SEcure Decentralised Intelligent Data MARKetplace

D4.3 Edge data processing and service certification – First version

Document Identification	
Contractual delivery date:	31/12/2023
Actual delivery date:	19/01/2023
Responsible beneficiary:	EGM
Contributing beneficiaries:	UCD, SURREY
Dissemination level:	PU
Version:	1.0
Status:	Final

Keywords:
Tools, Data processing

This document is issued within the frame and for the purpose of the SEDIMARK project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No.101070074. and is also partly funded by UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission or UKRI.

Document Information

Document Identification			
Related WP	WP4	**Related Deliverables(s):**	D3.1
Document reference:	SEDIMARK_D4.3	**Total number of pages:**	22

List of Contributors	
Name	**Partner**
Gilles Orazi Franck Le Gall Léa Robert Iheb Khelifi	EGM
Diarmuid O'Reilly Morgan Erika Duriakova Honghui Du Elias Tragos Qinqin Wang Aonghus Lawlor Neil Hurley	UCD
Tarek Elsaleh	SURREY

Document History			
Version	**Date**	**Change editors**	**Change**
0.1	26/09/2023	Gilles Orazi Franck Le Gall Léa Robert (EGM)	First version of document structure (table of content)
0.2	11/10/2023	UCD SIE	Introduction Subsection related to MLOps
0.3	15/11/2023	SURREY	Subsection related to Certification Services

Document History			
Version	**Date**	**Change editors**	**Change**
0.4	26/12/2023	EGM	Overall integration, update following reviews and cleaning
0.5	26/12/2023	EGM	Generate clean (no track change) version from v0.4
0.6	28/12/2023	ATOS	First Quality review
0.7	16/01/2024	EGM	Quality review comments corrected
0.8	17/01/2024	ATOS	Second Quality review
1.0	19/01/2024	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval date**
Reviewer 2	Tarek Elsaleh (SURREY)	22.12.2023
Reviewer 1	Arturo Medela (ATOS)	15.12.2023
Quality manager	María Guadalupe Rodríguez (ATOS)	18.01.2024
Project Coordinator	Arturo Medela (ATOS)	19.01.2024

Table of Contents

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under ROC curve
CPU	Central processing unit
DAG	Directed Acyclic Graph
Dx.y	Deliverable number y belonging to WP x
IoT	Internet of Things
IT	Internet Technologies
KPI	Key Performance Indicator
MAE	Mean Absolute Error
MCU	MicroController Unit
ML	Machine Learning
MLOps	Machine Learning Operations
NiFi	NiagaraFiles (Apache Software Foundation)
NGSI-LD	Next Generation Service Interface – Linked Data
NIST	National Institute of Standards and Technology
ONNX	Open Neural Network Exchange
QoS	Quality of Service
RDF	Resource Description Framework
RDFS	RDF Schema
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
SHACL	Shapes Constraint Language
SQL	Structured Query Language
UI	User Interface
W3C	World Wide Web Consortium
WP	Work Package

Abbreviation / acronym	Description
Wasm	WebAssembly

Executive Summary

This deliverable is a first version of the deliverable named “Edge data processing and service certification”.

The work has focused on identifying the building blocks required to develop a framework for deploying AI-based data processing and sharing modules at edge data sources, considering edge-cloud interactions following MLOps principles. It involves analysing various ML frameworks such as TensorFlow, PyTorch, tinyML, and edgeML, while addressing security and privacy concerns by implementing adaptive edge anonymization or tagging of sensitive data.

At this stage of the project these topics are still quite exploratory; this report focuses thus more on challenges, considered options and possible implementation choices. The last version of this deliverable, which is due M34 (July 2025), will provide all the details about the technical choices and their implementation.

After explaining the place of this work in the big picture of SEDIMARK (chapter 2), the core of the report is divided into two chapters:

- Chapter 3: explains the challenges and requirements for edge processing, with a special emphasis on the management of the Edge/Cloud interactions, especially by looking through various possible orchestration tools.
- Chapter 4: give insights on the role of MLOps, look through ML and MLOps frameworks and certification services.

1 Introduction

1.1 Purpose of the document

This document is the first version of a reporting of the work in WP4 about tools and processes for enabling data processing and sharing in an interoperable way at the data sources.

1.2 Relation to another project work

The work in WP2 “Requirements, architecture and interfaces”, reported in SEDIMARK_D2.1 [1] and SEDIMARK_D2.2 [2] so far, showed that the topics of edge computing for data quality, ML models and MLOps especially linked to federated learning are of special interest in this project. This has driven the work of T4.2 “Edge data processing and sharing”, reported in this deliverable.

