H2020 - INDUSTRIAL LEADERSHIP - Information and Communication Technologies (ICT)

ICT-38-2020: Artificial intelligence for manufacturing

Grant Agreement: 957362

Explainable Manufacturing Artificial Intelligence

WP3: Core Artificial Intelligence Bundles for Algorithm Lifecycle Management

D3.1: AI Bundles Methods and System Designs

Deliverable Leader: Suite5<Deliverable_leader>

Due Date: M12

Dissemination Level: Public

Version: D1.0

Short Abstract

This deliverable reports on the results produced through WP3 activities towards the delivery of the XMANAI Artificial Intelligence (AI) bundles designs and methods. It presents insights gained through the landscape analysis on research dimensions and technical advancements in the broader scope of AI pipelines and reflects upon the positioning of the XMANAI solution targeting explainable AI pipelines in the manufacturing domain. In this scope, the role of data models is explored and relevant standards are studied. The deliverable reports on the development of the XMANAI data model and presents its early concepts, relationships, and foreseen lifecycle management mechanisms. Finally, the detailed WP3 architecture design is discussed and each of its components is specified, including designed functionalities and offered methods, considered technologies for its implementation and indicative mockups.

Further Information: www.ai4manufacturing.eu

Document Log

Contributors	**Suite5**, TXT, Fraunhofer, POLIMI, AIDEAS, ATHENA, KBIZ, UBITECH, TYRIS
Internal Reviewer 1	INNOVALIA
Internal Reviewer 2	UNIMETRIK
Type	Report

History

Versions	Description
D0.1	Initial Table of Contents
D0.2	Initial draft of sections 2.2 and 2.3 with input from all partners
D0.3	Initial draft of section 3.3 with input from all partners
D0.4	Revision of sections 2.2, 2.3 and 3.3. Initial draft of sections 3.1, 3.2, 3.5
D0.5	Initial draft of WP3 architecture (section 4.1), Condolidated versions of sections 2 and 3.
D0.6	Initial draft of sections 4.2-4.12 by all technical partners.
D0.7	Full Deliverable Draft sent for internal review
R0.1	Revision of internal reviewer 1 (INNOVALIA)
R0.2	Revision of internal reviewer 2 (UNIMETRIK) based on R0.1
D0.8	Updated version addressing comments received during the internal review
F1.0	Final version submitted to the EC

Executive Summary

The XMANAI Deliverable D3.1 "AI Bundles Methods and System Designs" documents the results achieved through the activities performed in the context of all five WP3 (Core Artificial Intelligence Bundles for Algorithm Lifecycle Management) tasks in this first iteration. The purpose of this deliverable is to provide the draft design of the XMANAI AI bundles, in particular the processes that will be supported, the design of the components that will be implemented to support these processes, the foreseen high-level user interfaces and the rationale underpinning the relevant design decisions.

The deliverable first provides a thorough landscape analysis across the areas that were identified as the drivers of XMANAI AI bundles development and of the way the complete AI algorithm lifecycle will be handled. Insights from these areas are important towards the creation of a collaborative environment for managing not only individual AI models, but complete AI pipelines enriched with explainability aspects, targeting the manufacturing domain. In this context and building upon the insights gained through the XMANAI Deliverable D1.1, the current deliverable explores:

- **Data manipulation processes and technologies in the context of AI pipelines**, including any data preprocessing task needed to bring the raw input data to a format appropriate for insights extraction, through visualisations and/or machine learning models.
- **Visualisation and visual exploration processes, spanning data and model visualisations**, to gain insights on how different functionalities can be leveraged across the various steps of AI pipelines development, depending on the problem at hand and the stakeholder (data scientist, data engineer, business user) who creates or requires the respective visualisation.
- **MLOps (Machine Learning Operations)**, which offer good practices for building, deploying and managing Machine Learning (ML) systems, taking into consideration their differences from traditional (non-ML) development projects.

Apart from the above areas that are explored both in terms of research directions and practical / technical advancements, additional aspects of interest were also explored, including applicable orchestration engines and execution infrastructures, but also more targeted parts of the aforementioned areas, e.g. experiment tracking engines in the context of MLOps. Identified advantages, potential benefits or limitations and challenges associated with each of the studied domains (for **over 65 technologies and related processes**) are discussed and important considerations are highlighted to assist in the XMANAI design decisions.

The deliverable then discusses the role of data models in information systems development and their contribution towards data compatibility and consistency, highlighting the need to leverage data models in the scope of XMANAI explainable AI pipelines for manufacturing decision making. In order to develop the XMANAI data model, for which the graph modelling paradigm was considered more appropriate for extensibility and flexibility purposes, requirements were collected based on the one hand on the demonstrators data and on the other hand on a broader domain study that was performed on existing models and standards of the manufacturing domain.

Building upon the extracted insights, the first version of the XMANAI data model was designed. It comprises a total of **40 concepts** spanning across different aspects, processes and entities of the manufacturing domain, and will contribute towards data explainability within the AI pipelines. Each of the concepts is associated with certain properties and relationships among concepts are captured. **Almost 200 properties and more than 110 relationships** are defined to capture the underlying manufacturing knowledge, but more importantly rules and processes are designed to ensure the complete model's lifecycle is supported, including its extension and refinement as representations for additional domain data will be required as the project evolves.

The deliverable finally presents the WP3 architecture and its connection to the more high-level WP5 ("XMANAI Platform Architecture") architecture that is documented in D5.1 and provides insights and justifications for the way it is broken down into individual components, highlighting what each one

brings in the realisation of the XMANAI collaborative environment for managing the lifecycle of XAI pipelines aiming to bring new insights in the manufacturing decision making and operations. The architecture comprises a total of **11 components**, namely: (1) the Knowledge Graph Manager, (2) the Data Preparation Engine, (3) the XAI Model Engineering Engine, (4) the XAI Model Guard, (5) the Interactive Data Exploration and Experimentation Tool, (6) the Pipeline Designer, (7) the Experiment Tracking Engine, (8) the Pipeline Serving and Monitoring Engine, (9) the Results Visualisation Engine, (10) the XAI Models Explanation Engine, (11) the Execution and Orchestration Engine.

For each of the components, the deliverable provides the main functionalities and foreseen offered methods, their positioning within the architecture and their interactions with other components, as well as the technologies considered for their implementation. Indicative low-fidelity mockups of the core screens of the components that offer a user interface (i.e. for 10 out of the 11 components) are also presented.

The results presented in the current deliverable and the activities through which they were achieved, built upon the insights presented in D1.1 “State of the Art Review in XMANAI Research Domains”, D1.2 “XMANAI Concept Detailing, Initial Requirements, Usage Scenarios and Draft MVP”, D5.1 “System Architecture, Bundles Placement Plan and APIs Design” and (indirectly) D6.1 “Demonstrators Requirements”. All design activities in this context were performed in close collaboration with WP2 “Industrial Asset Management and Secure Asset Sharing Bundles”, which is responsible for the development of the Data Asset-related Bundles, and with WP5 “XMANAI Platform Continuous Integration”, which is responsible for defining the XMANAI platform architecture and delivering the final integrated XMANAI platform.

Finally, the detailed design and specifications documented in this deliverable shall guide the development activities in WP3 towards the initial release of the XMANAI AI Bundles that is expected on M18 as per the XMANAI DoA. The early implementation of such bundles will be documented in D3.2 “XMANAI AI Bundles – First Release”, and shall continue to proceed in close collaboration with the Asset Management-related Bundles delivered in WP2 and the overall integration activities of WP5.

Table of Contents

List of Figures

List of Tables

1 Introduction

The main purpose of this section is to provide a brief overview of the deliverable, introducing the purpose of the document, its main content and possible dependencies with other XMANAI tasks and activities.

1.1 XMANAI Project Overview

Despite the indisputable benefits that Artificial Intelligence (AI) can bring in society and in any industrial activity, humans typically have little insight about AI itself and even less concerning the knowledge on how AI systems make any decisions or predictions due to the so-called “black-box effect”. Many of the machine learning/deep learning algorithms are opaque and not possible to be examined after their execution to understand how and why a decision has been made. In this context, to increase trust in AI systems, XMANAI aims at rendering humans (especially business experts from the manufacturing domain) capable of fully understanding how decisions have been reached and what has influenced them.

Building on the latest AI advancements and technological breakthroughs, XMANAI shall focus its research activities on Explainable AI (XAI) in order to make the AI models, step-by-step understandable and actionable at multiple layers (data-model-results). The project will deliver "glass box" AI models that are explainable to a "human-in-the-loop", without greatly sacrificing AI performance. With appropriate methods and techniques to overcome data scientists’ pains such as lifecycle management, security and trusted sharing of complex AI assets (including data and AI models), XMANAI provides the tools to navigate the AI’s “transparency paradox” and therefore:

(a) accelerates business adoption addressing the problematic that “if manufacturers do not understand why/how a decision/prediction is reached, they will not adopt or enforce it”, and
(b) fosters improved human/machine intelligence collaboration in manufacturing decision making, while ensuring regulatory compliance.

XMANAI aims to design, develop and deploy a **novel Explainable AI Platform** powered by explainable AI models that inspire trust, augment human cognition and solve concrete manufacturing problems with value-based explanations. Adopting the mentality that “AI systems should think like humans, act like humans, think rationally, and act rationally”, a catalogue of **hybrid and graph AI models** is built, fine-tuned and validated in XMANAI at 2 levels: (i) baseline AI models that will be reusable to address any manufacturing problem, and (ii) trained AI models that have been fine-tuned for the different problems that the XMANAI demonstrators’ target. A bundle of **innovative manufacturing applications and services** are also built on top of the XMANAI Explainable AI Platform, leveraging the XMANAI catalogue of baseline and trained AI models.

XMANAI will validate its AI platform, its catalogue of hybrid and graph AI models and its manufacturing apps in **4 realistic, exemplary manufacturing demonstrators** with high impact in: (a) optimizing performance and manufacturing products’ and processes’ quality, (b) accurately forecasting product demand, (c) production optimization and predictive maintenance, and (d) enabling agile planning processes. Through a scalable approach towards Explainable and Trustful AI as dictated and supported in XMANAI, manufacturers will be able to develop a robust AI capability that is less artificial and more intelligent at human and corporate levels in a win-win manner

1.2 Deliverable Purpose and Scope

The current deliverable D3.1 “AI Bundles Methods and System Designs” documents the results that have been achieved during the first iteration of activities perfomed within all WP3 tasks towards the realisation of the XMANAI Artificial Intelligence bundles and algoritms’ lifecycle management. The activities through which these results were achieved, build on top of the insights presented in D1.1

"State of the Art Review in XMANAI Research Domains", D1.2 "XMANAI Concept Detailing, Initial Requirements, Usage Scenarios and Draft MVP", D5.1 "System Architecture, Bundles Placement Plan and APIs Design" and (indirectly) D6.1 "Demonstrators Requirements".

The purpose of this deliverable is to provide the (draft) design of the XMANAI AI bundles, in particular including the processes that will be supported, the design of the components that will be implemented to support these processes, the foreseen high-level user interfaces and the rationale underpinning the relevant design decisions. To support these design decisions, a thorough state of the art review in relevant research areas and technologies is performed and presented. The deliverable will also present the XMANAI approach regarding knowledge representation in the manufacturing domain and will document the XMANAI data model, which that will be integrated across various steps of the AI pipelines developed in the context of the AI bundles.

All activities documented in the present deliverable have been performed in close collaboration with WP2 "Industrial Asset Management and Secure Asset Sharing Bundles" and with WP5 "XMANAI Platform Continuous Integration", which defines the XMANAI platform architecture and will deliver the final integrated XMANAI platform.

The next WP3 deliverable D3.2 "XMANAI AI Bundles – First Release" will deliver the first release of the XMANAI AI bundles that is due on M18 and in accordance with the designs presented in this deliverable.

1.3 Impact and Target Audiences

The results presented in this document target mainly the technical users that develop the XMANAI Platform, as well as the researchers who support the solution. The business users are also influenced by the content, since in essence it describes, albeit in draft format, (part of the) XMANAI product to be built in the context of WP3 activities.

The state of the art insights presented in section 2 can be up to an extent considered understandable also by people without data science/ engineering background, however technical details are often provided and an understanding of machine learning and data analysis practices is expected. Section 3 on the other hand, assumes some background in data modelling but business users within the manufacturing domain are also in the target audience, since they have knowledge over the domain's common concepts, their properties and relations. Finally, section 4 requires technical background for the main presentation of the WP3 components and technologies, whereas the provided mockups are understandable to all involved stakeholders.

1.4 Deliverable Methodology

The information reported in this deliverable has been produced by the consortium members following the methodology described below:

- **For the state of the art review**, the relevant domains and categories to be studied were first identified and agreed upon. Assignments were then discussed and partners' contributions were collected in online collaboratively edited documents: (a) for the technologies and tools documentation and (b) for the methods in each domain/area. Although this was the first performed activity, the state of the art documents remained online and were refined and enriched as needed throughout this first iteration of WP3 activities, allowing for a feedback loop between the design process and the landscape analysis.
- **For the XMANAI data model consolidation,** the starting point was to examine data examples from the demonstrator partners and respective requirements towards the XMANAI platform. To gather this information in a cohesive way, the partners that act as technical support for the

technological development of the demonstrators reviewed the relevant material and discussed open points with the demonstrator partners to acquire concrete insights. At the same time a study on the existing models and ontology standards in manufacturing was performed by the project partners to understand the different manufacturing domains, their intrinsic differences, available standards/ vocabularies and diverse needs in terms of knowledge representation. An online collaborative document was used in order for partners to gradually provide input based on the insights from the demonstrators and the domain review in terms of required concepts, attributes and relationships. Dedicated discussions were held towards the collaborative definition of the required semantics, finally leading to the definition of the XMANAI model.

- **For the WP3 architecture, components and processes definition,** partners initially discussed and iteratively refined the architecture design. Once the architecture was finalised and the methods to be supported were agreed upon, the partners provided targeted input for each of the components and discussed foreseen interactions in order to ensure that a cohesive view is provided that can be leveraged towards implementation in the next steps.

1.5 Dependencies in XMANAI and Supporting Documents

The results presented in the current deliverable and the activities through which they were achieved, built upon the insights presented in D1.1 “State of the Art Review in XMANAI Research Domains”, D1.2 “XMANAI Concept Detailing, Initial Requirements, Usage Scenarios and Draft MVP”, D5.1 “System Architecture, Bundles Placement Plan and APIs Design” and (indirectly) D6.1 “Demonstrators Requirements”. All design activities in this context were performed in close collaboration with WP2 which is responsible for the XMANAI Industrial Asset Management and Secure Asset Sharing Bundles and with WP5 which defines the XMANAI platform architecture and will deliver the final integrated XMANAI platform. In this sense, the deliverable has strong connections to D2.1 which reports on the WP2 bundles methods and designs and to D5.1 that details the XMANAI services bundles and the high-level project architecture, as well as the user journeys within this architecture.

The current deliverable will serve as guide for the first release of the XMANAI AI bundles which will be delivered on M18 and will be documented in D3.2 “XMANAI AI Bundles – First Release”.

1.6 Document Structure

The remainder of this document is structured as follows:

- Section 2 presents insights from the state of the art review that was performed on the methods and technologies that will guide the design and development of the XMANAI AI Bundles that are delivered through the WP3 activities.
- Section 3 presents the XMANAI common data model and its lifecycle management processes. It presents relevant ontologies and standards that were studied, as well as the data model development approach and finally the concepts, relationships and properties foreseen in its first version.
- Section 4 presents the WP3 architecture and its connection to the WP5 high-level architecture. The overall approach and underlying methods leveraged for the development of the WP3 explainable AI pipelines are presented. For each of the WP3 components the provided functionalities are described and considered technologies for implementation are presented. Low-fidelity mockups are also provided for the main screens of the components that offer a user interface.
- Section 5 concludes the deliverable and discusses the next steps.

2 AI Bundles Landscape Analysis

2.1 Scope

According to the XMANAI DoA, WP3 *components will be used for data pre-processing, knowledge management, and representation as well as for the selection and the design of AI model execution paths that can run online or on the premises of manufacturers, running on top of the data management components of WP2 and taking as input the data stored and the AI models that will be developed and trained in WP4.* The WP is also *responsible for the execution environment of the XMANAI platform*.

In a nutshell, the WP3 development activities evolve around the provision of the XMANAI Artificial Intelligence Bundles. Machine learning models and methods providing explainability on these models' decisions constitute the essence of the WP3 offerings. Yet, in order for these models to be properly configured, trained, evaluated, utilised/ deployed, constantly monitored, assessed and refined as needed, numerous other processes need to be in place. Developing and leveraging robust and insightful AI pipelines that can assist manufacturers in their everyday operations and decision making processes, spans well beyond ML models' training by data scientists working in isolation. Figure 2-1 highlights the vastness and complexity of functionalities and infrastructures that surround the pure ML code when it comes to building a solid ML system.

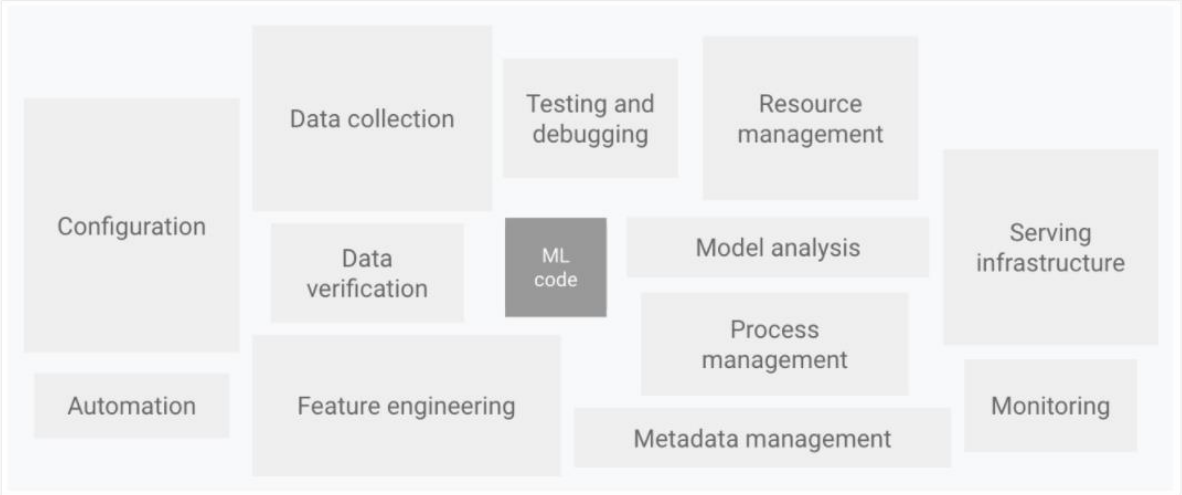

Figure 2-1: Elements from ML systems according to (Google, 2021)

In XMANAI the challenge is even greater, since explainability aspects are brought to the spotlight and their successful inclusion in this abundance of components and interactions is far from straightforward and established practices are not yet available.

As explained, not all of the processes shown in Figure 2-1 are within the scope of WP3. Some of them (e.g. data collection) are provided by WP2 components and others (e.g. the actual ML code of the trained models) will be provided through the WP4 activities. But WP3 is responsible for providing the "glue" among these and for handling all functionalities needed to transform the input data (after ingestion) to insights, i.e. for all aspects of the XMANAI AI pipelines. In order to do so successfully, a detailed landscape analysis was performed on the aspects affecting the development, management, serving, monitoring and evaluation of AI pipelines. These aspects can be roughly grouped under the following areas:

- Data manipulation in the context of AI pipelines, which considers the data preprocessing, cleaning and transformation processes needed to bring the initial raw data to the appropriate

form to be used for insights extraction, either through visualisations or used in ML models development and application.

- Machine learning, which lies at the core of the AI bundles and concerns all functionalities around machine learning and model explainability. It should be noted that the ML and explainability methods that are in the research scope of XMANAI have been studied and documented in D1.1 and will be further detailed in the actual models development activities that are part of WP4, and therefore only some additional, mostly technology-based, aspects are discussed here.
- Visualisation and visual exploration, which evolves around data visualisations and model visualisations, presenting how different visualization functionalities can be leveraged across the various steps of AI pipelines development, depending on the problem at hand and the stakeholder who needs the respective visual representation.
- MLOps, which constitute the intersection between ML and DevOps (development operations) and offer practices for building, deploying and managing Machine Learning (ML) systems.
- Orchestration engines and execution infrastructures, which are important aspects of the WP3 architecture and due to their nature are only explored from the technical point of view.

The current section presents the major findings from this state of the art analysis that was performed both in terms of theoretical foundations (presented in section 2.2) and of practical/technical building blocks (presented in section 2.3) that can be leveraged.

2.2 Methods

This section presents insights gained through the landscape analysis that was performed along the theoretical dimension of the areas that, as described in Section 2.1, guide the design of the WP3 services. Research papers, but also other online sources, including high-quality blogposts, methodology-oriented documentation material and white papers from core companies in the ML industry were studied to identify the state of the art, the challenges and future of AI pipelines in order to extract insights that will help shape the XMANAI positioning.

From the areas that were explored, as these were presented in the previous section, some are inherently more research-oriented and others face challenges mostly when it comes to their application, some are just starting to be explored by the academic community and others yield results dating back to the dawn of AI. An in-depth analysis of the complete AI spectrum is not possible to be even outlined, but is also not in scope here. Instead, the following subsections cover the wide range of functionalities and processes needed, but investigate deeper only those that are of interest to the AI bundles development.

An important aspect to clarify here is that in this “Methods” section (as opposed to the subsequent “Technologies” section), there is no reference to machine learning methods. It can be easily understood that an overview of the vast landscape of machine learning methods, models and algorithms is out of scope (impossible to be provided, and of little value as it would be too generic if done so), yet not conducting a state of the art analysis in the machine learning areas that are most relevant in XMANAI would be a significant omission. Indeed, this is not the case. D1.1 “State of the Art Review in XMANAI Research Domains” provides a detailed analysis of these areas, in particular of explainable AI and graph machine learning, which lie at the core of the project’s research and innovation scope. Furthermore, aspects related to the way these methods are supported and leveraged across the complete lifecycle pipelined are discussed here in the corresponding subsections, e.g. as part of sections 2.2.1 (MLOps), 2.2.2.2 (data preparation & ML) and 2.2.3.2 (visual exploration from the model perspective).

2.2.1 MLOps

MLOps is a set of practices for building, deploying and managing Machine Learning (ML) systems. Its name is derived from ML and operations (Ops), and it can be thought of as the intersection between ML and DevOps (development operations) practices.

DevOps refers to a set of principles that combine the work of software engineers with operational teams to create a combination of practices and tools that increase the organizations' ability to provide services with continuous delivery, minimal maintenance and maximum efficiency (Alla, 2021) Similarly, with the rapid growth in ML systems, MLOps has emerged as a solution that helps organizations successfully deploy ML models in production, manage risk, and ensure that applications are still in line with the business goals (Arrikto, 2021)

Therefore, MLOps practices aim to shorten development cycles, improve collaboration between teams, increase performance, scalability and security, and also increase return of investment of ML projects (Salama, 2021) . Another important area where MLOps plays a significant role in the ML lifecycle is model monitoring. Monitoring deployed models is crucial for continued provision of high quality ML services, and MLOps helps by monitoring model performance, monitoring metrics related to incoming data, detecting outliers and data drift, and also recognizing when models demand upgrades or retraining (Klaise, 2020). As such, it becomes evident that MLOps are relevant to all stages of AI-enabled pipelines and can significantly improve their reproducibility and robustness, accelerate their production serving and facilitate debugging.

MLOps supports the development and deployment of ML systems in the way DevOps supports data engineering. But there is a fundamental difference between deploying an ML model (MLOps) with deploying e.g. a web service: although in the latter a version of the code produces a version of the software, in the first case **a version of the code and a version of the data together** produce a version of the ML model. This dependence between models and data is an important aspect to consider when designing relevant services.

2.2.1.1 Steps and processes of ML development

Before diving into the principles and core capabilities of MLOps, it is important to establish a basic common understanding of the entire ML process for delivering a model to production. Even a quick search in the subject would easily reveal that there is no unique view on this, i.e. a concrete sequence of methods/tasks. Different data scientists/engineers and business users operating in different teams within different companies and domains, will each have their own view on this, adapted to their specific needs and the overall ML mentality. Some steps will be common across all (e.g. model training), whereas others may be combined/split or ultimately missing when comparing approaches.

The following 8 steps proposed by (Google, 2021) capture the core aspects of a data scientist's journey towards the successful delivery of an ML project and further outline or imply the required contributions of data engineers:

- **Data extraction**: During this step raw data are collected and processed in order to select and integrate relevant data from the various sources. If possible, the more data collected during this step (notwithstanding the need for high-quality data), the better the potential training outcomes will be, since more training examples increase the variety of information learned by the model. Data extraction (collection) is handled by WP2 services, but its significance should not be underestimated.
- **Data analysis**: The main objective of this step is to understand the available data for building the ML model. This step involves exploratory data analysis in order to extract insights about the data schema and characteristics, including the data distribution, identifiable trends or biases in the data. Added to that, features are constructed using feature engineering

techniques that will aid in the model implementation and accuracy of the predictions, by providing the models a deeper context. This is a crucial step as it dictates how the problem is going to be solved.

- **Data preparation**. This step involves scaling data into a more appropriate range, removing anomalies (outliers) that could potentially harm the models performance, but also splitting the data into training, validation and tests sets. During this step, data transformations and feature engineering can be applied to the model that solves the task, and the final output of this step are the data splits in the appropriate format.
- **Model training**. As the name suggests the models are prepared and trained in this step by the data scientist using the processed data from the previous step. The models are also tuned with hyperparameter tuning in order to get the best performing combination of parameters. During this step, the models are evaluated (using the validation set), in order to tune the model's hyperparameters, but also to prevent overfitting (when a model performs significantly better on a training set compared to a dataset never seen before). The output of this step is the best performing trained ML model.
- **Model evaluation**. During this step the ML model is evaluated using the test dataset produced during the data preparation phase, in order to assess the quality of the model. The output of this step is a set of chosen evaluation metrics that explain how well the model performs.
- **Model validation**. The model is compared to an appropriate baseline and if it exceeds the performance of the baseline model it can be considered ready for deployment.
- **Model serving**. At this step the trained model is integrated into the application and put into service. This is generally done by data engineers, as the job of the data scientist is typically considered completed after the model validation step.
- **Model monitoring.** The deployed model is constantly monitored in order to assess its predictive performance. If the performance falls below a certain threshold an iteration in the ML process may be invoked. Any bugs or unexpected model predictions are also reported, so that appropriate solutions can be established.

It should be stressed that the above steps do not consider ML explainability needs and processes, as this is currently an active research field, but not yet integrated in common practices.

2.2.1.2 MLOps Lifecycle and Priciples

MLOps aim to automate the entire process of designing, developing and serving an ML-enabled system in production (including support for the 8-steps process presented in the previous section, also referred to as data science steps for ML), making it significantly easier to deploy and maintain ML solutions.

The MLOps lifecycle consists of a total of seven steps which are shown in Figure 2-2. The first step in the MLOps lifecycle is the **ML development,** which corresponds to the aforementioned data science steps for ML. The **training operationalization** is then responsible for automating the process of deploying ML training pipelines. **Continuous integration** is where training pipelines are repeatedly executed upon trigger (e.g. when new data arrive or when code/settings change) or upon schedule. The output of this step is a trained ML model, ready for deployment. The **model deployment** step is responsible for automating the deployment process, whereas **prediction serving** refers to the actual serving of the deployed model in the production environment to get predictions. **Continuous monitoring** is responsible for checking the performance, efficiency and effectiveness of the deployed model, and finally, **data and model management** is responsible for providing support for auditability, traceability and compliance. At this step ML assets can also be shared and reused.

