the art: **Composite AI** [EmergingTech2020] [CompositeAI2023]. Essentially, as its name suggests, the objective of this tool is to combine different AI techniques in order to obtain substantial improvements in the results. It should be noted that ML and DL models require a large amount of labeled data to be sufficiently accurate, which is not always possible. Moreover, in many cases it may be even more convenient to use background knowledge models, NLP or symbolic AI among other tools [ColossalAI]. With this motivation arises the concept of composite AI, to carry out a combination of different AI techniques in order to take advantage of the strengths of each of them, and creating synergies to solve increasingly complex problems (without the need for such large amounts of data as in the case of ML/DL models), and faster [CompositeAI2020].

2.2.5 Differential Privacy and Homomorphic Encryption

In addition to the previously exposed techniques for PPML, it is also important to focus on other techniques that can be applied in combination with the above to add an additional layer of privacy in both data processing and analysis. That is, other Privacy Enhancing Technologies (PETs) for ML are presented below (see [Soykan2022] for further details in these techniques).

Differential Privacy (DP) belongs to the latest methods of preserving privacy. The main thought behind differential privacy is that absence of a single record does not impact the overall dataset characteristics. The implementation of differential privacy was proposed in [Dwork2014] with the additional ε parameter defining the level of privacy. In particular, this technique is of great interest when combined with DL modelling. For example, if it is applied to the weights or gradients of a model prior to sending it to the server in an FL scheme, an additional layer of privacy is added, which can be key since information from the original data can be extracted from the parameters which define a model.

Homomorphic Encryption (HE) is a cryptography paradigm that allows computation to be performed on the encrypted data. HE ensures that performing an operation on encrypted data and decrypting the result is equivalent to performing analogous operations without any encryption [Acar2018]. HE can be divided into:

- Fully Homomorphic Encryption (FHE): with FHE, business can analyze and process sensitive data while maintaining privacy and compliance controls. With this technology, internal or external parties can conduct data analysis and processing without requiring data to be exposed (decrypted) [Gentry2009], [IBM2021]. FHE allows to perform any number of operations so it is very expensive computationally in practice [OL2023]. Theoretically, any function can be computed.

- Somewhat Homomorphic Encryption (SWHE): it is a scheme more feasible in practice, but limits the number of operations that can be performed on encrypted data.

- Partial Homomorphic Encryption (PHE): allows only one type of operation to be performed (any number of times). It is a straightforward and efficient scheme despite its limitations.

In this context, two techniques that focus on privacy preservation, especially in order to enable secure collaboration between entities are as follows.

- **Secure Multi-Party Computation (SMPC)** aims to achieve the collaboration of different entities while preserving the privacy and confidentiality of their information. It is a generic cryptographic paradigm that enables computation process distribution across multiple parties, preserving any single party to reveal their private input and output data.

Funded by
the European Union

- **Secure Model Aggregation (SMA)**: in the same way as SMPC, the SMA goal in a FL scheme is to operate on encrypted models for the server so that when aggregation is performed, the individual contribution of each client involved is not known [SAFL2021].

The most important driver in the emergence of federated learning models is privacy. These techniques are developed in which privacy-aware approaches as training data is not shared among the collaborating parties, but it is arguable that existing solutions meet all the privacy requirements [Soykan2022]. PETs constitute a set of methods that can be used to improve privacy containing different building blocks including SMPC, HE, DP, as well as confidential computing using Trusted Execution Environments (TEE).

Tool	**Advantages**	**Disadvantages**
Federated Learning (FL)	- The data does not leave the device/institution that generates it (no exchange of raw data). - More diverse data to train the models. - Reduced latency and communication costs. - Continuous improvement of models using data from different clients. - Useful both with ML and DL models.	- Corruption of models by attackers. - Nodes may be intermittent. - Requires more power and memory of devices to train the model.
Split Learning (SL)	- No exchange of raw data. - Useful when each data owner has little data - Low memory cost. - Communication efficiency.	- Corruption of models by attackers. - Computationally complex when using huge amounts of data. - With the definition outlined in this study, it can be used only when the model is a neural network.
Differential Privacy (DP)	- Robust and simple algorithms. - Balance between privacy and utility. - Impossible to know if an individual is in the database.	- Large number of records required- - Noise can affect analysis.
Homomorphic Encryption (HE)	- Operate on encrypted data. - The result will also be encrypted.	- Very expensive computationally. - Very long computation time.

Table 01 Summary of advantages and disadvantages of four main privacy preserving techniques: FL, SL, DP and HE

Funded by the European Union

During this section the focus has been put especially on four widely used techniques that reflect the state of the art of privacy preserving tools: Federated Learning, Split Learning, Differential Privacy and Homomorphic Encryption. Although all of them are widely used techniques, it is important to be aware of the advantages as well as the disadvantages of using each of them as summarised and presented in Table 01.

2.3 - Evolution of Large-scale AI/ML Frameworks and Tools

The most popular and the most used frameworks and tools for centralised (ML) and deep learning (DL), with the dominance of Tensorflow (in general) and PyTorch (in research) were summarised in our work *"Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey"* [Ngu2019] published as the Open Access output of the DEEP-HybridDataCloud project EU H2020-777435 in 2019. Currently, the most popular ML/NN/DL frameworks and tools are presented in Table 01 with following short descriptions and functional notations.

Logo	ML/DL framework	GitHub Stars	ML	DL	Licence and Note
TensorFlow	Tensorflow	171.0 K	✓	✓	Apache 2.0
PyTorch	PyTorch	62.1 K	✓	✓	specific open-source licence for each module
K Keras	Keras	57.2 K		✓	Apache 2.0 built on top of Tensorflow 2
scikit learn	Scikit-Learn	52.7 K	✓		BSD 3-clause
dmlc XGBoost	XGBoost	23.7 K	✓		Apache 2.0 ensemble learning
	fast.ai	23.3 K		✓	Apache 2.0 wrapper for PyTorch
mxnet	MXNet	20.2 K		✓	Apache 2.0
飞桨 PaddlePaddle	PaddlePaddle	19.4 K		✓	Apache 2.0
DL4J	deeplearning4j	12.7K		✓	Apache 2.0 DL for JVM
AI Colossal-AI	Colossal-AI	6.7 K		✓	Apache 2.0, parallelism wrapper for Big DL models

Logo	ML/DL framework	GitHub Stars	ML	DL	Licence and Note
H2O.ai	H2O	6.1 K	✓	✓	Apache 2.0
[M]s MindSpore	MindSpore	3.2 K		✓	Apache 2.0 new and growing one

Table 02 The most popular framework and tools for centralised ML and DL

Tensorflow [Tensorflow] is an end-to-end open-source platform for ML and DNN. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that enables building and deploying ML-powered applications. It was originally developed by researchers and engineers working on the Google Brain team to conduct ML/DNN research. Today, Tensorflow is the most used DL framework, which allows the creation of optimised static graphs with eager execution to achieve more dynamic behaviour.

Keras [Keras] follows the motto "*Deep Learning for humans*". Keras follows best practices for reducing cognitive load: it offers consistent and simple APIs; provides clear and actionable error messages. Keras also has extensive documentation and developer guides. It declares itself as exascale machine learning. Built on top of TensorFlow 2, Keras is an industry-strength framework that can scale to large clusters of GPUs or an entire TPU pod.

PyTorch [PyTorch] follows the motto *"Tensors and Dynamic neural networks in Python with strong GPU acceleration"* with its most notable adoption of a Dynamic Computational Graph (DCG). The growing popularity of PyTorch can be clearly seen in Fig. 04, which presents the trend of the different frameworks used in the research implementations listed in the portal "*Papers with Code*" [Frameworks2022].

Scikit-Learn [Scikit-Learn] is an open-source, simple and efficient tool for predictive data analysis. it is accessible to everybody, and reusable in various contexts. Built on NumPy, SciPy, and matplotlib for ML. The project was started in 2007 and since then many volunteers have contributed. It is currently maintained by a team of volunteers.

XGBoost [XGBoost] is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements ML algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. The same code runs on major distributed environments (Kubernetes, Hadoop, SGE, MPI, Dask) and can solve problems beyond billions of examples. XGBoost has been developed and used by a group of active community members.

MXNet [MXNet] is a DL framework designed for both efficiency and flexibility. It allows the mixing of symbolic and imperative programming to maximize efficiency and productivity. At its core, MXNet contains a dynamic dependency scheduler that automatically parallelizes both symbolic and imperative operations on the fly. MXNet is portable and lightweight, scalable to many GPUs and machines. MXNet has support for

Python, Java, C++, R, Scala, Clojure, Go, Javascript, Perl, and Julia. Gluon (or GluonCV) is its common wrapper library for Computer Vision with PyTorch.

PaddlePaddle [PaddlePaddle] is an open-source (since 2016) Python library for enabling deep learning. It's the first independent research and development deep learning platform in China. It includes more than 200 pre-trained models which can help to accelerate development in industrial applications. As other ML/DL frameworks, it uses tensors to represent data and can be used in distributed learning tasks.

The Eclipse **Deeplearning4J** (DL4J) [DL4J] ecosystem is a set of projects intended to support all the needs of a JVM based DL application. This means starting with the raw data, loading and preprocessing it to building and tuning a wide variety of simple and complex DL networks. Because Deeplearning4J runs on the JVM, it can be used within a wide variety of JVM based languages other than Java, like Scala, Kotlin, Clojure and so on. The DL4J motto is *"Suite of tools for deploying and training DL models using the JVM"*.

H2O [H2O] is an in-memory platform for distributed, scalable ML. It uses familiar interfaces like Python, R, Scala, Java, JSON and the Flow notebook/web interface, and works seamlessly with big data technologies like Hadoop and Spark. H2O provides implementations of many popular ML algorithms including ensemble learning (XGBoost, Random Forests) as well as DNN algorithms. H2O also provides a fully automatic ML algorithm (H2O AutoML), i.e., in simple, unified interfaces to a variety of ML algorithms in H2O.

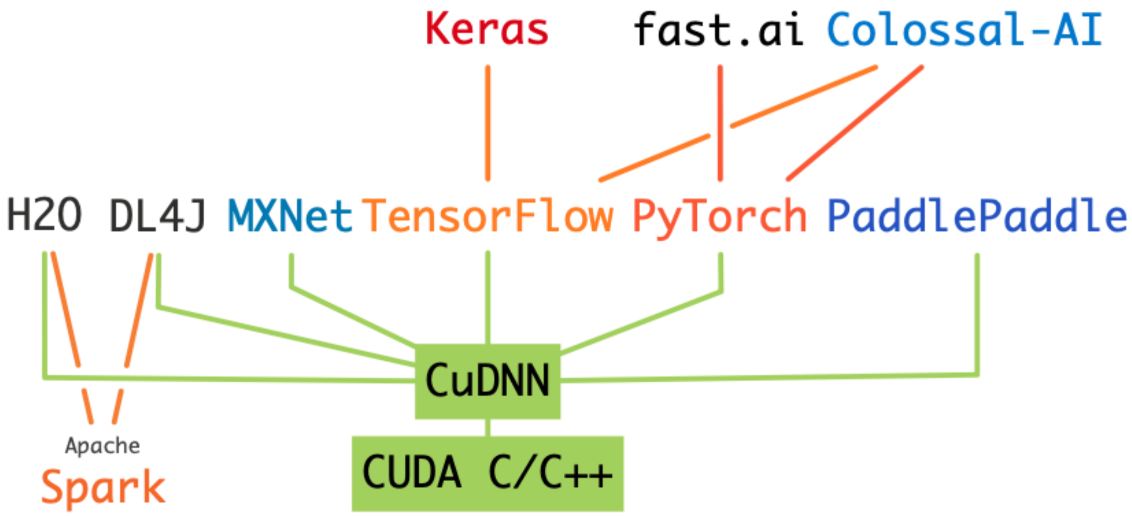


Fig. 03 Notable Neural Network and Deep Learning frameworks

The landscape of large-scale ML/DL frameworks and tools are dynamic and involved. **Tensorflow** is still the leading one with production-grade fame with **Keras** built on top of Tensorflow 2. **PyTorch** continues in its growing trend as a popular research framework. Except for the above-mentioned software products, a lot of other ML/NN/DL tools are here, for example, **fast.ai** [fast.ai] and **deeplearning4j**. Another popular DL framework like **CNTK** [CNTK] is no longer being developed, **Caffe2** [Caffe2] is in archive (Facebook), or **Chainer** [Chainer] is in the maintenance phase with limited development. New frameworks are also coming with growing popularity like **MindSpore** [MindSpore], which is a new open source DL training/inference framework that could be used for mobile, edge and cloud scenarios. Another interesting one in this unified direction is **Colossal-AI** [Colossal-AI], which provides a tool to write distributed Big DL models in an unified way.

The number of ML/DL frameworks and tools are still high and in a dynamic evolving state with the dominance of PyTorch in research (Fig. 04).