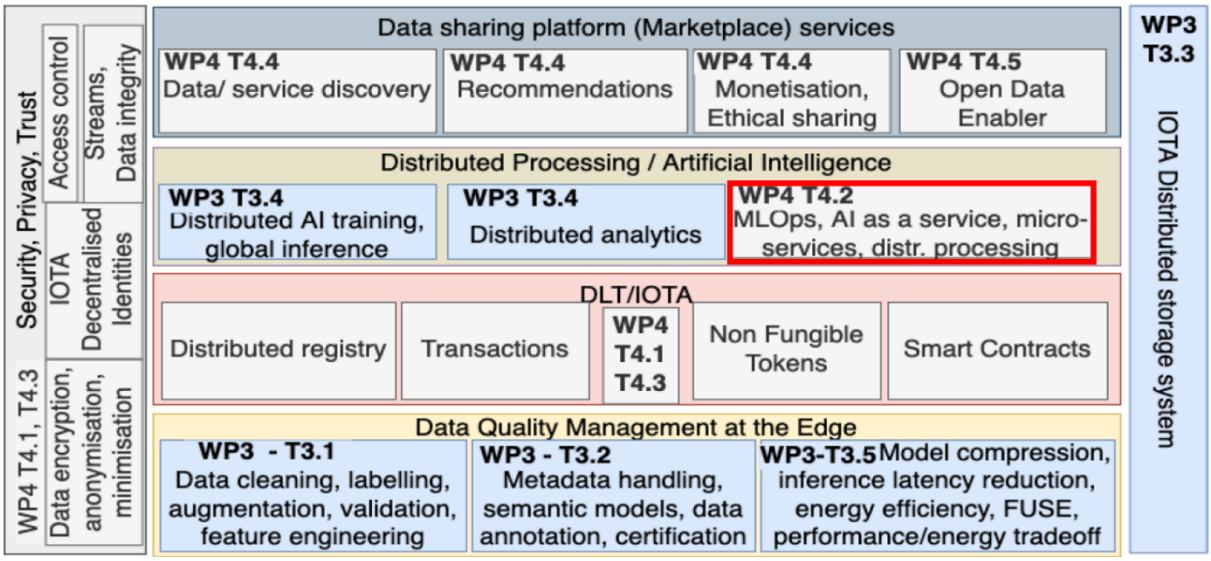

Figure 1: Positioning of distributed processing in SEDIMARK architecture

As one can see in the Figure 1, this task is at the heart of the SEDIMARK platform, in the layer dedicated to distributed processing and artificial intelligence. Its primary objective is to create a framework for the deployment of AI-driven modules that process and share data at edge data sources. It considers the interactions between edge and cloud systems while adhering to MLOps principles. It is thus in tight relationship with WP3 “Distributed data quality management and interoperability”.

1.3 Structure of the document

After a short introduction to the document (the current part) and a quick overview of the SEDIMARK platform (chapter 2), this document explores two aspects of the management of these data.

In chapter 3, the aspects related to the Edge/Cloud interactions are explored by first exposing the challenges and requirements for Edge computing (3.2) and then by looking at how they can be managed by using specific orchestration tools (3.3.1), or how dynamic processing can be implemented even at far edge (3.3.2), where networking and computing resources are very limited.

2 Report contributions to SEDIMARK environment

This deliverable presents the first draft of the work in Task 4.2 “Edge data processing and sharing”, presenting the first ideas about the architecture that will be developed within SEDIMARK to enable data processing and sharing at the edge. SEDIMARK is a decentralised system that will allow data providers to collect, clean and process their data at various stages, including at edge devices. This will be helpful in cases where large amounts of data are gathered at edge devices, so that once they are cleaned and processed, the communication and storage costs at the provider server is significantly reduced. Additionally, clean data at the edge will allow a more efficient machine learning techniques, both for training and for inference purposes. Techniques for data anonymisation at the edge will also be exploited to hide or remove sensitive information, thus either creating anonymised datasets that can be shared in the marketplace without privacy issues or training ML models that don’t reveal or leak private data.

Considering that there are many available frameworks used for training ML models, SEDIMARK also aims to provide a framework for ML model interoperability, exploiting existing well-known platforms. This will help providers to continue to use the frameworks they are familiar with, while at the same time they will be able to download/purchase models from the SEDIMARK marketplace and use them converting them into their preferred format/framework.

Figure 1 presents the SEDIMARK functional architecture that was described in deliverable SEDIMARK_D2.2 in detail. With orange highlights are the functional components that are part of this deliverable. These components are part of three different layers of SEDIMARK, security, data and intelligence layer. More details are given below:

- Data validation/certification: this component is described in Section 4.5 and is related with certifying that the data and ML models conform to the SEDIMARK ontology.
- AI model formatting: this component is described in section 4.4 and is related with enabling retraining and inference of ML models across different platforms.
- AI orchestrator: this component is described in sections 4.2 and 4.3, providing an overview of the frameworks used for training ML models, how the AI pipeline is orchestrated for enabling inference at the edge and how MLOps frameworks used within SEDIMARK help to easily convert trained ML models to deployment.
- Data processing dashboard: this is described in section 3.3, describing orchestration tools for data processing that will enable edge-cloud interactions.