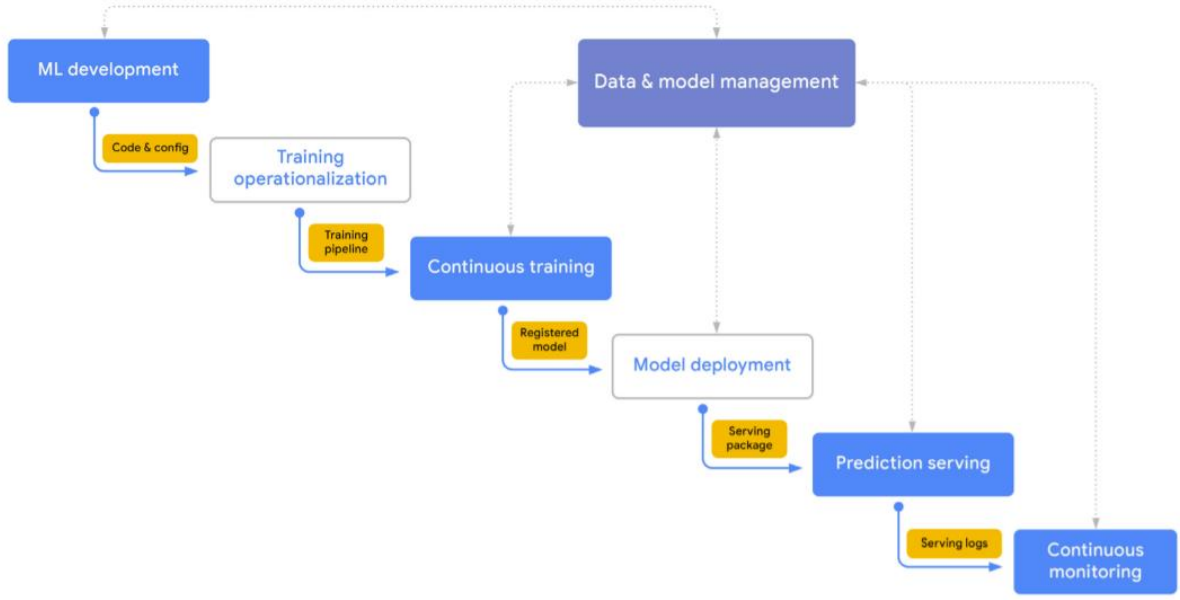

Figure 2-2: The MLOps process from (Salama, 2021)

Although these seven steps, potentially with variations in terminology and/or exact tasks' breakdown, are followed by any company providing an ML-enabled product/ service, the level of automation in their realization varies. Three distinct levels, representing three stages of MLOps automations, have been defined: manual implementation, ML pipeline automation, and continuous integration/continuous delivery of principles.

Manual Implementation (level 0)

The manual implementation refers to a setup in which MLOps principles are manually applied. This level is script-driven and iterative. All steps described in section 2.2.1.1 are manually implemented, and engineers manually integrate the trained models into the required application. The common way to process is to use Rapid Application Development (RAD) tools, such as Jupyter Notebooks (Visengeriyeva, 2021). Data scientists build, train and test their models, create a model class that will be pushed to a code repository, and then engineers integrate this model into an application or system. Any change in the data or settings will lead to a new manual model development, having to repeat the entire process again.

Another important characteristic of level 0 is the disconnection between data scientists and engineers. First, data scientists are responsible for processing the raw data, performing data analysis, building training and validating the model. Then in the next phase the engineers are responsible for manually deploying the model and for making the required model services available in production.

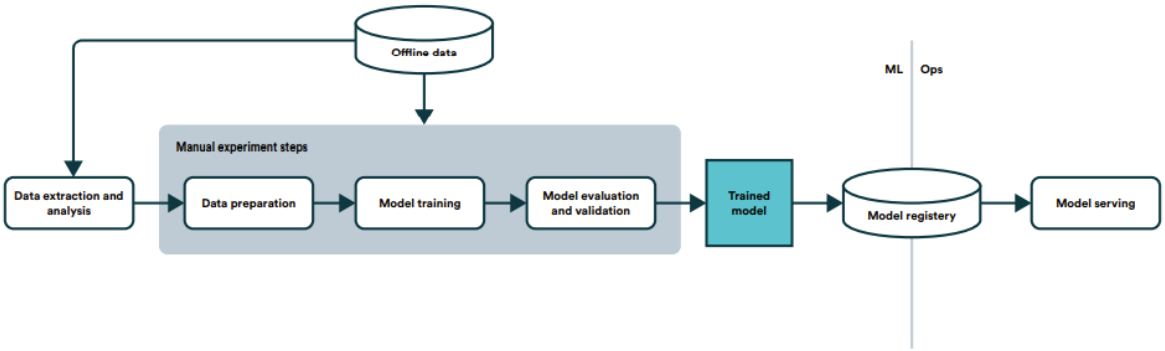

Figure 2-3: MLOps Level 0 from (Valohai, 2021)

There are several challenges that arise with this level of implementation (Google, 2021):

- There is no Continuous Training (CT) and therefore, models must be frequently retrained manually so that they are up to date with new data that arrive.
- There is no active performance monitoring, which might result in poor performance and efficiency.
- There are also infrequent release iterations, due to the time and resource requirements for creating and deploying an updated ML model.
- The deployment process is only concerned with the trained model as a prediction service, rather than deploying the entire AI pipeline (Google, 2021).

ML pipeline automation (level 1)

This setup contains ML pipelines in order to automate the process of training ML models and therefore achieve continuous delivery of model prediction service. In order to achieve this, the steps of data and model validations are automated, along with pipeline triggers and metadata management (Visengeriyeva, 2021).

The first characteristic of this level is quick experimentation, as all steps from section 2.2.1.1 are fully automated and orchestrated. This level:

- Enables faster experimentation and facilitates moving AI pipelines from development to production.
- Allows for continuous training of the production models, as the training procedure is fully automated and can be repeated when new data arrive.
- Provides continuous delivery of models in production, as the model deployment step, which serves the trained models as prediction services is automated.
- Enables deploying the whole training pipeline (in contrast with level 0 where only a trained model was deployed), therefore the trained model can automatically and recurrently run to serve the prediction service.

Continuous Integration (CI) / Continuous Delivery (CD) of Pipelines (level2)

In the final level, a CI/CD system performs fast and reliable AI model deployments in production, where the components of the ML models, the training pipelines and the data are now automatically built, tested and deployed.

An important aspect that should be highlighted is that even at this higher level of automation, an experimentation stage is foreseen, during which the data scientist can iteratively try out new ideas, ML algorithms, architectures and feature engineering components.

After this stage, the source code of the ML pipeline steps is pushed to a source repository. Continuous Integration allows to build source code and run various tests, and then output the pipeline components (packages, executables, and artifacts) to be later on deployed. Continuous delivery deploys the artifacts produced in CI stage to the target environment, and outputs the deployed pipeline with the new model. Automated triggering allows the pipeline to be automatically executed in production based on either a schedule or in response to any trigger, and outputs the retrained model that is pushed to the model registry. Model continuous delivery then provides the trained model to be used as a prediction service, where its performance is monitored by appropriate statistics.

The main advantage of this setup, compared with level 1, is automated pipeline deployment with testing and monitoring. Rapid pipeline creation and deployment along with continuous delivery of model services, allows engineers to keep up with significant changes in the data (requiring new models and pipelines), while also capitalizing on the new trends and architectures.

Figure 2-4 depicts a level 2 implementation of an AI pipeline, according to the approach proposed by Google who also coined the three MLOps levels. Figure 2-5 shows an adaptation of the original model by Valohai, to highlight the fact that there is no single best practice, but the adopted pipeline for managing AI pipelines should be appropriately designed to address the needs of the team responsible for the AI project and the context in which it is developed and served.

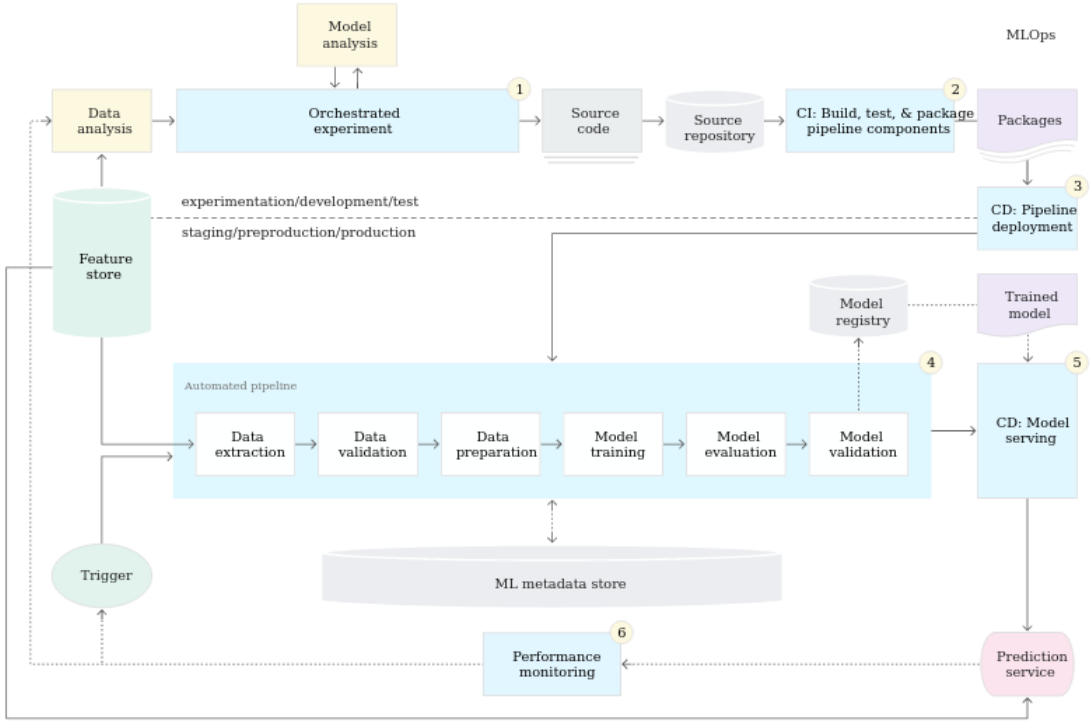

Figure 2-4: Level 2 MLOps from (Google, 2021)

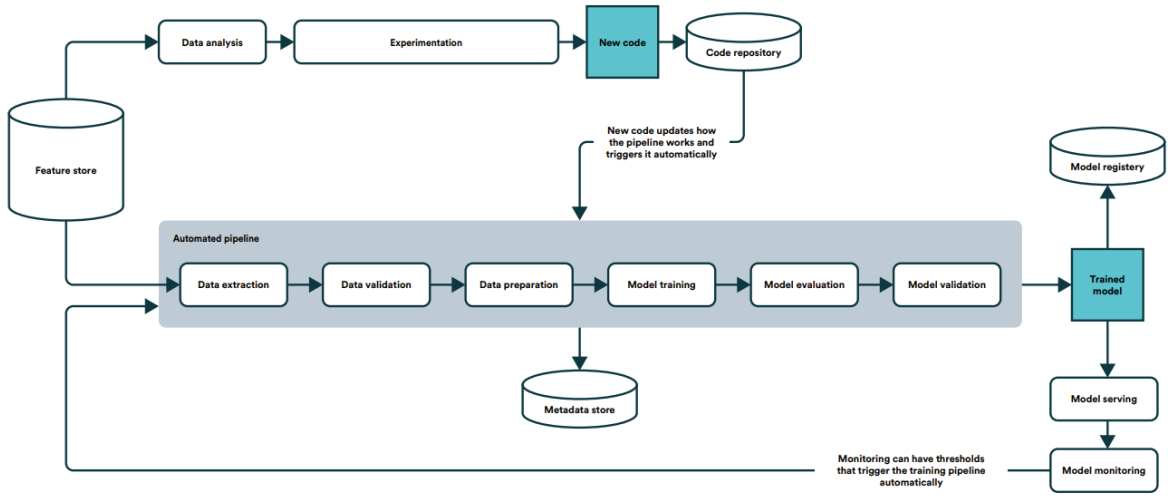

Figure 2-5: Level 2 MLOps from (Valohai, 2021)

There is a set of additional technical capabilities that need to be established to achieve the full capacity of MLOps levels 1 and 2 that were previously mentioned and can also be seen in Figure 2-4 and Figure 2-5. These are briefly examined below in terms of important processes and methods. Relevant tools and technologies will be explored in section 2.3.

- **Feature Stores:** Feature stores are centralised repositories that contain standardized data processed into features. Relevant technologies are examined in section 2.3.4.1, but it is important to understand the processes that these tools bring/ facilitate in the AI pipelines lifecycle. Discovery and reuse of the available features to any step within the pipeline, from experimentation, to continuous training, to online serving (Valohai, 2021) is a common need in such settings. Therefore, having guarantees that the same features used for training are the same ones used for serving, while avoiding having similar features with different definitions, can improve collaboration between data scientiests and data engineers and facilitate debugging.
  One of the main advantages of having such processes in place is the way "temporal operations" in AI pipelines are facilitated. As explained in the beginning of section 2.2.1, **a version of the code and a version of the data together** produce a version of the ML model. Therefore, data scientists often need to:
  - Make changes to already created features, i.e. in a sense perform a rollback.
  - Monitor how features evolve over time and timely detect drift or training-serving skew.
  - Go back in time and retrieve a value, a challenging process considering that features are generated through other operations (information extraction and combination from raw data, aggregations, etc.). Without the ability to retrieve a point in time snapshot, this would be an extremely time-consuming and custom process.

  Feature governance, including transparency and auditing, is also enabled through the usage of such technologies, highlighting the fact that automation of some traditionally manually handled processes is gaining attention, as the use of AI finds its way into the operations of more domains and stakeholders.

- **Experiment Tracking**: Developing performant machine learning often involves iterating over several experiments. These experiments need to be tracked for comparability and reproducibility. Experiment tracking is the process of managing all the different experiments and their components, such as parameters, metrics, models and other artifacts and metadata (code and data versions) and it enables us to (Google, 2021):

- Organize all the necessary components of a specific experiment.
- Reproduce past results (easily) using saved experiments.
- Log iterative improvements across time, data, ideas, teams, etc.

ML Experiment (and metadata) Tracking is foundational to all other MLOps capabilities, as it enables reproducibility of results. Additionally, it allows organization of all the necessary components of a specific experiment and logging of improvements and ideas.

- **Model Monitoring:** One of the most important aspects of the AI pipeline is continuous monitoring of the efficiency and effectiveness of the production model. This is important, because as there are data changes over time, the properties of the model start to deviate (since it was trained with a different set of datasets), therefore the model's performance degrades.

  The first category of monitoring tests should be around features and data in general. Data version and dependency changes should be monitored at all times, along with data invariants and numerical stability (Visengeriyeva, 2021). As suggested by (Breck, 2017), there are several tests for features and data that can be implemented. First of all, feature expectations should be captured in a schema, by calculating statistics and adjusting them on domain knowledge, while also capturing invariants (Hsu, 2003). (Section 3 will provide more insights on how XMANAI leverages data models in this context). Correlation coefficients should also be calculated in order to select the features that bring the most value; since using all features is costly and should be avoided. Outlier detection should also be performed as well as feature attributions change. This procedure also helps erase the problem of data drift. Data drift describes a growing skew between the original dataset used to train the model and the dataset that is now used in production for predictions (Salama, 2021) .There are two types of skews, schema and distribution skews. Schema skew is when the training and production data do not have the same schema, and distribution schew is when the training data feature distribution deviates from the production data feature distribution ((Klaise, 2020),(Muralidhar, 2021)). Alerts should be generated when the data and schema don't match, or when there are undesired values (eg. NaNs). Moving on, the cost for each feature should also be monitored by calculating computing resources (latency and RAM) required; since keeping each feature (even not important ones) is also an ongoing maintenance burden (Sculley, 2014). There should also be sufficient time to allow for proper handling of new feature developments, in order to care for privacy controls (eg. sensitive user data), especially for data sources that have not previously been used in ML.

  The second category for monitoring should be around model development, where for example A/B experiments can help monitor offline/online metric relationship (Breck, 2017). A/B tests can also help assess the impact of model staleness by regularly testing against older model versions. Added to that, comparisons between simple baseline models and tests on slices of new data will confirm the functionality and benefits of the model, as well as justify the costs (Breck, 2017) .

  Finally, not only the model effectiveness should be monitored, but also the model serving (infrastructure) effectiveness. This includes metrics like resource utilization (CPUs, GPUs, memory), latency, throughput and error rates (Visengeriyeva, 2021). Integration tests on a ML pipeline should also run both continuously as well as with new releases of models (Breck, 2017) .

It becomes evident that AI pipelines cannot be thought of as a set of independent (or even close to) steps. Processes are closely linked and interdependent: effective monitoring in production requires mechanisms across all steps, from data ingestion to result consumption, reproducibility requires changes in the way data and models are handled, etc and this is why adopting MLOps practices can make a difference.

2.2.2 Data manipulation in AI Pipelines

Data preparation (manipulation) is considered as one of the most critical phases on any applied machine learning (ML) process. Data preparation, or data preprocessing as it is also referred to in literature, constitutes the method of transforming and preprocessing the collected raw data into the most appropriate form for the modeling and analysis phase. It is a fundamental and crucial phase and usually constitutes the most time consuming phase of the overall process (TowardsDataScience, 2021).

Data preparation has been a long-standing challenge in data science to avoid incorrect results and misleading conclusions obtained from inaccurate and coarse data (Berti-Equille, 2019). The importance of the required data preparation phase is easily acknowledged by the nature of ML algorithms, some of which impose the hard requirement of the input data being numerical, even when this is not their "natural" representation. On top of that, many ML algorithms impose requirements on the nature and structure of input data, such as expected data types, their scale and the relationship between the various included variables, depending on their implementation or scope. Additionally, raw data usually contain erroneous information, noise, outlier and missing values that decrease their value and should be properly corrected in order to get the maximum value and most accurate results. Finally, in some cases, raw data tend to be rather complex with high dimensionality which can again decrease their value in the applied ML process.

Data preparation incorporates multiple steps that differentiate on each separare ML process mainly due to the differences in the initial raw data types and structures but also due to the different requirements imposed by each ML algorithm for the required input data. Nevertheless, the various steps of the data preparation can be categorized as follows (Brownlee, 2020):

- **Data Cleaning**: The specific tasks refer to the methods applied for the identification and correction of erroneous information or errors in the data.
- **Feature Selection**: The specific tasks refer the methods applied for the identification of the most relevant variables for the specific ML process.
- **Data Transforms:** The specific tasks refer to methods applied for the change of the scale or distribution of the variables.
- **Feature Engineering**: The specific tasks refer to methods applied for the derivation of new variables from the existing data.
- **Dimensionality Reduction**: The specific tasks refers to the methods applied for the creation of compact projections of the existing data.

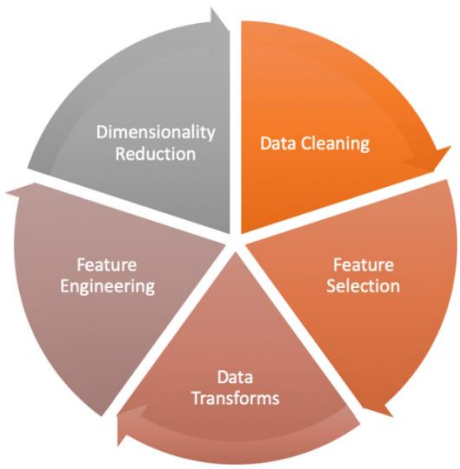

Figure 2-6: Data Preparation for Machine Learning

Data Cleaning is an umbrella of tasks related to the detection, correction or even removal of corrupted or erroneous information from the data. It aims at eliminating the errors or possible noise included in the data towards the increase of their accuracy, correctness and completeness, as well as at maximizing their usability, quality and integrity. Data cleaning constitutes a crucial part of the data preparation as it corrects or removes incomplete or incorrect information of the corresponding data, setting also the basis for the next steps of the data preparation towards the effective data analysis. To fix or eliminate the errors different phases are performed. At first, the identification of the possible errors is performed that provides insights of the included information as well as of the corrective actions that should be performed. In the second step, the identified errors are corrected or removed and missing values are imputed where needed. In the final step, the corrected or imputed data are verified as an evaluation of the performed data cleaning operation. It should be noted that all the steps performed require advanced expertise and deep knowledge of the domain from which the information is included in the data in order the data cleaning process to be an effective and efficient process. While the performed actions are dependent to each specific domain and to the information included in the data, there are several techniques which can be performed in most of the cases:

- *The outliers detection and the correction of these outlier*s with a regular or reasonable value. To achieve this, a variety of basic statistic methods are applied in order to identity the various anomalies, extreme values or outliers. Depending on the nature of the data and the distribution of the included values multiple algorithms can be utilised (Agrawal, 2015; Thumudu, 2020):
    - The nearest-neighbor based algorithms such as k-NN, Local Outlier Factor, Connectivity-based Outlier Factor, Local Outlier Probability, Influenced Outlierness, Local Correlation Integral.
    - Clustering based algorithms such as Cluster Based Local Outlier Factor and Local Density Cluster based Outlier Factor.
    - Classification Based Algorithms such as Neural Networks, Bayessian Networks, Decision Trees.
    - Statistic Based Techniques such as Parametric Techniques and Non-Parametric Techniques.
- *The identification of duplicate values or records and their removal*. As it obvious, any columns that contain the exact same value on all records of the data or duplicate records have no significant value on an analysis. Hence, it is considered as standard approach that duplicates are removed from the data.
- *The identification and correction of the missing values.* The specific action is considered the most critical and challenging one. As it obvious again, empty values especially on mandatory fields or columns are decreasing the value of the performed analysis. The specific action is related to their identification at first and as a second step their removal from the data or the imputation of values with techniques such as basic statistic methods (mean, median, most frequent value) or with a predefined value for non-time series data, Last Observation Carried Forward (LOCF) and Next Observation Carried Backward (NOCB) and Linear Regression for time-series data and more advanced data imputation through ML models such as the k-Nearest Neighbours and MissForest algorithms (Kaggle, 2021).

From the above it becomes evident that data cleaning cannot and should not be considered as a process needed only prior to the data ingestion, i.e. data cleaning does not fall exclusively under WP2 methods, as in many cases the right choices depend on the underlying analysis task and therefore on the data scientists' choices, which may even vary for the same data used in different contexts and models.

Data harmonisation is sometimes considered a special case of the data cleaning functionalities. It describes the process of combining data from different data sources and harmonising the individual features of a dataset, resulting in a more comparable dataset. Harmonisation is often used in the same

context as data simplification and standardisation. While data standardisation is more related to the conformity of the data, data harmonisation improves the consistency of the data provided. In data standardisation, the information is converted into national, regional or international standards. Data simplification, on the other hand, eliminates unnecessary and redundant information from the data.

Data harmonisation is in general perceived as a process of data integration: Information from different data sources is integrated into a single system and after that point the data type, file types or any naming conventions are not important.

A typical example is the harmonisation of date formats. Several countries use different date formats (Figure 2-7 presents in a visual way the problem), thus a common format needs to be agreed upon when data are expected to be ingested in the same system and processed by the same components.

YYYY-MM-DD	1970-01-01
MM/DD/YYYY	01/01/1970
DD. Month YYYY	01. January 1970
DD Month YY	01 January 70
...	...

Figure 2-7: Example of various presentation formats of a date

The way a data model can help in this direction prior to data ingestion is discussed in D2.1 as well as in section 3 of the current deliverable that provides insights into the way XMANAI leverages data models. Still, in subsequent steps of AI pipelines, non-obvious data harmonisation actions may be required and more importantly derivative data should be appropriately harmonised to ensure that they can also be properly integrated with the original (raw) data as well as with other derivative data from the same or other pipelines.

Feature selection is a group of methods applied for the effective selection of the subset of features incorporated into the data which are considered as the most relevant and with the highest value for the upcoming data analysis. To this end, driven by the domain knowledge and expertise of the data scientist, the features which are not considered as relevant and that could compromise the quality of the performed analysis and as a consequence of the produced results are eliminated. The feature selection methods can be classified into three major categories:

- *Filter methods* (Landset, 2015) which utilise independent techniques from the subsequent ML algorithms that will be used in the data analysis. In these methods, the appropriate set of features are selected through an evaluation criterion or a score which is used to the assess the degree of relevance of each feature to a target value. The most widely used techniques are based on Pearson's Correlation, Linear Discriminant Analysis (LDA), Analysis of Variance (ANOVA) and Chi-Square.
- *Wrapper methods* (Venkatesh, 2019) which utilise a predictive trained model to evaluate the accuracy of a subset of features to perform the feature selection. In this case, the evaluation is performed through the classification of the relevance of the selected subset. The most widely used techniques are the forward selection, backward elimination and the recursive feature elimination.
- *Embeded methods* (Venkatesh, 2019) which perform the search for the optimal subset of features through a classifier that is a combined space of features subsets and hypotheses. Embeded methods combine the advantages of Filter and Wrapper methods to achieve the aspired selection. The most widely used techniques are Lasso regression and Ridge regression.

Data Transforms refers to the set of methods utilised to transform the type of input data or the distribution of the included variables. As data can be classified into numeric data types (typically integer and float numbers) and categorical data types (ordinal with rank ordering, nominal without

rank ordering and boolean) there is a need to transform the input values into the appropriate type that is suitable for the upcoming data analysis but also in a compliant input type for the selected ML algorithm. To this end, the main categories of data transforms can be grouped as follows (Brownlee, 2020):

- *Discretization Transform*: In this method, a numeric variable is encoded as an ordinal variable hence changing its distribution in order to be used into a data analysis.
- *Ordinal Transform*: In this method a categorical variable is encoded as an integer variable, hence encoded to numbers before it is used in the data analysis.
- *One Hot Transform*: In this method, a categorical variable is encoded as an boolean variable, before it is used in the data analysis.
- *Normalization Transform*: In this method, the input variable is normalized hence scaled in the range from 0 to 1.
- *Standarization Transform*: In this method, the input variable, provided that it has a Gaussian probability distribution, is scaled to a standard Gaussian distribution with a mean of observed values 0 and a deviation set to 1.
- *Power Transform*: In this method, the numerical input variable's distribution is altered to follow the Gaussian probability distribution.
- *Quantile Transform*: In the method, the numerical input variable's distribution is altered to a different distribution such as the Gaussian or uniform probability distribution.

Feature Engineering is a group of methods that facilitate the generation of new features from the existing data. Feature Engineering is typically used to improve the performance of the trained ML model by introducing new features that could expose the hidden relationships between the existing variables. While data transforms are also part of feature engineering in most cases, there are additional approaches to unveil new information from the available data.

These additional methods include the addition of a new variable on the set of existing variables whose values are set automatically based on a set of rules, such as adding a boolean value on the new feature when several conditions are met or to use statistical methods such as the mean or median value for the values of the new feature. In addition to this, the most widely used approach is the Polynomial Transform method. In this method, the new feature's value are generated by raising the input values of a variable to an exponent such as its square value or the multiplication of the values of two different features. These features are referred to as Polynomial Features and they are utilised to increase the performance of the utilised ML algorithms, nevertheless their usage depends on the selected algorithm and the distribution of values of the input data while requiring extended domain knowledge of the input data.