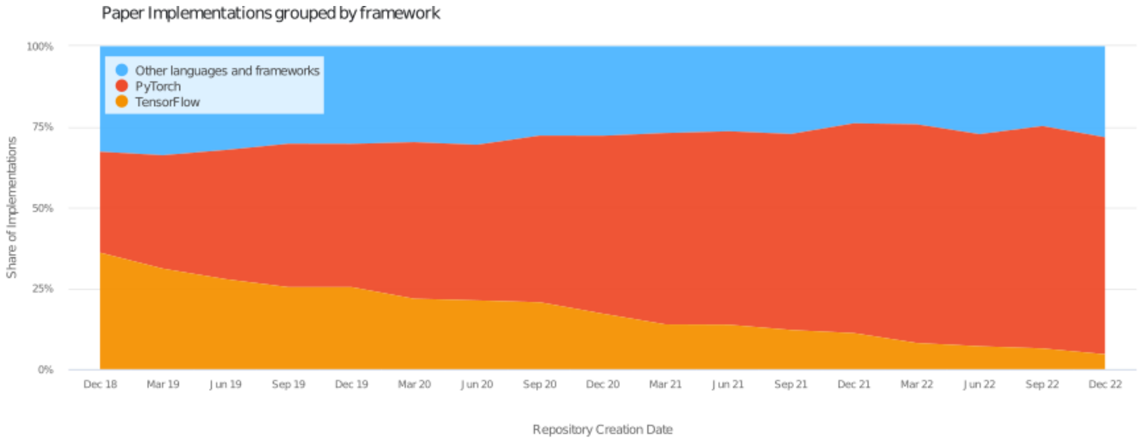


Fig. 04 Research paper implementation grouped by framework. Extracted from *Papers with code* [Frameworks2022].

In the following sections we jump from the traditional centralised ML/DL frameworks and tools to the distributed and federated learning ones, highlighting the main libraries involved. All of these frameworks are privacy-aware, preserving federated learning. However, their privacy implementations are in various realisation levels such as planned, partial or full level. Several of them provide more assurance and protection like secured/encrypted mechanism or peer-to-peer networking among data owners and data scientists. The landscape is highly dynamic with development changing, new or planned feature integrating, product merging and so on.

The list of federated learning (FL) frameworks is presented in Table 02 and the list of Homomorphic Encryption (HE) and Differential Privacy (DP) libraries is presented in Table 03. The short notation of each framework includes its popularity (by number of GitHub Stars), special features, open-source licence and a note. The description of each framework is presented in the following subsections after the tables. Abbreviations in the Table header are: Homomorphic Encryption (HE), Differential Privacy (DP), Secured/Encrypted (SE), Peer-to-Peer networking (P2P).

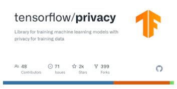
Logo	Federated Frameworks	GitHub Stars	SE	P2P	Licence and Note
TensorFlow Federated	Tensorflow/Federated	2.0 K			Apache 2.0
 tensorflow/privacy Library for training machine learning models with privacy for training data 48 Contributors 71 Issues 2k Stars 399 Forks	Tensorflow/Privacy	1.7 K			Apache 2.0
TFEncrypted	Tensorflow Encrypted	1.1 K	✓		Apache 2.0
PySyft	OpenMined/PySyft	8.5 K			Apache 2.0
PyGrid	OpenMined/PyGrid	615		✓	Apache 2.0
	OpenMined/SyMPC	83	✓		MIT
FATE	FATE	4.8 K			Apache 2.0
FedML	FedML	2.3 K			Apache 2.0
	Flower	1.9 K			Apache 2.0 wrapper
Rosetta	LatticeX-Foundation/Rosetta	504			LGPL 3.0
Open Federated Learning (OpenFL) - An Open-Source Framework For Federated Learning python 3.6 | 3.7 | 3.8 · Ubuntu (latest) passing · Windows (latest) passing · docs passing · downloads 11k · docker pulls 6.7k · pypi v1.4 · slack @openfl · License Apache 2.0 · cite citation · Open in Colab	Intel/OpenFL	391			Apache 2.0
Federated Learning	IBM Federated Learning	367			Specific open-source licence
	NVIDIA/Flare	284			Apache 2.0
	CapePrivacy/TF-Trusted	86			Apache 2.0

Table 03 Notable Federated Learning Frameworks

Logo	Framework	GitHub Stars	HE	DP	Licence
	HElib	2.9 K	✓		Apache 2.0
	Microsoft/SEAL	2.8 K	✓		MIT
	OpenMined/TenSEAL	504	✓		Apache 2.0
	Paillier	476	✓		GPLv3
	OpenFHE	199	✓		BSD-2-Clause
	Opacus	1.3 K		✓	Apache 2.0
	Diffprivlib	667		✓	MIT
	OpenMined/PyDP	361		✓	Apache 2.0

Table 04 Homomorphic Encryption and Differential Privacy Libraries (12.2022).

2.3.2 Secure and Private Tensorflow family

Tensorflow Federated [TF-Federated] is a framework for implementing FL. It is an open-source framework for ML and other computation on decentralised data. Tensorflow Federated enables developers to simulate the included FL algorithms on their model and data, as well as to experiment on novel algorithms. The building blocks provided by Tensorflow Federated can also be used to implement non-learning computations, such as aggregated analytics over decentralised data. The framework consists of two layers of interfaces 1) Federated Learning API, which is a set of high-level interfaces to apply the included implementations of FL and evaluation to the existing Tensorflow models; 2) Federated Core API is a set of low-level interfaces for concisely expressing novel federated algorithms by combining Tensorflow with distributed communication operator within strongly-typed functional programming environment. Tensorflow Federated claims to be architecture-agnostic, which means the ability to compile all code into an abstract representation and as a result, it can be deployed in a diverse environment.

Tensorflow Privacy [TF-Privacy] is a Python library, which includes implementations of TensorFlow optimizers for training ML models with privacy for training data.

Tensorflow Encrypted [TF-Encrypted] is a framework for Encrypted Machine Learning in Tensorflow. It allows the design and implementation of private ML within distributed systems. Tensorflow on its own offers an optimised engine for executing local and distributed computations as well as a high-level interface for expressing these computations. Tensorflow Encrypted provides a basic multi-party computation (MPC) type and passive security under single corruption, which relies on two cryptographic primitives, namely additive secret sharing and secure channel between all participants [Dahl2018]. Tensorflow Encrypted can be seen as a bridge between Tensorflow and the Microsoft SEAL library.

Here are several interesting but less notable frameworks in the secure and private FL such as Tensorflow-based **Rosetta** [Rosetta], or CapePrivacy **TF-Trusted** [TF-Trusted], which allows the execution of several Tensorflow models inside an Intel SGX device.

2.3.3 The OpenMined Syft family

Syft is OpenMined's open-source stack that provides secure and private Data Science in Python [OpenMined]. Syft decouples private data from model training, using techniques like FL, differential privacy, and encrypted computation. The OpenMined Syft family of frameworks is the most popular approach compared to the previous Tensorflow family frameworks. Syft family of frameworks represents a vision of an ecosystem for creating and developing private, AI-powered solutions. The OpenMined Syft family comes with Duet [PySyft]**,** which provides fast prototyping tools such as Jupyter notebook in order to facilitate the user experience.

PySyft [PySyft] is the flagship of the OpenMined Syft family. PySyft augments DL frameworks for servers and IoT with privacy-preserving capabilities. PySyft has several other co-frameworks for different platforms, such as KotlinSyft [KotlinSyft] (Kotlin library for Android), SwiftSyft [SwiftSyft] (Swift library for iOS) or Syft.js [Syft.js] for web and Node, built in Javascript.

PyGrid [PyGrid] is the software providing a peer-to-peer network of data owners and data scientists who can collectively train AI models using PySyft. PyGrid contains Hagrid (HAppy GRID!), which is a command-line tool that speeds up its deployment.

The Syft family also contains **TenSeal** [TenSEAL], which is a library for homomorphic encryption operations on tensors, built on top of Microsoft SEAL [MS-SEAL] to enhance the family software product by cryptography capacity. **SyMPC** [SyMPC] is a library which extends PySyft (version ≥ 0.3) with SMPC support. It allows computing over encrypted data, and to train and evaluate neural networks. **PyDP** [PyDP] is a Python library which provides ε-differential privacy algorithms, such as the laplace mechanism. The features of SyMPC and PyDP are planned to be progressively integrated into the main framework PySyft.

PySyft-Tensorflow [PySyftTF] is a fusion collaboration attempt between the Syft family and Tensorflow family. The framework (56 GitHub stars) brings secure, private DL to Tensorflow, however, it will be deprecated soon in favour of PySyft as recently announced in [PySyftTF] GitHub repository. We can see in this fusion-shattering example the dynamical evolving state in this applied research area.

Funded by
the European Union

2.3.4 Other Federated Frameworks and Tools

Other libraries that are interesting to highlight here are those that implement techniques for homomorphic encryption, as well as those that implement differential privacy algorithms. Some of them are presented below.

Related to Homomorphic Encryption (HE):

- **Microsoft SEAL** [MS-SEAL]: Microsoft SEAL is an easy-to-use open-source (MIT licensed) HE library developed by the cryptography and privacy research group at Microsoft. The library is written in modern standard C++ and is easy to compile and run in many different environments.

- **HElib** [HElib] is an open-source software library that implements homomorphic encryption. It supports the Brakerski Gentry Vaikuntanathan FHE scheme (BGV) scheme with bootstrapping and the approximate number Cheon-Kim-Kim-Song homomorphic encryption scheme (CKKS) scheme. HElib also includes optimizations for efficient homomorphic evaluation, focusing on effective use of ciphertext packing techniques and on the Gentry-Halevi-Smart optimizations.

- **Paillier** [Paillier] is a Python 3 library implementing the Paillier Partially Homomorphic Encryption. The homomorphic properties of the Paillier cryptosystem are 1) Encrypted numbers can be multiplied by a non encrypted scalar; 2) Encrypted numbers can be added together; 3) Encrypted numbers can be added to non encrypted scalars.

- **OpenFHE** [OpenFHE] is an open-source library for open-source for performing Fully Homomorphic Encryption. It includes efficient implementations for the most common FHE schemes such as BGV and Brakerski-Fan-Vercauteren scheme (BFV) for integer arithmetic and CKKS for real-numbers arithmetic, among others. This library is written in C++.

Related to Differential Privacy (DP):

- **Opacus** is a library that enables training PyTorch models with differential privacy [Opacus]. It supports training with minimal code changes required on the client, has little impact on training performance, and allows the client to online track the privacy budget expended at any given moment.

- **Diffprivlib** is IBM general-purpose library for experimenting with, investigating and developing applications in, differential privacy [Diffprivlib].

Turning back to the libraries that implement federated learning architectures and algorithms, the following libraries are worth mentioning:

- **Intel OpenFL** [OpenFL] OpenFL is a Python 3 framework for FL. It is designed to be a flexible, extensible and easily learnable tool for data scientists. OpenFL is hosted by Intel, aims to be community-driven, and welcomes contributions back to the project. It supports aggregation algorithms like FedAvg, FedProx, FedOpt, and FedCurv (presented in works [Mah2017], [Li2020], [Reddi2020], [Shoham2019]) with Tensorflow, PyTorch implementations as well as implementation with other DL frameworks.

- **Flower** [Flower] is a Python library which provides a unified approach to FL, analytics, and evaluation. It is a FL framework that offers a stable language and ML framework-agnostic implementation of a FL system. The framework allows for the rapid transition of existing ML training pipelines into a FL setup to evaluate their convergence properties and training time in a federated setting. Another advantage of Flower is its support for extending FL implementations to mobile and wireless clients, with heterogeneous compute, memory and network resources [Beutel2020]. Flower federates any workload, any ML framework, and any programming language. Flower provides API wrapper for Tensorflow, Tensorflow Lite, PyTorch, PyTorch Lightning, Hugging Face, MXNet, JAX, and Scikit-Learn.

The list of FL frameworks is long [LF-AI-Data-Landscape], which reflects the high interest of the research community to address the secured privacy concerns. Many of them have promising features for federated learning in a dynamic fast development stage. We can list here several of them such as **FATE** [Fate] (Federated AI Technology Enabler) is an open-source project hosted by the Linux Foundation; **FedML** [FedML] is an open-source library whose aim is to facilitate research and production related to federated learning; **IBM Federated Learning** [IBM-FL] is a Python framework for FL in an enterprise environment; **NVIDIA Flare** [Flare] (Federated learning application runtime environment); **Sherpa.ai FL**: framework for FL and Differential Privacy includes both DNN and classical ML approaches [Ludwig2020]. Different use cases are presented in [Rodriguez2020] together with the analysis of the software functionalities and a review of other frameworks; and many more similar products. The evolution has been starting, however these frameworks are currently in a highly dynamic continual development state, i.e., fast changing with many improvements and incompatibility issues.

2.4 - Summary of AI/ML and Privacy Preserving Tools

This is a highly dynamic and evolving environment with a lot of frameworks and tools, which indicates intensive research interests in the PPML area aiming to address siloed and unstructured data including privacy and regulation of data sharing (like GDPR, CCPA) and incentive models for data transparent ecosystems.

- Theoretically, FL has all CL/ML tasks like supervised learning, unsupervised learning or reinforcement learning including neural network implementation. Practically, current implementations of FL libraries and frameworks are in various levels of quality and realisation (e.g., the lack of DL layer types) in comparison with the implementation of classical ML/DL frameworks.

- The highly evolving state of frameworks and tools for distributed and FL leads to the unstable development of intelligent software. The most frequent obstacle is the inconsistency among library versions, which can break up or slow down the development phase of AI models. The appreciated use of the containerization approach may significantly help to overcome this obstacle.