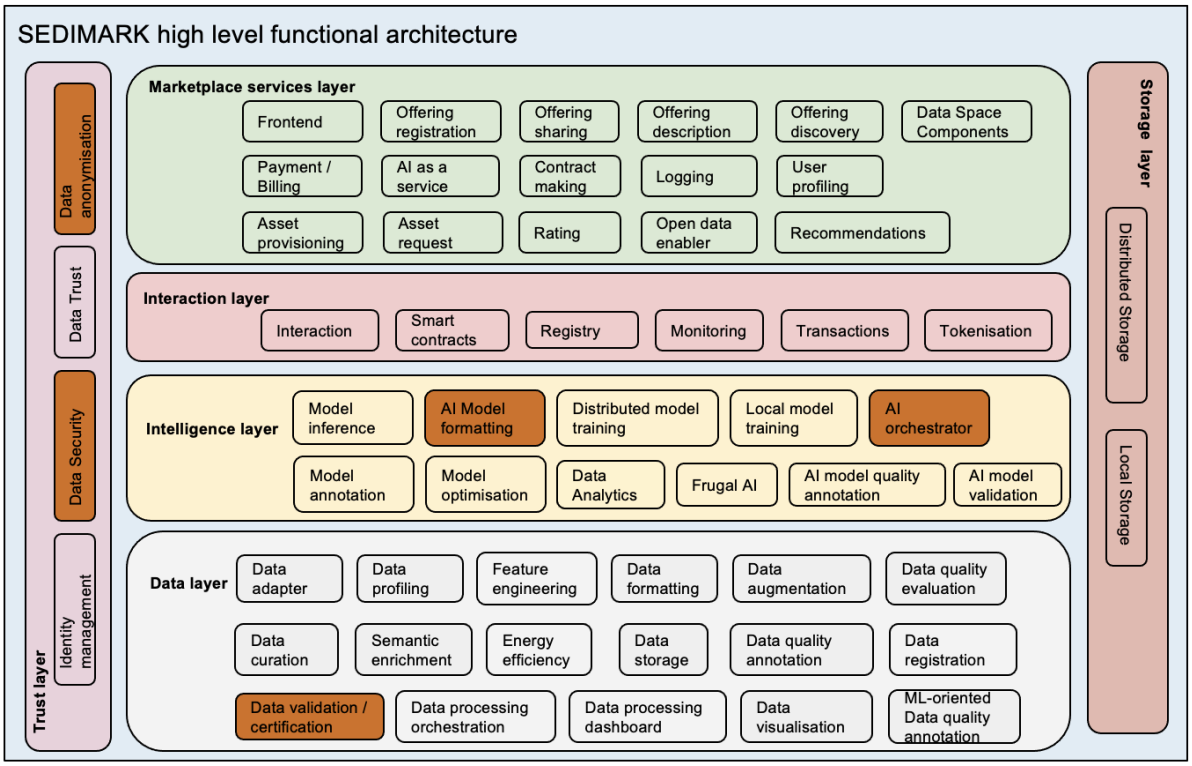

Figure 2: The SEDIMARK functional architecture. Orange highlights functional components that are being part of this deliverable.

3 Architecture for Edge data processing

3.1 Introduction

Edge processing, also known as edge computing, refers to the practice of processing data near the source of generation rather than relying on a centralized cloud-based system. In traditional computing models, data is sent to a remote datacentre or cloud for processing and analysis. However, edge processing brings computational capabilities closer to the "edge" of the network, which is typically where data is generated. In the context of SEDIMARK, this paradigm is in use for distributed AI as well as for potential delocalization of some of the data processing pipeline processes.

The environment in which Edge data processing is performed raises multiple constraints that need to be handled, such as privacy preservation, response time, throughput, and resource consumption (e.g., CPU, memory, energy, bandwidth), while the latter may influence the monetary cost. In the following, some of these requirements are to be considered for the SEDIMARK assets (e.g., Artificial intelligence (AI) model and service assets).

- Bandwidth: While edge data processing reduces the need for transmitting all data to the cloud (federated learning) or between nodes (gossip learning), there is still a need for network connectivity. Limited bandwidth can affect data and synopsis transmission to and from the edge.
- Computing resources: Edge devices often have constrained processing capabilities (e.g., memory and storage). Therefore, running complex and massive data processing tasks on such devices can be challenging. One of the envisaged solutions within SEDIMARK to cope with this issue is to use sampling and efficient data processing methods.
- Privacy: In the SEDIMARK decentralized environment, privacy naturally arises since personal and sensitive data will be processed, from which real insights about individual behaviour, health, or relationships can be inferred.
- Data quality: The data provided within SEDIMARK can be noisy, duplicated, or incomplete. Ensuring data quality and extracting knowledge from potentially imperfect data is a challenge that needs to be handled. To do so, SEDIMARK will provide curation techniques to address imperfect data and improve its quality.