Dimensionality Reduction refers to the set of methods utilised to generate a representation of the input data into a space with a lower number of dimensions. Typically, the dimensionality of the input data is expressed by the number of variable included. Hence, as more input variables are included, the dimensionality of the input grows. However, this creates problems into the data analysis as complexity is also increased. Dimensionality reduction comes as an alternative to feature selection to increase the quality of the performed data analysis and of the produced results. The methods preserve the most important features of the input data by creating a projection of the data. In general, the various dimensionality reduction methods can be classified as follows:

- *Matrix Factorization* methods which performs the reduction of the dimensionality by breaking down the data matrix into its core parts, which are then ranked and selected in order to maintain the most suitable representative ones. The most commonly used methods are the Principal Component Analysis (PCA) and the Singular Value Decomposition (SVD) methods.
- *Manifold Learning methods* which generate compact projections of high-dimensional data utilizing high-dimensionality statistic methods. The most commonly used methods are

Kohonen Self-Organizing Map (SOM), Sammons Mapping, Multidimensional Scaling (MDS) and t-distributed Stochastic Neighbor Embedding (t-SNE).

- *Linear Dimensionality Reduction (LDA)* method that is ML approach for dimensionality reduction which leverages this predictive modeling algorithm for multiclass classification. LDA performs dimensionality reduction by finding the linear combination of the predictors in which the between-group variance was maximized relative to the within-group variance.

Dimensionality reduction will be further explored in the context of visualizing high-dimensional data in section 2.2.3.1.

2.2.3 Visualisation & Visual Exploration

Visualization and Visual Exploration can be defined in two separate directions. Visual exploration from **the data point-of-view** is the process of understanding the characteristics of the available data using appropriate visualization techniques. Visual exploration from **the model point-of-view** describes the process of explaining the model's behavior, i.e., understanding what a Machine Learning model has learned throughout the training phase, using appropriate visualization techniques. Both directions share the common characteristic of incorporating *a suitable visualization technique* for helping the user understand part of a Machine Learning process. Their difference is whether they focus on explaining the data or explaining the model.

2.2.3.1 Visual exploration from the data point of view

Visual exploration from the data point-of-view aims at providing a visual description of the data. Typically the visualization is the outcome after an appropriate summarization has been applied to the raw data. Each visualization technique is compatible with a specific type of input data. Most of the methods we presented apply to generic tabular data; different types of input data, such as images or plain text, require specific techniques. Furthermore, some visualization techniques are applicable either to categorical or continuous data. Since it is possible to transform contiguous features to categorical using quantization, the methods for categorical features can be applied to all cases using this intermediate step.

2.2.3.1.1 Numerical descriptive statistics

Numerical descriptive statistics aim to describe the characteristics of an input feature through a numerical value. Below, we present an overview of the essential descriptive statistics. Location measures provide answers to questions about the general location of the data. The sample mean (1) is the most well-known and most commonly used location measure. A robust to outliers alternative for measuring the location is the median (2).

$$\bar{x} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}}^{\mathrm{N}} \mathrm{x_i} \tag{1}$$

$$median(x) = \begin{cases} x_{(n+1)/2}, if\ n\ is\ odd \\ \frac{1}{2}\left(x_{(n/2)} + x_{(n/2+1)}\right), if\ n\ is\ even \end{cases} \tag{2}$$

The following two numerical values describe the scale or spread of the data; Variance (3) is the measure of deviation from the mean. The mean absolute deviation (4) is also a measure of spread, which is more robust to outliers. Finally, the inter-quartile computes the range between the smaller 25% of the values and the higher 75%.

$$var(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 = \frac{1}{N}\sum_{i=1}^{N} {x_i}^2 - \mu^2 \tag{3}$$

$$MAD(x) = median(|x_i - median(x)|) \tag{4}$$

$$IQR = Q_3 - Q_1 \tag{5}$$

The last class of descriptive statistics defines the shape of a specific feature. The skewness (4) describes whether a feature x is uniformly spread around the mean value. For example, positive skewness means that the distribution x is more spread to the right side of the mean (longer right tail). Galton's measure of skewness (7) is a robust-to-outlier alternative. Finally, the kurtosis (8) measures how often x takes on values that are considerably larger or smaller than its standard deviation.

$$skew(x) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i - \mu}{std(x)}\right)^3 \tag{6}$$

$$Galton'smeasure = \frac{(Q_3 - Q_2) - (Q_2 - Q_1)}{Q_3 - Q_1} \tag{7}$$

$$kurt(x) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i - \mu}{std(x)}\right)^4 \tag{8}$$

2.2.3.1.2 Visualizations of the descriptive statistics

Based on the numerical values described above, different visualizations can be created to facilitate understanding of the input data.

2.2.3.1.2.1 Bar plot

A bar plot is a collection of bars with different heights. The heights of the bars are proportional to the values that they represent, for instance, the number of occurrences n_j of the attribute v_j in data $x_1, \dots, x_n$. If we compare populations of different sizes, it is convenient to use the relative frequency of an attribute $f_j = \frac{n_j}{n}$.

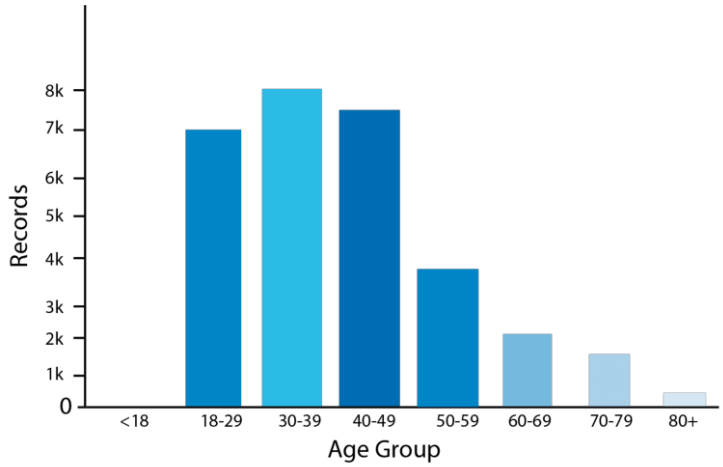

Figure 2-8 Illustration of a bar plot graph

2.2.3.1.2.2 Histogram

The histogram plot is the extension of the bar plot for continuous features. The range of observations are firstly divided into k-non overlapping and successive bins $B_1, B_2, \dots, B_k$. Then, we count the

number of observations that fall into each bin. There are multiple ways to define the successive bins. The simplest and most frequent one is dividing the range of values into equal size bins.

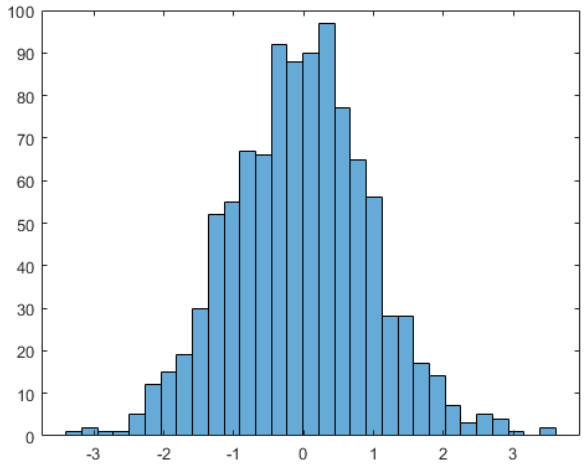

Figure 2-9 Illustration of a histogram graph

2.2.3.1.2.3 Kernel density plot

Often, we can make the hypothesis that the distribution of the observations is continuous and smooth. By default, a histogram uses a discrete number of bins that lead to a discontinuous graph. Histograms, therefore, will generally misrepresent distributions that are continuous and smooth. To remedy this shortcoming, we can estimate the density as a continuous function and then plot our estimate. To assess the probability density, we can use a kernel density estimate:

$$p(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i)$$

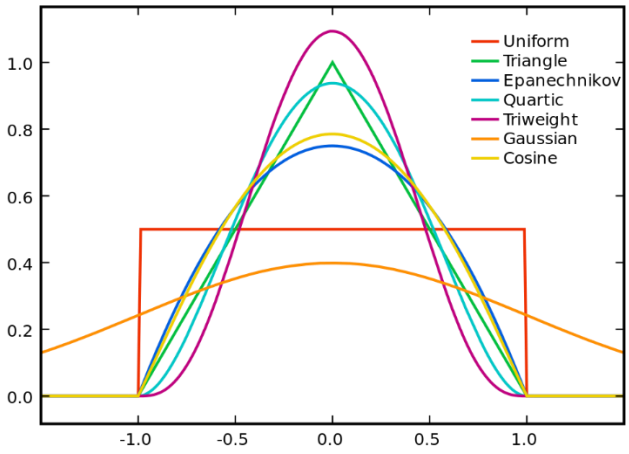

Figure 2-10 Graph of smooth histogram with kernel density estimation

2.2.3.1.2.4 The violin plot

The Violin plot combines a box plot and a rotated/reflected kernel density plot that is attached to both sides of the box plot. It describes robust (box plot) and non-robust but more informative (kernel density plot) aspects of the samples in a single plot.

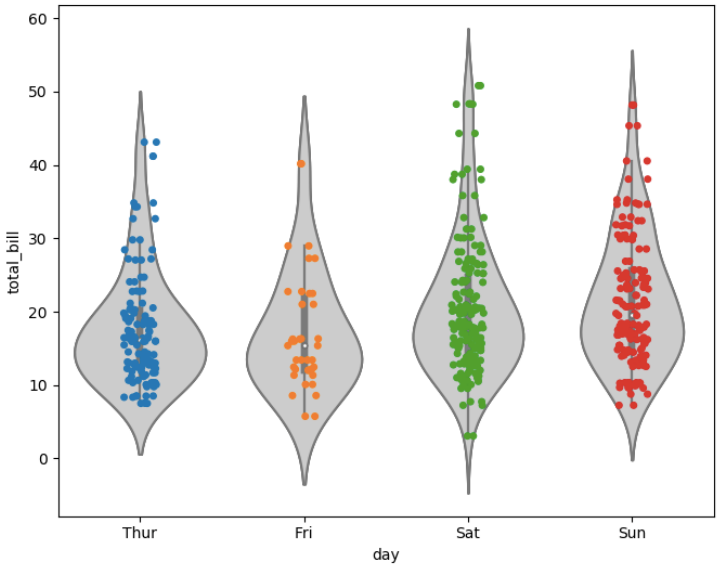

Figure 2-11 Illustration of a violin plot

2.2.3.1.3 Descriptive statistics to capture correlations among features

The statistics we described above show characteristics of a single-dimensional feature. Still, they cannot capture the relationship between variables, i.e., how one variable changes in relationships with the value of another variable. The numerical quantity that measures this correlation is called covariance (1), and its normalized alternative (values in the range -1 to 1) is the Pearson correlation (2). Finally, there are cases where two variables depend on the other but in a nonlinear fashion. This can be numerically computed through a nonlinear transformation g, before applying the Pearson correlation formula, as in (3).

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y - \bar{y}) \quad (1)$$

$$\rho(x,y) = \frac{cov(x,y)}{std(x)std(y)} \quad (2)$$

$$\rho\big(g(x),g(y)\big) = \frac{cov\big(g(x),g(y)\big)}{std\big(g(x)\big)std\big(g(y)\big)} \quad (3)$$

As before, a practitioner can easily understand the correlation between two features through an appropriate visualization. Figure 2-12 shows many examples of relationships between two attributes and the corresponding Pearson coefficient. In Figure 2-13, we present the so-called pair-plot matrix, a sequence of pair-plots among all features. The plot in the $i-th$ row and $j-th$ line is the pair plot between attributes x_i and x_j. Hence, in the diagonal of the matrix, we plot the histogram of each feature.

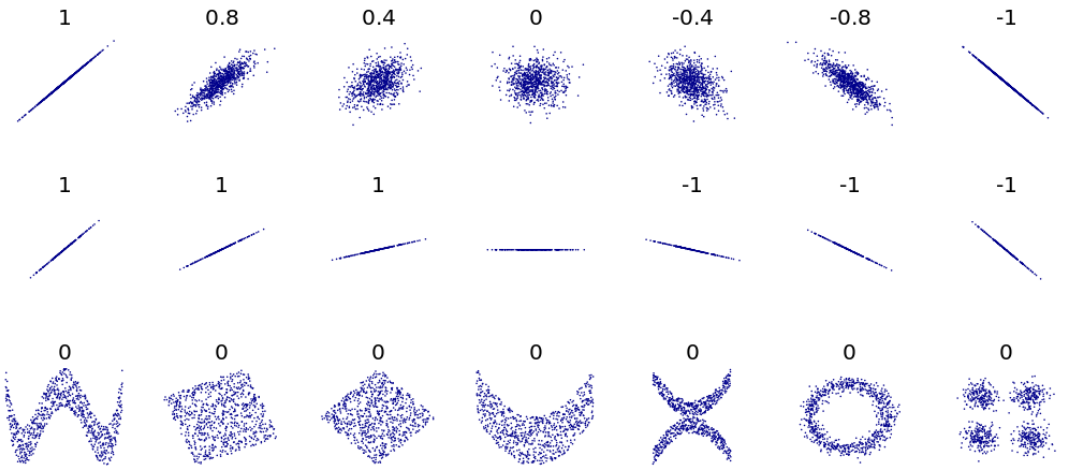

Figure 2-12 Illustration of correlations between two variables, where the Pearson coefficient varies from -1 and 1.

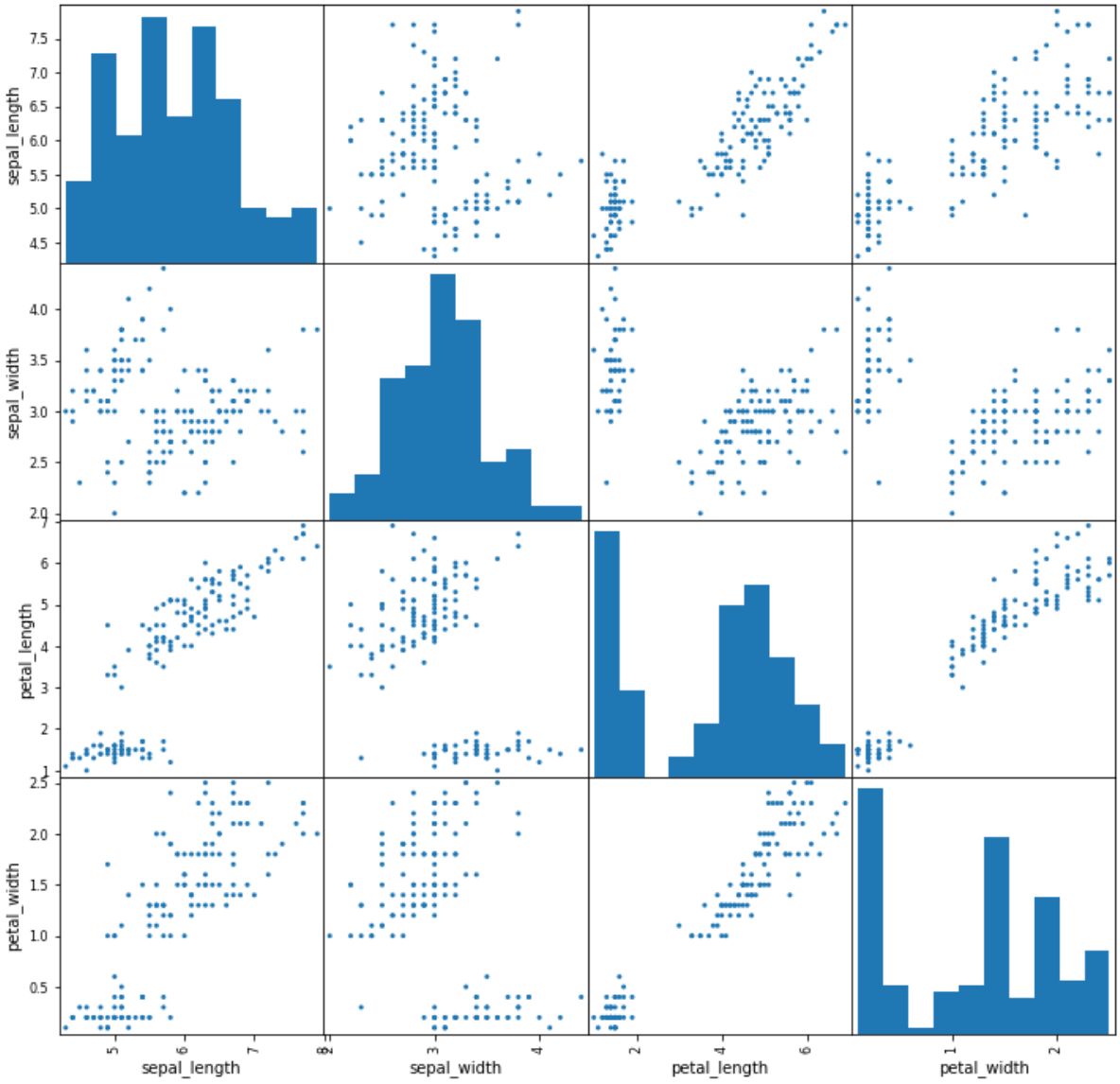

Figure 2-13 Example of a table of pair-wise plots between all the features

2.2.3.1.4 Dimensionality Reduction

Dimensionality reduction is about representing the data in a lower-dimensional space so that specific properties of the data are preserved as much as possible (Izenman, 2008). Dimensionality reduction can visualize high-dimensional data if the plane is chosen as the lower-dimensional space. This section will present three nonlinear dimensionality reduction methods; PCA., t-SNE, and UMAP.

Dimensionality reduction is a way to project a high-dimensional input to a lower-dimensional space.

$$X_{low} = f_1(X), where\ X_{low} \in R^{D_1} and\ X \in R^{D_2}\ and\ D_1 < D_2$$

A way to measure the accuracy of the procedure is to reproject the data back to the initial dimension and calculate the error (distance) between the initial and the reprojected ones. Principal component analysis (PCA) is the best way in terms of mean square error for applying a linear projection.

In many cases, to lower the dimension without much information being lost, it is imperative to use a nonlinear projection. In the general case, any f_1 that projects the input to a lower dimension can be used for dimensionality reduction. For example, we may train a Neural Network with two parts; the first one lowering the dimension f_1 and a second one $f_2$for reprojecting to the initial dimension. We can train by minimizing the reprojection loss $L = || f_2(f_1(X)) - X ||$, and then isolate the f_1 part as the dimensionality reduction function. Apart from this generic framework that can be trained from scratch, there are two more cases of particular interest, t-SNE, and UMAP.

The t-SNE algorithm (L.J.P. van der Maaten, 2008) is a nonlinear dimensionality reduction approach. At a very high level, t-SNE calculates a similarity measure between pair of instances in both the high-dimensional and the low-dimensional space. It quantifies the difference between those two instances to minimize it. Hence, t-SNE is a so-called similarity preserving technique. In general, t-SNE is a helpful technique, although sometimes its performance suffers from large datasets and is very sensitive to outliers.

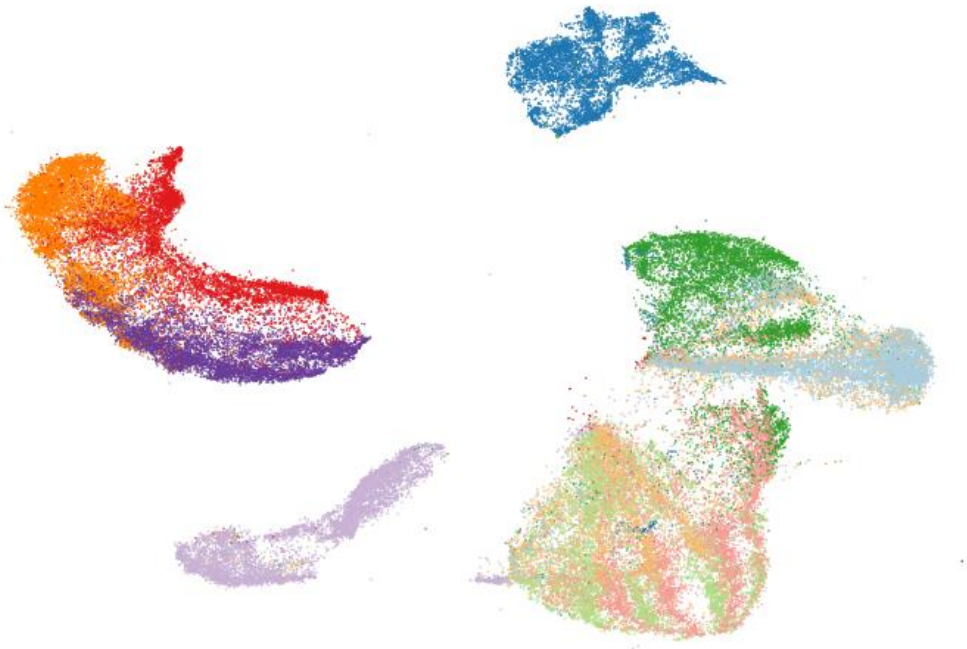

Figure 2-14 Illustration of t-SNE projection to the fashion MNIST dataset

Uniform Manifold Approximation and Projection (UMAP) (McInnes, 2018) is a dimension reduction technique that can be used for visualization similarly to t-SNE, but also for general nonlinear dimension reduction. The algorithm is founded on three assumptions about the data:

1. The data are uniformly distributed on a Riemannian manifold
2. The Riemannian metric is locally constant (or can be approximated as such)
3. The manifold is locally connected

From these assumptions, it is possible to model the manifold with a fuzzy topological structure. The embedding is found by searching for a low-dimensional data projection with the closest possible topological structure. In general, UMAP has some significant advantages over competitive methods. To name some of them, UMAP is generally quicker than t-SNE, can better preserve the global structure, and enables progressively add new data to an already existing projection.

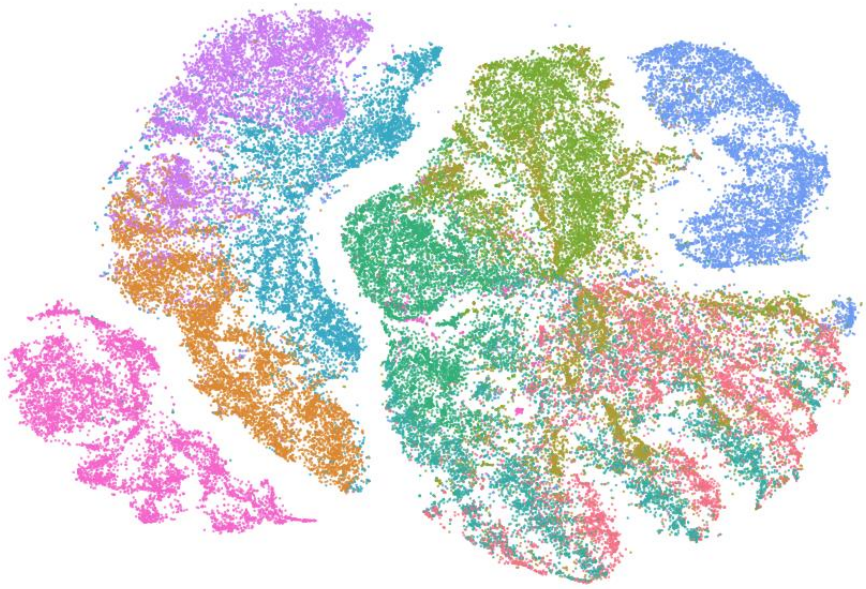

Figure 2-15 Illustration of the UMAP projection to the fashion MNIST dataset

2.2.3.2 Visual exploration from the model point-of-view

Visual exploration from the model point-of-view covers the visualization techniques that explain how a black-box model works. There is a wide range of different explainability techniques (Arrieta, 2020), with the XMANAI Deliverable D1.1 having already provided an extensive review of these methods. In this chapter, we don't focus on an ample enumeration of the existing explainability techniques. Instead, we focus on the visualization techniques employed by some of these methods to make the explanation more conceivable.

2.2.3.2.1 Feature effect

Feature effect visualization techniques aim at provide intuition on the impact (importance) of each feature on the output prediction (Apley, 2017). In mathematical terms, if $y = f(x)$ is the black-box model, feature effect techniques try to isolate the impact of a specific feature to the output $y = \tilde{f}(x_i)$. The appropriate visualization technique varies depending on the input type. If the input is tabular data, the appropriate visualization technique is a simple one-dimensional graph where some dots are added on the horizontal axis to represent where many input points exist, as in Figure 2-16. Another way to explain a specific input feature is to depict its importance in the models accuracy. Here we aim to visualize how much the accuracy of a black-box model would have changed if a particular feature was absent.

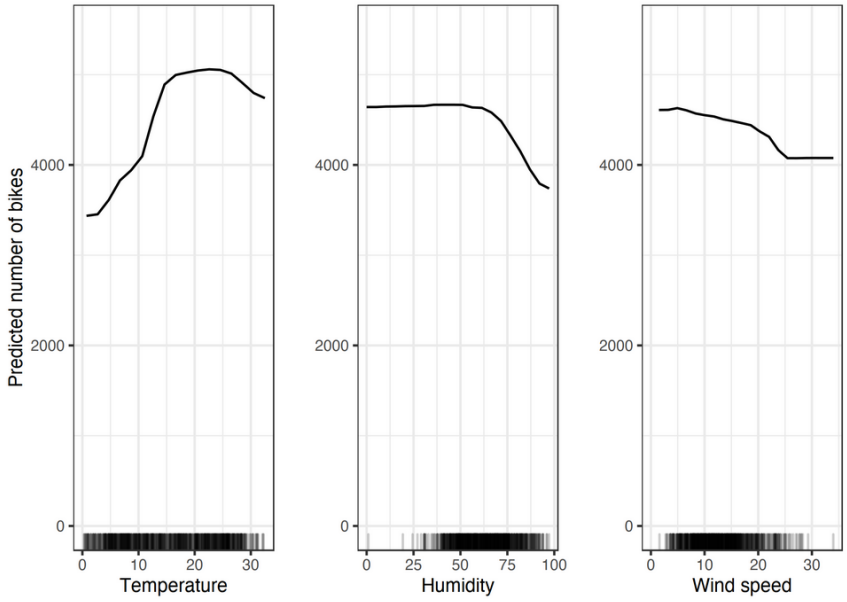

Figure 2-16 Feature effect visualization when the input is simple tabular data. Image taken from https://christophm.github.io/interpretable-ml-book/pdp.html

Figure 2-17 illustrates an appropriate visualization for feature importance. In case input data are in the form of an image the feature importance in measured per example (not globally) and is visualized with an appropriate saliency map.

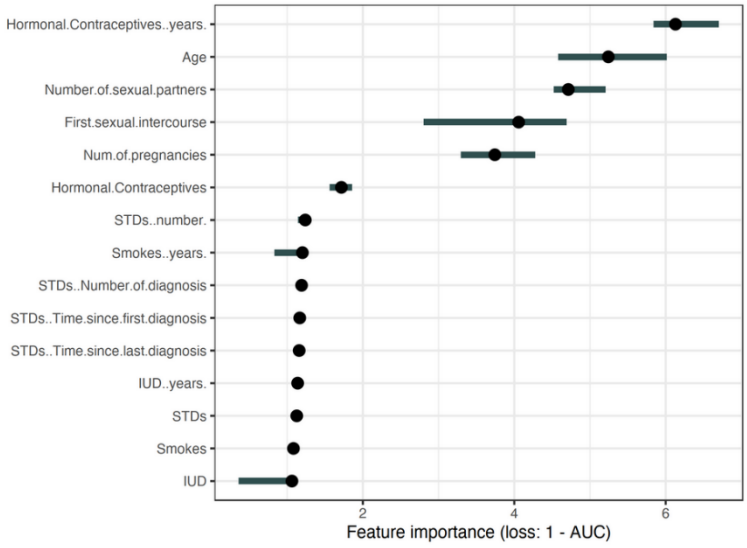

Figure 2-17 Feature importance graph. Image taken from https://christophm.github.io/interpretable-ml-book/feature-importance.html

In Figure 2-18, we illustrate such a visualization computed using the SHAP values (Lundberg, 2017).