- Requirements on high computational power including the requirement on high memory consumption due to the underlying cryptographic and secure technologies such as Homomorphic Encryption (HE) and Differential Privacy (DP).

- In the special case of Fully Homomorphic Encryption (FHE) the requirement on computational power and memory consumption is exceptionally high, which leads to, in many cases, impossible realisation of secured distributed model training (data at use) and a lot of difficulties also in the inference (data at use). The reason is the tremendous overhead of FHE schemes.

- Here is a lack of access to such high computational power, which enables the secure AI research realisation within AI communities of practitioners.

- The requirement for the network communication (V as Volume) is not so high, because of distributed computing on distributed data, especially with Differential Privacy (DP). Usually, the exchanged data among clients and servers are models, model parameters and metadata for model updates. However, network communication and network performance can be a bottleneck for distributed learning when the model updates are large and/or intensive (V as Velocity).

- The realisation of the networking elements like model aggregation, parameter aggregation is not stable with current FL architecture realisations (vertical and/or horizontal FL architectures). Here is an obstacle to configure such a distributed learning network composed, even private, of servers and clients at large scale for AI/ML practitioners.

Requirements on secure assurance for sensitive data protection in distributed computing environments are for all three data states: data at rest, data in motion and data in use. The concrete requirements are different from application to application. Some of them require the assurance of all states, others can require certain states based on their need.

3 - Landscape of AI/ML Lifecycle Management

Efficient development of reliable AI/ML applications and services is based on **software development methodologies** such as Development and Operations (DevOps) and Machine Learning Operations (MLOps). DevOps can be seen as an extension of Agile software development principles [Agi2022], [AgilePF], which is a set of practices aimed at improving the effectiveness of software development practitioners, teams, and organisations. **DevOps** combines software development (Dev) and IT operations (Ops) aiming to shorten the development life cycle and provide continuous integration and continuous delivery (CI/CD) with high software quality assurance. DevOps encompasses culture and people within an organisation, with the goal of improving collaboration between the development team and information technology (IT) operations. DevOps components are:

- Plan,
- Code,
- Build,
- Test,
- Release,
- Deploy,
- Operate and
- Monitor.

Key features of DevOps architecture are:

- Automation,
- Collaboration,
- Integration and
- Configuration management.

However, there is no *"one size fits all"* and, as a result, there is a wide variety of DevOps tools [Bha2022].

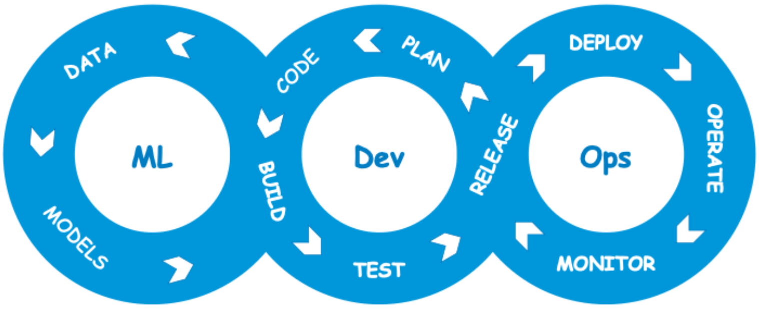


Fig. 05 Machine learning operations MLOps = ML+Dev+Ops [Ngu2022]

MLOps is modelled on the existing discipline of DevOps. It is a set of practices that aims to deploy and maintain ML models in production reliably and efficiently [GoogleMLOps2020] (Fig. 05). DevOps and MLOps tools help organisations to deal with the challenges that come with the implementation of DevOps practices. Apart from MLOps, in recent years, there is also the rise of DataOps, ModelOps, and AIOps.

- MLOps (Machine Learning Operations) applies DevOps principles to developing, deploying, and managing ML models.
- ModelOps is a subset of MLOps that enable organisations to operationalize ML models
- DataOps is the practice of applying DevOps principles to data engineering and management.
- AIOps (Artificial Intelligence for IT Operations) uses AI and ML techniques to automate IT issues such as detection, diagnosis, and resolution.

The main differences between DataOps and MLOps are as follows.

- DataOps is a process-oriented methodology to improve the quality of data, increase the efficiency of analytics, and reduce the time cycle of data analytics. DataOps improves the availability, accessibility, and integration of data.
- MLOps combines operations with ML. It automates and streamlines the entire ML lifecycle, from production to development, deployment to retraining, encompassing DevOps practices such as Continuous Integration and Continuous Deployment (CI/CD) for efficient model management. The aim of MLOps is to facilitate the deployment of ML models in production environments.

Implementation of DevOps and MLOps tools in organisations in a mature operational state can not be done in a short time. It requires a cooperative effort in shifting culture to break down communication silos, which will enable better software and ensure enhanced transparency in the entire chain [Ngu2022].

According to [GoogleMLOps2020], ML and other software systems are similar in continuous integration of source control, unit testing, integration testing, and continuous delivery of the software module or the package. However, in ML, there are a few notable differences:

- CI (Continuous Integration) is no longer only about testing and validating code and components, but also testing and validating data, data schemas, and models.
- CD (Continuous Delivery) 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) is a new property, unique to ML systems, that is concerned with automatically retraining and serving the models.

One next crucial thing to deal with ML is **metadata**, which is data about experiments and model training runs such as debugging, visualising, monitoring model training in order to get to the best model. It is a good practice to log anything (alias metadata) that happens during the ML run such as: data version, code version, environment configuration, hyperparameters, training metrics and losses, hardware metrics (CPU, GPU, TPU), memory consumption during training/inference, evaluation and test metrics (f-score, acc, roc-auc on test and validation datasets), model predictions and domain specific metadata. MLOps need to deal also with above-mentioned metadata as well as pipeline metadata.

Furthermore and in general, **governance** is the backbone of MLOps [Governance2022]. Governance initiatives in MLOps fall into categories: data governance (for ensuring appropriate use and management of data) and process governance (well-defined processes to ensure the life cycle of the model, and all records have been kept).

3.1 AI Infrastructure Landscape

With the increasing availability of huge amounts of data, the ML pipelines for large-scale learning tasks can be exploited to provide an additional level of challenge. It is a common opinion that such tasks cannot be managed by all users to exploit and utilise computational resources in large-scale and distributed e-Infrastructures properly. Most of them, in fact, have domain knowledge in specific fields, but lacking technological or infrastructure knowledge. Therefore, support by the infrastructure layer must break down the complexity of the task and allow scientists to focus on their respective activities, i.e. modelling of the problem, evaluating and interpreting the results of the intelligent algorithms [ALG2020]. There is a complex demand for AI development accompanied with the need for massive resources to support it. The advent of cloud technology offers a powerful infrastructure for AI and ML development and deployment, which is well-known as **AI Infrastructure**.

Data engineers, data scientists, analysts, and anyone working in any kind of a data role have to juggle an ever-increasing number of scheduled tasks [WMR2021] [Zeydan2022]. **Workflow Management Platforms (WMP)** help automate all the processes. One notable moment of the entrant modern open-source WMP for data engineering automation started in 2014 with Airbnb's Airflow project.

- The Airflow [Airflow] project is written in Python, and its workflows are created via Python scripts as **DAGs**. Workflow orchestration is based on the principle "*configuration as code*". Airflow works best with workflows that are mostly static and slowly changing. It is not a streaming solution, but it is often used to process real-time data, pulling data off streams in batches.

- Spotify's Luigi [Luigi] is another similar project. Luigi is a Python package that builds complex pipelines of batch jobs. Spotify uses Luigi to build long-running pipelines of thousands of tasks that stretch across days or weeks. Luigi doesn't use DAGs but it connects *"tasks"* and *"targets"* as **pipelines** [WMR2021].

However, Airflow in particular and early entrance WMPs in general, were not designed to execute data pipelines directly like ML/AI code in the learning cycle required in the Data Science process (Fig. 06). ML/AI code specifies the need to pass data and metadata between the tasks, to be dynamic and parameterized, to run in parallel such as in hyper-parameter tuning manner or distributed learning, or the requirement for a more flexible and low latency scheduler [Hewage2022]. This is the reason for the emergence of **the next generation** of modern WMP like MLflow, MetaFlow, Prefect, Dagster and other products grouped as **MLOps platforms** [DataCamp2022]**.** The common aim of all MLOps platforms is MLOps management within AI Infrastructure.

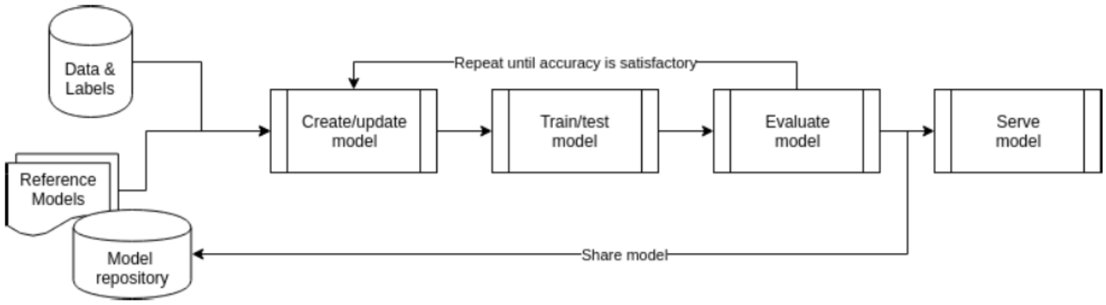


Fig. 06 Simplified learning life cycle for a ML application [ALG2020].

The AI Infrastructure landscape report [AI-Infra2022] was released in October 2022. In the report a number of notable products have been analysed to define and validate at what degree they comply and support the different categories/features as main components of their workflow and workload management environment/system. From that original list, the most notable open-science products are selected based on works [Hewage2022] and appended relevant MLOps platforms in Table 05. A list of these categories (sorted by the feature importance from AI infrastructure management with MLOps products) together with their brief feature description is as follows:

- **Orchestration (O)** coordinates and manages the individual workflows that resulted from disassembling the end-to-end ML pipeline.
- **Distributed Training (DT):** when the workload to train a model is split up and shared among multiple mini processors, called worker nodes. These worker nodes work in parallel to speed up model training. Typically used for training deep neural networks in deep learning [DistLearn].
- **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 for building the model, choosing an algorithm to build the model and eventually building the model [KDNuggets-MDV].
- **Model Testing/Validation (MTV)** is about accurately checking (e.g. unit tests) the expected behaviour of the model and providing metrics/plots to summarise the performance on a validation/test dataset.
- **Model Inference (MI)** is the process of inferring results from input(i.e. live and unseen) data on a fully trained model.
- **Model Deployment (MDP)** is the process of putting/launching the ML model to a production environment to be used for the intended purpose.

- **Experiment Tracking and Metadata Store (ETMS)** is a main mechanism to log relevant metadata and results (often when building a model the team has to try different models, hyperparameters and/or training/test data sets). It helps to evaluate, identify, and reproduce experiments.

- **Data Versioning and Management (DVM)** is the process of handling changes (from different teams) in the datasets (used as input in a training model) through a specific version number. A tool to manage and organise these different generated versions is essential to enable reproducibility.

Except for the above-mentioned features, there are also more desired features such as Feature Management (FM) and Model Versioning and Management (MVM). FM is also called in several MLOps product documentations a Feature Store (FS), and MVM as a Model Store (MS). These features can stay along as a MLOPs product feature or they can be seen as a part of the ETMS feature.

Product (platform)	GitHub Stars	O	DT	CM	MDV	MTV	MI	MDP	ETMS	DVM
MLflow	13.4 K						✓✓	✓✓	✓✓	
Kubeflow	12.2 K	✓✓	✓✓		✓✓				✓✓	
Prefect	10.9 K	✓✓								✓✓
Dagster	6.2 K	✓✓						✓✓		✓✓
MetaFlow	6.2 K	✓✓							✓✓	✓
Pachyderm	5.7 K	✓✓	✓	✓✓	✓✓	✓✓	✓✓	✓	✓	✓✓
ClearML	4.0 K	✓✓	✓✓	✓✓	✓✓	✓	✓✓	✓✓	✓✓	✓✓
Seldon core	3.6 K	✓✓				✓✓	✓✓	✓✓	✓	
Polyaxon	3.2 K	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓
Flyte	3.1 K	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓	✓✓
ZenML	2.6 K	✓✓		✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓
TFX	1.9 K	✓✓	✓✓	✓✓	✓✓	✓	✓✓	✓✓	✓	✓✓
MLeap	1.4 K	✓✓						✓✓		
MLRun	898	✓✓	✓✓	✓	✓✓	✓	✓✓	✓✓	✓✓	✓✓

Table 05 Notable Open Source MLOps Platforms

Abbreviations in the Table header: Orchestration (O), Distributed Training (DT), Code Management (CM), Model Development (MDV), Model Testing/Validation (MTV), Model

Inference (MI), Model Deployment (MDP), Experiment Tracking & Metadata Store (ETMS), Data Versioning and Management (DVM).

In Table 05, 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 without any limitations). The reason is the fast changing and evolving state of MLOps products in the AI Infrastructure landscaping. The number of desired features/abilities are increasing, for example, there is a tendency to break DVM into DVM, FM, and MVM or a tendency to break ETMS into FS, MS and the rest of metadata. Such similar tendencies can be temporary (or not temporary based on concrete product development), which may reflect in their documentations evolving with time.