These aspects depend on the framework used to handle the processing distribution as well as the way processes are implemented. In this report, focus is on the tooling which is investigated in the following section.

3.2 Edge-Cloud Orchestration tools

There exist many edge-cloud orchestration platforms. Identified open-source platforms have been analysed to evaluate how they could support the handling of a data pipeline distributed over cloud and edge. They are the following:

- FogFlow: FogFlow is a FIWARE enabler to orchestrate data processing flows between cloud and edge. It uses intent based programming. For example, for service consumers, they can specify which type of results are expected under which type of Quality of Service (QoS) within which geo-scope; for data providers, they can specify how their data should be utilized by whom. In FogFlow, orchestration decisions are made to meet those user-definable objectives during the runtime. It became NGSI-LD compliant in September 2022, but no further updates have been observed in the roadmap since then.

- Apache Airflow: Apache Airflow is a platform to programmatically author, schedule and monitor workflows. It uses Python language to batch processing workflow run at regular intervals. It does not aim at processing event streams. However, coupling with a service bus having storage capabilities such as Apache Kafka allows for periodic processing of stream fragments. Airflow interface is mainly provided for workflows activation and monitoring. Coding in python remain mandatory for workflows definition. It builds on Kubernetes to provide auto-scaling.
- Mage.ai: Mage.ai aims at simplifying the Apache Airflow experience. It remains coding based, allowing Python, R and SQL in the same data pipeline while the User Interface (UI) focuses on monitoring workflows execution. Both batch and stream processing are allowed. Pipelines can be configured through the set of global variables. Distributed processing is part of the roadmap, considering Ray as a distributed execution framework layer for parallel processing and Dask as Python parallel computing library.
- Apache NiFi: Apache NiFi also aims at implementing workflow defined as DAG. However, in contrast to Airflow, it provides aa highly configurable web-based interface to define the workflow which can consider either stream or batch processing. Hundreds of existing connectors enable the ingesting of data from almost any kind of source. External scripts or executables can be called thus making Apache NiFi completely customizable.
- MiNiFi: Apache MiNiFi is a sub project of Apache Apache NiFi meant to collect data and process data on the edge. Java (heavier) and C++ (lighter) flavors are provided. Both are however too large to be executed on a low power, microcontroller based far edge device.
- StarlingX: StarlingX is an edge cloud infrastructure targeting security, ultra-low latency, and extremely high service uptime which are requirements from the industrial Internet of Things (IoT). The underlying hardware layer is expected to run Yocto Linux, whereas scalability and orchestration are managed by Kubernetes and OpenStack frameworks, making StarlingX an heavy player.
- OpenNebula: OpenNebula is an open-source framework made to create multi-provider hybrid & edge clouds. It focuses on the virtual infrastructure layer and while deployment of containers and microVMs, it does not address the data processing layer.
- EdgeXFoundry: EdgeXFoundry focuses on IoT related use cases. It abstracts IoT protocols (sensors, actuators and others) and provides device management (administration and maintenance of IoT devices deployed on the field) capabilities. While there are still developments on-going, the number of tested devices and protocol adapters is relatively limited.

Based on this rapid analysis, two main options have emerged for consideration in the project:

- Mage.ai, with questioning on the need for customised user interface to ease management and configuration of pipeline and evaluate distribution capabilities over Dask/Ray.
- NiFi, with MiNiFi running locally but with questions on the ability to execute AI distributed models.

3.3 WebAssembly on MCU

WebAssembly (Wasm) is a binary instruction format that serves as a portable compilation target for programming languages, enabling deployment on web browsers and various environments. When used on microcontroller units (MCUs) like the STM32 L4 series, WebAssembly opens up several possibilities.

It can be of interest to run some of the processing algorithms for data quality as close as possible to the sensor, the extreme case being to run them *in the sensor*, or more precisely by the microcontroller embedded in it. The idea is to be able, from the cloud, to push some *hook functions* to be run on the collected data by the MCU of the sensor, each time a new data point is acquired. To enable such a capability, the WebAssembly technology will be used, this is explained in SEDIMARK_D3.1 [3].

From the Edge-Cloud interaction point of view, this imposes to be able to manage a list of processing *hooks*, and push/remove them on a list of far edge nodes. This may involve some improvement, of the chosen orchestration tool. This will be further investigated.

4 MLOps

Machine Learning Operations or MLOps comes as an inspiration from the DevOps world. It aims to unify the ML system development and the ML system operations like Model Registry, Model Tracking, Model Evaluation or Model Exposure.