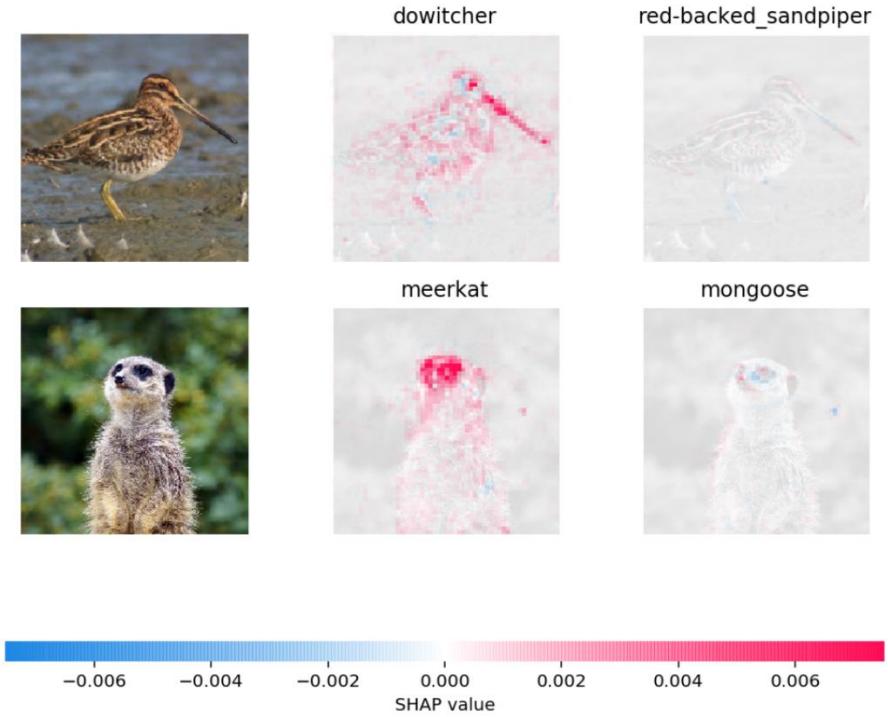

Figure 2-18 Saliency map visualization using the SHAP values method.

Correspondigly, if the input is plain text, we use a dedicated Natural Language Processing technique (Leila Arras, 2017) for producing an appropriate visualization, as in Figure 2-19.

true	predicted	N°	Notation: -- very negative, - negative, 0 neutral, + positive, ++ very positive
		1.	do n't waste your money .
		2.	neither funny nor suspenseful nor particularly well-drawn .
		3.	it 's not horrible , just horribly mediocre .
		4.	... too slow , too boring , and occasionally annoying .
		5.	it 's neither as romantic nor as thrilling as it should be .

Figure 2-19 Feature Relevance example in the N.L.P. domain. Stronger red corresponds to words that contributed more to the prediction. Image take from (Leila Arras, 2017).

Finally, another class of explainability techniques targets on searching the dataset for characteristic examples (prototypes) of a class, but also some notable minorities (criticism). In Figure 2-20, we illustrate an example of prototypes and criticisms, using the technique developed by (Been Kim, 2016).

Figure 2-20 Prototypes and criticisms illustration. Image taken from (Been Kim, 2016).

2.3 Technologies

This section presents insights gained through the landscape analysis that was performed along the technical dimension of the areas that, as described in Section 2.1, guide the design of the WP3 services. It should be noted that there is not a one-to-one mapping along those high-level areas and/or the exact areas that were explored and presented in Section 2.2 with the ones presented here. This can be easily explained by the fact that research dimensions and applied solutions are heavily influencing each other, yet they cannot coincide. This section presents a selection of tools and technologies relevant to the XMANAI WP3 development scope which can be leveraged either directly (i.e. used in the project) or indirectly (i.e. help identify the advantages, difficulties and challenges that come along with each design approach and decision), thus assisting to shape the WP3 offerings.

The exact information that will be presented depends also on the nature of the corresponding tools/ technologies, but mainly includes the following:

- Name and link to Website
- License
- Basic info, including the supported data types and formats (when applicable), the supported platforms and operating systems (when applicable), the supported programming languages (when applicable)
- Core features and functionalities
- Main advantages and main limitations/ disadvantages

It should be noted that an exhaustive list of all relevant technologies and tools is not possible to be provided, as the AI landscape is vast and constantly evolving, as can be seen from the glimpse provided by Matt Turch in (Turck, 2021). Therefore, in order to provide a comprehensive set of technologies that will serve is shaping the XMANAI solution, most of the technologies selected for inclusion in this section are open source or have an open source release. Apart from offering the potential of directly leveraging an open source tool, focusing on open source solutions also facilitated information gathering, as detailed features and limitations of commercial platforms are not typically accessible online.

2.3.1 Data manipulation in AI Pipelines

As previously explained, data manipulation includes several important aspects in the development of robust and insightful AI pipelines, spanning across data preparation for analysis, data harmonisation to enhance integration and improve understanding, data curation for improved quality under a specific analysis task, data versioning to facilitate debugging etc.

2.3.1.1 Data Harmonization and Integration

The tools in this category concern the process of data integration and data mapping. It should be stressed that data ingestion in XMANAI is handled through the WP2 services. Therefore the scope of this analysis is any data enrichment and/or harmonization process that cannot be performed a priori, (i.e. before data ingestion and storage), but is dependent on the particular analysis to be performed or is required in the context of the new/derivative data being generated through the AI pipelines.

Table 2-1: Data Harmonisation & Integration Technologies

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Karma: https://github.com/usc-isi-i2/Web-Karma	**License:** Apache 2.0 **Supported data types & formats:**	• Concerns data integration mostly • Schema mapping of tabular sources to ontologies • Supports hierachical data sources	• Integration of ontologies • UI available • Data Integration through WebAPI	• Needs maybe an additional step for graph data preparation

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	hierarchical data structures **Supported platforms:** Linux, Mac, Windows **Language specific:** -	• Batch processing • Data Transformation available to transform multiple data formats into common formats • Publish data in RDF or store in a database		
iQvoc: https://github.com/innoq/iqvoc	**License:** Apache 2.0 **Supported data types & formats:** SKOS, RDF, OWL **Supported platforms:** Web **Language specific:** -	• Tool for managing vocabularies • Thesauri • Taxonomies • Classification schemes • Subject heading systems • Supports linked data	• Use of SotA technology • Publishing feature • Multilingual interface • UI available	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
RMLMapper: https://github.com/RMLio/rmlmapper-java	**License:** MIT License **Supported data types & formats:** csv, json, xml, sparql, rdb **Supported platforms:** Java / Python supported platforms **Language specific:** Java, Python	• RML based mapping • Most common data structures supported • Generating Linked Data	• Popular RDB Supported • WebAPIs providing • Many output formats • Metadata generation	• RML Knowledge • No NoSQL support • Loads all data in memory (might be problematic for big datasets) • UI not available
Talend: https://www.talend.com/products/talend-open-studio/	**License:** Apache 2.0 **Supported data types & formats:** structured data **Supported platforms:** Windows / Max, on premise or cloud **Language specific:** Java	• Concerns data integration mostly (ingest data from any data source) • Big data mapping tool • RDBMS connectos • SaaS Connectors • Data discovery • Data quality checks	• Support dynamic schemas • 100+ connectos available • UI available	• Open Source edition with limited scheduling & streaming features

2.3.1.2 Data Preparation & Curation

Data preparation is a important part of any ML process as it is responsible for transforming the raw data into informative features that bring value to model training. It is a process that can vary singificanlty among the different tasks, thus a wide range of functionalities should be covered. This section gathers the tools that can be employed for data preparation as well as data curation purposes,

that among others include the organization, cleaning, description and integration of data from different sources.

Table 2-2: Data Preparation & Curation Technologies

Name	Basic Info	Core features	Main Advantages	Main Limitations
Dask: https://dask.org/	**License:** BSD 3-Clause **Language specific:** Python	• Dask is an open-source Python library that lets you work on arbitrarily large datasets and dramatically increases the speed of your computations through parallel computation.	• Supports almost all functionalities of Numpy, Pandas, Scikit-Learn • Supports build-in task scheduling with Task Graphs • Can be integrated to distributed computing environments e.g. Kubernetes • Lazy loading and execution • Supports end-to-end pipelines • Massive community	• Needs some expertise to avoid bugs
Apache Spark: https://spark.apache.org/	**License:** Apache 2.0 **Language specific:** Python, Scala, R	• Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching and optimized query execution for fast queries against data of any size	• Works in distributed environments (YARN, MESOS, Kubernetes etc.) • Support distributed data datasets of tabular data • Integrated SQL • In-memory data processing • Lazy loading and execution • Supports processing on streaming data • Many built-in functions for column manipulation • Supports Python UDFs • Has its own ML lib • Supports integrated end-to-end pipelines • Task visualization via UI • Part of Apache family • Massive community	• In-memory processing can be expensive • Spark job requires manual optimization and expertise • Back pressure handling • Supports only near real-time processing (micro-batches) of live data • No built-in scheduler for automating jobs
Optimus: https://hi-optimus.com/	**License:** Apache 2.0 **Language specific:** Python	• Python library for easy, fast, parallelized and scalable data cleansing, exploration and Machine Learning Models creation.	• Supports many backends – Pandas, Dask, CuDF, Dask_Cudf • Visualization of data with Bumblebee • Fast execution & easy to implement • Clean API with many handy functions for data curation • Open Source	• Supports only tabular data • Does not support lazy-loading and lazy-execution • Cannot load data from a database – needs intermediate step • Does not support scheduling

Name	Basic Info	Core features	Main Advantages	Main Limitations
Pandas: https://pandas.pydata.org/	**License:** BSD 3-Clause **Language specific:** Python	• Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool	• Largest collection of dataFrame manipulation tools • Very nice documentation • Huge community • Easy to use and debug • Many ready-to-use tools for data cleaning, harmonization, canonicalization etc	• Slow execution on big data (bad scaling). Needs to run sequentially in a batched fashion • Manually write code for batch processing of the dataset
CuDF: https://github.com/rapidsai/cudf/	**License:** Apache 2.0 **Language specific:** Python	• cuDF is a Python-based GPU DataFrame library for loading, joining, aggregating, and filtering data. It's part of the Rapids libraries	• Integrated with dask to scale to big data • Rapidly growing • Same API of Pandas • Can read from Google cloud and Amazon S3 • Works directly on GPU	• Needs dask integration to scale
Koalas: https://github.com/databricks/koalas	**License:** Apache 2.0 **Language specific:** Python	• Koalas is a python library that implements the pandas DataFrame API on top of Apache Spark	• No need to familiarize with a different library, since Koalas just implement Pandas on top of Spark, in order to be suitable for big data Single code base for plain pandas ("small" data handling) and for big data manipulations	• Requires a separate Apache Spark installation

2.3.1.3 Data Versioning

The tools in this category focus mainly on processing the data and tracking the different versions that occur. However, tracking the data preparation pipelines as well as models is also provided by some tools such as DVC and Pachyderm. The primary intended user is a Data Scientist.

Table 2-3: Data Versioning Technologies

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Pachyderm: https://www.pachyderm.com/	**License:** Pachyderm Community License **Supported data types & formats:** images, text **Supported platforms:** cloud, SaaS hub of company **Language specific:** Go, Python (Scala planned for future)	• Build pipelines to process data of different commits (versions) • Track which models were used on which data and produced which results • Combine input data of different repositories • Use any object store that has S3 compatible API	• Language and framework agnostic, so any tooling can be used for creating machine learning workflows • Reproducible results • Automation: pipelines automatically triggered when new data arrive (integration with CI/CD systems) • UI available	• Community version has limitations on scaling (up to 16 pipelines) and parallelism (up to 8 workers per pipeline)
DVC: https://dvc.org/	**License:** Apache 2.0 **Supported data types & formats:** N/A	• Track and save data and machine learning models (data storage) • Compare model metrics among experiments • ML pipeline framework	• Language- & framework-agnostic (supports Python, R, Julia, Scala Spark, custom binary, Notebooks,	• Issues arise when running DVC on windows • Some pipeline stages may only work on some

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	Supported platforms: platform agnostic **Language specific:** Python	(built-in way to connect ML steps into a DAG and run the full pipeline end-to-end) • DVC is a command line tool, it can also be used as a python library	flatfiles/TensorFlow, PyTorch) • Storage agnostic (Amazon S3, Microsoft Azure Blob Storage, Google Drive, Google Cloud Storage, Aliyun OSS, SSH/SFTP, HDFS, HTTP, disc to store data) • Compatible with Git • Makes data science projects reproducible by creating lightweight pipelines • UI available (DVC studio)	operating systems and require certain software packages to be installed (even though DVC is platform-agnostic)
Superb AI: https://www.superb-ai.com/	**License:** MIT **Supported data types & formats:** images, video (in paid plans) **Supported platforms:** N/A **Language specific:** Python	•Data management: partition, filter, search through data • Automated data labeling • Quality assurance on labeling	•Python sdk to manage suite from code • cli • Integrates with pachyderm • User management • UI available	• Application on image or video data only • On-premise version is not existing currently.

2.3.2 Visualisation & Visual Exploration

This section follows the same approach with the sub-section presenting the relevant visualisation methods, i.e. sub-section 2.2.3. As explained, visualization and visual exploration can be examined from **the data point-of-view** (the process of understanding the characteristics of the available data using appropriate visualization techniques) and **the model point-of-view** (the process of explaining the model's behaviour using appropriate visualization techniques).

2.3.2.1 Data-level Visualization

This category focuses on libraries for statistical visualizations of the data and their characteristicts. These libraries offer different approaches for creating interesting and comprehensible plots to get insights on the data, while some (e.g. UMAP and HyperTools that will be presented) support also the visualization of high dimesnsional data.

It should be noted here that there are numerous JavaScript libraries that allow developers to create beautiful, often interactive, charts. These libraries can render a wide range of charts that present the required information in a nice way and XMANAI will utilize one or multiple such libraries to provide aesthetically pleasing visual results to the end users.

However, it is not in scope here to examine these libraries, since there are many production ready solutions and the selection will depend on the overall look and feel of XMANAI and the technologies that will be used in frontend development. The scope here is to examine the technologies that address the needs of data scientist instead, which are often overlooked but are extremely important to ensure the right tools are offered for in-depth data understanding and flexible exploration. Table 2-4 therefore presents libraries in this direction.

Table 2-4: Data Visualisation Technologies

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Plotly: https://github.com/plotly/plotly.py	**License:** MIT **Supported data types & formats:** tabular, timeseries **Supported platforms:** N/A **Language specific:** Python	• High level, interactive & browsed based visualization library.	• Interactive plots • Large variety of plots (machine learning, geographical, scientific, statistical, and financial data) • Integration with pandas • Supports jupyter notebooks • UI available with Dash extension only	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
Bokeh: https://github.com/bokeh/bokeh	**License:** BSD-3-Clause **Supported data types & formats:** tabular, timeseries **Supported platforms:** N/A **Language specific:** Python	• High level, interactive & browsed based visualization library.	• High-performance interactivity over large/streaming datasets • Visualize network graphs • Supports jupyter notebooks	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
HoloViews: https://github.com/holoviz/holoviews	**License:** BSD-3-Clause **Supported data types & formats:** tabular, timeseries **Supported platforms:** N/A **Language specific:** Python	• High level library for data analysis & visualisation • Tabular data • Gridded data (multidimentional) •Geometry data	• Bundling together raw data with semantic metadata • Multi-dimensional containers, interactive plots, web dashboard • Support for large/streaming data • Supports jupyter notebook	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
MatPlotLib: https://github.com/matplotlib/matplotlib	**License:** PSF **Supported data types & formats:** tabular, timeseries **Supported platforms:** N/A **Language specific:** Python	• Cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy	• Static, animated & interactive visualizations	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
Altair: https: //github.com/altair-viz/altair	**License:** BSD-3-Clause **Supported data types & formats:** tabular, timeseries	• Declarative statistical visualization library • Data encoding & aggregations	• Built around pandas dataframes	• Based on Vega and Vega-Light (vega datasets are installed as well)

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	Supported platforms: N/A **Language specific:** Python			
UMAP: https://github.com/lmcinnes/umap	**License:** BSD-3-Clause **Supported data types & formats:** tabular, timeseries **Supported platforms:** N/A **Language specific:** Python	• Implementation of UMAP dimensionality reduction technique (similar to t-SNE)	• Visualization of no linear dimesnional reduction (manifolds)	• For the visualization, tools like matplotlib datashader holoviews are needed
HyperTools: https://hypertools.readthedocs.io/en/latest/index.html	**License:** MIT **Supported data types & formats:** tabular, timeseries **Supported platforms:** N/A **Language specific:** Python	• Visualizing and manipulating high-dimensional data • Dimensionality reduction based, visual exploration of high-dimensional data	• Built on top of matplotlib, seaborn and sklearn	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
Seaborn: https://github.com/mwaskom/seaborn	**License:** BSD-3-Clause **Supported data types & formats:** tabular, timeseries **Supported platforms:** N/A **Language specific:** Python	• Relational plots, Distributions, Categorical plots, Regression fits • Matrix plots (heatmap, hierarchically-clustered heatmap) • Multi-plot grids (conditional, pairwise, joint)	• Nice aesthetics	• Static plots

2.3.2.2 Model-level Visualisation

It should be noted that the technologies considered in this section are also examined in D1.1 "State of the Art Review in XMANAI Research Domains" that presented an in-depth analysis of explainability methods and approaches. The scope here is not to repeat that analysis, but to explore in more detail some prevalent tools and frameworks for the extraction of technically-oriented insights to assist in the architecture design process.

The category under which these tools appear, i.e. visualization instead of machine learning, should also be clarified: Most of the tools in this category focus on exploring (e.g. TensorBoard) and interpreting (LIME) the models and their predictions. Visualization is an important part of these procedures, as explanations are more comprehensible when visualized. Thus, even though the tools mentioned in the table do not explicitly concern visualization techniques, they are included here

because they utilize visualizations which are either built-in or supported by libraries such as matplotlib. The primary intended user is again the data scientist.

Table 2-5: Model Visualisation Technologies

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
TensorBoard: https://www.tensorflow.org/tensorboard	**License:** Apache 2.0 **Supported data types & formats:** text, images, audio **Supported platforms:** N/A **Language specific:** N/A	• Traking and visualizing metrics such as loss and accuracy • Visualizing the model graph (ops and layers) • Viewing histograms of weights, biases, or other tensors as they change over time • Projecting embeddings to a lower dimensional space • Displaying images, text, and audio data • Profiling TensorFlow programs	• Collaboration & reproducibility (TensorBoard.dev is a free public service that can be shared) • Visualizing Tensorflow complex computation graphs • Load directly in notebooks • UI available	• Tensorflow specific
Alibi: https://github.com/SeldonIO/alibi	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Many model-agnostic explanation techniques • Implementations of black-box, white-box, local and global explanation methods for classification and regression models	• Large collection of model-agnostic explanation methods	•Limited support in deep learning models • Visualization tools are needed.
SHAP: https://github.com/slundberg/shap	**License:** MIT **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Implementation of SHAP explanation technique	• Implementations for trees, NLP, Deep learning • Implements a Model agnostic approach	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
LIME: https://github.com/marcotcr/lime	**License:** BSD-2-Clause **Supported data types & formats:** text, images, tabular **Supported platforms:** N/A **Language specific:** Python	• Implementation of LIME explanation technique	• Any black box classifier (tabular data, image) • Simple regression	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
Yellowbrick: https://github.com/DistrictDataLabs/yellowbrick	**License:** Apache 2.0 **Supported data types & formats:** N/A	• Nice tools for model selection • Combines sk-learn, matplotlib to produce visualisation of machine	• Works with pandas dataframe • Works in jupyter notebooks and python scripts	• Limited support for model-explanation techniques (practically offers

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	Supported platforms: N/A **Language specific:** Python	learning workflows • Feature visualization and selection • Model visualization		only Feature Imortance)
Captum: https://github.com/pytorch/captum	**License:** BSD-3-Clause **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Counterfactual explanations • Data exploration • Model exploration	• Provides web interface for visualisations (Captum Insights)	• Pytorch specific
Tensorflow-ModelAnalysis: https://github.com/tensorflow/model-analysis	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Evaluating-Comparing tensorflow models • Visualize metrics • model validation against others and baselines	• Suitable for large amount of data • Performs computations in a distributed way • Can be extended to support other frameworks (not just tensorflow) • UI available	• Tensorflow specific • Requires Apache beam, apache arrow
Lucid: https://github.com/tensorflow/lucid	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Neural networks interpretability and visualisation	• Compatibility with tensorflow	• Currently not supporting tensorflow 2 • Research code, not production code
what-if-tool: https://pair-code.github.io/what-if-tool/	**License:** Apache 2.0 **Supported data types & formats:** text, tabular, image **Supported platforms:** N/A **Language specific:** Python	• Counterfactual explanations • Data exploration • Model exploration • Model comparison (up to 2)	• Interactive interface • Nice plots for understanding the data and explaining the model • Supports tensorflow and other frameworks • Classification and regression models • UI available	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
ELI5: https://github.com/eli5-org/eli5	**License:** MIT **Supported data types & formats:** tabular, text, image **Supported platforms:** N/A **Language specific:** Python	• Explain weights & predictions of sklearn, XGBoost, LightGBM, CatBoost & lightning classifiers & regressors. • Explain predictions of Keras image classifiers with GRAD-CAM • Implementation of LIME, Permutation Importance	• Simple tool to explain & visualize models & predictions • Supports variety of models	• Limited support for model-explanation techniques (LIME, Permutation Feature Importance, GRAD-CAM)

2.3.3 Machine Learning

Machine learning lies at the core of the AI pipelines, and it is thus important that any decisions around their architecture design take into consideration the requirements imposed by the relevant libraries and frameworks, as well as the advantages and limitations that these technologies bring in order to ensure that XMANAI will select, leverage and provide an appropriate selection/combination to cover its stakeholders' needs.

2.3.3.1 Machine learning and deep learning libraries and frameworks

Machine learning is not a new domain and an exhaustive list of applicable tools cannot be provided. Therefore, some of the most mature and popular solutions are provided here, ensuring that deep learning and graph machine learning solutions are also considered in order to have insights across all the applicable ML spectrum in XMANAI.

Table 2-6: ML/DL Technologies

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Scikit-learn	**License:** BSD 3-Clause **Supported data types & formats:** NumPy arrays, SciPy sparse matrices, Pandas dataframe **Supported platforms:** Linux, macOS, Windows **Language specific:** Python	• Built on NumPy, SciPy, and matplotlib • Supervised Models • Datasets • Parameter Tuning • Feature Selection • Dimensionality Reduction • Cross-validation • Ensemble Methods • Feature Extraction • Clustering	• Well-established community & support • Easy to use and integrate • Extensive documentation	• No GPU support • Categorical variables need preprocessing
Tensorflow	**License:** BSD 3-Clause **Supported data types & formats:** all native Python types	• Flexible and easy training • Parallel Neural Network Training • Feature columns • Support for statistical distributions	• Support for both CPU and GPU • Well-established community & support • Easy to use • Effective & Scalable execution • Easy experimentation	• Lack of symbolic loops • Limited Windows support • Only NVIDIA GPU support

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	Supported platforms: Linux, macOS, Windows **Language specific:** Python, Javascript, C++, Java		• Abstraction & flexibility	• Low performance in speed & usage
Keras	**License:** MIT **Supported data types & formats:** NumPy arrays, TensorFlowDataset o bjects, Python generators **Supported platforms:** Linux, macOS, Windows **Language specific:** Python	• Extensive implementations of commonly used neural-network building blocks • Simplified programming for deep neural network code • Support for convolutional and recurrent neural networks	• Can run on top of multiple frameworks such as Tensorflow, CNTK, PlaidML, Theano, R • Facilitates fast experimentation with deep neural networks • Efficient development of deep learning models • Well-established community & support	• Lack of pre-trained models to experiment • Inefficient error handling and reporting • Low level API errors can be hard to handle and resolve
Spark MLlib	**License:** Apache 2.0 **Supported data types & formats:** dataframes **Supported platforms:** Linux, macOS, Windows on top of HDFS **Language specific:** Java, Scala, Python, R	• Distributed and iterative computations • High Performance • Scalability	• Implementation ML algorithms • Easy to use and integrate • Well-established community & support • Multiple programming languages support • Well-established community & support	• Requires a Spark installation • Limited number of implemented algorithms
PyTorch	**License:** BSD **Supported data types & formats:** NumPy arrays **Supported platforms:** Linux, macOS, Windows **Language specific:** Python, C++, Java	• Supports automatic differentiation of tensors • Suppors computations on tensors • Support of custom data loaders • Enhanced computational graphs development	• Easy to learn • Rich APIs • Support for CPU & GPU • Scalable distributed training and performance optimization	• Lack of native support for visualization • Production mode requires an API server
XGBoost	**License:** Apache 2.0 **Supported data types & formats:** LIBSVM, csv	• ML algorithms implementations under Gradient Boosting framework • Support for parallel tree boosting • High efficiency, flexibility and portability	• Easy to learn • Decrease of feature engineer need • Fast to interpret • Minimization of outliers impact	• Hard to tune, many hyperparamters

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	Supported platforms: Linux, macOS, Windows **Language specific:** C++, Java, Python, R, Julia, Perl, and Scala		• Support for large sized datasets	
Theano	**License:** BSD 3-Clause **Supported data types & formats:** All native Python types **Supported platforms:** Linux, macOS, Windows **Language specific:** Python	• Enables the definition, optimization and evaluation of mathematical expressions that include multi-dimensional arrays • Efficient symbolic differentiation	• High Performance • Easy to use • Efficiency and Scalability	• Support discontinued
Spark GraphX	**License:** Apache 2.0 **Supported data types & formats:** GraphFrames **Supported platforms:** Linux, macOS, Windows on top of HDFS **Language specific:** Java, Scala, Python, R	• Support of number of graph algorithms and builders to perform graph analytics tasks • Ease of programming • Efficiency in graph problems	• Provides a set set of pre-built graph algorithms • High performance • Fault tolerance • Ease of use	• Requires a Spark installation
StellarGraph	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** Linux, macOS, Windows **Language specific:** Python	• State-of-the-art algorithms implementations for graph machine learning • Built on TensorFlow 2 and its Keras high-level API, as well as Pandas and NumPy. • Inter compatibility with Neo4j	• Cross-platform support with Keras and Scikit-Learn • Ease of use • Modularity • Extensibility • High performance	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
Pytorch Geometric	**License:** MIT **Supported data types & formats:** N/A **Supported platforms:** Linux, macOS, Windows **Language specific:**	• PyTorch Graph Neural Network Library • Support of various methods for deep learning on graphs and other irregular structures • Multi GPU support • Distributed graph learning support • Benchmark datasets availability	• Easy to use • Extensive API support • Support for multiple state-of-the-art GNN architectures • High performance	• Requires PyTorch

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	Python			
Spektral	**License:** MIT **Supported data types & formats:** N/A **Supported platforms:** Linux, macOS, Windows **Language specific:** Python	• Based on the Keras API and TensorFlow 2 • Implementation of the core layers of graph deep learning • Support utilities for the representation, manipulation and transformation of graphs to graph deep learning projects	• Supports graphs representation • Extensive implementation of message passing layers • Support for graph pooling • Popular graph dataset are included	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*

2.3.3.2 Hyperparameter tuning

This category contains tools which are utilized for hyperparameter tuning, a process explained in section 2.2.1. The primary intended user is the data scientist and these tools facilitate/ accelerate the process of finding the optimal model hyperparameters, e.g. through parallelization and state-of-the-art sampling algorithms.