The list of all existing MLOps platforms is flourishing, so we can talk about fast expansion of them. Except for MLOps platforms, here are a long list of similar products oriented to ML life-cycle management such as Microsoft FLAM [FLAM], Neural Network Intelligence (NNI) [NNI], Epistasis Lab TPOT [TPOT] or Orchest [Orchest] (currently only beta version). The boundaries between MLOPs platforms and ML life-cycle management products are also not clear and not strict, whereas in general, MLOps products are more closely related to AI Infrastructure management [AI-Infra2022] [NeptuneAI2022] in comparison with ML life-cycle management products. It is suitable to note several products are oriented just to selected features, for example, DVC [DVC] is for data version control.

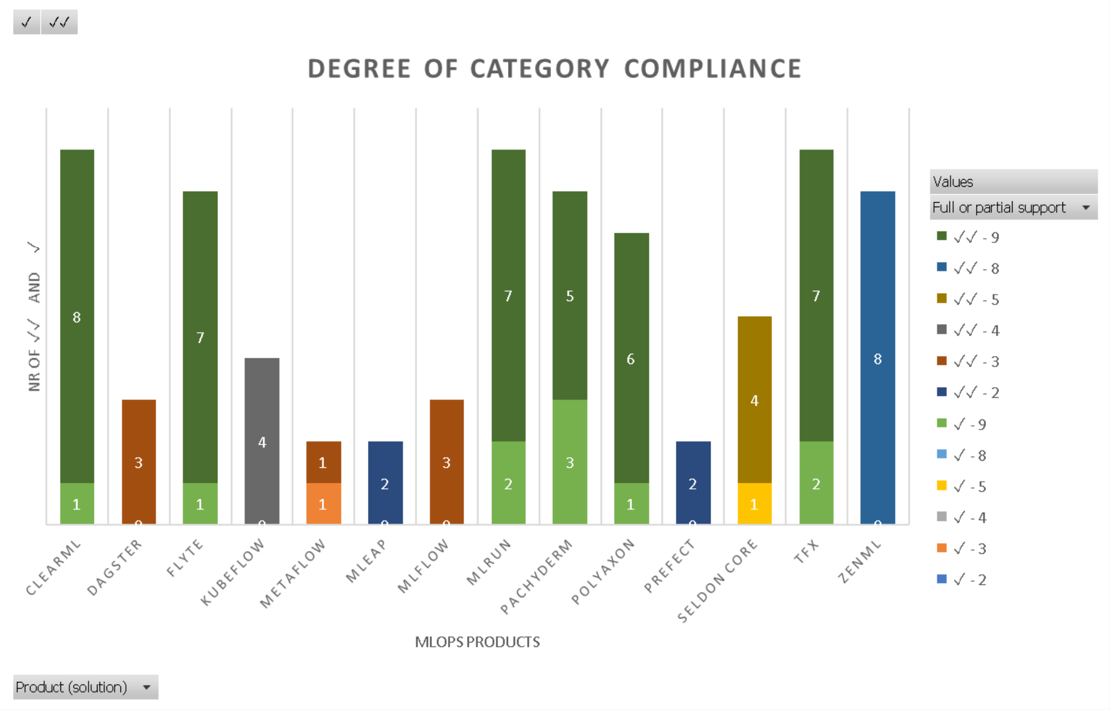


Fig. 07 Calculated degree of the category compliance for MLOps products

The list is long and dynamically involved as all of these products are fast evolving in the recent time [RockeScience2022]. The most common features of the most notable open-source MLOps products presented in Table 05 are as follows.

- All products presented in Table 05 are open-source under Apache 2.0 licence. The exception is Pachyderm with its specific open-source licence.
- All of 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 Jupyter Notebook (except for MLFlow and MLeap). PyTorch is also supported by almost all of the products (except for Pachyderm and MLeap) [NeptuneAI2022-Menzli].
- Minority supported languages from a part of these products are R, Java, Scala, Go, C++, shell and Jsonnet.

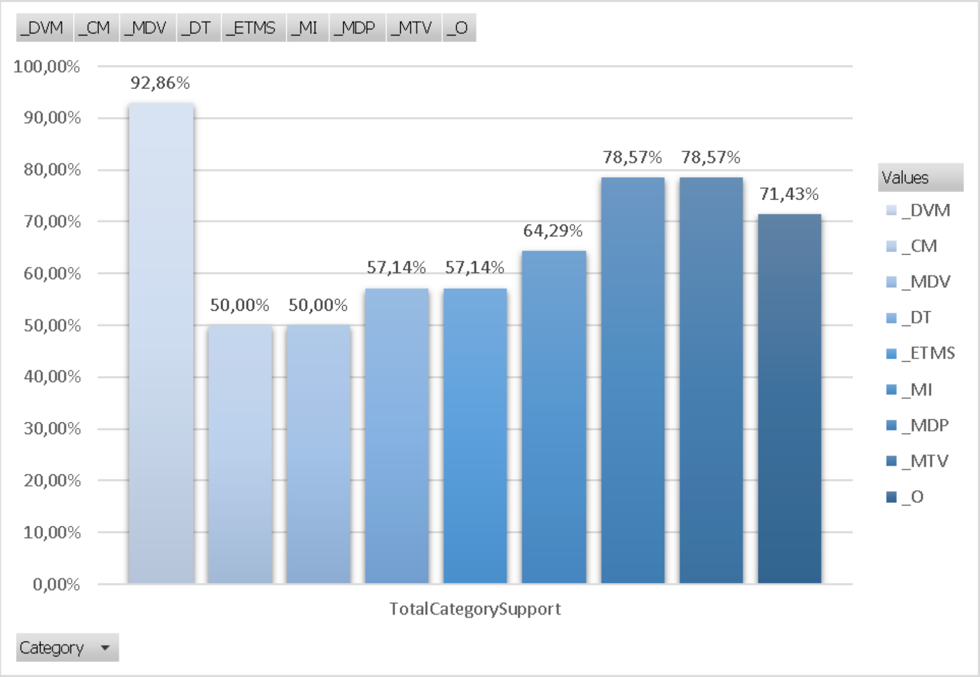


Fig. 08 Calculated percentage score of category support levels

Even platforms with a similar scope have different concepts and strategies, making them hard to compare directly. Table 05 consists of **14 open-source MLOps** products analysed across **9 features/abilities** (O, DT, CM, MDV, MTV, MI, MDP, ETMS, DVM). The overall product's score (degree of category compliance) results is shown in Fig. 07:

- ClearML, TFX, MLRun and ZenML supported partly or fully all the categories, scoring the highest degree of compliance.
- A moderated level compliance (i.e. > 66%) was achieved for ML products like: Pachyderm, Flyte and Polyaxon.
- Whereas, Seldon core and Kubeflow recorded a medium score (i.e. 50%).

Funded by the European Union

- Eventually, a low level compliance (i.e. <22%) was observed for MLflow, MLeap, Prefect, Dagster, and MetaFlow products.

Not all above-mentioned MLOps features (O, DT, CM, MDV, MTV, MI, MDP, ETMS, DVM) in Table 05 are fully supported by all presented MLOps products. A graph with the current status of each category support level is given in Fig. 08. The partially-supported state is frequent as well as currently no-support or in the development plan.

- The category DVM is the most supported among all products;
- The category CM and MDV are the least supported ones;
- The remaining categories are at a moderate level (>50%) supported by the selected products.

From these above-mentioned 14 MLOps products (Table 05), Padycherm, ClearML, Polyaxon are more or less commercial MLOps products with an open source free/community version/licence. The limitations of their free open source versions are in the number of users, the resource scale and supported levels of services.

AI infrastructure with a MLOps product for management can provide various auxiliary software products (available as Docker images) oriented to specific needs of their users like label annotator such as CVAT (Computer Vision Annotation Tool) [CVAT], blockchain node/client like Polkadot [Polkadot] or Advanced Ethereum Client [Parity-Ethereum]. These blockchain nodes (clients) require the same secure and distributed technology for collaborative communication as federated learning.

3.2 MLOps Platforms

In the following sections, we present a concise presentation of the most notable in the next generation of MLOps products for AI Infrastructure management for intelligent softwares based on product vendor documentations and the most recent AI landscape evaluation reports [AI-Infra2022] [NeptuneAI2022].

3.2.1 MLflow

Logo	mlflow
URL	https://mlflow.org/
Motto	An open source platform for the ML lifecycle
Open-Source	https://github.com/mlflow/mlflow/ (13.4K GitHub stars)

MLflow [MLflow] 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). MLflow's current components 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.

Insights: MLflow can be classified into the group of model metadata storage and management MLOps products. It has the following main supported features (Table 05): Model Inference (MI) and Experiment Tracking and Metadata Store (ETMS). It is great as a basic ML lifecycle platform to manage the whole ML lifecycle that includes experimentation, reproducibility, deployment, and a central model registry. The tool is library-agnostic, meaning usable with any ML library and in any programming language.

3.2.2 Kubeflow

Logo	Kubeflow
URL	https://www.kubeflow.org/
Motto	The ML toolkit for Kubernetes. Kubeflow is the cloud-native platform for ML operations - pipelines, training and deployment.
Open-Source	https://github.com/kubeflow/kubeflow (12.2K GitHub stars)

The Kubeflow project [Kubeflow] 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.

Insights: Kubeflow has the following supported features (Table 05): 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.

3.2.3 Prefect

Logo	PREFECT

URL	https://www.prefect.io/opensource/
Motto	You should love your workflows again
Open-Source	https://github.com/PrefectHQ/prefect (10.9K GitHub stars)

The Prefect 2 [Prefect2] is the second-generation dataflow coordination and orchestration platform from Prefect. Prefect 2 has been designed from the ground up to handle the dynamic, scalable workloads that the modern data stack demands. It is powered by Prefect Orion, a brand-new, asynchronous rules engine.

Insights: Prefect has two major concepts, Workflow and Task as the next data orchestration tool for Python. These concepts are quite similar to an Airflow DAG and nodes inside DAG. However, Prefect claims that it is much better than Airflow [Prefect-vs-Airflow] thanks to the result of years of experience working on Airflow and related projects with an user-friendly, lightweight 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 any time, with any concurrency, for any reason. A schedule is a predefined set of start times with a flow parameter to define workflow time dependency.

3.2.4 Dagster

Logo	dagster
URL	https://dagster.io/
Motto	An orchestration platform for the development, production, and observation of data assets.
Open-Source	https://github.com/dagster-io/dagster (6.2K GitHub stars)

Dagster [Dagster] is an orchestrator that's designed for developing and maintaining 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 machine learning pipelines (feature pipelines, training pipelines, and batch inference pipelines).

Insights: Dagster and Prefect (both launched in 2018) are often compared to each other in the context of the Airflow next generation [Airflow-Prefect-Dagster]. 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.

3.2.5 Metaflow

Logo	METAFLOW
URL	https://metaflow.org/
Motto	A framework for real-life data science
Open-Source	https://github.com/Netflix/metaflow (6.2K GitHub stars)

Metaflow [Metaflow] 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 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, providing a framework to structure Python code as a directed acyclic graph (DAG) 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 either locally or distributed using AWS Batch.

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

3.2.6 Pachyderm

Logo	
URL	https://www.pachyderm.com/
Motto	Data-Centric Pipelines and Data Versioning
Open-Source	https://github.com/pachyderm/pachyderm (5.7K GitHub stars)

Pachyderm [Pachyderm] is cost-effective at 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:

Funded by the European Union

- Data-driven pipelines automatically trigger based on detecting data changes.
- Immutable data lineage with data versioning of any data type.
- Autoscaling and parallel processing built on Kubernetes for resource orchestration.
- Uses standard object stores for data storage with automatic deduplication.
- Runs across all major cloud providers and on-premises installations.

Insights: Pachyderm combines data lineage with end-to-end pipelines on Kubernetes. It's available in three versions, Community Edition (open-source), Enterprise Edition, and Hub Edition (still a beta version). Pachyderm has moved some components of Pachyderm Platform to a source-available limited licence. 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 repo 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 a number of files.
- Provenance - to track the dependencies and relationships among datasets.

3.2.7 ClearML

Logo	CLEAR ML
URL	https://clear.ml/
Motto	ClearML - Auto-Magical CI/CD to streamline your ML workflow. Experiment Manager, MLOps and Data-Management
Open-Source	https://github.com/allegroai/clearml/ (4.0 K GitHub stars)

ClearML [ClearML] declares to turn a ML experiment into MLOps with only 2-lines-of-code and it can easily develop, orchestrate, and automate ML workflows at scale. ClearML comes with Free (up to 3 users), Pro, Scale and Enterprise price plans.

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

- Experiment Manager - automagical 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 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 05), for example, to track and upload metrics and models, reproduce experiments, create bots that send Slack messages based on experiment behaviour, manage data (store, track, and version control), remotely execute experiments on any compute resource you have 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.

3.2.8 Seldon Core

Logo	SELDON
URL	https://www.seldon.io/solutions/open-source-projects/core
Motto	An open source platform to deploy ML models on Kubernetes at massive scale
Open-Source	https://github.com/SeldonIO/seldon-core (3.6K GitHub stars)

Seldon Core [SeldonCore] declares itself as the standard open-source platform for rapidly deploying ML models on Kubernetes at 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.