With the increasing demand to bring machine learning models to production, MLOps provides a set of practices to streamline and automate the end-to-end ML lifecycle. In this chapter we will present a state of art of current ML and MLOps Frameworks as well as the standards that describe the ML models.

4.1 Definition

MLOps is the compound of machine learning and operations, aiming to standardize and streamline the end-to-end workflow of building, deploying, and monitoring machine learning models in a production environment. This ensures models are robust, scalable, and reliable and helps bridge the gap between Data Science and IT teams.

4.2 ML Frameworks

TinyML [4] refers to a growing field and community focused on enabling machine learning inference on extremely low-power, memory-constrained microcontrollers. These microcontrollers are typically used in edge devices, such as IoT sensors. Such devices can process data locally, making real-time decisions without needing to communicate with a central server, thus saving bandwidth and reducing latency.

TensorFlow [5] is an open-source machine learning framework developed by Google Brain. It provides a comprehensive suite of tools, libraries, and community resources that helps researchers and developers build and deploy ML-powered applications easily. In a distributed marketplace, TensorFlow can handle tasks ranging from recommendation systems on the server-side to real-time data processing on edge devices.

PyTorch [6] is an open-source machine learning library developed by Facebook's AI Research lab. It's known for its dynamic computational graph, which makes it particularly favourable for research. PyTorch can be utilized to train and deploy models for various tasks in the marketplace, such as personalization, demand forecasting, and fraud detection. With the help of ONNX (Open Neural Network Exchange), PyTorch models can also be converted and served on platforms that don't natively support PyTorch.

EdgeML [7] is a project from Microsoft Research that aims to bring machine learning to edge devices. This local processing can enhance user experience through faster decision-making and also ensures that sensitive data can be processed without being sent to a central server, enhancing privacy.

4.3 MLOps Frameworks

MLOps has gained a lot of traction in recent years due to the increasing need to bring machine learning models easily from development to production. This has led to the emergence of various frameworks, platforms, and tools specifically designed to address the challenges in the ML lifecycle. Here's an overview of some of the prominent MLOps frameworks.

MLFlow [8] is an open-source platform designed to manage the end-to-end machine learning lifecycle. It includes tools for tracking experiments, packaging code into reproducible runs, and

sharing and deploying models. MLFlow is not constrained to Kubernetes and runs in the environment where the user chooses.

KubeFlow is an open-source project from Google that provides a set of purpose-built components for deploying, monitoring, and operating ML systems on Kubernetes [9]. It has many features like pipeline creation, multi-framework support, training jobs orchestration, and serving.

Neptune.ai is a metadata store for MLOps, which allows teams to track, organize, and collaborate on ML projects. Like MLFlow it features experiment tracking, model registry, integration with popular ML frameworks.

The field of MLOps is rapidly evolving, with a plethora of tools and frameworks emerging to cater to various challenges in the ML lifecycle. The choice of an MLOps framework often depends on specific organizational needs, existing tech stack, and the complexity of ML workloads.

In this project the MLFlow framework will be used based on the majority recommendations coming from industry and its availability as an open-source solution.

4.4 Framework-agnostic ML model description

Interoperability of neural network models between frameworks is a key development area within the ML research community. A plethora of frameworks exist for defining and training neural network models, with PyTorch [10], TensorFlow [11], and JAX [12] being the most popular from a research and development perspective. There are two broad modes of framework interoperability. On the one hand, a user might wish to deploy a model defined and trained in a framework optimized for inference at scale. On the other hand, it might be preferable to distribute a model definition in a common format that enables continued training within a framework of choice. This is especially the case within distributed or federated learning, where the sharing of raw Python code can present a security risk.

In general code written using the syntax and abstractions of one framework cannot be easily ported to another framework. As sown in [13], there are many converters between individual frameworks, but still the picture is incomplete, since there are many cases where no converter exists between two frameworks (i.e. between Theano [14] and caffe2 [15]). Additionally, there can be converters from i.e., framework 1 to framework 2, but no converters for the opposite conversion from framework 2 to framework 1, as in the case of ONNX to torch using onnx2torch, but no converter from torch to ONNX. Given the rapid pace of development, maintaining converters is a problem, and many frameworks may lack equivalent operators, and thus they will have to be re-implemented by the converter developer [16]. Small differences in the implementation of neural network components between frameworks might also result in differing model behaviour when models are ported from one framework to another, while it is noted by [16] that these converters can often introduce subtle bugs and security problems.

In the case of model deployment and inference, most of the popular frameworks contain a module or method for porting code to Open Neural Network Exchange (ONNX) [17], a common intermediary depiction which represents the network as a language agnostic graph, that can then be compiled and deployed in one of several inference run-times. SEDIMARK will allow for the export of models to ONNX format for the purpose of inference. However, ONNX does not fully support the retraining of models.