Table 2-7: Hyperparameter Tuning Technologies

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Optuna: https://optuna.org/	**License:** MIT **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Pythonic search for hyperparameters • Search large space using pruning • Parallelize hyperparameter searches • State of the art efficient optimization algorithms for searching parameters • Visualizations of hyperparameter analysis	• Framework agnostic (it can be integrated with: AllenNLP, BoTorch, Catalyst, Chainer, fast.ai, Keras, LightGBM, MLflow, MXNet, pycma, PyTorch, scikit-learn, scikit-optimize, skorch, Tensorflow, XGBoost) • Lightweight • Easy to implement • Scalable • CLI • Combination with mlflow to track all hyperparameters and metrics	• UI not available
Hyperopt: http://hyperopt.github.io/hyperopt/	**License:** Copyright (c) 2013, James Bergstra All rights reserved. **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Distributed and asynchronous hyperparameter optimization • Available algorithms: random forests, TPE, Adaptive TPE	• Scalable (parallelized by Apache Spark or MongoDB) • Pre-defined list of models, and preprocessing steps (function: any_classifier(), and any_preprocessing())	• Challenging to use directly, requires the optimization procedure and search space to be carefully specified (extension to HyperOpt called HyperOpt-Sklearn) • Documentation could be better • Visualizations

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
				are very basic • UI not available
scikit-optimize: https://scikit-optimize.github.io/stable/	**License:** BSD **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Implements several methods for sequential model-based optimization • Built on top of NumPy, SciPy and Scikit-Learn • Fast and effective hyper-parameter tuning	• Offers convergence plots to monitor the optimization procedure • Stores and loads optimization results • Provides wrappers for scikit-learn Gridsearch using Bayesian Optimization • Able to search spaces for real, integer and categorical data. • Can be used for model built with other libraries (e.g. tensorflow) • Open source	• Does not perform gradient-based optimization

2.3.4 MLOps

Machine learning operations (MLOps) aim on providing an end-to-end machine learning process to design, build and manage testable and reproducible machine learning code. There are many tools that have emerged the past few years due to the increasing interest in MLOps among data scientists, ML engineers and data engineers. There are tools that focus on a specific task of the ML process and others that try to capture the whole lifecycle of a machine learning project. Subsections 2.3.4.1-3 provide insights on tools and technologies across three different aspects of MLOps, whose role was also discussed in section 2.2.1.

2.3.4.1 Feature stores

Tools in this category facilitate mainly the operations of data scientists and data engineers, by providing functionalities such as:

- Feature joins from different sources in a point-in time-correct way
- Storage connectors for batch (BigQuery, GCS) and streaming data sources (Kafka)
- Feature availability in a scalable, consistent and performant way for training and inference
- Sharing in a controlled manner among projects & separating across organisations (when the same platform is used)

Feature stores can save time on data preparation and feature engineering, enable shareability and **versioning** of data assets, and support various data modalities, including tabular data, images, and text.

Table 2-8: Feature Stores

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Hopswork: https://www.hopsworks.ai/	**License:** GNU affero general public license 3.0 **Supported data types & formats:** N/A **Supported platforms:** Any	• Feature management, validation, documentation, sharing and insights through rich metadata • Store feature dataset as feature group using HSFS library. • Store training datasets • Retrieval of features groups and training datasets easily	• Online and offline feature store • Connect to feature store from outside Hopsworks • Implementation (hsfs library) for spark and python environments • UI available	• Hopsworks Feature Store is a component of the larger Hopsworks data science platform

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	Language specific: Python, Scala, Java	and fast • Supports integration with azure, databricks, AWS • Allow point-in-time correct and consistent access to feature data (time travel).		
Feast: https://feast.dev/ https://github.com/feast-dev/feast	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** Any, Google cloud platform is natively supported **Language specific:** Python	• Able to serve large volumes of historical features for model training • Able to serve features to online models through a low latency gRPC API (with Go and Java SDKs) • Supports deployment in Kubernetes, Spark, Terraform (but not required) • Actively developed, plans can be found in https://feast.dev/blog/a-state-of-feast/	• Configuration in .yml file • Can be combined with other tools & technologies: Big Query (offline), Big Table (online), Redis (low-latency), Apache Beam (feature engineering), • Storage connectors with implementations for Cassandra and Redis Cluster.	• Feast is a Python library for feature retrieval and not computation, so this needs to be handled by another process/tool • Primarily built around Google products • Feature discovery not supported • Feature engineering & validation not supported • Streaming data not supported by default • UI is not available
MLRun (part of iguazio MLOps platform) https://github.com/mlrun/mlrun	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** Any **Language specific:** Python	• Feature and artifact registry/search • Tracking and versioning • Supports full ML pipeline deployment	• Supports Lineage • Supports streaming data • UI available	• Open Source version does not support built-in services , (like spark, Presto, Dask erc) nor project resource management • Online feature serving is also not available in the open source version
PyRasgo: https://github.com/rasgointelligence/PyRasgo	**License:** GNU affero general public license 3.0 **Supported data types & formats:** N/A **Supported platforms:** Snowflake AWS, Cloud platforms **Language specific:** Python	• Rasgo is a feature store for preparing, understanding, and deploying features using a cloud data warehouse. The free python package, PyRasgo, interacts with the Rasgo API and provides: • Feature profiling • Feature importance evaluation • Feature and target variable relationship visualization • Feature tracking and versioning • Feature publishing and sharing	• Implementation for python environment (and notebooks) • UI available • Supports feature engineering • Supports time travel • Supports streaming data	• The free python package is just a connector to Rasgo (requires login, which is free for individuals and small teams). • Still in alpha development.

2.3.4.2 Experiment tracking

Experiment management and tracking includes recording hyperparameters, metrics, artifacts and data, supporting configuration locally or on remote machines. The primary intended user is the Data Scientist, however most of the experiment tracking tools provide useful features for both Data Scientists and Data Engineers.

It should be noted that most of the tools in this category can be used for more than just logging parameters, code and artifact such as polyaxon, amazon sagemaker and verta.ai (that are presented in the table below) which can be used for model deployment and orchestration as well. However, the focus in Table 2-9 is on their experiment tracking functionality.

Table 2-9: Experiment Tracking Technologies

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Sacred: https://github.com/IDSIA/sacred#sacred	**License:** MIT **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Keep track of all the parameters of your experiment • Easily run your experiment for different settings • Save configurations for individual runs in a database • Reproduce your results	• Easy integration • Extensive experiment parameterization • Reproducibility (lightweight by using heuristics to capture the source code) • Randomness (new seed for each run) • Bookkeeping (experiment publishes collected information in the form of events, to which observers can subscribe) • Integration with tensorflow	• CLI logger (not adapted for team collaboration) • Limited to python scripts • Not much support for organizing and analyzing results • No automation for reproducing experiments (researcher has to manually reconstruct the environment, copy the stored files and rune with the saved config parameters) • UI not available
Polyaxon (v 1.9.5): https://polyaxon.com/docs/	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** On premise or on any cloud platform **Language specific:** N/A	• Hyperparameter tuning •Running jobs in parallel (to find the best model) • Model registry • Collaboration (access-control and team management) • Visualizations • Orchestration (schedule jobs/experiments, manage resources)	• CLI • API • Integration with steamlit, notebook, tensorboard, vscode • Customizable and extensible • Works with any framework or library • Open source • Serverless and Kubernetes native • Reproducibility, scalability, data autonomy, maximum resource utilization • UI available	• Polyaxon does not deploy the operators required for running distributed jobs to keep the deployment process lightweight. In order to use a distributed jobs operator, you need to make sure that your namespace/ cluster has the operator deployed or you should deploy the operator(s) before starting an execution.
ML Flow: https://mlflow.org/	**License:** Apache 2.0	• Projects: Package data science code in a format to reproduce runs on any platform	• Scalable (apache spark) • Team collaboration (compare parameters	• No user management

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	Supported data types & formats: N/A **Supported platforms:** all **Language specific:** Python, R, Java	• Models: Deploy ML models in diverse serving environments • Store, annotate, discover, and manage models in a central repository	and results) • Manage and deploy models from a variety of ML libraries to a variety of model serving and inference platforms • UI available	
Guild AI: https://guild.ai/	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** Linux, MaxOS, Windows **Language specific:** Any language or framework	• Compare and Analyze Runs • Tune Hyperparameters • Automate Pipelines • Train and Backup Remotely • Publish and Share Results • Jupyter Notebook Integration	• Does not require to modify your existing code • Does not require additional software or systems like database or containers • Code diffs • Python API • UI available • CLI	• No user management • Does not support notebooks' tracking • Does not integrate easily with other libraries
Neptune: https://neptune.ai/	**License:** proprietary **Supported data types & formats:** text, image, audio **Supported platforms:** N/A **Language specific:** Python, R	• Metadata store for MLOps (log and display any ML metadata you care about) • Compare experiments and models • See ML experiments live • Reproducible and traceable experiments • Collaboration (share results or visualizations by sending a link) • Query experiment and model training metada programmatically • Model Registry	• Notebook versioning • User management • Experiment Organization • Notebook Diffs (Automatically snapshots Jupyter notebooks) • Grouping Experiments • Scales to Millions of Runs • Integrations (scikit, tensorboard, sacred, catalyst, scikit-optimize, ray, hiplot) • Track EDA (log images and charts) • Experiment dashboard directly to a pandas DF • Can integrate MLflow with Neptune • UI available	• Subsrciption for teams (individuals: free)
Amazon sagemake: https://github.com/aws/sagemaker-python-sdk	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** N/A **Language specific:** Python	• Reconstruct an experiment • Collaboration: build on experiments conducted by peers • Trace model lineage for compliance and audit verifications • Create and manage machine learning pipelines • Track the lineage of machine learning workflows • Model registry: Versioning, artifact and lineage tracking	• Supports variant ML framework: Scikit-learn, TensorFlow, XGBoost, PyTorch, SPARKML, etc • Scaleable • Open source	• UI not available • A bit complex, without a strong programming background

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
		• Analyze and preprocess data, tackle feature engineering, and evaluate models.		
Atlas OSS (beta): https://docs.atlas.dessa.com/en/latest/	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** cloud, local **Language specific:** Python	• Job queuing & scheduling (experiment variations) • Keep every single experiment, complete with code and any saved items	• Built-in integrations (tensorboard) • Reproducibility • Collaborate across your team and user access controls • CLI • Local Scheduler for job orchestratrion • UI available	• Python-based (SDK) • Beta
verta.ai: https://www.verta.ai/	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** AWS, Azure, Google cloud, vmware **Language specific:** Python	• Track and visualize ML experiments • Visualize and compare model versions • Keep code, data, configuration and environment variables for model reproducibility • Share experiments and results with the team • Model registry (stage best model for release, model validation, automatically track model version) • Model deployment • Model monitoring (quality metrics, data quality)	• Supports most popular libraries and frameworks (TF, Pytorch, Scikit-learn, spark, R, AWS, Google Cloud) • Integrates with kubeflow, ANACONDA, amazon sagemaker • Secure team colaboration • Integration with CI/CD pipelines (Jenkins, GitOps) • UI available	• Model deployment is limited, whereas monitoring is not available in community version • Model validation and CI/CD automation not supported in free version

2.3.4.3 ML automation (CI/CD/CT)

The tools in this category can support more than just the processes of ML automation (CI/CD/CT). Experiment tracking, model comparison, resource management and model deployment are among the functionalities that may also be provided by such tools, since in order to achieve ML automation, other features need to be supported as well. The focus in Table 2-10 is on the ML automation aspects, but the broader scope is also considered and presented to ensure that the full potential of these tools is reflected.

Table 2-10: ML Automation Technologies

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Seldon core: https://github.com/SeldonIO/seldon-core	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** Cloud (AWS EKS, Azure AKS, Google GKE, Alicloud, Digital Ocean and Openshift) **Language specific:**	• Converts ML models (Tensorflow, Pytorch, H2o, etc.) or language wrappers (Python, Java, etc.) into production REST/GRPC microservices • Easy way to containerise ML models • Powerful and rich inference graphs made out of predictors, transformers, routers, combiners, and more. • Metadata provenance to ensure each model can be traced back to its respective	• Framework and language agnostic • Full lifecycle management: updating, scaling, monitoring & compliance	• Built to run on kubernetes • UI not available

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
	Python, R, Java, NodeJS, Go	training system, data and metrics • Microservice distributed tracing through integration to Jaeger for insights on latency across microservice hops. • Secure, reliable and robust system maintained through a consistent security & updates policy		
ClearML (former Trains): https://clear.ml/	**License:** Apache 2.0 **Supported data types & formats:** tabular, audio, image, text **Supported platforms:** N/A **Language specific:** Python	• Tracking command-line parameters, hyperparameters, models and other artifacts, results (metrics and other data) • Logs data and any preprocessing on them (data storage) • Stores feautures, models, artifacts locally or on cloud • Visualizations • Order of task execution is managed by queues • Manages resources and cluster allocation • Scales up to the need dynamically	• Supports frameworks like TF, Pytorch, Keras, Fast.ai, Scikit-learn • Storage options(file systems, S3, GCS, azure) • Integration with popular libraries • Provides flexibility by giving the user the choice to change the configuration of saved models • Features are easily and fast served and deployed to production • Code can be executed on any machine, the entire execution environment is recreated when needed (e.g. from local to remote machines)	*No applicable information available online. If selected for usage within XMANAI, further investigation will be required.*
Kubeflow: https://www.kubeflow.org/	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** local, cloud (Kubernetes required) **Language specific:** domain specific language based on Python	• Track and compare experiments • Model training with software such as Jupyter notebooks • Provides end-to-end workflows that speed development (build, train and deploy) • Hyperparameter tuning during training • Kubeflow Pipelines can be used to create reproducible workflows	• Scalability and portability • Supports frameworks such as TF and Pytorch • Available distributions for AWS, Azure. GCP, IBM cloud, Nutanix carbon, • UI available	• Currently doesn't have a dedicated tool for CI/CD (use Pipelines component instead) • Needs Kubernetes

2.3.5 Execution Infrastructures

Execution infrastructures are important components for the WP3 architecture. Execution dependencies and communications are considered as directed binary relations from one logical component to another. Based on the amount for outsourcing required by the physical positioning of the functional software components, different solutions can be considered for building the execution infrastructure. For the scope of the XMANAI project and according to the needs highlighted by the demonstrators, three main different execution infrastructures types have been considered, identifying specific suitable tools: full-cloud, hybrid and on-premise solutions.

2.3.5.1 Full-cloud solutions

In **full-cloud solutions**, every aspect of the pipeline is outsourced; this includes data storage, computational power, and tools for visualization. Full-cloud solutions are often presented as off-the-shelf solutions, characterized by a set of commonly adopted composition patterns; this applies to both of the considered cloud providers: Google and Amazon. This class of solutions may be preferred by entities with a moderately sized infrastructure or with poor scalability. Both the considered cloud providers feature elastic services able to adapt to the consumer's demand; in practical terms, choosing between Google and Amazon boils down to preference and convenience, and the pricing of the specific use case.

Table 2-11: Execution Infrastructures (full-cloud)

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Google Cloud: https://cloud.google.com/	**License:** Paid **Language specific:** No specific language	• Google Cloud Platform is a suite of Google's public cloud computing resources and services. Specialized in high compute offerings like Big Data. Google's primary service is Compute Engine.	• Excellent integration with other Google services and with kubernetes • Fast I/O • Strong data analytics and storage • Facilitates easy collaboration • Good portability and open source integration • Specialized in big data and ML	• Majority of components based on Google proprietary tech; no real control over Virtual Machines • Complex transition away from the platform to another vendor • Fewer features/services • Good portability and open source integration • Fewer global data centers
Amazon Cloud service: https://aws.amazon.com/ec2/	**License:** Paid (free trial) **Language specific:** No specific language	• AWS is a secure, cloud service developed and managed by Amazon. It dominates the public cloud market by offering a range of cloud-based products and services. The flagship compute service of AWS is Elastic Compute Cloud, or EC2.	• Extensive range of infrastructure applications • Highly flexible • Easy transition for users with existing digital infrastructure • Free tier available • Greater control over security • Scalability • Cost-effective pricing model • Rapid deployment	• Visualization is expensive • Hybrid options available, but not a priority • Potentially long migration times from legacy systems

2.3.5.2 Hybrid solutions

In **hybrid solutions**, the consumer's infrastructure oversees data storage and visualization, while the computational tasks are entrusted to cloud providers. This includes for example subservices of AWS (Amazon Web Services), namely AWS Lambda or AWS Elastic Compute Cloud, which differ in how the execution is managed on the cloud: AWS Lambda hosts containerized microservices, readily available to be invoked via an API, akin to remote procedure call; on the other hand, AWS EC2 processes computational jobs on the cloud by means of a remote customizable virtual machine with several configuration parameters, including OS, size of RAM, and number of cores. The core elements of these

solutions are often composed with surrounding AWS entities in standardized fashions, resulting in commonly shaped architectural patterns that these components are designed to fit into.

Table 2-12: Execution Infrastructures (hybrid)

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Amazon Lambda: https://aws.amazon.com/lambda/	**License:** Paid (free trial) **Language specific:** No specific language	• Extend other AWS services with custom logic • Build custom backend services • Bring your own code • Completely automated administration • Built-in fault tolerance • Package and deploy functions as container images • Automatic scaling • Fine-grained control over performance • Connect to shared file syst-Run code in response to Amazon CloudFront requests	• Reduced Cost of Execution • Serverless computing. • Run code without provisioning or managing infrastructure. • Connect to relational databases • Orchestrate multiple functions • Integrated security model • Trust and integrity controls • Only pay for what you use • Flexible resource model	• No Control Over Environment • Complex Call Patterns
Amazon Elastic Compute Cloud: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html	**License:** Paid (free trial) **Language specific:** No specific language	• Virtual computing environments • Preconfigured templates for your instance • Various configurations of CPU, memory, storage, and networking capacity • Secure login information • Storage volumes for temporary data that's deleted when you stop, hibernate, or terminate your instance • Persistent storage volumes for your data • Multiple physical locations for your resources • A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups • Static IPv4 addresses for dynamic cloud computing. • Metadata, known as tags, that you can create and assign to your Amazon EC2 resources Virtual networks you can create that are logically isolated from the rest of the AWS Cloud.	• It is constantly evolving both in terms of features and user-friendliness. • Can be used for installing Web Application and Middle application • Can be used for migrating the data from one source to another source • Provides scale up and scale down capability by managing instance type • Instance variety	• Complex autoscaling • Limited information for the resources managed by Amazon EC2. • Networking is not flexible • UI for EC2 service is a little complex

2.3.5.3 On-premise solutions

In **on-premise solutions**, data storage and computational power are located in the consumer's infrastructure. This solution minimizes costs attributed to external services; however, it requires a sufficiently powerful infrastructure in order to be able to store large volumes of data and carry out the necessary computations. This solution may also be preferred by entities with particular concern for security and non-disclosure in regards to data and architectural details, as it does not involve sending or sharing data with a third-party storage provider, nor opening any doors from any part of the architecture. Unlike the previous classes of solutions, on-premise solutions are most often customized to fit the specific application they are immersed into and, while the single components may be common to many solutions, their interactions are specific to the area of interest.

Table 2-13: Execution Infrastructures (on-premise)

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Local Docker: https://docs.docker.com/	**License:** Apache 2.0 **Language specific:** No specific language	• Local Docker enables developers to locally build, share, and run containerized applications and microservices. Docker Desktop includes Docker Engine, Docker CLI client, Docker Build/BuildKit, Docker Compose, Docker Content Trust, Kubernetes, Docker Scan, and Credential Helper.	• Full control over the execution environment of the application. • Reduced risk and increased reliability of tests. • Less effort for maintaining environments for the application. • Easy updating of an existing application and environment. • "Infrastructure-as-Code" approach. The infrastructure is not created manually any longer but is a result of an automated process.	• Increased complexity due to an additional layer. • Managing a huge amount of containers is challenging. • The containers share the same kernel and are therefore less isolated than real VMs. • Introducing Docker can be a demanding and time-consuming task.
Apache Airflow: https://airflow.apache.org/	**License:** Apache 2.0 **Language specific:** Python	• Open-source task scheduler that allows users to programmatically author, build, monitor, and share workflows in the cloud.	• It allows users to create workflows with high granularity and track the progress as they execute • Make it easy to do potentially large data operations • Highly flexible, since was designed to work within an architecture that is standard for nearly every software development environment. • Highly scalable, both up and down	• Highly dependent on Python • The web user interface is arcane • Some risk involved with long-term community support of a free set of software
Argo CD: https://argo-cd.readthedocs.io/en/stable/#what-is-argo-cd	**License:** Apache 2.0 **Language specific:** YAML	• Declarative, GitOps continuous delivery tool for Kubernetes • Makes application deployment and lifecycle management automated, auditable, and easy to understand	• Automated deployment of applications to specified target environments • Ability to manage and deploy to multiple clusters • Rollback/Roll-anywhere to any application configuration	• Requires Kubernetes • All microservices need to be containarised

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
			committed in Git repository • Automated or manual syncing of applications to its desired state • Webhook integration (GitHub, BitBucket, GitLab)	

2.3.6 Orchestration Engines

Orchestration Engines are responsible for the orchestration, namely the coordination and management, of the XMANAI pipelines. The following table summarized two main tools that work as container orchestration platforms that run on top of either virtual or physical machines.

Table 2-14: Orchestration Engines

Tool name	Basic Info	Core features	Main Advantages	Main Limitations
Kubernetes: https://kubernetes.io/	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** All **Language specific:** No specific language	• Kubernetes is an open-source platform for orchestrating containers (typically Docker containers). It manages the lifecycle and networking of containers that are scheduled to run. It includes 5 main components for developers: 1) Node - Virtual Machines or physical hardware, 2) Pods - the smallest unit that you can ask Kubernetes to run, 3) Deployments - a set of declarative statements about how to deploy a pod, and how Kubernetes should manage it, 4) Services - an in-cluster load-balancer for pods, 5) Ingresses - exposes HTTP/HTTPS routes from outside the cluster to your services.	• Powerfool tool for managing services and resources • Safety • Scalability • Wide community • Makes shipping services easy •Providea a web UI for monitoring	• Not user friendly, steep learning curve • Difficult to optimize and debug
Docker-swarm: https://docs.docker.com/engine/swarm/	**License:** Apache 2.0 **Supported data types & formats:** N/A **Supported platforms:** All **Language specific:** No specific language	• A Docker Swarm is a group of either physical or virtual machines that are running the Docker application and that have been configured to join together in a cluster.	• A lot easier to install and reason about • Built into the official Docker CLI • More lightweight and has less moving parts • Compatibility with docker-compose	• May be deprecated in some years in favor of Kubernetes • No web UI • Limited functionalities

2.4 Remarks and considerations

Sections 2.2 and 2.3 provided insights gained through the state of the art analysis performed in the areas that will guide the design and development of the WP3 AI bundles. Although findings, challenges and other remarks were briefly discussed per each area or even at the level of specific methods and

tools when needed, there are certain considerations that reflect upon the AI pipelines as a whole and are thus important to be considered when designing the XMANAI positioning.

Consideration 1: AI pipelines are not a collection/sequence of individual steps

As stressed in the MLOps methods section, there are many dependencies among the steps of ML development and deployment in production systems. As (Valohai, 2021) states "To get a truly scalable business model around ML usually requires that you step back and think of your product in terms of the pipeline instead of a single instance of a trained model.". Data exploration and understanding precedes model training and biases in training data, stemming from the initial raw data, will propagate in the system if unnoticed (Breck, 2017). Governance audits in production environments are significantly facilitated when checkpoints for data distributions are in place and tracing erroneous predictions can be extremely difficult without safeguards in the feature extraction step. Explainability methods can reveal the most impactful features for a trained model and querying a model against specific selected instances can provide information on the way the model behaves and help the debugging process both during the experimentation and at the production phases. The expected data volumes (whether data in the real setting will come in batches or real time actions are required etc.), are important parts of the production serving phase, and should also not be overlooked during the exploration phase. This will ensure that features which are not timely available for prediction will be excluded during training. A relevant factor to consider here is that even in MLOps automation levels 1 and 2, experimentation which is by definition a multi-step process is not precluded, or even neglected. **Experimentation is inherent in data science** and is a process that spans across all steps from data understanding, to data exploration, to model exploration and testing.

This connection between the AI development steps and processes is naturally reflected on the technical design of the underlying functionalities. At the same time, components in AI pipelines development are preferred to be modularized and containerized to promote reusability (with the exception of purely EDA tools, such as notebooks) (Google, 2021), thus making the AI pipelines design process challenging.

Consideration 2: A trained model is not the last step of an AI pipeline

According to (Valohai, 2021) "enforcing MLOps best practices makes reproducibility a priority". One of the main advantages such practices stress is that problems and questions that emerge in production can be traced back to ensure that involved stakeholders can get the answer they need and/ or effectively and timely debug the issue.

Indeed, multiple times during the presented landscape analysis, pipeline monitoring aspects were highlighted, not only because MLops allow the increase of the level of automation, but also because of the common misconception that a well trained model, i.e. evaluated and ensured to perform well according to the defined metrics of interest, is the end goal of an AI pipeline. Model performance in production can differ from that in the experimentation settings or it can gradually deteriorate due to numerous reasons, so establishing feedback loops and ensuring all involved stakeholders are aware that the process does not end after successfully training and deploying a model, is critical.

It should be noted here that XMANAI is in a way designed to ensure that the ML model cannot be the end of the journey even for the data scientists, due to the provision of the model explainability methods that constitute a permanent link between the way the model was trained, what it has learnt and how this translates in production applications.

Consideration 3: Explainability is not yet part of MLOps

ML model explainability (or interpretability) refers to the ability to explain what happens to the ML model between input and output; how it makes decisions, and why. There are three main aspects of interpretability which are: transparency, ability to question, and ease of understanding (Onose, 2021). ML model explanations can be either global (overall explanation of the model's behaviours), or local

(explanations of individual predictions) (Molnar, 2021). To this direction, D1.1 provided a thorough review of explainable AI methods and tools and showed the importance of explainability in allowing humans to trust the decisions of the ML models, while also holding the model accountable for any wrong or rogue decision. Compliance with company policies, industry standards and government regulations are also among the benefits explainability can offer. Finally, explainability can also help model performance, as it allows data scientists to understand how the model works, thus assisting in the optimization of its parameters.

Having listed so many advantages, it may be anticipated that integrating explainability techniques, at least to some extent, in AI pipelines would be an already addressed need in the flourishing MLOps landscape. However, the state of the art review did not yield significant results. XAI and MLOps are two domains gaining attention from both the scientific and the commercial communities, but no established patterns and practices exist yet for their intersection.