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 when things go wrong;
- Use model explainers to understand why certain predictions were made.

Seldon Core can also be classified into the group of model deployment and serving MLOps products.

3.2.9 Polyaxon

Logo	polyaxon

Funded by
the European Union

URL	https://polyaxon.com/
Motto	MLOps Tools for Managing & Orchestrating the ML LifeCycle
Open-Source	https://github.com/polyaxon/polyaxon (3.2K GitHub stars)

Polyaxon [Polyaxon] is a platform for building, training, and monitoring large scale DL applications with the aim to solve reproducibility, automation, and scalability for ML applications. Polyaxon deploys into any data centre, cloud provider. It supports all the 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).

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, cloud provider, and can be hosted and managed by Polyaxon. When it comes to orchestration, Polyaxon can maximise the usage of the cluster by scheduling jobs and experiments via their 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 with 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, changelogs, and so on.

3.2.10 Flyte

Logo	LF AI & DATA GRADUATE PROJECT
URL	https://flyte.org/
Motto	Kubernetes-native workflow automation platform for complex, mission-critical data and ML processes at scale
Open-Source	https://github.com/flyteorg/flyte (3.1K GitHub stars)

Flyte [Flyte] is open-source Kubernetes-native workflow automation platform for complex, mission-critical data and ML processes at scale. It has been tested at Lyft, Spotify, Freenome, and others. Flyter'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
- Data-aware and strongly-typed
- Resource-aware and deployments at scale

Insights: Flyte is a 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 creates 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. 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 totaling 100,000 executions every month. Flyte can also be classified into the group of testing and maintenance MLOps products.

3.2.11 ZenML

Logo	ZenML
URL	https://docs.zenml.io/
Motto	Build portable, production-ready MLOps pipelines
Open-Source	https://github.com/zenml-io/zenml (2.6K GitHub stars)

ZenML [ZenML] is an extensible, open-source MLOps framework for creating portable, production-ready MLOps pipelines. It's 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 favourite tools in one place to tailor a workflow that caters to specific needs.

Insights: In ZenML, a Stack represents a set of configurations for user-selected MLOps tools and infrastructure. For instance:

- Kubeflow for ML workflow orchestration;,
- Amazon S3 bucket as an 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 onto computing workloads for ZenML users.

3.2.12 Tensorflow Extended

Logo	TensorFlow Extended
URL	https://www.tensorflow.org/tfx
Motto	Tensorflow Extended (TFX) is an end-to-end platform for deploying production ML pipelines
Open-Source	https://github.com/tensorflow/tfx (1.9K GitHub stars)

Tensorflow Extended (TFX) 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. Both the components themselves as well as the integrations with orchestration systems can be extended. TFX components interact with a ML Metadata backend that keeps a record of component runs, input and output artifacts, and runtime configuration. This metadata backend enables advanced functionality like experiment tracking or warm-starting/resuming ML models from previous runs.

TFX libraries include

- TensorFlow Data Validation (TFDV),
- TensorFlow Transform (TFT),
- TensorFlow Model Analysis (TFMA),
- TensorFlow Metadata (TFMD),
- ML Metadata (MLMD).

TFX requires Apache Beam is 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, which is the initial post-beta release of TFX, which provides stable public APIs and artifacts along with nightly built packages. TFX popularity is still not in the top but growing thanks to the popularity of the Tensorflow family.

3.2.13 MLeap

Logo	mleap
URL	https://combust.github.io/mleap-docs/
Motto	Deploy ML Pipelines to Production

Open-Source	https://github.com/combust/mleap (1.4K GitHub stars)

MLeap [MLeap] aims to deploy ML data pipelines and algorithms not to be 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)

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

3.2.14 MLRun

Logo	MLRun
URL	https://www.mlrun.org/
Motto	ML automation and tracking
Open-Source	https://github.com/mlrun/mlrun (898 GitHub stars)

MLRun [MLRun] is an open MLOps platform for quickly building and managing continuous ML applications across 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 computation resources and to break the silos between data, ML, software, and DevOps/MLOps teams and to enable collaboration and fast continuous improvements.

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 support the following MLOps tasks:

- Project management and CI/CD automation,
- Ingest and process data,
- Develop and train models,
- Deploy models and applications,
- Monitor and alert.

MLRun ecosystem supports the newest data stores, development tools, services, platforms such as:

- Data stores: object (S3, GS, az), 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 (e.g., Kubernetes engine on Docker Desktop), Docker, Linux/KVM, NVIDIA DGX;
- CI/CD: Jenkins, Github Actions, Gitlab CI/CD, KFP.

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

3.3 Summary of MLOps Platforms and AI Lifecycle Management

The MLOps scene is exploding with a multitude of products available in order to address different tasks concerning the ML lifecycle. The rapid expansion of MLOps platforms is only one part of the overall AI landscape. Although at the first glance all MLOps platforms have the same goal of supporting the ML lifecycle, when entering in the details, these platforms have different goals, which opens up space for different platforms to take different approaches. The number of all existing (widely just so-called) MLOps products is much higher (more than 250 listed software products) [RockeScience2022] than the above presented top open-source ones [DataCamp2022]. There are also other approaches to classify MLOps products, e.g., into sub-groups with narrowed purposes as follows.

- Model metadata storage and management.
- Data and pipeline versioning.
- Hyperparameter tuning.
- Run orchestration and workflow pipelines.
- Model deployment and serving.
- Production model monitoring.

The boundaries among MLOps product sub-groups are also blurry (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 a MLOps platform and it can be tempting to look for the popularity (e.g., the score of GitHub stars) or the completeness of features/abilities [Thoughtworks2021], e.g., Orchestration (O), Distributed Training (DT), Code Management (CM), Model Development (MDV), Model Testing/Validation (MTV), Model Inference (MI), Model Deployment (MDP), Experiment Tracking & Metadata Store (ETMS), Data Versioning and Management (DVM) [AI-Infra2022]. Newly appearing products with more supporting features may have less stars just because they have a shorter lifetime on the market, but it does not mean 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:

- Although product popularity is a good indicator to filter out low-ranked products, which are new or not so good quality, popularity is not inline with the number of supported features of MLOps products.
- All products are evolving and adding new features over time.
- Higher range of features can come at a cost of inflexibility and higher cost to maintain.
- Many use cases do not require a wide range of features.
- If required features are missing, it is often sufficient to add supplementary tools.

The best compromise (or balance) between popularity and the number of supported features belong to Pachyderm, ClearML, and Polyaxon (see Table 05)**.** Pachyderm has the most partially supported features (in-progress evolving). However, it is oriented to be commercial with an open source free version. The similar commercial plans with open source free version are with ClearML and Polyxagon, which has the highest number of supported features.

The next interesting ones are Flyte, TFX (pre-release beta version), 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 ML/AI and the same is valid for MLOps products.

Also according to the Neptune.ai comparison report released in December 2022 [NeptuneAI2022], a competent MLOps platform has to have (at least) the following features/abilities:

- **Orchestration (O)**: ability to spin up a service (compute, network and storage management) based on that model artefact to train and consequently to deploy them at the end of the process.

- **Model Development (MDV)**: ability to build model artefacts that contain all the information needed to preprocess data and generate a result.

- **Model CI/CD/CT** including Model Deployment (MDP) and Model Inference (MI): ability to mark models as ready for staging and production, and run them through a CI/CD/CT process.

- **Experiment Tracking and Metadata Store (ETMS)**: ability to track the code that builds them, and the data they were trained and tested on.

- **Data Versioning and Management (DVM)**: 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 as well as deploying and monitoring a model. An MLOps platform can be thought of as a collection of tools for achieving the tasks involved in getting value

Funded by the European Union

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 organisation as a single platform for different use cases.

In the context of AI landscape, AI/ML lifecycle management including AI infrastructure, the DEEP-Hybrid-DataCloud framework (in short DEEP framework, developed as the output of the DEEP-HybridDataCloud project EU H2020-777435) is a distributed architecture to provide ML practitioners with a set of tools and cloud services that cover the whole ML development cycle: ranging from the models creation, training, validation and testing to the models serving as a service, sharing and publication. The DEEP framework allows transparent access to existing e-Infrastructures, concretely the European Open Science Cloud (EOSC), effectively exploiting distributed resources for the most compute-intensive tasks coming from the ML development cycle. Moreover, the DEEP framework provides scientists with a set of cloud-oriented services to make their models publicly available, by adopting a serverless architecture and a DevOps approach, allowing an easy share, publish and deploy of the developed models [ALG2020]. Currently, the DEEP framework is actively supporting use cases coming from the EOSC, the EOSC DIH, the EGI-ACE (like INRAE and JRC) and now iMagine Horizon Europe projects.

The AI4EOSC project will be built on top of the DEEP-Hybrid-DataCloud outcomes and the EOSC compute platform and services. It will deliver an enhanced set of advanced services for the development of Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) models and applications in the EOSC. These services will be bundled together into a comprehensive platform providing advanced features such as distributed, federated and split learning; novel provenance metadata for AI/ML/DL models; event-driven data processing services or provisioning of AI/ML/DL services based on serverless computing. The main outcomes of the AI4EOSC project will serve as a catalyst for researchers, facilitating the collaboration, easing access to high-end pan-European resources and reducing the time to results; paired with concrete contributions to the EOSC exploitation perspective, creating a new channel to support the build-up of the EOSC Artificial Intelligence and Machine Learning community of practice.

4.- Methodology for Platform Requirements Specification

The AI4EOSC platform, that is deeply described in deliverable D3.2. Initial high-level architecture specification, is an integration of several technologies to provide advanced features to train and develop machine learning and deep learning models. It is based in the previous DEEP platform [DEEP-Platform] and will deliver new high-level services and functionalities, targeting direct exploitation by scientific teams, allowing them to reduce the time to results and increase productivity by building better analytics tools, products, and services leveraging AI/ML/DL, with focus on advanced features like federated learning, split learning or distributed training. Initially, the platform has to meet the onboarded use-case applications needs. For that, in this section we describe the methodology followed by WP6 to collect all the use case requirements, information that is key to WP3, WP4 and WP5, to properly design and develop the AI4EOSC platform.

A requirement is a service, function or feature that a user needs. We can categorise a requirement into two different categories:

- Functional: what it does (functionality, features)
- Non-functional: how well it performs against defined parameters.

Requirements evolve and emerge during the project. Moreover, as one of the objectives of the WP3 is to incorporate user input as the basis of the services development through the whole development and implementation processes, carrying out a co-design methodology, we need a methodology to avoid unnecessary rework and to manage complexity of requirements specification.

In this section we describe the proposed methodology for gathering user requirements for the three use cases of the project: Agrometeorological forecasts, Integrated plant protection scenario, and Automated Thermography. The actual data collection for requirements is done by WP6 and the results are collected in D6.1 “Analysis of user applications, collection of requirements”.

4.1.- Requirement Analysis Methodology

The proposed approach in requirements gathering is inspired by the DSDM Agile Project Framework [AgilePF]. Thus, the requirements will be formulated by focusing on personas and looking at the framework from the point of view of the users of the framework (use cases and future new users).

The requirements elicitation process will take place by following five steps, as depicted in Figure 9:

- Step 1: Gather information from the Use Case and the architecture of the application.
- Step 2: Define the Epics of the application.
- Step 3: Define the User Stories resulting from the Epics. The User Stories are always presented using the point of view of different user profiles (Personas).

Funded by
the European Union

- Step 4: Formulate the Requirements which results from the User Stories.
- Step 5: Define which AI4EOSC platform component (tool) is involved in each Requirement.

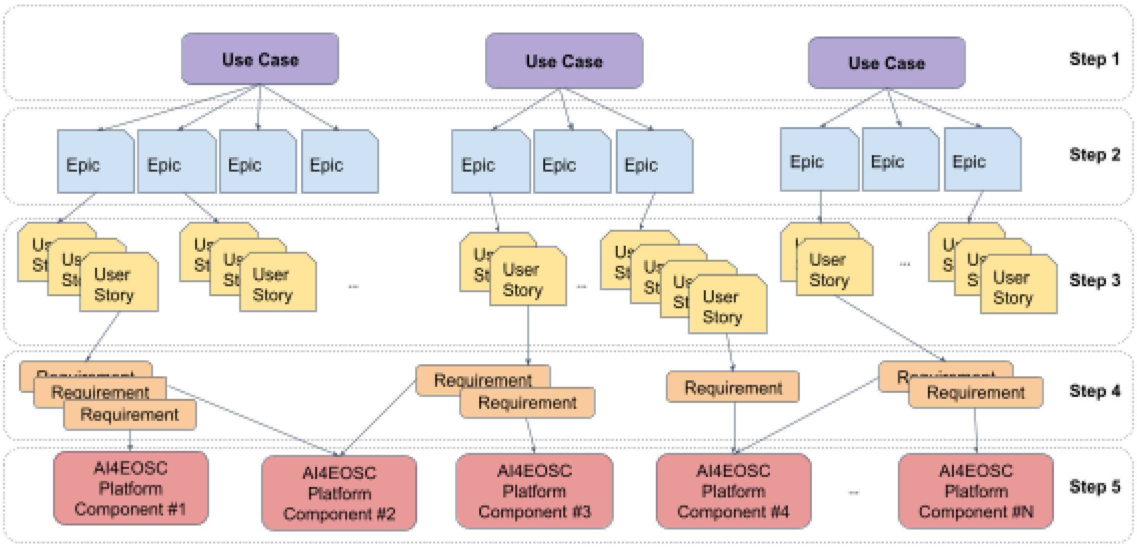

Use Case
Use Case
Use Case
Step 1
Epic
Step 2
User Story
Step 3
Requirement
Step 4
AI4EOSC Platform Component #1
AI4EOSC Platform Component #2
AI4EOSC Platform Component #3
AI4EOSC Platform Component #4
AI4EOSC Platform Component #N
Step 5

Fig. 09 Flowchart of the proposed methodology.