As seen in the table from [13], most conversions between frameworks are based on the MMdNN project [18], which is an attempt to define a “Universal Converter” for deep learning

models to allow both inference and re-training of ML models across different frameworks. MMdNN converts model formats to an “Intermediate Representation”, and from that, converts the model to the target platform format. However, MMdNN is only focusing on a subset of deep learning models and currently is not maintained on GitHub, with its latest commit more than 3 years ago.

There are ongoing attempts to remedy this problem in a more holistic manner - for instance the commercial effort Ivy [19] from unify.ai aims to offer code transpiration between frameworks, though requires an Application Programming Interface(API) key for use and did not appear to work out of the box when tested. There is currently not fully inclusive, open-source solution to the problem of transferring models between frameworks for continued training.

A recent update to the popular framework Keras [20] will support code written in Keras being run with either Jax, PyTorch or TensorFlow as a backend. Both inference and continued model training are supported. Though the Keras API imposes limitations on the class of models that can be defined within it, for most general use-cases it proves sufficient.

For the time being, SEDIMARK will build upon the newly released Keras Core Python package to offer a degree of interoperability for the purpose of further training models in distributed settings (see Figure 3 below). Models defined in this format can be seamlessly exported to either JAX, TensorFlow or PyTorch, though the reverse is not true. While allowing for a choice of back-end and providing a common format for participants in the SEDIMARK distributed learning ecosystem, this does effectively restrict users to the Keras syntax and abstractions when defining their models.

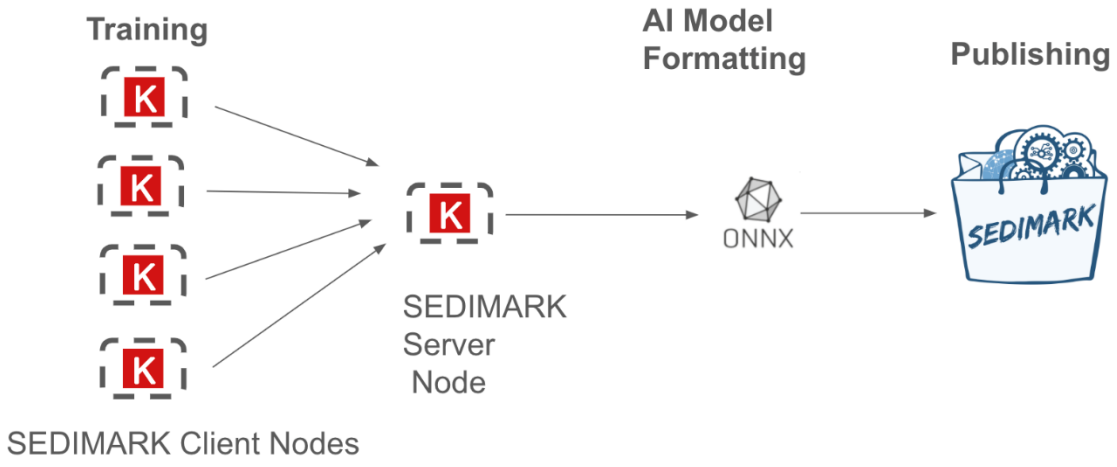

Figure 3: Model formatting process within SEDIMARK.

Going forward SEDIMARK will continue to explore alternatives, such as the ONNX training-runtime, which is still undergoing development. Keras-core has already been implemented as the model format within both SEDIMARK distributed training components (Fleviden and Shamrock), and performance will be tested with nodes running each of the three underlying frameworks it supports. The ability to port models to ONNX will be available on both Fleviden and Shamrock to allow for immediate sharing/publishing of trained models within the wider SEDIMARK marketplace.

5 Certification Services

Assets advertised through offerings made available through the Marketplace will be expected to vary in terms of quality and performance. Although, it is mandatory to pass a minimum set of requirements for them to be minimally viable and exchangeable assets. This would involve checking for compliance:

- Assets with standards specified by the Marketplace.
- Connectors with the minimal set of operations.
- Use of Assets in accordance with license or policy restrictions.

Figure 4 illustrates the categorization of certification services for a particular type of Asset.