Consideration 4: Collaboration is a key success factor, but not straightforward to achieve

Insights from the literature review and the technologies exploration are clear: Collaboration among the stakeholders involved in the different stages of the development and deployment of AI pipelines is crucial. It accelerates understanding, prevents mistakes, facilitates debugging and easier transition from experimentation to production serving and finally enables thorough performance evaluation and meaningful feedback loops. In reality, collaboration is a requirement and AI-enabled products cannot be put in production without it being satisfied. Making this collaboration easier across all steps in the most productive way for all involved actors is the real challenge.

Workflow management systems allow multiple agents to work towards achieving a common goal by enabling communication between them. MLOps can be considered in this context as AI workflow management systems for automating the lifecycle of AI models. MLOps improves communication and collaboration between business experts and data scientists to achieve effective management of the AI models lifecycle.

Notwithstanding the advantages gained from this collaboration, collaboration may also increase the level of complexity in the required functionalities. Starting from technical aspects such as granting access rights to different stakeholders for the various assets (datasets, models, pipelines, experiments, analysis results) to difficulties like ensuring that all relevant stakeholders have signed off on the model. Collaboration in terms of information flows can only have a positive effect, but collaborative decision making introduces new risks and challenges if involved stakeholders are not provided with the information they require in a comprehensible manner for their needs and background.

Consideration 5: Visualisation is not a step, it is an integral part of almost all steps of an AI pipeline

This is mostly a conclusion that should be leveraged in the design of the XMANAI AI Bundles, rather than a consideration, in the sense that it emerges as an indisputable fact across all the areas that were studied. Visualisation in other software development processes may be considered as a process closer to the end result, mainly addressing the end users, but reality in data analytics is different. From data exploration and outlier detection, to model training and experiments comparisons, to production performance evaluation and explainability insights, to training-production data skew monitoring, different visualisations are leveraged. Data scientists, data engineers and business users all require targeted and diverse visualisation functionalities to be available, as visual understanding, interpretation and even questioning of input data, models, and results is extremely powerful and significantly more intuitive for users.

Consideration 6: The landscape of methods and tools is vast, but technical limitations should not be underestimated.

Finally, although the AI-specific aspects are important (e.g. AI development mentality, cross-team collaboration and all the AI pipeline development challenges already mentioned), designing and

implementing the XMANAI AI bundles also brings the more traditional software development challenges. Scalability issues and in-memory analysis requirements should be studied and understood to put in place a robust technical solution. There is an abundance of technologies that can assist in bringing the XMANAI services to life, but selecting the ones that will complete the puzzle of required functionalities without conflicts, ensuring the right balance between development and integration effort, whilst attending to all the needs of the manufacturing domain in XAI-enabled decision making, will be challenging.

3 Industrial Asset and Knowledge Graphs Modelling

3.1 Scope

Having high-quality data to build upon is an indisputable requirement to create insightful AI pipelines, yet the importance of having common underlying structure and semantics for these data may not always be that obvious. Indeed, even without a common data model, data scientists can still explore, query and learn how to understand the available data, train and evaluate machine learning models etc. However, the availability of a data model can significantly facilitate and accelerate these processes and make them less error prone. Therefore XMANAI will leverage data models almost across all steps of its AI pipelines, sometimes to enhance the respective processes and others to provide functionalities that would be otherwise impossible to achieve (at least at the same level of quality).

Using a common data model during data ingestion allows the stakeholders that have knowledge over the data (e.g. business users) to pass this information in an asynchronous manner to the teams that will be responsible for data analysis. Semantics stay with the data, thus **facilitating understanding** for the data scientists even in the first steps of exploration. Spotting anomalies in the data, developing an intuition as to the expected distributions, creating meaningful visualisations, anticipating required transformations and thinking of potentially useful combinations, are among the processes that become easier based on this availability of data insights, thus **accelerating everyday operations of data scientists** when it comes to exploratory data analysis.

Going a step further, a data model is not only related to the input data, but can be also leveraged for derivative data generated during data pre-processing, feature engineering and finally results extraction from the AI models and analysis processes. In this way, data across the AI pipelines are enriched with semantics conveying to stakeholders what the data are about and how they can/should be handled. Having data that essentially "carry their meaning" across the AI pipeline steps thus allows XMANAI to achieve **data explainability**, thus contributing to the overall explainability of the XMANAI AI pipelines.

Numerous technical advantages also stem from ensuring that data conform to a common data model. Discrepancies and changes in the way input sources provide data, either due to errors or to performed updates, might go unnoticed and propagate to subsequent steps where **debugging** will be harder. Having a common data model allows the implementation of more advanced (in a way smarter) **data validation** rules, enhances and up to an extent guarantees data integrity and thus significantly facilitates production deployment and end-to-end monitoring of pipelines by data engineers. Furthermore, data manipulation operations in terms of data manipulation can be enabled or disabled depending on the data types/units of the data being used. The same information can be leveraged in the way visualisations are configured, in the way feature extraction is performed and in the way ML models' hyperparameters are configured.

Finally, having this common understanding of the data being used and enforcing certain actions, rules, quality and security tests based on commonly agreed unpon structures, does not only facilitate the operations of each stakeholder role in XMANAI, but provides the foundations for a more productive **collaboration** among business users, data scientists and data engineers.

3.2 Background

A data model defines the way data can be organized, both in terms of how data are structured, defining which properties each entity should have, and how different data points relate to each other. The role of a data model is to support the development of information systems through the description of data definitions and formats, which ensures data compatibility and consistency across the whole system. Different types of technologies can be used for defining a data model, in the case

of XMANAI project, Graph Database modelling is considered to guarantee flexible relationships between data points in opposition to relational databases.

A graph is a data structure usually comprising nodes and edges but the way that they operate can have some differences thus their classification can be defined as: directed graphs, where the edges connecting the nodes have a set direction (these are the most commonly used graph types for graph databases); mixed graphs, where some edges are directed and others are not; graphs where the edges are not directed at all and graphs where the edges have a weight or "cost" to traverse. The graphs themselves can then also be distributed in different categories, for example, a complete graph, where each pair of vertices is joined by an edge, a finite graph, where the sets of vertices and edges are finite or a planar graph, where the edges and vertices can be drawn in a two-dimensional plane and no edges intercept each other.

In order to find the best approach to follow in the XMANAI project, pros and cons of each of the methods presented were analysed to understand which one is the most appropriate for the kind of data the project handles.

RDF style graph database

An RDF-Store, or Triple-Store style graph focuses on the relationships between the data, and it follows a much stricter group of rules and structure, since it comes down to the one defined by RDF triples: a group of three data points, a subject, a predicate and an object. The RDF triples can then be used to define the structure of the graph database itself through ontologies that will list the different subjects that can be defined, properties that can be used for linking the data that is going to be later stored in the database and the kinds of objects expected to be linked to each property.

Advantages: provides inferences, from the ontologies defining its structure; it's much easier to derive information from the graph if the subjects defined in the ontology of the database can describe relationships with other subjects without a direct link through an RDF triple. For example, if the concept of animal is defined in the ontology and the concept of dog is defined as well, then it's not necessary to create a triple for every individual dog to be linked to the animal concept, it can be done through a subclass definition in the ontology schema for the graph; allow semantic searches which can be very helpful for specific human machine interactions and query systems as well as machine to machine interoperability and is intimately related to the semantic web in general.

Disadvantages: higher cost to traverse edges which tends to be logarithmic in nature.

Labelled Property graph database

A Labelled property graph is a data representation approach that is structured as nodes, properties and relationships. A node contains zero or more properties, where properties are key-value pairs, and two nodes are connected by a directed relationship (edge) that can also contain zero or more properties.

Advantages: tend to be better optimized for graph traversals (less steps are needed from one node to another node) or through the usage of certain algorithms the costs can vary. This enables the processing of graph data to be done at potentially faster rates which can be crucial depending on the manufacturing processes required by the user; Its similarity to other programming language data structures provide an easier integration with different applications that might need to use the database, making it also easier to implement without much data manipulation after retrieval of the data from the database, which can also improve response times from the services developed around the database.

Disadvantages: Lack of ability to provide indexations for all nodes, which makes direct access to nodes based on attribute values not possible, so even if the graph traversal is easier in Labelled Property

Graphs, it can be more difficult to find a particular node, even though after that, finding the nodes related to that one is made easier.

Both approaches have advantages, so the final point of comparison was the flexibility of the graphs. Since Labelled Property graphs have flexible schemas to define their structure or can even be schema free, they offer much more flexibility of which data can be stored and even scalability if there is any need to expand the data, whereas the RDF storage is not easily expandable since it relies on ontologies as Schemas for defining its data structure. This makes Labelled Property Graph style database a more appropriate choice for the data model of the XMANAI project.

3.3 Methodology

For deriving the XMANAI data model, the methodology presented in this section was followed. It comprises five steps with two complementary steps, as showed in Figure 3-1.

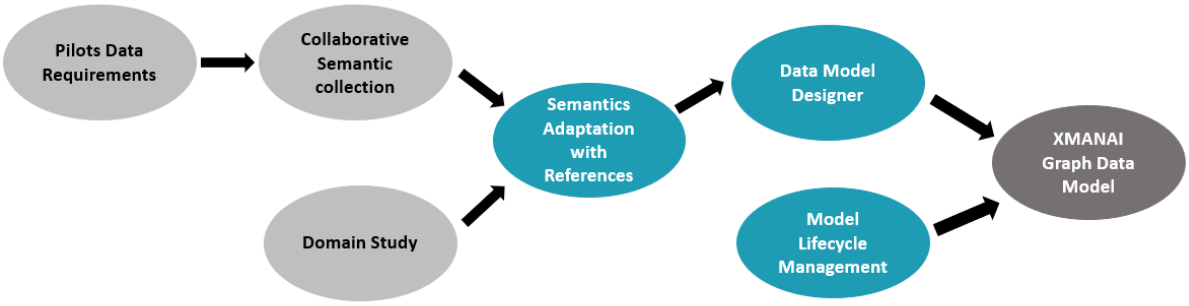

Figure 3-1: XMANAI Data Model Methodology

The steps are described as follows:

- **Collection of demonstrators data requirements:** the starting point of the definition of the data model was done taking in consideration the demonstrators' data examples and their requirements towards XMANAI platform. To gather this information in a cohesive way, the partners that are responsible for the technological development of the demonstrators together with the demonstrator partners, built on the demonstrators' data profiling presented in the XMANAI Deliverable D1.2 and followed a template document that is available for consultation under the Chapter 3 of XMANAI Trial Handbook.
- **Collaborative semantic collection:** Project partners got together and collaboratively tried to define semantics and vocabularies for the manufacturing scenarios. This exercise was made focusing mainly on the demonstrator scenarios and requirements since the manufacturing industry scope is wide, and it is unmanageable to assess all the possibilities.
- **Manufacturing domain study:** At the same time a study on the existing models and ontology standards in manufacturing was performed by project partners to understand the different manufacturing domains and their needed intrinsic differences. The study is presented in the following section of this document.
- **Semantics Adaptation with references:** Taking the two previous steps as inputs, an adaptation and extension on the ontologies collected and elected as the most suitable ones for XMANAI, was made by partners so that all the pilots' domains could be addressed, without losing the manufacturing background.
- **Data Model Designer:** With the semantic information collected and defined, it became possible for the project partners to start drafting a data model suitable to cover the XMANAI demonstrators domains and specificities.
- **Model lifecycle management:** In this step, project partners created a set of rules to guarantee the maintenance of XMANAI data model during its entire lifecycle.

- **XMANAI Graph Data model:** Finally in the last step, the processes are finalised with the achievement of the defined goal and the early data model of XMANAI project that will be refined in the next months was defined.

3.4 Domain Study

This section presents the state-of-the art review made by project partners for existing relevant models, standards, ontologies and schemas. It also presents existing knowledge representations for manufacturing semantics and vocabularies.

3.4.1 ISO 10303

ISO 10303[1] was based on the Product Data Exchange Specification (PDES) which was submitted to ISO in 1988. PDES was a data definition effort intended to improve interoperability between manufacturing companies to improve productivity. ISO 10303 begun its development in 1984 with the purpose of having a "single, complete, implementation-independent Product information model", but because of its complexity it had to be broken into smaller parts. Around 1994/95 ISO published the initial release as an international standard and today it has expanded to many more parts that have been added over the years, including the AP 203 Configuration controlled 3D design which is one of the most important ones as it is supported by many CAD systems.

ISO 10303 is a standard for computer-interpretable data regarding 3D models for product manufacturing. The official title for this standard is "Automation systems and integration - Product data representation and exchange", informally it is known as STEP which stands for "STandard for the Exchange of Product model data". It enables the interchange of data between different systems used in the manufacturing process, namely several CAx (Computer-aided Technologies) systems, from 3D modelling in CAD (Computer-aided design) software to the final step of production with CAM (Computer-aided manufacturing) software and it can also be used for analytics tasks in product management tools.

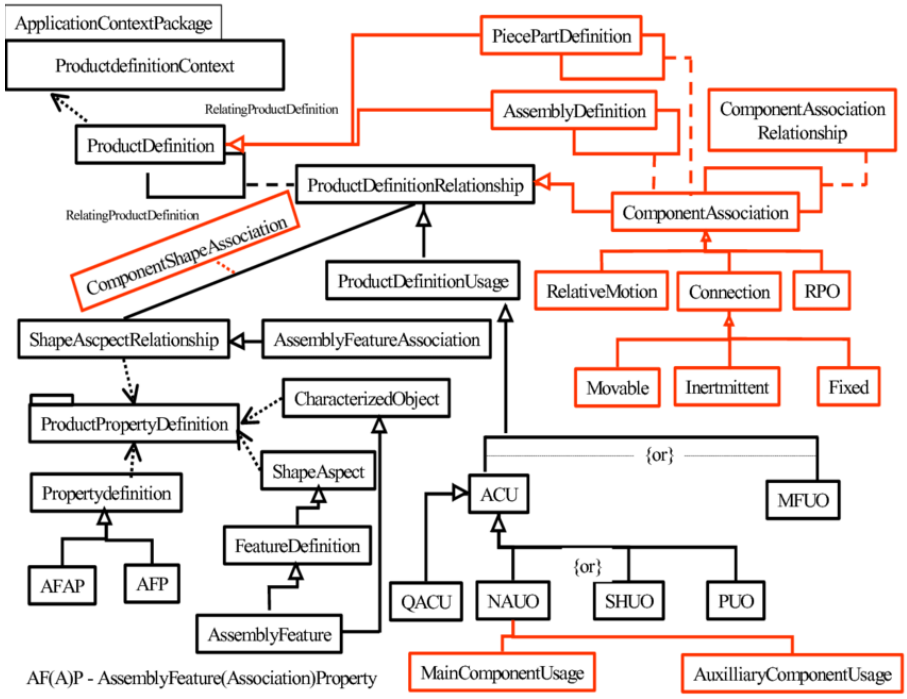

Figure 3-2: Overview of ISO 10303 Part Structure in UML Notation (Rachuri, 2003)

[1] https://www.iso.org/standard/66654.html

There are three main areas ISO 10303 can be applied to: design, manufacturing and lifecycle support, which requires an organized structure to understand how they can be utilized. The structure of ISO 10303 is divided in many parts, each part organized in one of three groups namely, the environment, the integrated data models and the top parts groups.

The environment group describes the modelling language used for ISO 10303, called EXPRESS and the data structures that can be used to exchange data from models. Those data structures are either STEP-File, STEP-XML or via shared database access using SDAI (standard data access interface). The final purpose of the environment group is the methodology and framework for conformance testing.

The Integrated data models group consists of the definitions for the generic resources of products, being the most significant ones the fundamentals of product description and support, the geometric and topological representation, the representation structures and the product structure configuration. Then there are many other resources like Materials, Visual representation, Mathematical descriptions and Numerical analysis. There are also application resources which include assembly model for products, computational fluid dynamics data, modelling commands for exchanging 2D CAD models, and many other tools.

The Top parts group define more specific APs for well-defined combinations and configurations to represent particular data models of an engineering or technical application, for example, Configuration controlled 3D designs of mechanical parts and assemblies, Sheet metal die planning and design, Core data for automotive mechanical design processes, and many more.

3.4.2 ISO 15926

The ISO 15926[2] is a standard for **data integration, sharing, exchange**, and hand-over between computer systems. Specifically, it has been conceived for the representation of process plant life-cycle information according to a common standard, to promote interoperability among industrial automation systems for process plants.

At the beginning of 1990s, the need of developing a European standard data model for industrial lifecycle information, that boosted the creation of the Standard ISO 10303, enhanced the development also of ISO 15926, deriving from the previous one but more focused on process industries. The current Standard is the result of more than twenty years of collaborative activities, when experts from various domains have been involved and a number of existing classes and libraries have been standardized to be added in the final version.

In 2003, "Part 2 – Data Model" of ISO 15926 was published and it contains specifications for a generic data model that can support all disciplines, supply chain company types and life cycle stages. It was shortly followed by "Part 1 - Overview and fundamental principles" where the main purpose of ISO 15926 is explained. Last release "Part 10 - Conformance testing" was published in 2019 but, however, not all the expected 13 parts are finalized yet.

[2] https://www.iso.org/standard/70694.html

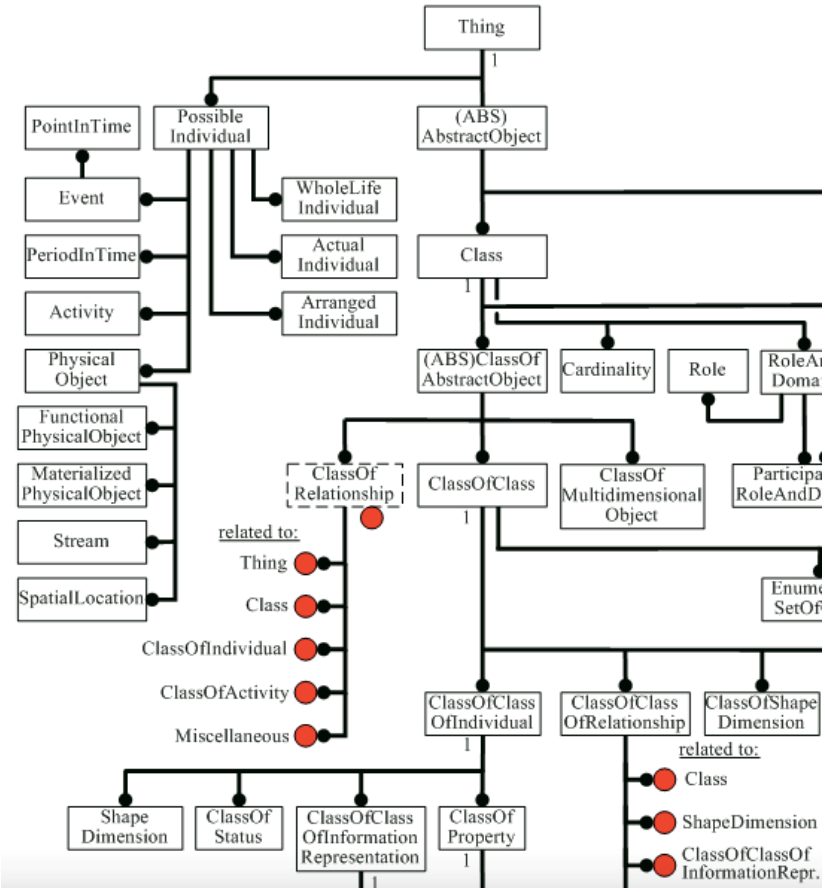

Figure 3-3: Detail of "Possible Individual" and related relationship[3]

The plant representation is specified by a generic, conceptual data model designed to be used in conjunction with reference data: the scope of the 4D (spatio-temporal) data model is to cover the entire lifecycle of a facility and its components.

A wide number of industry domains can be modelled according to the provided standard, for instance, hydrocarbon process and conditioning systems, injected gas and water conditioning and injection systems, oil and gas product transport systems, safety and control systems, electricity generation and its supply systems, steam generation and its supply systems, structures, buildings and accommodation.

Items are modelled according to the guidelines provided by ISO 15926 in Part 2 and the entire product life-cycle of materials is described. It starts from the preliminary phases when requirements for production and transportation are specified, material functions are defined, and the market availability of required materials and equipment is verified. It includes also the specification for the installation and commissioning of plant equipment, of the production and process operations (taking into account also processes conditions and consumption, yields and quality of processes material), while the last stage is maintenance and replacement of equipment.

To complement the model, reference data are provided in Part 4 where a proper vocabulary of reference data is set-up and a set of initial reference data is provided. Classes and relationship can be described based on an ontology approach, consistent with the W3C Web Ontology Language (OWL), as specified in "Part 8 - Implementation methods for the integration of distributed systems: Web

[3] https://15926.org/topics/data-model/index.htm#diagram_for_top_level

Ontology Language (OWL) implementation", where rules to translate the data model into the RDF and OWL languages are provided.

3.4.3 X3D ontology

To better approach the data structure and database defined, ontologies are required to enable the knowledge representation offering well-established standards. Ontologies can describe objects as well as classes and properties of that same objects making them a common solution for content description through several applications and domains. The ontologies also allow to represent different features of 3D content such as geometry, structure, appearance and animation but commonly each ontology only covers part of the listed features.

Extensible 3D (X3D) Ontology[4] is an RDF (*Resource Description Framework*)/RDFS (*Resource Description Framework Schema*) /OWL (*Web Ontology Language*) specification providing a representation of a X3D architecture through a document composed of arbitrarily ordered Terminological knowledge which describes concepts and relations between them; Relational knowledge that provides hierarchies and properties of relations; Assertional knowledge that describes facts regarding objects using concepts formalized by Terminological and Relational knowledge. This structure allows to represent a model into subject-predicate-object triples, that then can be queried using SPARQL protocol with RDF Query Language.

The goal of the ontology is to provide flexible integration of the X3D standard with the semantic web technologies currently in place. Hence, the X3D ontology is generated automatically from a schema XML which is described in the X3D Unified Object Model.

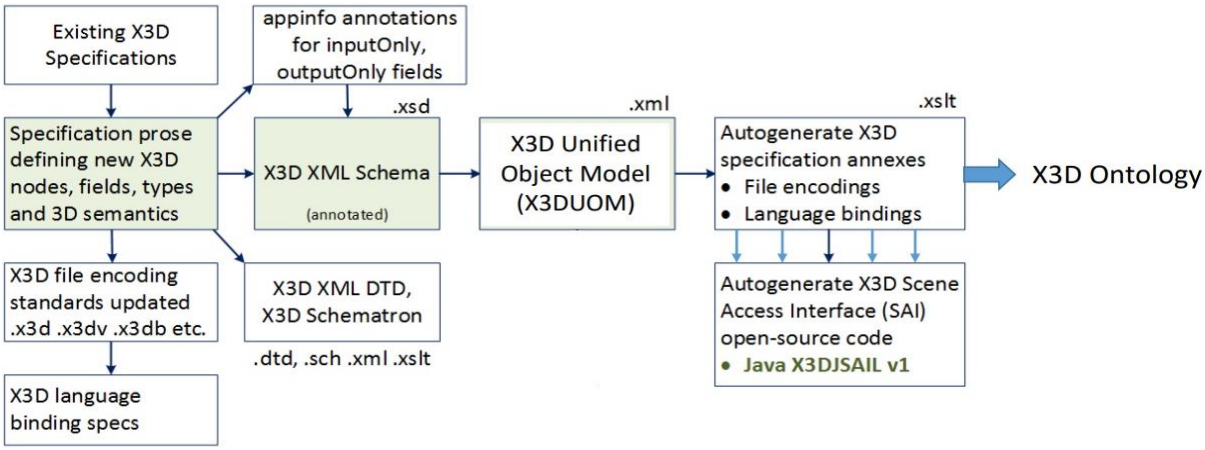

Figure 3-4: X3D specification (Brutzman, 2020)

Being an XML schema, this document contains a list of nodes, interfaces and fields required for the X3D, information related to the hierarchy of nodes and fields as well as each field data type. This approach combines both semantic and syntactic elements of X3D models with the support of metadata to integrate with Semantic Web.

X3D includes a wide range of features that can be used for industrial design in engineering and scientific visualization, CAD and architecture, Geospatial, Human animation, 3D printing and 3D Scanning, AR/MR/VR. Today X3D has several applications in medical visualization, training and simulation, multimedia, entertainment, education, etc. The ontology contains the full set of data types and graph parent-child relationships available in the X3D, which provides the capability of query any

[4] https://www.web3d.org/x3d/content/semantics/semantics.html

X3D model using besides the already existing properties, additional derivative ones to facilitate those same queries like *HasChild*, *hasGeometry* with SPARQL.

3.4.4 I40KG

Due to the rising popularity of digitizing processes, components and complete production lines, there has been a lot of research of norms, standards and specifications for Industry 4.0 systems. Because of this, an overwhelming amount of data was needed to be sorted through and evaluated for each specification. I40KG attempts to create a structuring approach to better organize the relevant entities and explicitly outline their interlinks and attributes to better aid both domain experts and newcomers.

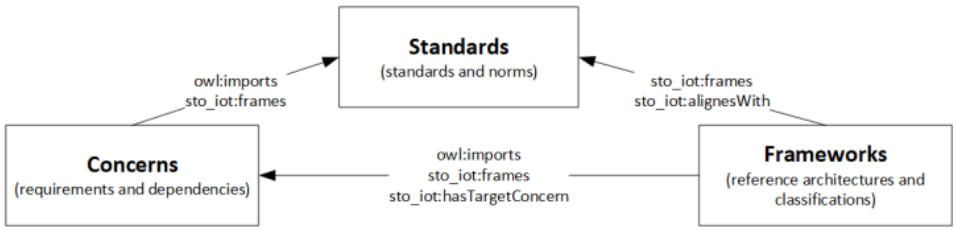

Figure 3-5: The three partitions of the I40KG[5]

In order to keep up with the dynamic and evolving domains in the digital transformation of manufacturing systems (Industry 4.0), this structure was proposed in the form of a semantically annotated knowledge graph for related standards, norms and reference frameworks. It provides a Linked Data-compliant collection of annotated, classified reference guidelines supporting newcomers and experts alike in understanding how to implement Industry 4.0 systems. The proposal is a publicly available knowledge graph that contains the latest state of Industry 4.0 specifications with respect to standards, reference frameworks as well as key requirements. The inter-linked nature of the content and its various relations to outside topics led to the design of an RDF-based knowledge graph. The goal is to be able to retrieve the following types of information: "Where can additional information about a certain topic be found?", "Which specification is most appropriate for establishing a secure data exchange between Industry 4.0 devices?" and "What are the requirements related to a specific Industry 4.0 challenge and where can appropriate guidance to solving them be found?"

As a knowledge graph based in RDF, I40KG has an ontology describing all the relationships between objects/predicates[5].