In the next subsections we will analyse deeply each one of the steps.

Step 1: Use case definition

The first step of the requirements elicitation process is to individually analyze and collect relevant technical information from the Use Cases. The relevant information is, but not limited to:

- Data structure, acquisition, and storage
- Data processing (for cleansing and training), in terms of:
  - Hardware requirements: GPUs, CPUs, RAM, Storage
  - Software requirements: Tools involved, i.e. application architecture (and limitations detected in those tools) and execution environment (CLI, GUI - Notebooks, etc.)
- Model inference: Non-functional requirements such as time constraints
- Model distribution (license, packaging, adherence to FAIR principles, etc.)

In AI4EOSC, WP6 has been in charge of interviewing each use case to know in depth all of them. For that, a set of questions organized in 7 different categories has been prepared, gathering information from data sources, data preparation, modelling (registry, serving and monitoring), code and CI/CD. This information is collected and properly analyzed and presented in D6.1 “Analysis of user applications, collection of requirements”.

Step 2: Epics

An ***epic*** is a high-level description of a usage situation that can be broken down into a number of smaller tasks (User stories). Epic descriptions can be represented mainly with a short title and a brief description, and must be related to a use case and have an unique identifier. Epics can also be versioned, thus adapting to the evolution of the use case. Table 06 collects all the fields of an epic, with a short description and an example of each of them.

Field name	Explanation	Example
Epic ID	A unique id with the format <Use Case No.>.E<Epic No.>	UC1.E01
Title	A short sentence representing the Epic	Train an AI model for carrot vs parsnip detection
Description	A short description, usually one or two paragraphs	To efficiently detect if an image is a carrot or a parsnip using the AI4EOSC platform with our mobile application, we need to train an AI model. Images acquired via the cameras need to be preprocessed, labeled and stored for long-term use.
Use Case	The related use case	UC1
Owner	The person in charge of it	John Doe (john@doe.com)
Version	A sequence of numbers that identifies the state of the epic	1.0

Table 06 Template for Epics.

Step 3: User Stories

A ***User Story*** is a requirement expressed from the perspective of an end-user (*Persona*). Starting from an epic, this is divided into several user stories that focus on a unique feature or functionality, clearly mention the *persona* and are described as a one phrase statement.

A ***Persona*** is a fictional character that represents a potential user. Personas need to be defined right at the beginning of this step, because the user stories are defined for the different personas identified in each use case. For example, the Application architect, the AI Model developer, the Infrastructure provider, the AI4EOSC Platform admin, or the

Funded by
the European Union

Application end user for each use case can be possible profiles directly involved with the AI4EOSC platform. The proposed list of personas has been aligned with WP6, and the details together with a description of each one of the profiles observed is available in D6.1 “Analysis of user applications, collection of requirements”.

As a… <persona> **when** <performing an action>
I want To… <do something> (requirement)
So that I can… <achieve some goal> (task)

As a(n)… application manager **when** deploying my service
I want To… deploy Virtual Machines
So that I can… access computing power on-demand

Fig. 10 Schema of a user story and an example

Knowing the actors of the scene, we can now proceed to describe the user stories. Figure 10 represents the typical structure of a user story, and an example of it. Each user story can be seen as a short request that expresses the needs of a *Persona*. Table 07 collects all the information related with a user story, and exemplifies all of the fields considered.

Field Name	Explanation	Example
User Story ID	An unique id with the format <Epic No.>.S<Story No.>	UC1.E01.US01
User Story Title	A short sentence representing the User story	Provision computing resources for AI model training
Priority	One of the following: ● Must have: The system (AI4EOSC platform) must implement this user story to be accepted. ● Should have: The system should implement this user story: some deviation from the requirement as stated may be acceptable. ● Nice to have: The system should implement this user story, but may be accepted without it.	Must Have
Persona	The fictional character that will act	Application Architect

	as the end user of this story. The Personas must be described in advance.	
Description	As a <persona> when <performing an action> I want to <to something> so that <I can achieve some goal>	As the “Application Architect” I need to have access to computing resources with at least 16 GB of RAM and a GPU so that I can train an AI model
Need/Purpose	The objective that the persona wants to achieve through the story	Train the AI model on dynamically provisioned computing resources
Epic ID	The Epic related with the user story	UC1.E01
Use Case	The use case related with the user story	UC1
Owner	The person in charge of it	Jane Doe (jane@doe.com)
Version	A sequence of numbers that identifies the state of the user story	1.0

Table 07 Template for User Stories.

Step 4: Requirements

In this step user stories are transformed into requirements. A requirement is a functionality that a user (Persona) needs (involving at least one technological component of the AI4EOSC platform). Requirements are organized in different categories and have a level of priority and status associated. Each requirement needs to be associated with AI4EOSC Platform Components, those that will be in charge of supplying the requirement. Ideally requirements should be mapped to a single (or few) components to reduce the granularity. Table 08 collects all the data related to a requirement, and exemplifies it.

Field Name	**Explanation**	**Example**
Requirement ID	An unique id with the format <Use Case No.>.Req<Requirement No.>	UC1.Req01
Title	A short sentence representing the requirement	Access to computing power

Funded by
the European Union

Priority	One of the following: ● Must have: The system (AI4EOSC platform) must implement this requirement to be accepted. ● Should have: The system should implement this requirement: some deviation from the requirement as stated may be acceptable. ● Nice to have: The system should implement this requirement, but may be accepted without it.	Must Have
Required for (User Story)	The user story related with the requirement	UC1.E01.US01
Category	One of the following tentative list: ● Computing requirements ● Storage requirements ● Data transfer requirements ● Software Quality Assurance ● Monitoring requirements	Computing requirements
Description	Describe the intention of the requirement	I need a virtual machine powerful enough to train an AI model
Rationale	Justification of the requirement, why is this needed.	To train the AI model, computing resources are needed
Tool	AI4EOSC Platform Component in charge of supplying the requirement	AI4EOSC Training Dashboard
Acceptance Criteria	Metrics, objectives required to achieve the expected outcome of the requirement	16Gb of RAM and a CPU at least.
Supporting Materials	References that illustrate and explain the requirement	-
Tentative scheduling	Tentative scheduling of accomplishment	M8
Status	One of the following: ● Defined ● In Progress	Defined

Funded by the European Union

	- Implemented - In Testing - Cancelled	
Use Case	The use case related with the requirement	UC1
Requester	The person who requests	Jane Doe (jane@doe.com)
Owner	The person in charge of it	John Doe (john@doe.com)
Version	A sequence of numbers that identifies the state of the requirement	1.0

Table 08 Template for Requirements.

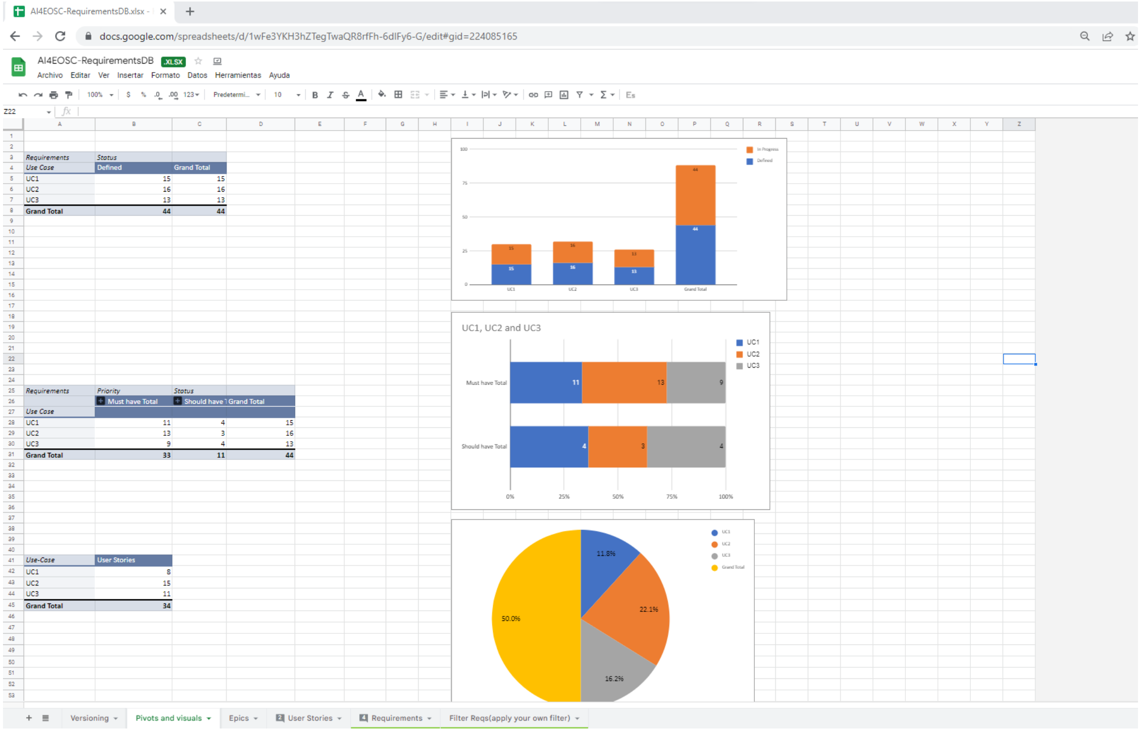
Step 5: Assets

In this step, all requirements from the Use Cases are consolidated and mapped to the AI4EOSC assets (i.e. the key tools of the AI4EOSC platform, such as Infrastructure Manager, PaaS Orchestrator, etc., as described in D3.2 “Initial high-level architecture specification”). In this phase existing requirements may be reformulated in a more detailed way; and additional requirements may be added. Requirements for the assets will be mapped to relevant Use-Case requirements.

4.2.- Implementing the methodology

To effectively implement the described methodology, we rely on a living document, stored in a shared space, with the information of personas, epics, user stories and requirements. With this approach, user communities from each use case and technical teams from AI4ESOC (mainly WP4, WP5 and WP6) can interact among these shared document, that acts as a repository of requirements to analyze and fine-tune them in an iterative approach, thus fostering co-design.

The document has been created as a Shared Google Spreadsheet available to the involved AI4EOSC partners, so that they can access, analyze and verify the needs of the use cases. The document contains 4 main sheets, where data is structured in Epics, User Stories and Requirements, all of them quantitatively represented in a visual sheet as Figure 11 shows. This scenario leads us to maintain a continuously updated set of requirements for the AI4EOSC platform.



AI4EOSC-RequirementsDB.xlsx

docs.google.com/spreadsheets/d/1wFe3YKH3hZTegTwaQR8rfFh-6dIFy6-G/edit#gid=224085165

AI4EOSC-RequirementsDB .XLSX

Requirements	Status	
Use Case	Defined	Grand Total
UC1	15	15
UC2	16	16
UC3	13	13
Grand Total	44	44

Requirements	Priority	Status	
	Must have Total	Should have T	Grand Total
Use Case			
UC1	11	4	15
UC2	13	3	16
UC3	9	4	13
Grand Total	33	11	44

Use-Case	User Stories
UC1	8
UC2	15
UC3	11
Grand Total	34

Fig. 10 Screenshot of the visual sheet of the requirements repository

Detailed information about the actual collection and requirements from the use cases, based on the methodology described here, is provided in D6.1 “Analysis of user applications, collection of requirements”.

5.- Conclusions

This document has provided a comprehensive and up-to-date overview of the most relevant technologies available in the field of AI/ML/DL. Several frameworks, tools and solutions have been analysed and presented in summary in this report, which will be a key input for the design and specification of the AI4EOSC platform architecture. It will improve decision-making in the next steps of the project, not only for WP3 but also for other work packages such as WP4 and WP6. As mentioned, the aim of the deliverable is not to give a winning solution for the project at this point, but to analize the available solutions with the perspective of the project, as described in section 3.3 of this document.

Moreover, the document has described the methodology to collect the requirements of the AI platform taking as the main input the requirements of its initial users, the three use cases of the project, from WP6. As the requirements might evolve during the project, we have also proposed a co-design approach to maintain a live repository of requirements to track the evolution of the needs of the three use cases during the project. This work is actually complemented with the information presented in D6.1 “Analysis of user applications, collection of requirements”, where the methodology is used as the basis for the actual requirements gathered directly from the three use cases: Agrometeorology, Integrated Plant Protection and Automated Thermography.