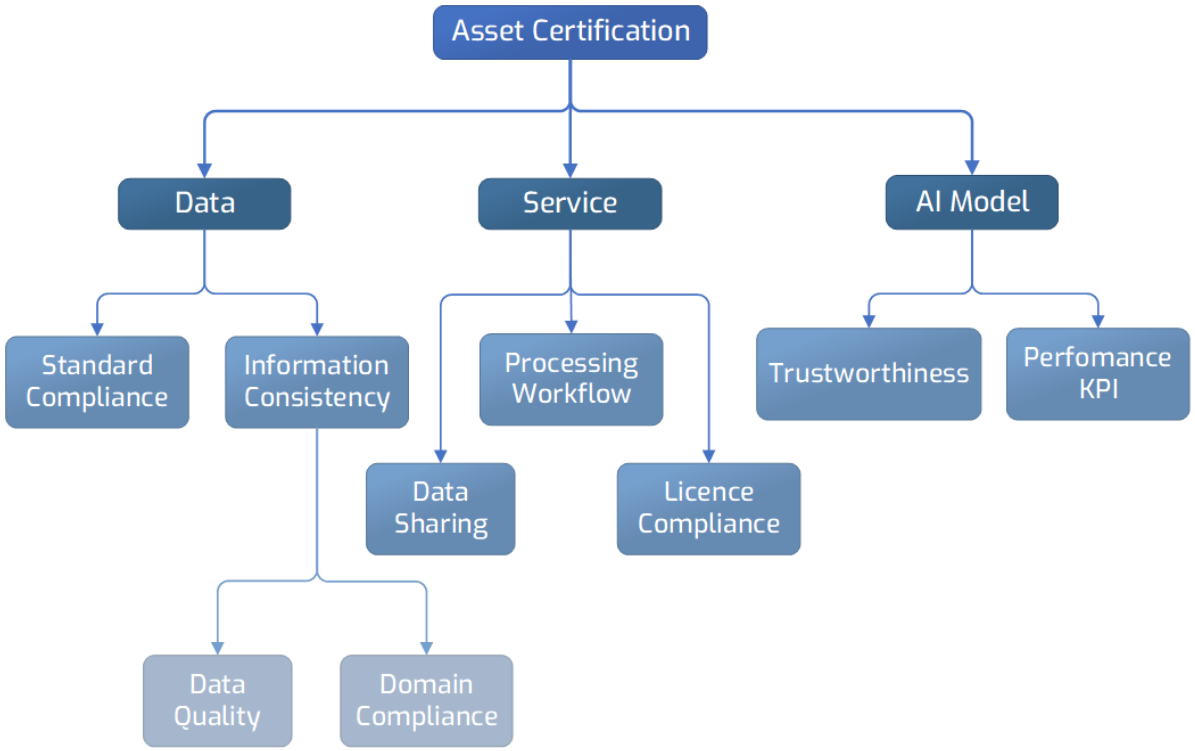

Figure 4: Taxonomy of Certification services for Assets within SEDIMARK.

5.1 Data Assets

For data assets, their formatting, annotation, and enrichment need to comply with the information model standards specified by SEDIMARK. Information regarding both metadata and data needs to be assessed for consistency. This will involve checks for quality and compliance within domain-specific parameters. In deliverable SEDIMARK_D3.3, the process for validating compliance with standards is described. In addition, within the technologies to be used, one potential technology relevant to the evaluation of graph-based data representation is SHACL (Shapes Constraint Language) which is a W3C specification aimed at validating the compliance of a graph by checking its "shape" comply to the expectation. With regards to information consistency, a subset of generic data quality metrics defined in Deliverable 3.1 will be used, as well as metric governed by domain-specific restrictions or ranges.

5.2 Service Assets

For Service Assets, the set of operations available through their corresponding interfaces will be checked to ensure that they conform to the standards adopted for data sharing interfaces. A dedicated test suite will be developed, and API specification templates be provided for self-testing. For example, NGSI-LD is currently the main interface for Data Sharing, therefore the technical specification on validating NGSI-LD platforms will be used as a reference [21]. Processing workflows will also be checked for following data quality and integrity requirements. This will include validating processing pipeline stages and any limitations in terms of inputs and outputs, and statistics relating to service quality declarations. It will also check Data Assets are used in compliance with the license's terms of use. Rajbahadur et al. [22] provide a case study of how dataset usage from public domain can potentially result in non-compliance, and what AI engineers in turn should be conscience of when processing and how results are then shared or exchanged.

5.3 AI Model Assets

For AI Model Assets, their performance KPIs will be checked to meet minimum requirements. This could include aspects such as Accuracy, Loss, Confusion Matrix, AUC (Area Under ROC curve), Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R Square. It will also assess the “trustworthiness” of the AI model, based on factors such as processing and provenance transparency, as according to Gardner et al. [23], trustworthiness is a property that “demonstrates fulfilment of its promise by providing evidence of dependability in the context of use, and end users have awareness of its capabilities during use”. According to the National Institute of Standards and Technology (NIST) [24], the main aspects of AI trustworthiness are validity and reliability, safety, security and resiliency, accountability and transparency, explainability and interpretability, privacy and fairness with mitigation of harmful bias.

6 Conclusions

In conclusion, this document has presented a comprehensive overview of the work conducted in relation to Edge data processing and service certification with a focus on analysing existing toolset to provide a framework for deploying AI-based data processing and sharing modules at edge data sources. This initiative is carried out with a keen consideration of edge-cloud interactions and follows the principles of MLOps.