[5] https://i40-tools.github.io/I40KG/docs/index.html

General	**Name**	Industry 4.0 Knowledge Graph (I40KG)
	DL Expressivity	SHOIF(D)
	Licence (R1.1)	Creative Commons 3
	Size	44 classes, 35 object properties, 22 data properties, 1335 individuals
	Standards and Norms (R1.2)	338 standards and standard parts, 49 ISO standards, 67 IEC standards, 11 DIN standards
	Frameworks	18 reference frameworks divided into 138 classification sections
	Concerns	160 interrelated Industry 4.0 concerns in 6 categories
	External Links (F3, I3)	286 to DBpedia resources, 271 to Wikipedia pages
	Reasoning	4.257 derived triples
	Total size	16.447 unique triples without derived ones
Reuse	**Reused Ontologies** (I2, R1)	DCTERMS, DCELEMS, PROV, DUL, FOAF, OM, etc.
	Reused ODPs	Componency ODP
Documentation	**Element description** (F2, R1)	By means of `rdfs:label`, `rdfs:comment`, `skos:prefLabel` and `rdfs:isDefinedBy`
	`Ontology Documentation`	http://i40.semantic-interoperability.org/sto/
Conventions	**Naming pattern**	CamelCase notation for the schema and Ada for instances
	Linked Data (R1.3)	5 Star Linked Data
Multilinguality	**English labels for all terms**	`rdfs:label` and `rdfs:comment` with the @en notation
Availability	**PersistentURI** (F1)	https://w3id.org/i40/sto
	Serialisations (I1)	Turtle, RDF/XML
	GitHub (A1)	https://github.com/i40-Tools/I40KG/
	LOV (F4)	http://lov.okfn.org/dataset/lov/vocabs/sto
	OntoPortal (A2)	http://iofportal.ncor.buffalo.edu/ontologies/STO
	Licence	Creative Commons 3.0
	VoCol Instance (A2)	http://vocol.iais.fraunhofer.de/sto/

Figure 3-6: Relevant aspects of the I40KG and related resources[5]

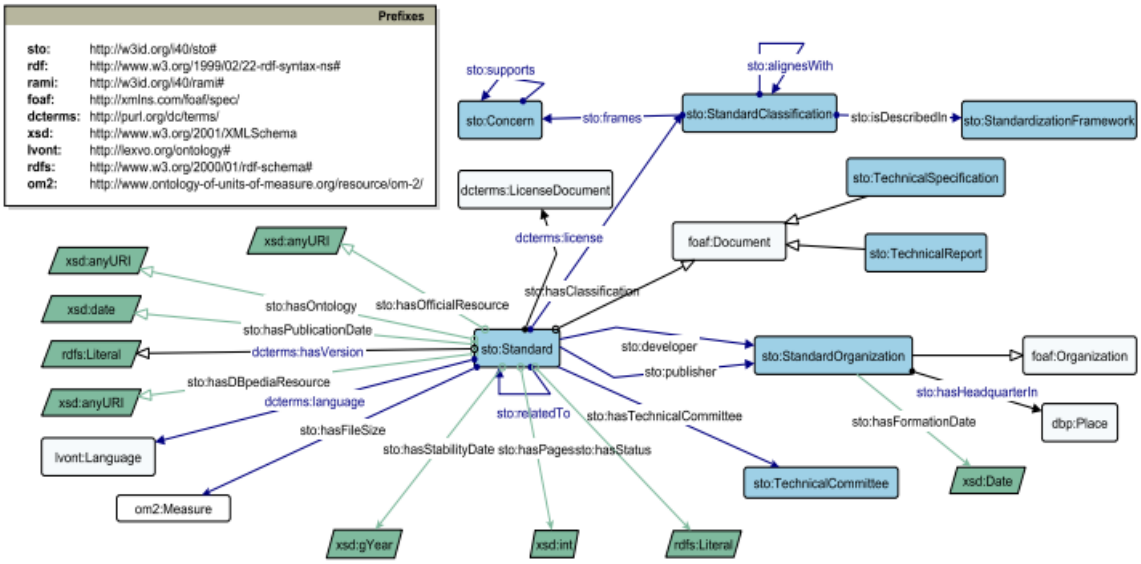

Figure 3-7: Core classes and properties of the Standards Module[5]

3.4.5 SENSR

Created by the OMG (Object Management Group, Inc.), SENSR[6] was made for aiding the interpretation of sensor data, since currently manufacturers of hardware sensors rely on "arcane data sheets" to describe the format of data supplied by their sensors, each with their own data sheet format, and errors and ambiguities can and are likely to occur in these situations.

SENSR provides a metamodel for describing the form of serialized data streams emitted by sensors and describing how that data should be interpreted by a client to derive the intended meaning of the

[6] https://www.omg.org/spec/SENSR/#spec-versions

data. The intent is that a manufacturer's sensor products can be characterized by a model expressed in the proposed metamodel, making it easier for clients to derive the intended meaning of the data emitted by the sensors, however, this metamodel does not concern itself with how the transmission of the data is made from the sensor to the client. This is done through a model provided by the manufacturer as a document called an Electronic data Sheet (EDS), which is a machine consumable document such that it can be generated by the manufacturer and parsed by the consumer. OMG provides an illustration of how the EDS may be used, as follows:

- Equipment is required to offer sensor data to one or more unknown consumers.
- A consumer wishing to receive the data will access the EDS for the sensor product. The delivery of the EDS may occur at any time prior to configuration of the consumer.
- Guided by the EDS, the consumer is adapted to parse the data provided by the sensor. This configuration may occur at any time prior to the communication with the sensor.
- The consumers and the hardware will communicate over a pre-determined protocol, such as Bluetooth, mDNS + TCP/IP, or other to be determined channels.
- The consumer begins to receive data from the sending hardware, and can interpret the data in the way that is intended.

3.4.6 ONTO-PDM

ONTO – PDM (Panetto, 2012) is an ontology-based approach for facilitating systems interoperability in a manufacturing environment. The proposed ontological model is a facilitator for interoperating all application software that share information during product lifecycle. Specifically, due to the knowledge that must be embedded in the manufacturing enterprises, all technical data are stored based on a common model. This ontology framework focuses into formalising all the technical data, and contributes to the definition of a product ontology embedded into the product, making it interoperable with applications, and minimizing the loss of semantics. The implementation of the ontology has been done through its translation into the OWL language for its use with the Protégé, ontology development and instantiation environment, and its translation into an Entity/Relationship model, for its implementation into a Database Management System (DBMS).

The proposed methodology provides insights for generalising a knowledge representation process to feed concepts into an ontological model. To achieve this objective conceptualising, merging and reusing knowledge embedded into existing standards for product technical data (ISO 10303) and ERP/MES data (IEC 62264), is needed to formalize a Product Ontology (ONTO-PDM) for enterprise applications interoperability, centred to the product. The implementation of the ontology has been done through its translation into the OWL language for its use with the Protégé ontology development and instantiation environment and its translation into an Entity/Relationship model for its implementation into a Database Management System (DBMS).

ISO 10303 standard focuses on STREP PDM (Product Data Management) schema, which is a reference information model for the exchange of central, common subset of the data being managed within a PDM system. It represents the intersection of requirements and data structures from a range of STEP Application Protocols, all generally within the domains of design and development of discrete electro/mechanical part and assemblies. The IEC 62264 standard concerns the information related to the interface between plant production scheduling, operation management and plant floor coordination. To take into account the various exchanged information, the IEC 62264 standard defines a set of eight sub-models that specify all concepts for enterprise-control integration. These can be appropriately adopted to setup the Product Ontology for interoperability. The proposed conceptualisation is based on one interpretation of the eight sub-models.

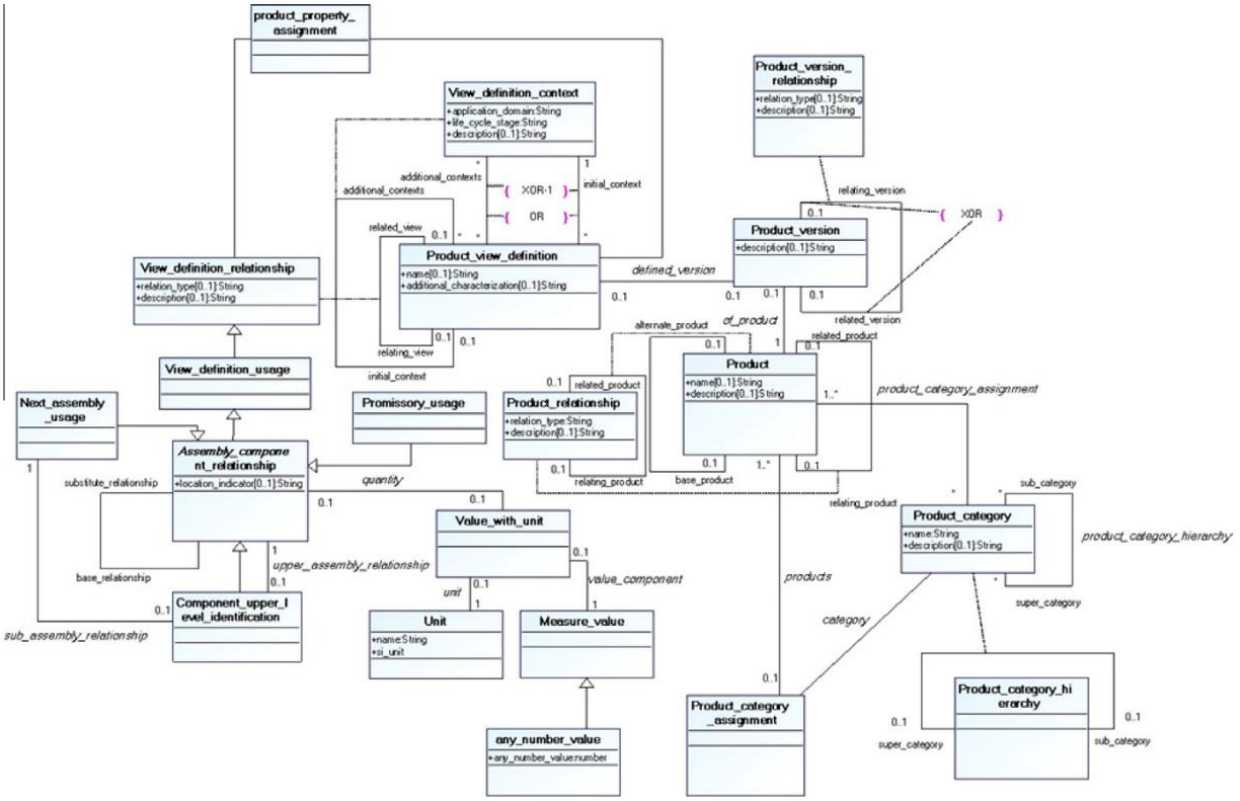

Figure 3-8 Assembly structure module conceptualized in UML (Panetto, 2012)

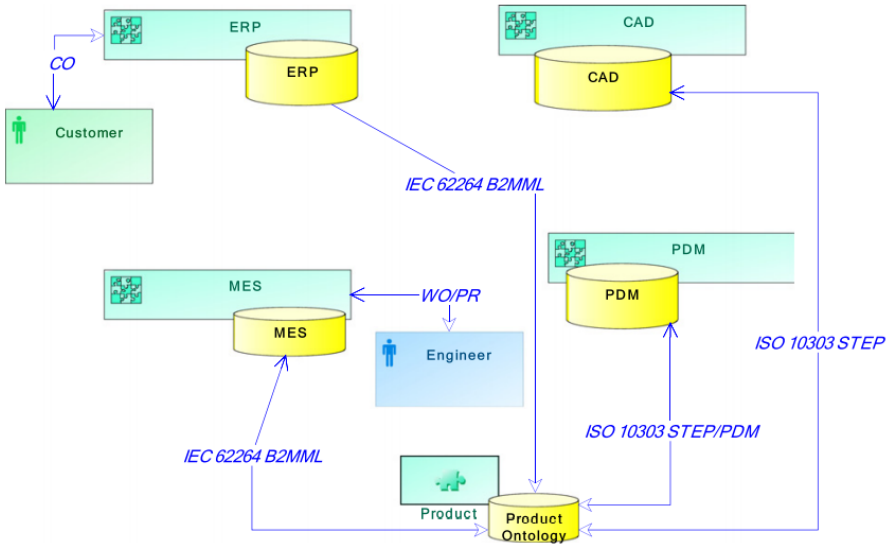

Figure 3-9 Product-centric interoperability architecture (Panetto, 2012)

In order to formalise and verify the mappings between the selected standards, First Order Logic (FOL) patterns to define the semantics of each construct of the standards conceptual models, are proposed, such as class, attribute, association and aggregations, generalization and hierarchies. FOL is a language characterized by a formal specification of the semantics that allows expressing structured knowledge in one hand and promotes the implementation of reasoning support in the other hand. First-Order-Logic (FOL) predicates utilized, where each predicate formalizes mappings between STEP PDM concepts and IEC 62264 ones, represented by a FOL axioms, thus providing an unambiguous representation of knowledge. Starting from the realization, demonstrated by mapping that the

IEC62264 models contain the information included in STEP PDM, the specific information of STEP PDM is merged in B2MML ontology, to build an ontological model that is able to store all product technical data and information, consistent with both standards.

Each manufacturing setting was equipped with its enterprise systems (i.e.Windchill PDM, ProEngineer CAD and SAP R/3 for DiMeG or Flexnet MES and Sage ERP X3 for AIPL), dedicated to specific tasks (engineering tasks or manufacturing ones) and provided by a particular vendor. In this product-centric information system, these heterogeneous applications were forced to interoperate with the product, to store and to draw the pertinent product information on it.

The proposed Product Ontology is capable to support the information exchange between the product and the many applications that interact with it, thus figuring a "product-centric" scenario. In such "product-centric" scenario all information flows are centralised to the product ontology that acts, then, as a mediator between all involved enterprise systems.

3.4.7 VFDM

The Virtual Factory Data Model (VFDM) (Terkaj, 2012) is a common data model for the representation of factory objects related to production systems, resources, processes and products that exploits the advantages of an ontology-based data modelling. The VFDM was the outcome of VFF (Virtual Factory Framework) in 2012, a European research project aimed at developing an integrated virtual environment to enable the interoperability between software tools supporting the factory processes along all the phases of its lifecycle.

The VFDM is built upon the advantages of an ontology-based data modelling, and can be considered as the shared meta-language providing a common definition of the data that are shared among the software tools connected to the framework. Specifically, the VFDM aims at formalizing and integrating the concepts of product, process, production resource and building as handled by the digital tools supporting the factory life-cycle phases. Moreover, the VFDM is designed to exploit already existing technical standards and extends their definitions to represent the characteristics of a manufacturing system in terms of the products to be manufactured, the manufacturing process they must undergo and the resources entitled to operate the different manufacturing operations.

The architecture of VFDM can be seen in Figure 3-10.

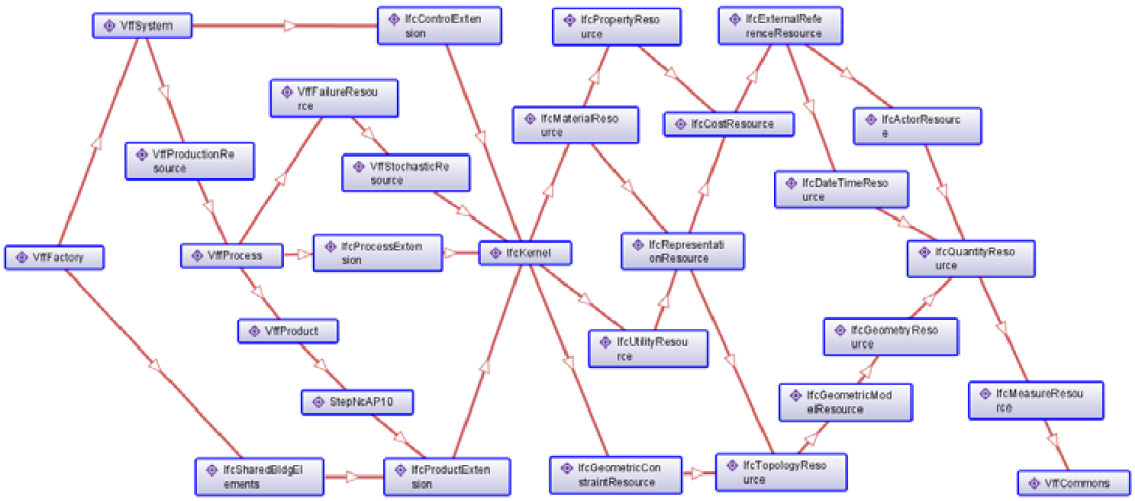

Figure 3-10: Architecture of VFDM (Terkaj, 2012)

At a functional level, VFDM aims at formalizing and integrating key industry-related concepts, specifically for building, product, process and production resources handled by the digital tools supporting the factory life-cycle phases and production system:

- Building: physical structure and constructions of a factory (e.g. walls, columns, etc.).
- Product: the production output of the factory.
- Process: the set of actions executed by a system that take place during the manufacture and that affect or transform the product.
- Production Resource: the resources needed during the processes to transform the product, such as human operators, machines, production lines, conveyors, etc.
- Production System: the transformation system (e.g. manufacturing system, assembly system) that affects a product by means of physical resources and/or human resources within a process.
- Factory: the factory seen as whole during its lifecycle.

The current version of VFDM is mainly based on the IFC and STEP-NC standards that were translated into a set of ontologies The Entities in the IFC standard are mapped to OWL Classes in the VFDM. Most of the classes derived from IFC are specializations of two fundamental classes named IfcTypeObject and IfcObject, both being subclasses of IfcObjectDefinition. The former class is the generalization of any thing or process seen as a type, the latter seen as an occurrence. IfcObject has the following subclasses: IfcProduct, IfcProcess, IfcResource, IfcControl, IfcActor, IfcGroup. IfcProduct represents the occurrence of a generic object that can be related to a geometric or spatial context (e.g. a manufactured products, machine tools, transport systems, etc.). IfcProcess defines a process that can be used to transform an input into output (e.g. an assembly operation, machining operation, etc.). IfcResource represents the information related to resource needed to execute a process. IfcControl is the generalization of the concepts that control or constrain the use of products, processes, or resources. IfcActor defines the actors or human agents that are involved in a project. IfcGroup is the generalization of any group. The subclasses of IfcTypeObject (i.e. IfcTypeProduct, IfcTypeProcess, IfcTypeResource) can be paired with the corresponding subclasses of IfcObject.

The previously described generic classes can be exploited to model a wide range of manufacturing systems while taking into consideration both physical and logical aspects. A key feature of the IFC standard consists in the availability of predefined relationships that characterize the data by specifying assignments, associations, connections, and decompositions according to the required level of details. For example, these relationships allow to explicitly define the following links:

- Assignment of a process step to the production resource that can execute it
- Nesting of a process into its sub-processes and decomposition of a machine tool into its components
- Precedence constraints between the processes
- Input and output entities of a process

3.4.8 CMData

While there are existing product data exchange-oriented standards covering different parts of a product's history, this data model is defined as a full version of existing standards focused on production planning and supply chain collaboration. Furthermore, this proposal complements and responds to new business needs for cloud collaboration in industry. The goals of CMData (Andres, 2021) are the following ones:

- To enhance data interpretation.
- To improve sharing between enterprise legacy systems and cloud environments.
- To identify the necessary data for planning activities in a collaborative way.

CMData considers two approaches. Firstly, an academic approach, focused on the literature review of restocking, manufacturing and delivery planning problems. Secondly, an industrial approach, in charge of identifying the plans followed by companies to carry out very complex and concrete solutions that are not addressed in the academic field.

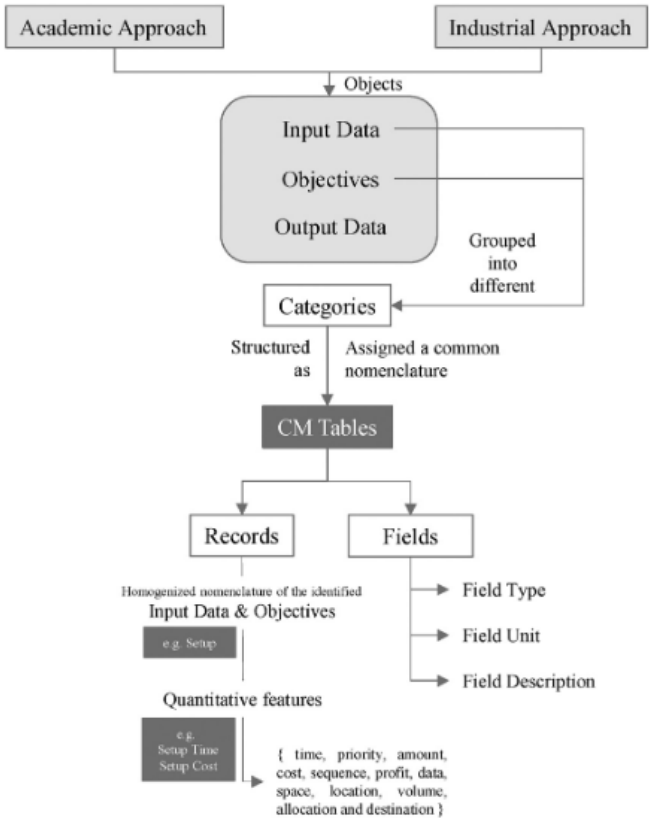

Figure 3-11: Hierarchical Structure of CMData (Andres, 2021)

3.4.9 MASON

MASON (MAnufacturing's Semantics ONtology) (Lemaignan, 2006) is an ontology based on and inherited from the Web Ontology Languages (OWL) ontologies. It proposes three concepts: entities, operations and resources.

- Entities. Its purpose is to specify the product and give an abstract view of it.
- Operations. They are related to the description of processes.
- Resources. They refer to all manufacturing resources.

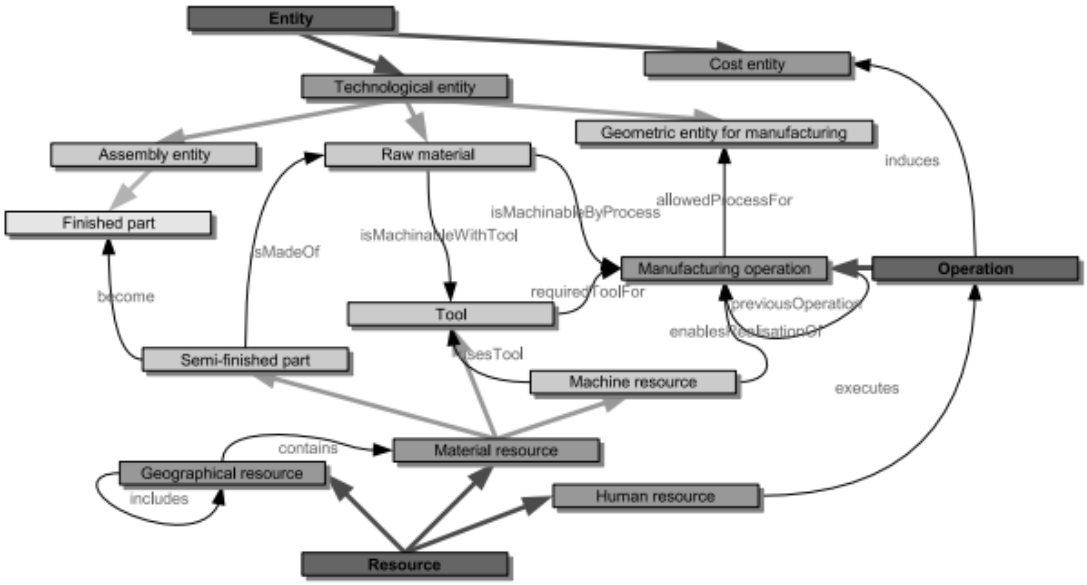

Figure 3-12: Overview of MASON (Lemaignan, 2006)

3.4.10 ISO 21838

This standard was developed in response to the demand from many quarters for ontology-based solutions to the problem of **semantic interoperability across networks of information systems**. The demand arised particularly from large organizations and consortia of organizations in areas such as bioinformatics, healthcare, the manufacturing industry, and military and government administration, where information systems, created independently, need to exchange data in such a way that meaning is preserved.

One of the best ways of representing networks is graph data modelling, where objects and connections are represented in shape of graph, equipped with nodes and relationships with properties and labels. A big problem for knowledge graphs is the tendency to generate silos: actually, if the design involves different actors (each one developing the structure to suit their data needs), it often happens that the final models overlap, presenting aspects in common that cannot be merged. The result in very many instances is the need for a considerable amount of manual effort, which often takes the form of creating an entirely new knowledge graph artifact, which then forms a new silo in its own right.

The biomedical community firstly proposed a solution that was then adopted by other sectors (defence and intelligence, for instance), based on the concept of top-level ontologies and spoke ontologies inheriting information from the previous level of the chain. ISO 21838[7] incorporates such techniques and provides the standard for the creation of suites of interoperable ontology models, designed for graph data modelling: part 1 identifies the requirements which must be satisfied by a top-level ontology, part 2 certifies that Basic Formal Ontology satisfies these requirements.

3.4.11MKG

Manufacturing knowledge (MK) (He, 2019) for the reuse of product-related production problems (PP) in industrial decision making has been extensively studied in the academic field. However, related practices and studies are fragmented and insufficient. MKG for PP proposes, based on a framework of

[7] https://www.iso.org/standard/71954.html

relationships, to build a manufacturing knowledge graph (MKG) based on ontologies in order to integrate different MKs. It aims to represent the details between knowledge entities.

This framework contains three layers:

- Lower layer. It integrates various types of knowledge as well as problems solved in the past.
- Intermediate layer. It establishes relationships and correspondences between decisions to help the decision-maker to address a specific problem based on all the related integrated knowledge.
- Upper layer. It simulates an interface that has as input a concrete problem and offers as output concrete solutions to deal with it.

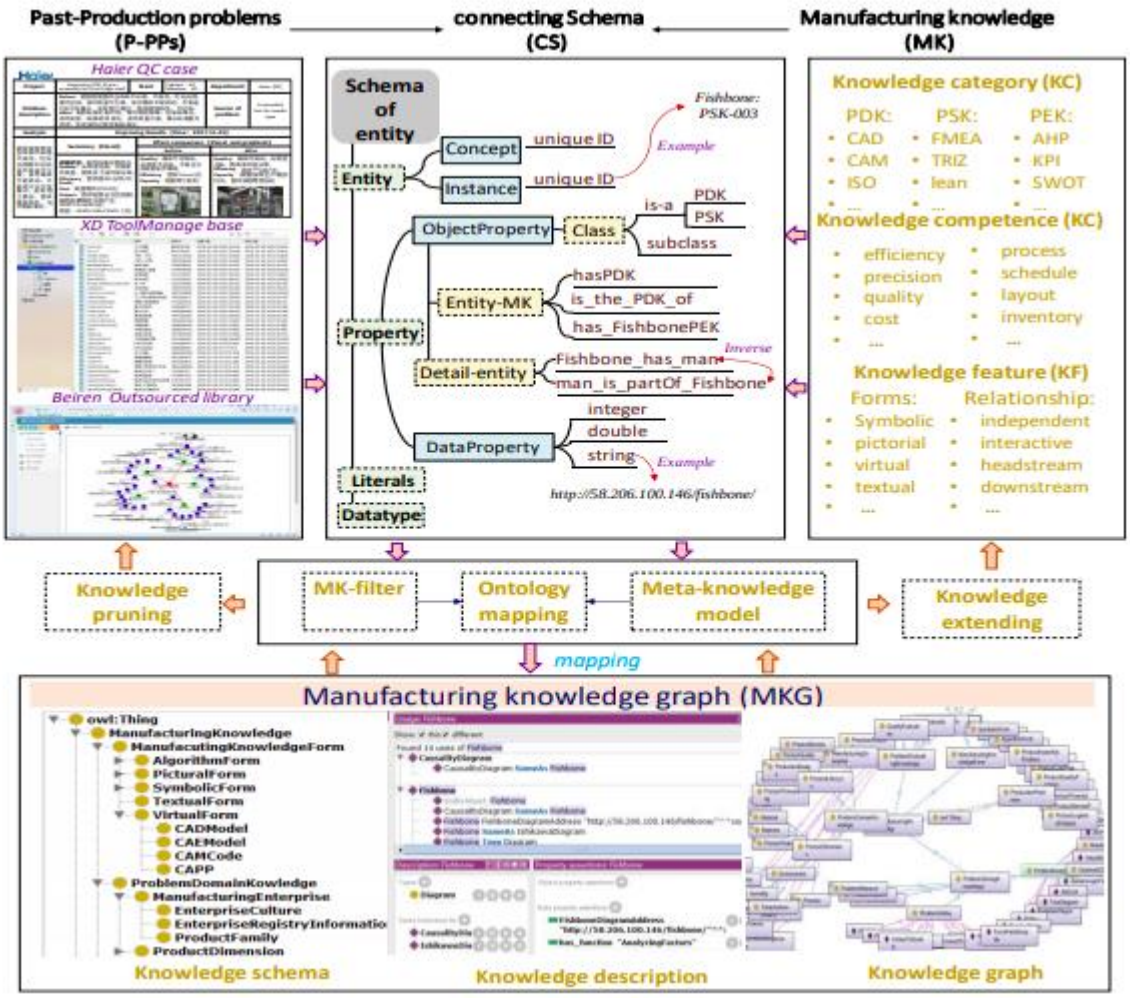

Figure 3-13: Detail of MKG (He, 2019)

3.4.12AMO

Since the development of the manufacturing industry rapidly increases and becomes more competitive, quality and cost are significant factors that require to be studied by the manufacturers. These factors are affected closely by the process and the material. The Additive Manufacturing Ontology (Ali, 2019) is an ontology designed to represent the Additive Manufacturing (AM) Product Life Cycle (PLC). Specifically, AMO is a modular ontology that employs Basic Formal Ontology (BFO) as its top level, while also drawing from the Common Core Ontologies (CCO) and three other ontologies from the Coordinated Holistic Alignment of Manufacturing Process Ontologies (CHAMP) that

represent manufacturing processes: the Manufacturing Process Ontology, the Design Ontology, and the Testing Process Ontology. This combination of various ontologies makes sure that AMO exploits normally used terms and definitions and consequently, increases the prospects that AMO can be re-used and combined with other ontologies. In general, AMO uses eight process entities in which some of these entities are imported from the Design Ontology and the Manufacturing Process Ontology.