References

[Acar2018] Acar, A., Aksu, H., Uluagac, A.S. and Conti, M., 2018. A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys (Csur), 51(4), pp.1-35. https://dl.acm.org/doi/abs/10.1145/3214303

[ALG2020] García, Á.L., De Lucas, J.M., Antonacci, M., Zu Castell, W., David, M., Hardt, M., Iglesias, L.L., Moltó, G., Plociennik, M., Tran, V. and Alic, A.S., et al. 2020. A cloud-based framework for machine learning workloads and applications. IEEE Access, 8, pp.18681-18692. https://doi.org/10.1109/ACCESS.2020.2964386

[Alzubaidi2021] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L., 2021. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8(1), pp.1-74. https://doi.org/10.1186/s40537-021-00444-8

[Belouadah2021] Belouadah, E., Popescu, A. and Kanellos, I., 2021. A comprehensive study of class incremental learning algorithms for visual tasks. Neural Networks, 135, pp.38-54. https://doi.org/10.1016/j.neunet.2020.12.003

[Beutel2020] Beutel, D.J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., de Gusmão, P.P. and Lane, N.D., 2020. Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390. https://arxiv.org/pdf/2007.14390

[Botvinick2019] Botvinick, M., Ritter, S., Wang, J.X., Kurth-Nelson, Z., Blundell, C. and Hassabis, D., 2019. Reinforcement learning, fast and slow. Trends in cognitive sciences, 23(5), pp.408-422. https://doi.org/10.1016/j.tics.2019.02.006

[Cespedes2022] Céspedes Sisniega, Jaime, and López García, Álvaro. "Frouros: A Python library for drift detection in Machine Learning problems." *arXiv preprint arXiv:2208.06868* (2022). https://doi.org/10.48550/arXiv.2208.06868

[Dahl2018] Dahl, M., Mancuso, J., Dupis, Y., Decoste, B., Giraud, M., Livingstone, I., Patriquin, J. and Uhma, G., 2018. Private machine learning in Tensorflow using secure computation. Privacy Preserving Machine Learning, NeurIPS 2018 Workshop, Montréal, December 8. https://ppml-workshop.github.io/ppml18/slides/56.pdf

[Dargan2020] Dargan, S., Kumar, M., Ayyagari, M.R. et al. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. Arch Computat Methods Eng 27, 1071–1092 (2020). https://doi.org/10.1007/s11831-019-09344-w

[Dong2020] Dong, X., Yu, Z., Cao, W., Shi, Y. and Ma, Q., 2020. A survey on ensemble learning. Frontiers of Computer Science, 14, pp.241-258. https://doi.org/10.1007/s11704-019-8208-z

[Dong2021] Dong, Shi, Ping Wang, and Khushnood Abbas. "A survey on deep learning and its applications." Computer Science Review 40 (2021). https://doi.org/10.1016/j.cosrev.2021.100379

[Dua2016] Dua, S. and Du, X., 2016. Data mining and machine learning in cybersecurity. https://www.amazon.com/Data-Mining-Machine-Learning-Cybersecurity/dp/1439839425 . CRC press. ISBN 978-1439839423.

[Dwork2014] Dwork, C. and Roth, A., 2014. The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science, 9(3–4), pp.211-407. https://doi.org/10.1561/0400000042

[Emmert2022] Emmert-Streib, F. and Dehmer, M., 2022. Taxonomy of machine learning paradigms: A data-centric perspective. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 12(5), p.e1470. https://doi.org/10.1002/widm.1470

[Geng2009] Geng, X., Smith-Miles, K. (2009). Incremental Learning. In: Li, S.Z., Jain, A. (eds) Encyclopedia of Biometrics. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-73003-5_304

[Gentry2009] Gentry, C., 2009. A fully homomorphic encryption scheme. https://www.proquest.com/docview/305003863?pq-origsite=gscholar&fromopenview=true. Stanford university.

[Hewage2022] Hewage, N. and Meedeniya, D., 2022. Machine Learning Operations: A Survey on MLOps Tool Support. arXiv preprint arXiv:2202.10169. https://arxiv.org/abs/2202.10169

[Heg2019] Hegedűs, I., Danner, G. and Jelasity, M., 2019, June. Gossip learning as a decentralized alternative to federated learning. In IFIP International Conference on Distributed Applications and Interoperable Systems (pp. 74-90). Lecture Notes in Computer Science, Vol. 11534, ISBN 978-3-030-22495-0. Springer, Cham. https://doi.org/10.1007/978-3-030-22496-7_5

[Lheureux2017] L'heureux, A., Grolinger, K., Elyamany, H.F. and Capretz, M.A., 2017. Machine learning with Big data: Challenges and approaches. IEEE Access, 5, pp.7776-7797. https://doi.org/10.1109/ACCESS.2017.2696365

[Li2020] Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A. and Smith, V., 2020. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems, 2, pp.429-450. https://proceedings.mlsys.org/papers/2020/176

[Ludwig2020] Ludwig, Heiko, et al. "Ibm federated learning: an enterprise framework white paper v0. 1." *arXiv preprint arXiv:2007.10987* (2020). https://arxiv.org/abs/2007.10987

[Mah2017] McMahan, B., Moore, E., Ramage, D., Hampson, S. and y Arcas, B.A., 2017, April. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). PMLR, 2017, pp. 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html .

[Ngu2019] Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., López García, Á., Heredia, I., Malík, P. and Hluchý, L.,: Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. Artificial Intelligence Review, Volume 52, Issue 1, pp. 77-124, ISSN 0269-2821, DOI 10.1007/s10462-018-09679-z. Springer Nature, 2019, CC BY 4.0. Springer Nature research highlights in Computer Science 2019. https://link.springer.com/article/10.1007%2Fs10462-018-09679-z

[Ngu2022] Nguyen, G.: Introduction to Data Science. The Edition of University Textbooks on Informatics and Information Technologies, 2022, Spektrum STU Publishing, ISBN 978-80-227-5193-3. (STU AIS access). https://elvira.fiit.stuba.sk/library/pdf-viewer/765297eb-be49-4013-857e-ad3438726942

[OL2023] Lytvyn, O., Nguyen, G.: Secure Multi-Party Computation for Magnetic Resonance Imaging Classification. 14th International Conference on Ambient Systems, Networks and Technologies (ANT 2023). Procedia Computer Science, ISSN 1877-0509, *accepted paper*.

[Reddi2020] Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S. and McMahan, H.B., 2020. Adaptive federated optimization. arXiv preprint arXiv:2003.00295. https://arxiv.org/abs/2003.00295

[Rodriguez2020] Rodríguez-Barroso, Nuria, et al. "Federated learning and differential privacy: Software tools analysis, the sherpa. ai fl framework and methodological guidelines for preserving data privacy." *Information Fusion* 64 (2020): 270-292. https://doi.org/10.1016/j.inffus.2020.07.009

[SAFL2021] Kwing Hei Li, Pedro Porto Buarque de Gusmão, Daniel J. Beutel, and Nicholas D. Lane. 2021. Secure aggregation for federated learning in flower. In Proceedings of the 2nd ACM International Workshop on Distributed Machine Learning (DistributedML '21). Association for Computing Machinery, New York, NY, USA, 8–14. https://doi.org/10.1145/3488659.3493776

[Sainzpardo2023] Sáinz-Pardo Díaz, Judith, and López García, Álvaro. Study of the performance and scalability of federated learning for medical imaging with intermittent clients. Neurocomputing. Volume 518, pp. 142-154, ISSN 0925-2312, DOI 10.1016/j.neucom.2022.11.011, 2023, Elsevier, CC BY 4.0. https://www.sciencedirect.com/science/article/pii/S0925231222013844

[Shearer2000] Shearer, Colin. The CRISP-DM model: the new blueprint for data mining. Journal of data warehousing, 2000, vol. 5, no 4, p. 13-22.

[Shoham2019] Shoham, N., Avidor, T., Keren, A., Israel, N., Benditkis, D., Mor-Yosef, L. and Zeitak, I., 2019. Overcoming forgetting in federated learning on non-iid data. arXiv preprint arXiv:1910.07796. https://arxiv.org/abs/1910.07796

[Shokri2015] Shokri, R. and Shmatikov, V., 2015, October. Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (pp. 1310-1321). https://doi.org/10.1145/2810103.2813687

[Soykan2022] Soykan, E.U., Karaçay, L., Karakoç, F. and Tomur, E., 2022. A Survey and Guideline on Privacy Enhancing Technologies for Collaborative Machine Learning. IEEE Access, 10, pp.97495-97519. DOI: 10.1109/ACCESS.2022.3204037. https://ieeexplore.ieee.org/abstract/document/9875277

[SuW2022] Su, W., Li, L., Liu, F., He, M. and Liang, X., 2022. AI on the edge: a comprehensive review. Artificial Intelligence Review, pp.1-59, ISSN 0269-2821, DOI 10.1007/s10462-022-10141-4. Springer Nature, 2022. https://doi.org/10.1007/s10462-022-10141-4

[Verbraeken2020] Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T. and Rellermeyer, J.S., 2020. A survey on distributed machine learning. Acm computing surveys (csur), 53(2), pp.1-33. https://doi.org/10.1145/3377454

[Zeydan2022] Zeydan, E. and Mangues-Bafalluy, J., 2022. Recent Advances in Data Engineering for Networking. IEEE Access. https://doi.org/10.1109/ACCESS.2022.3162863

Links

[Agi2022] What is Agile Software Development? https://www.agilealliance.org/agile101/. Accessed 05.11.2022.

[AgilePF] The DSDM Agile Project Framework. https://www.agilebusiness.org/dsdm-project-framework.html. Accessed 28.12.2022

[AI-Infra2022] AI Infrastructure Landscape, 2022 https://ai-infrastructure.org/ai-infrastructure-landscape/ . Accessed 17.11.2022.

[Airflow] Apache Airflow - A platform to programmatically author, schedule, and monitor workflows. https://github.com/apache/airflow . Accessed 07.12.2022.

[Airflow-Prefect-Dagster] Navid P.: Airflow, Prefect, and Dagster: An Inside Look. 2022. https://towardsdatascience.com/airflow-prefect-and-dagster-an-inside-look-6074781c9b77 . Accessed 06.01.2023.

[APPI] Japan's data protection law, the Act on the Protection of Personal Information, 2019 https://www.ppc.go.jp/files/pdf/Act_on_the_Protection_of_Personal_Information.pdf . Accessed 11.11.2022.

[Bha2022] S. Bhanda, R. Badam: Comprehensive List of DevOps Tools 2022. https://www.qentelli.com/thought-leadership/insights/devops-tools . Accessed 5.11.2022.

[Caffe2] Caffe2 is a lightweight, modular, and scalable deep learning framework. https://github.com/facebookarchive/caffe2 . Source code now lives in the PyTorch repository https://github.com/pytorch/pytorch/ Accessed 20.11.2022.

[CCPA] California Consumer Privacy Act. 2018. https://oag.ca.gov/privacy/ccpa . Accessed 11.11.2022.

[CNTK] CNTK - Microsoft Cognitive Toolkit (CNTK), an open source deep-learning toolkit. https://github.com/microsoft/CNTK . Accessed 20.11.2022.

[Chainer] Chainer - A flexible framework of neural networks for deep learning. https://github.com/chainer/chainer . Accessed 23.11.2022.

[ClearML] ClearML - Auto-Magical CI/CD to streamline your ML workflow. Experiment Manager, MLOps and Data-Management. https://github.com/allegroai/clearml/ . Accessed 26.12.2022.

[ColossalAI] A Unified Deep Learning System for Big Model Era. https://github.com/hpcaitech/ColossalAI . Accessed 23.11.2022.

[CompositeAI2020] Composite AI in Action with ELEMENT. https://blackswantechnologies.ai/media-room/composite-ai/. Accessed 23.11.2022.

[CompositeAI2023] What is Composite AI & Why is it Important in 2023? https://research.aimultiple.com/composite-ai/. Accessed 23.11.2022.

[CVAT] Computer Vision Annotation Tool - an interactive video and image annotation tool for computer vision. https://github.com/opencv/cvat . Accessed 25.12.2022.

[Dagster] An orchestration platform for the development, production, and observation of data assets. https://github.com/dagster-io/dagster . Accessed 28.12.2022.

[DataCamp2022] 17 Top MLOps Tools You Need to Know. 2022 December. https://www.datacamp.com/blog/top-mlops-tools . Accessed 03.02.2023.

[DEEP-Platform] The DEEP Hybrid Datacloud Platform. https://deep-hybrid-datacloud.eu/the-platform/. Accessed 28.12.2022

[Diffprivlib] https://github.com/IBM/differential-privacy-library . Accessed 5.11.2022.

[DistLearn] Distributed training with Azure Machine Learning, 10.2022. https://learn.microsoft.com/en-us/azure/machine-learning/concept-distributed-trainin. Accessed 07.12.2022.

[DL4J] deeplearning4j (DL4J) Suite of tools for deploying and training deep learning models using the JVM. https://github.com/deeplearning4j/deeplearning4j . Accessed 23.11.2022.

[DVC] Data Version Control (DVC) Data Version Control | Git for Data & Models | ML Experiments Management - command line tool and VS Code Extension. https://github.com/iterative/dvc . Accessed 26.01.2023.

[EmergingTech2020] What's New In Gartner's Hype Cycle For Emerging Technologies, 2020 https://www.forbes.com/sites/louiscolumbus/2020/08/23/whats-new-in-gartners-hype-cycle-for-emerging-technologies-2020/?sh=19bd9b7fa46a. Accessed 31.01.2023

[EnsembleSKL] Ensemble methods, scikit-learn library. https://scikit-learn.org/stable/modules/ensemble.html. Accessed 23.11.2022.