The report delves into the challenges and requirements associated with edge processing, addressing critical aspects such as bandwidth, computing resources, privacy, and data quality. It provides insights into the orchestration tools governing Edge/Cloud interactions, including an exploration of WebAssembly on MCU. Two frameworks being mage.ai and Apache NiFi/MiNiFi have been identified as of particular interest.

Then MLOps environments have been studied. MLflow has been identified as a best candidate for models' construction and training while ML frameworks such as TensorFlow, PyTorch, tinyML, and edgeML are candidates for models' execution. Additionally, the document outlines a framework-agnostic ML model description and introduces certification services.

Finally, in respect with certification of dataset and services, initial analysis has focuses on tools relevant for graph-based data models such as one built on RDFS or NGSI-LD.

It is essential to note that, at this point, the document primarily addresses challenges, considered options, and potential implementation choices, recognizing that the final version will provide in-depth details on the technical decisions made and their subsequent implementation. As the SEDIMARK project progresses, this preliminary deliverable sets the foundation for the subsequent phases, emphasizing the exploratory nature of the current topics and paving the way for a more comprehensive and detailed report in the future.

7 References

[1] SEDIMARK, “D2.1 Use cases definition and initial requirement analysis,” 2023, June.

[2] SEDIMARK , “D2.2 SEDIMARK Architecture and Interfaces. First version,” 2023, September.

[3] SEDIMARK, “D3.1 Energy efficient AI-based toolset for improving data quality. First version,” 2023.

[4] K. P. P. R. S. I. O. L. A. L. Rakhee Kallimani, “TinyML: Tools, Applications, Challenges, and Future Research Directions,” no. https://arxiv.org/abs/2303.13569, 2023.

[5] A. A. P. B. E. B. Martín Abadi, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015.

[6] A. a. G. S. a. M. F. a. L. A. a. B. J. a. C. G. a. K. T. a. L. Z. a. G. N. a. A. L. a. D. A. a. K. A. a. Y. E. a. D. Z. Paszke, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” vol. 2019, no. Advances in Neural Information Processing Systems 32.

[7] K. W. N. L. G. X. Zhihe Zhao, “EdgeML: An AutoML Framework for Real-Time Deep Learning on the Edge,” vol. https://doi.org/10.1145/3450268.3453520, 2021.

[8] A. C. A. D. A. D. A. G. S. A. H. A. K. C. M. S. M. T. N. P. O. M. P. A. S. F. X. M. Z. R. Z. J. Z. a. C. Z. Andrew Chen, “Developments in MLflow: A System to Accelerate the Machine Learning Lifecycle,” Vols. In Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning (DEEM'20)., 2020.

[9] M. S. S. M. Aditya Pandey, “Deployment of ML Models using Kubeflow on Different Cloud Providers,” vol. 2206.13655, no. https://doi.org/10.48550/arXiv.2206.13655, 2022.

[10] [Online]. Available: https://PyTorch.org/.

[11] [Online]. Available: https://www.TensorFlow.org/.

[12] [Online]. Available: https://jax.readthedocs.io/en/latest/.

[13] ysh329, “Deep learning model convertors,” [Online]. Available: https://github.com/ysh329/deep-learning-model-convertor.

[14] [Online]. Available: https://github.com/Theano/Theano.

[15] [Online]. Available: https://caffe2.ai.

[16] Z. M. G. C. K. L. T. X. L. &. C. C. Deng, “Differential Testing of Cross Deep Learning Framework {APIs}: Revealing Inconsistencies and Vulnerabilities,” in *32nd USENIX Security Symposium*, 2023.

[17] [Online]. Available: https://onnx.ai/.

[18] Y. C. C. Z. R. Q. T. J. X. L. H. &. Y. M. Liu, “Enhancing the interoperability between deep learning frameworks by model conversion. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering,” in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020.

[19] D. P. F. F. F. J. S. &. C. R. I. Lenton, “Templated deep learning for inter-framework portability,” [Online]. Available: https://doi.org/10.48550/arXiv.2102.02886.

[20] [Online]. Available: https://keras.io/.

[21] ETSI ISG CIM, “GR 030 Validation of NGSI-LD test Platform and Examples of uses,” 2023.

[22] E. T. L. Z. D. L. B. C. Z. M. (. D. M. G. Gopi Krishnan Rajbahadur, “Can I use this publicly available dataset to build commercial AI software? -- A Case Study on Publicly Available Image Datasets,” 03 11 2021.

[23] C. R. K. S. C. a. S. A. Gardner, “Contextualizing End-User Needs: How to Measure the Trustworthiness of an AI System.,” 2023. [Online]. Available: https://doi.org/10.58012/8b0v-mq84. [Accessed December 2023].

[24] NIST (National Institute of Standards and Technology), “AI Risk Management Framework,” 2023. [Online]. Available: https://www.nist.gov/itl/ai-risk-management-framework.