[EUguidelines] European Commission. EU guidelines on ethics in artificial intelligence: Context and implementation. 2019. URL:

https://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_BRI(2019)640163 . Accessed 23.11.2022.

[fast.ai] The fastai deep learning library. https://github.com/fastai/fastai . Accessed 23.11.2022.

[Fate] FATE (Federated AI Technology Enabler). https://github.com/FederatedAI/FATE . Accessed 5.11.2022.

[FedML] FedML, Federated Learning/Analytics and Edge AI Platform. https://fedml.ai/. Accessed 11.11.2022.

[FLAM] FLAM a fast library for AutoML and tuning. https://github.com/microsoft/FLAML . Accessed 28.01.2023.

[Flare] NVIDIA Federated Learning Application Runtime Environment. https://github.com/NVIDIA/NVFlare . Accessed 24.11.2022.

[Flower] Flower - A Friendly Federated Learning Framework https://github.com/adap/flower. Accessed 5.11.2022.

[Flyte] Flyte - Kubernetes-native workflow automation platform for complex, mission-critical data and ML processes at scale. https://github.com/flyteorg/flyte . Accessed 25.12.2022.

[Frameworks2022] Papers with code. Trends on the paper implementations grouped by framework. https://paperswithcode.com/trends. Accessed 14.11.2022.

[GDPR] General Data Protection Regulation. 2016. https://eur-lex.europa.eu/eli/reg/2016/679/oj . Accessed 14.11.2022.

[GoogleMLOps2020] Cloud Architecture Center - MLOps: Continuous delivery and automation pipelines in machine learning. https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning , 2020. Accessed 27.12.2022.

[GoogleUnifying2022] Kazmaier G.: Building the most open data cloud ecosystem: Unifying data across multiple sources and platforms. https://cloud.google.com/blog/products/data-analytics/building-most-open-data-cloud-all-data-all-source-any-platform, October 11, 2022. Accessed 7.12.2022.

[Governance2022] Treveil M., Heidmann L.: What Is MLOps? Chapter 4. Governance. November 2020, Publisher(s): O'Reilly Media, Inc., ISBN 9781492093619. https://www.oreilly.com/library/view/what-is-mlops/9781492093626/ch04.html Accessed 15.12.2022.

[H2O] H2O is an Open Source, Distributed, Fast & Scalable Machine Learning Platform. https://github.com/h2oai/h2o-3 . Accessed 3.12.2022.

[HElib] Open-source software library that implements homomorphic encryption (HE). https://github.com/homenc/HElib . Accessed 21.11.2022.

Funded by
the European Union

[IBM2021] Fully Homomorphic Encryption on IBM Cloud Hyper Protect Virtual Servers, https://www.ibm.com/cloud/blog/fully-homomorphic-encryption-on-ibm-cloud-hyper-protect-virtual-servers . Accessed 6.11.2022.

[IBM-FL] IBM Federated Learning, https://github.com/IBM/federated-learning-lib . Accessed 11.11.2022

[JAX] JAX: Autograd and XLA https://github.com/google/jax . Accessed 24.01.2023.

[KDNuggets-MDV] YaramS.: Machine Learning Model Development and Model Operations: Principles and Practices, 10.2021. https://www.kdnuggets.com/2021/10/machine-learning-model-development-operations-principles-practice.html . Accessed 07.12.2022.

[Keras] Keras - Deep Learning for humans. https://github.com/keras-team/keras . Accessed 20.11.2022.

[KotlinSyft] OpenMined/KotlinSyft - The official Syft worker for secure on-device machine learning. https://github.com/OpenMined/KotlinSyft . Accessed 17.11.2022.

[Kubeflow] Machine Learning Toolkit for Kubernetes https://github.com/kubeflow/kubeflow . Accessed 15.12.2022.

[Metaflow] Build and manage real-life data science projects with ease! https://github.com/Netflix/metaflow . Accessed 26.12.2022.

[MindSpore] MindSpore is a new open source deep learning training/inference framework that could be used for mobile, edge and cloud scenarios. https://github.com/mindspore-ai/mindspore . Accessed 17.11.2022.

[MLeap] Deploy ML Pipelines to Production. https://github.com/combust/mleap . Accessed 26.12.2022.

[MLflow] MLflow: A Machine Learning Lifecycle Platform, https://github.com/mlflow/mlflow/ . Accessed 15.12.2022.

[MLRun] Machine Learning automation and tracking. https://github.com/mlrun/mlrun . Accessed 26.12.2022.

[MS-SEAL] Microsoft SEAL is an easy-to-use and powerful homomorphic encryption library. https://github.com/microsoft/SEAL . Accessed 17.11.2022.

[MTu2020] Resilience and Vibrancy: The 2020 Data & AI Landscape, https://mattturck.com/data2020/ . Accessed 11.11.2022.

[MTu2021] Red Hot: The 2021 Machine Learning, AI and Data (MAD) Landscape, https://mattturck.com/data2021/ . Accessed 11.11.2022.

[MTu2022] Matt Turck at venture capital firm FirstMark with early-stage enterprise and B2B investing, https://firstmarkcap.com/team/matt-turck/ . Accessed 11.11.2022.

[MXNet] MXNet - Lightweight, Portable, Flexible Distributed/Mobile Deep Learning with Dynamic, Mutation-aware Dataflow Dep Scheduler; for Python, R, Julia, Scala, Go, Javascript and more. https://github.com/apache/mxnet . Accessed 20.11.2022.

[NeptuneAI2022-Menzli] Menzli A.: Best End-to-End MLOps Platforms: Leading Machine Learning Platforms. https://neptune.ai/blog/end-to-end-mlops-platforms . Accessed 28.12.2022.

[NeptuneAI2022] Czakon, J.: The Best MLOps Tools and How to Evaluate Them, 01.12.2022. https://neptune.ai/blog/best-mlops-tools . Accessed 02.12.2022.

[NNI] Neural Network Intelligence - An open source AutoML toolkit for automate machine learning lifecycle, including feature engineering, neural architecture search, model compression and hyper-parameter tuning.https://github.com/microsoft/nni . Accessed 15.12.2022.

[NNIreview] Garvin Li: NNI review article from Zhihu. https://nni.readthedocs.io/zh/stable/sharings/nni_autofeatureeng.html . Accessed 06.01.2023.

[LF-AI-Data-Landscape] LF AI & Data Foundation Interactive Landscape, 2023. https://landscape.lfai.foundation/ . Accessed 12.01.2023.

[Luigi] Luigi - Python module to build complex pipelines of batch jobs. https://github.com/spotify/luigi . Accessed 07.12.2022.

[Opacus] Opacus a library that enables training PyTorch models with differential privacy https://github.com/pytorch/opacus . Accessed 5.11.2022.

[OpenFHE] Open-Source Fully Homomorphic Encryption Library https://github.com/openfheorg/openfhe-development . Accessed 24.01.2023.

[OpenFL] Open Federated Learning (OpenFL) - An Open-Source Framework For Federated Learning. https://github.com/intel/openfl . Accessed 23.11.2022.

[OpenMined] OpenMined - A WORLD WHERE EVERY GOOD QUESTION IS ANSWERED. https://www.openmined.org/ . Accessed 23.11.2022.

[OpenMinedDP] Open Mined - Use Cases of Differential Privacy https://blog.openmined.org/use-cases-of-differential-privacy/. Accessed 05.12.2022.

[orchest] Build data pipelines, the easy way https://github.com/orchest/orchest . Accessed 26.01.2026.

[PaddlePaddle] PArallel Distributed Deep LEarning: Machine Learning Framework from Industrial Practice https://github.com/PaddlePaddle/Paddle . Accessed 24.01.2023.

[Pachyderm] Data-Centric Pipelines and Data Versioning. https://github.com/pachyderm/pachyderm . Accessed 28.12.2022.

[Paillier] Paillier - A library for Partially Homomorphic Encryption in Python. https://github.com/data61/python-paillier . Accessed 21.11.2022.

[Parity-Ethereum] Parity-Ethereum - The Fastest and most Advanced Ethereum Client. https://github.com/openethereum/parity-ethereum . Accessed 25.12.2022.

[Polkadot] Polkadot - the multichain vision for Web3, secures a growing ecosystem of specialized blockchains. https://polkadot.network/ . https://github.com/paritytech/polkadot . Accessed 25.12.2022.

[Polyaxon] MLOps Tools For Managing & Orchestrating The Machine Learning LifeCycle. https://github.com/polyaxon/polyaxon . Accessed 26.12.2022.

[Prefect2] Prefect2 - The easiest way to coordinate your dataflow. https://github.com/PrefectHQ/prefect . Accessed 15.12.2022.

[Prefect-vs-Airflow] Why Not Airflow? https://docs-v1.prefect.io/core/about_prefect/why-not-airflow.html#overview . Accessed 06.01.2023.

[PyDP] PyDP - The Python Differential Privacy Library. https://github.com/OpenMined/PyDP . Accessed 22.11.2022.

[PySyft] OpenMined/PySyft - Data science on data without acquiring a copy. https://github.com/OpenMined/PySyft . Accessed 17.11.2022.

[PySyftTF] PySyft - Tensorflow. https://github.com/OpenMined/PySyft-TensorFlow. Accessed 14.11.2022.

[PyTorch] PyTorch - Tensors and Dynamic neural networks in Python with strong GPU acceleration. https://github.com/pytorch/pytorch . Accessed 20.11.2022.

[RockeScience2022] A review of +250 MLOps tools and solutions. https://rocketscience.one/mlops-software-review/ . Accessed 06.01.2023.

[Rosetta] LatticeX-Foundation/Rosetta - A Privacy-Preserving Framework Based on TensorFlow. https://github.com/LatticeX-Foundation/Rosetta . Accessed 22.11.2022.

[RLOptim] Reinforcement Learning and Optimization (Knowledge Engineering Institute, Autonomous University of Madrid) https://www.iic.uam.es/en/artificial-intelligence/reinforcement-learning/. Accessed 23.11.2022.

[Scikit-Learn] scikit-learn: machine learning in Python https://github.com/scikit-learn/scikit-learn . Accessed 20.11.2022.

[SeldonCore] Seldon Core: Blazing Fast, Industry-Ready ML. https://github.com/SeldonIO/seldon-core . Accessed 26.12.2022.

[SplitLearningMIT] MIT Media Lab's Split Learning: Distributed and collaborative learning - Distributed deep learning and inference without sharing raw data. https://splitlearning.mit.edu/. Accessed 5.11.2022.

[Syft.js] OpenMined/Syft.js - The official Syft worker for Web and Node, built in Javascript. https://github.com/OpenMined/syft.js/ . Accessed 17.11.2022.

[SyMPC] A SMPC companion library for Syft. https://github.com/OpenMined/SyMPC . Accessed 17.11.2022.

[SwiftSyft] OpenMined/SwiftSyft - The official Syft worker for iOS, built in Swift. https://github.com/OpenMined/SwiftSyft . Accessed 17.11.2022.

[TenSEAL] OpenMined/TenSEAL - A library for doing homomorphic encryption operations on tensors. https://github.com/OpenMined/TenSEAL . Accessed 17.11.2022.

[Tensorflow] Tensorflow - An Open Source Machine Learning Framework for Everyone. https://github.com/tensorflow/tensorflow . Accessed 20.11.2022.

[TF-Encrypted] Encrypted Deep Learning in Tensorflow https://tf-encrypted.io/ https://github.com/tf-encrypted/tf-encrypted . Accessed 19.11.2022.

[TF-Federated] Tensorflow Federated: Machine Learning on Decentralized Data https://www.tensorflow.org/federated https://github.com/tensorflow/federated . Accessed 19.11.2022.

[TF-Privacy] Library for training machine learning models with privacy for training data. https://github.com/tensorflow/privacy . Accessed 19.11.2022.

[TF-Trusted] capeprivacy/tf-trusted to run most Tensorflow models inside of an Intel SGX device. https://github.com/capeprivacy/tf-trusted . Accessed 22.11.2022.

[TFX] Tensorflow Extended (TFX) - end-to-end platform for deploying production ML pipelines. https://github.com/tensorflow/tfx . Accessed 22.11.2022.

[Thoughtworks2021] Dawson R., et al.: Guide to Evaluating MLOps Platforms. 2021. https://www.thoughtworks.com/what-we-do/data-and-ai/cd4ml/guide-to-evaluating-mlops-platforms . Accessed 28.12.2022.

[TPOT] TPOT - Tree-based Pipeline Optimization Tool. Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming. https://github.com/EpistasisLab/tpot . Accessed 28.01.2023.

[VentureBeat2022] Why do 87% of data science projects never make it into production? https://venturebeat.com/ai/why-do-87-of-data-science-projects-never-make-it-into-production/ . Accessed 5.11.2022.

[WMR2021] Workflow Management Review: Airflow vs. Luigi, 2021. https://www.upsolver.com/blog/workflow-management-review-airflow-vs-luigo. . Accessed 07.12.2022.

[XGBoost] XGBoost - Scalable, Portable and Distributed Gradient Boosting (GBDT, GBRT or GBM) Library, for Python, R, Java, Scala, C++ and more. Runs on single machine, Hadoop, Spark, Dask, Flink and DataFlow. https://github.com/dmlc/xgboost . Accessed 20.11.2022.

[ZenML] ZenML - Build portable, production-ready MLOps pipelines. https://zenml.io. Accessed 26.01.2022.