AMO was developed as a mid-level ontology that can be re-used for various types of additive manufacturing and it seemed to be suitable for PLC applications. Moreover, the properties interrelating objects of different ontologies were only defined directly in the AMO at the instance level. In such way, the generality of AMO was established. For the sake of completeness, an application ontology extending AMO with new classes related to different fields has been developed. This indicated that AMO was able to be re-usable to utilize the application ontology; while, new object properties were not necessary to be created.

3.4.13SIMPM

The "Semantically Integrated Manufacturing Planning Model" (Šormaz, 2019) takes care of a specific aspect of manufacturing, the **process planning**, that needs special attention in the context of the new manufacturing strategy, where manufacturing resources (both hardware and software) may be shared, be on-demand, be hosted in cloud services, ecc. The Manufacturing Process Planning (MPP) deals with the selection of suitable manufacturing processes and related machine and tools for a particular product design.

The purpose of the SIMPM ontology is to model three fundamental constraints of the MPP: variety, time, and aggregation. The ontology proposed in the paper is a foundation (or upper-level) ontology, built with most general and reusable concepts, while domain dependent details are captured in the core or lower ontologies. It is proposed as a collection of OWL (Ontology Web Language) axioms, which may provide upper level semantics for capturing the knowledge of manufacturing process planning.

The primary goal of the SIMPM foundation ontology is to link planning variables from one aggregation dimension to another by establishing logical links. In process planning, several entities interact each other: to describe a task it is required a list of operations (e.g. cutting, joining, molding, bending, covering), a list of agents (either machine or human), a set of tools, and the final goal. In addition, every machinable feature of a part design is linked to suitable manufacturing processes, compatible machine and tool to use. SIMPM enhances dynamic process plan, since any change in the design level can easily be translated to required modifications in corresponding plans by identifying the affected portion of plan using these links.

3.4.14 SSN Ontology

The Semantic Sensor Network (SSN) ontology[8] is an ontology for **describing sensors** (S) and their observations (O), the involved procedures, the studied features of interest, the samples (S) used to do so, and the observed properties, as well as actuators (A). SSN follows a horizontal and vertical modularization architecture by including a lightweight but self-contained core ontology called SOSA (Sensor, Observation, Sample, and Actuator) for its elementary classes and properties. With their different scope and different degrees of axiomatization, SSN and SOSA are able to support a wide range of applications and use cases, including satellite imagery, large-scale scientific monitoring, industrial and household infrastructures, social sensing, citizen science, observation-driven ontology engineering, and the Web of Things.

[8] https://www.w3.org/TR/vocab-ssn/

Currently sensors represent the major source of data available on the Web today, so it is required to manage such data not simply as values, but also to interpret them, understanding the meaning behind. Any value should be equipped with information about the sensor that has generated it, the environment around it, the technique used for catching the information, etc.

The objective of SSN is to combine the standards provided by the Open Geospatial Consortium (OGC) to describe sensors and observation with those designed by W3C for describing interconnected graphs of data, including also actuators and actuation.

3.4.15 MCCO

The aim of the Manufacturing Core Concepts Ontology (Usman, 2011) was to provide an ontological foundation to distribute the manufacturing knowledge among the production and the design domain of the product lifecycle. Mainly, the communication across different domains that is necessary for machines is referred to in MCCO as interoperability. The term of Interoperability indicates "the ability to share technical and business data, information and knowledge seamlessly across two or more software tools or application systems in an error free manner with minimal manual interventions". Moreover, the underlying technological base of the MCCO has provided an understanding of production quality, cost and timescales which potentially determined additional connections to a business perspective.

MCCO was formed by identifying a core set of concepts formalized in heavyweight logic. Fundamentally, heavyweight ontologies define concepts, control their use, capture knowledge and provide a route to share across design and production and provide preferably reasoning capability compared to the databases with fixed form and format. Common logic (ISO/IEC 24707:200) based Knowledge Frame Language (KFL) provided by Highfleet was used for heavyweight formalization. Moreover, three levels of specialization, the generic, the product lifecycle generic and the domain specific level have been identified for the MCCO. Feature concepts and other concepts in MCCO have been specialized at these three levels. Overall, the feature concepts were used to: i) show the implementation of the multi-level ontology and the core concepts and ii) provide a path of communicating, identifying and sharing knowledge across design and production domains. Finally, MCCO was loaded in the Integrated Ontology Development Environment (IODE) in which the five ontology aspects (1. Capturing of semantics, 2. Controlling concepts, 3. Capturing Domain Knowledge, 4. Inferring Knowledge, 5. Route to knowledge sharing through MCCO) have been tested.

3.4.16 Context Ontology

An ontology-based approach for context modeling is proposed in the industrial domain in (Giustozzi, 2018). Specifically, for industrial production in Industry 4.0, items like sensors, devices (which both of them generate a massive amount of data) and enterprise assets are connected to each other as well as to the Internet. In this approach, the execution of industrial processes depends on i) their internal state, ii) user interactions and iii) the context of their execution, in order to be context-aware and provide added-value information to increase the monitoring of operations and their performance.

The proposed Context Ontology approach represents the significant general concepts and the relations in the industrial domain to attain context representation and context reasoning. These fundamental context types are based on the location, the identity, the time and the activity which are able to characterize the situation of a particular entity. These context types reply to questions of who, what, when, where, and additionally, they can deduce other contextual information. The main ontologies of the Context Ontology consisted of a combination of other smaller ones and can be classified according to the subject of their conceptualization in three categories: general or core ontologies, domain ontologies and application ontologies. Particularly, core ontologies including the

Time ontology, the Location ontology and the Sensor ontology are global and domain-independent conceptualizations that can be reused across various domains of knowledge. Domain ontologies include concepts that describe industrial processes like the Process ontology, the Resource ontology and the Situation ontology. In addition, the domain ontologies model common concepts and aspects for certain domains, which can be reused for more specific scenarios. The Context Ontology is written in OWL which is developed by W3C, the ontology is developed with the SWRLAPI and the OWLAPI; while, smaller ontologies are based on the Semantic Web Rule Language (SWRL3). To conclude, the proposed ontology is generic and extensible to accommodate a wide spectrum of manufacturing services by providing structures for context-related concepts, rules and their semantics.

3.4.17 BigQuery Export schema for Google Analytics

The manufacturing domain covers a wide range of activities, not all of them tightly linked to the core manufacturing process. Google Analytics is a web analytics service that provides statistics and basic analytical tools for search engine optimization (SEO) and marketing purposes and in this context it is considered relevant to the XMANAI scope to ensure that data coming from the manufacturers' online presence (i.e. related to their websites) will be also considered. BigQuery is a solution provided by Google towards a fully-managed enterprised cloud-based data warehouse supported with machine learning, geospatial analysis and business intelligence.

Data, including those coming from the Google Analytics services, are stored using a columnar storage format optimized for analytical queries. It should also be noted that database transaction semantics are supported, therefore making the underlying utilised schema interesting for exploration.To export data from BigQuery to be used in customer models or to be joined with other data sets, BigQuery provides export from Google Analytics with streaming export (BigQuery Export https://support.google.com/analytics/answer/9358801).

Exports follows a schema that can be done for the current day (*intraday events*) providing the most recent data however might not have the complete set of data due to limitations regarding processing late events and failed uploads are some of the examples, or can be exported daily representing a complete day (*events*) of data being a more stable dataset. For each intraday event a table is created with the format "ga_sessios_intraday_AAAAMMDD" containing per session all the information, this type of import is automatically made three times per day overwriting the previous one when a new one is generated. When all data are collected and processed for the current day, is then created a table with the format "ga_sessions_AAAAMMDD" which will substitute the intraday events, aggregating all the values collected through the intraday events in a single record per session.

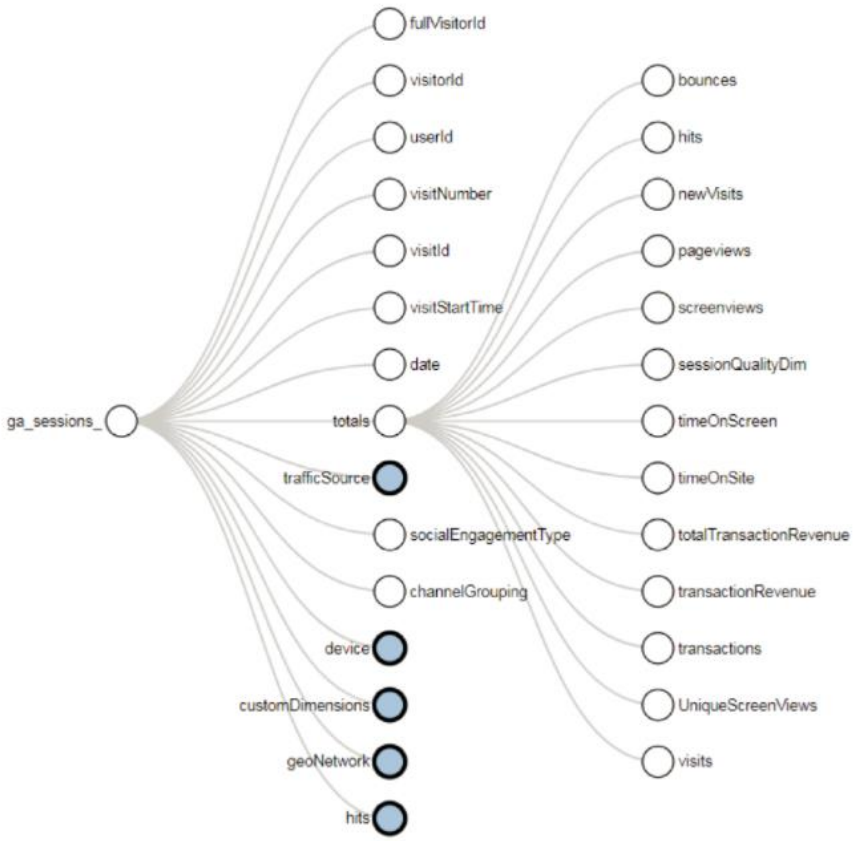

Figure 3-14: Tree visualization of the Google Analytics 360 BigQuery Export Schema from (Marciszewski, 2018)

3.5 XMANAI Graph Data Model

The following sections describe the rationale adopted in the creation of XMANAI Data Model.

3.5.1 Generic definitions

The standard definition of a data model is a representation of the structure of data that is going to be stored on a traditional table-based database. It also describes the tables and relationships between tables. Although different databases will be used in XMANAI, the Data Model describes the **nodes**, **properties of nodes** and **relationships between nodes**. The nodes in a graph data model are based on **concepts** that can represent some object or notion that reflects reality in a digital form. They contain different **properties** to attempt to mirror the concept that will be stored, from shape and size, to scheduling times and productivity metrics. These nodes and their properties can be created from a structure defined in the graph data model.

Much like in the physical world, some concepts have to interact with one another, therefore there must be a process through which **relationships** are created. For example, if a machine in a production line is producing a certain object, this object has to have a path to reach a consumer, this interaction may involve many steps and imply that there is a relationship between the machine and a distribution system, in the case of a graph data model, this would be represented as a relationship between nodes, where the machine in the production line node has a relationship that describes how it feeds the distribution system node. The relationships between nodes can also be defined with properties on their own, since some relationships between two concepts can be different depending on the context where they are represented.

That said, the main modelling terms of the XMANAI Data Model have been defined as follows:

- **Concept:** A concept represents an object, entity or abstraction of some utility in the real world. In the case of a physical object this can be a very direct description of the physical attributes of the object and its purpose, in cases of abstract representations there is a need for consensus on how they are defined to be able to extract greater utility out of them.
- **Category:** The data model concepts are categorised into conceptual Categories that are interrelated so that there can be a general overview of the way it is structured, to make it more easily understood and readable.
- **Relationship:** A relationship is defined as a connection between concepts, it is, how they are related to one another. It can be defined by 5 types:
    - normalRelationship: concepts that are related by properties.
    - equivalent: one concept is equal to the other.
    - partOf: one concept is a subset of the other.
    - extensionOf: one concept is a superset of the other.
    - resultOf: one concept is a result of the other.

  Each Relationship contains further descriptors to better define how two concepts interact with each other.
- **Property:** A property is the basic data format in which concepts and relationships can be described. Usually, the data types of each property are defined apriori, being the most commonly used ones string, integer, date or more complex ones like lists of elements. Typically, a concept has multiple properties which will define and differentiate concepts.

3.5.2 Model design

In this section the first version of XMANAI Data Model is presented along with semantics and vocabularies supported. Note that the main focus on the first version XMANAI Data Model was to collect the semantics that covers the XMANAI demonstrators' data needs. The Data Model will be continuously improved and completed in the next versions, during the course of the project. Table 3-1 provides the list of concepts and the related details of the XMANAI Data Model.

Table 3-1: Concepts of XMANAI Data Model

Concept	Description	Conceptual category
3DScanner	A laser scanning device that scans an object (e.g. sphere) and extract data points, using specific measurement parameters e.g. lateral density, direction density, and exposure time	3D representation
MeasurementPosition	The specific position generated by the 3DScanner, for each one of the measurements, assigned to the 3-D space (x, y, z)	3D representation
DataPoint	The specific position generated by the 3DScanner, for each one of the measurements, assigned to the 3-D space (x, y, z)	3D representation
DataPointCloud	A set of data points in space, that represent a 3D space or an object	3D representation
SurfaceDirection	The specific direction of the object's surface with respect to the axial direction of the laser, for each one of the measurements, assigned to the 3-D space (I, j, k)	3D representation
PositionDeviation	The deviation between the measured and the actual position values. If deviation>0 the measured value exceeds the reference value.	Quality control & KPI

Concept	Description	Conceptual category
ObjectDiameter	Diameter of the object under investigation	Quality control & KPI
Group	Set of Assets in a production line	Machine Monitoring
Asset	Each operation of the product line	Machine Monitoring
Identifier	Relationship of the part produced by each asset and the engine derivative	Machine Monitoring
Event	Events that happened at an Asset. Used to extract cycle times	Machine Monitoring
Incident	Status of the assets	Machine Monitoring
Accumulator	Counter for each part produced by an asset	Machine Monitoring
Shift	Work/Monitoring scheduling	Machine Monitoring
Components	Information about the status of ech component	Production
ProductionPlant	Location/Description of the Production Plant	Production
ProductionPlanning	Planning of the production/Production expectations	Production
ProductionData	Historical production data	Production
Inventory	Number of parts available at a production plant	Production
Shipping	Shipping status	Production
Market	Marketplace where items will be/were sold at	Market (Product and Customer)
Sale	Transaction for a product sold from a market to a customer	Market (Product and Customer)
Customer	User history of transactions and customer information	Market (Product and Customer)
Session	Analytics data for products seen or purchased by the customer per log in session	Market (Product and Customer)
Product	Product code, available stock, quantity sold	Market (Product and Customer)
ProductInfo	Size, Price	Market (Product and Customer)
Channel	The type of channel used to purchase the product (from physical shop to online e-commerce platform)	Market (Product and Customer)
ClickstreamData	Clickstream data generated by Google Analytics tags embedded in the e-commerce platforms	Market (Product and Customer)
TrafficSource	Information about the Traffic Source from which the session originated, including “Click Info” [The physical table is nested inside the ClickstreamData]	Market (Product and Customer)
Device	The device used to navigate [Regarding devices used in ecommerce transactions, the physical table is nested inside the ClickstreamData]	Market (Product and Customer)

Concept	Description	Conceptual category
Hits	It contains details of any types of hits occurred, such as the data source, the time, the latency, the type… [The physical table is nested inside the ClickstreamData]	Market (Product and Customer)
Visitor	It contains the mapping between the visitor that generates the stream of data and the registered customer, in case of any.	Market (Product and Customer)
TransactionStatus	The status of a particular transaction	Market (Product and Customer)
Station	Quality Control grouping for machines in the same area/production line	Quality control & KPI
Traceability	Historical data of previous Quality Control verifications	Quality control & KPI
Status	Approval or dismissal of a part	Quality control & KPI
Quality	Quality quantification of a part	Quality control & KPI
Mapping	Flexible mappings for quality standards for different parts	Quality control & KPI
Requirements	Basic Quality Requirements for a part	Quality control & KPI
Sensor	Device, agent (including humans), or software (simulation) involved in, or implementing, a Procedure to get a Result.	Sensor data

Table 3-2 provides the list of Relationships between Concepts and its related details, that are considered in the first version of the XMANAI Data Model.

Table 3-2: Relationships of XMANAI Data Model

Concept	Related Concept	Relationship Type	Relationship Title	Relationship Description
Asset	Group	partOf	belongsToGroup	The associated Group of the Asset
SaleOrder	Market	normalRelationship	isPurchasedWhere	The market where the sales order is purchased
	Channel	normalRelationship	isPurchasedWith	The channel used to purchase the sales order
	Customer	normalRelationship	isPurchasedBy	The customer who purchases the sales order
	Product	extensionOf	includes	The Product included in the sales order
	Session	normalRelationship	isPurchasedDuring	The session during which the sales order is purchased
	Status	normalRelationship	statusIs	The status of the sales order
Channel	SalesOrder	normalRelationship	channelFor	The sale order that is performed using the channel

Concept	Related Concept	Relationship Type	Relationship Title	Relationship Description
	Customer	normalRelationship	usedBy	The customer who uses the channel
	Visitor	normalRelationship	isNavigatedBy	The visitor who performs the navigation
	Session	normalRelationship	isNavigatedDuring	The session during which the navigation is performed
	Device	normalRelationship	isNavigatedWith	The device used to perform the navigation
ClickstreamData	Visitor	extensionOf	generatedByVisitor	The Visitor that generates the stream of data
	TrafficSource	extensionOf	generatedByTraffic	The Traffic Source that generates the stream of data
	Device	extensionOf	generatedByDevice	The Device that generates the stream of data
	Hits	extensionOf	generatedByHits	The Hits that generate the stream of data
	Session	extensionOf	generatedDuring	The Session during which the stream of data is generated
Visitor	ClickStreamData	partOf	generates	The stream of data generated by the visitor
	Customer	equivalent	registeredAs	The registered customer who performs the navigation
TrafficSource	ClickStreamData	partOf	generates	The stream of data generated by the visitor
Device	ClickStreamData	partOf	generates	The stream of data generated by the Device
	Session	normalRelationship	isUsedDuring	The session during which the device is used
Hits	ClickStreamData	partOf	generates	The stream of data generated by the Hits
Market	SalesOrder	normalRelationship	placeOfSalesOrder	The sale order that is purchased in the Market
	Customer	normalRelationship	placeOfCustomer	The customer who used the Market for purchase orders/navigate
	Session	normalRelationship	placeOfSession	The session realised in the market

Concept	Related Concept	Relationship Type	Relationship Title	Relationship Description
Customer	SalesOrder	normalRelationship	purchases	The sale order purchased by the customer
	Visitor	equivalent	visitAs	The visitor associated to the customer
	Market	normalRelationship	PurchasesIn	The market where the Customer purchases sales orders and navigates
	Channel	normalRelationship	uses	The channel used by the customer
	Session	normalRelationship	starts	The session started by the customer
Product	SalesOrder	partOf	belongsToSalesOrder	The associated Sale Order of the Product
	ProductInfo	equivalent	isSpecifiedBy	The detailed information associated to the product
ProductInfo	Product	equivalent	specifies	The product which the detailed information refers to
Session	SalesOrder	normalRelationship	isSessionOfTheOrder	The sale order purchased during the session
	ClickStreamData	partOf	generates	The stream of data generated during the session
	Device	normalRelationship	isSessionOfTheDevice	The device used during the session
	Customer	normalRelationship	isSessionOfTheCustomer	The customer who starts the session
	Market	normalRelationship	isSessionOfTheMarket	The market where the session happens
Incident	Asset	normalRelationship	isIncidentOfTheAsset	An incident is referred to a specific asset
Accumulator	Asset	normalRelationship	isAccumulatorOfTheAsset	An accumulator is referred to a specific asset
Event	Asset	normalRelationship	isEventOfTheAsset	An eventis referred to a specific asset
Identifier	Asset	normalRelationship	isIdentifierOfTheAsset	An identifier is referred to a specific asset

Concept	Related Concept	Relationship Type	Relationship Title	Relationship Description
3DScanner	DataPoint	resultOf	actualResult	Measurement position (data point) is the actual result/outcome of the concept 3DScanner for a specific measurement.
	DataPointCloud	resultOf	actualResult	Data point cloud is the final set of data points, which extracted from the concept 3DScanner, and represent the object under investigation.
	SurfaceDirection	resultOf	actualResult	Surface direction defines the orientation of the surface during a specific measurement of the 3DScanner concept
	ObjectDiameter	resultOf	actualResult	Object diameter is the 3DScanner concept actual result/outcome of the radius measurement.
	PositionDeviation	resultOf	actualResult	The main KPI of the measurement performance of the 3DScanner
DataPoint	SurfaceDirection	extensionOf	extensionOfSurfaceDirection	The Measurement position (data point) is an extension of the Surface direction
	DataPointCloud	partOf	partOfDataPointCloud	The Measurement position (data point) is a part of the data point cloud.
	3DScanner	partOf	partOf3DScanner	3DScanner is the generator (parent) of the data points.
	PositionDeviation	partOf	partOfPositionDeviation	Position deviation is defined by the difference of the measurement position and the associated nominal value.
	ObjectDiameter	partOf	partOfObjectDiameter	The Measurement position (data point) is a part of Object diameter; Diameter is a line segment connecting two data points, going through the center of a circle.

Concept	Related Concept	Relationship Type	Relationship Title	Relationship Description
DataPointCloud	SurfaceDirection	extensionOf	extensionOfSurfaceDirection	Data point cloud is an extension of the surface direction
	DataPoint	extensionOf	extensionOfMeasurementPosition	Data point cloud consists of a set of Measured positions (data points)
	3DScanner	resultOf	resultOf3DScanner	Data point cloud is the actual result/outcome of the 3DScanner concept measurements.
	PositionDeviation	normalRelationship	relevantToPositionDeviation	Data point cloud is relevant to the position deviation concept; Position deviation is the difference between the measured data points of the cloud and the nominal values.
	ObjectDiameter	extensionOf	extensionOfObjectDiameter	The Data point cloud concept is an extension of the Object diameter concept; To measure the object diameter the data points from the cloud are needed.
SurfaceDirection	DataPoint	partOf	partOfMeasurementPosition	Surface direction is a part of Measurement position (data point); defines the surface orientation from where the data point is extracted.
	3DScanner	partOf	partOf3DScanner	Surface direction is a part of 3DScanner; defines the surface orientation where the laser from the 3DScanner concept is scanning.
	DataPointCloud	partOf	partOfDataPointCloud	Surface direction is a part of the Data point cloud; defines the surface orientation for each one of the data points that are included in the cloud.
	PositionDeviation	partOf	partOfPositionDeviation	Surface direction is part of Position deviation; defines the surface orientation for each one

Concept	Related Concept	Relationship Type	Relationship Title	Relationship Description
				of the data points used for the position deviation extraction.
	ObjectDiameter	normalRelationship	relevantToObjectDiameter	Surface diameter is relevant with the Object diameter concept; Surface direction is needed for the data points measurement, from which object diameter is calculated.
PositionDeviation	DataPoint	normalRelationship	relativetoMeasurementPosition	Position deviation is relative to measurement position (data point); It is the difference between the measured data point and the actual value of the data point
	SurfaceDirection	normalRelationship	relativetoSurfaceDirection	Position deviation is relative to Surface direction concept
	3DScanner	normalRelationship	Relativeto3DScanner	Position deviation is relative to 3DScanner concept; It is extracted from the laser measurements of the 3DScanner and the nominal data point values.
	DataPointCloud	normalRelationship	relativetoDataPointCloud	Position deviation is relative to the Data point cloud concept; It is the difference between the measured data points of the cloud and nominal values.
	ObjectDiameter	normalRelationship	relativeToObjectDiameter	Both concepts need the data point measurement
ObjectDiameter	DataPoint	normalRelationship	relativeToMeasurementPosition(data points)	Object diameter can be calculated using a group of data points
	SurfaceDirection	normalRelationship	relativeToSurfaceDirection	Object diameter is calculated using a group of data points, that measured through the use of surface direction
	DataPointCloud	normalRelationship	relativeToDataPointCloud	Object diameter is calculated using the data point cloud

Concept	Related Concept	Relationship Type	Relationship Title	Relationship Description
	3DScanner	resultOf	resultOf3DScanner	Object diameter is an outcome/output of the 3DScanner
	PositionDeviation	normalRelationship	relativeToPositionDeviation	Object diameter concept is relevant with Position deviation concept; Both need the data point measurement in order to be calculated
DataPoint	MeasuremenPosition	equivalentTo	equivalentTo	DataPoint is an equal semantic for MeasuremenPosition
MeasuremenPosition	DataPoint	equivalentTo	equivalentTo	MeasuremenPosition is an equal semantic for DataPoint
Components	Inventory	extensionTo	extensionToInventory	Components are an extension of inventory due to the fact that this concept add information about status of the diffferent parts included in inventory.
	ProductionPlant	normalRelationship	isStoredInAProductionPlant	A component belongs to a specific production plant
	ProductionPlanning	normalRelationship	isPlannedToBeProduced	The component planned to be produced
Inventory	Components	partOf	partOfComponents	Almost all Inventory properties are included in Components
	ProductionPlant	normalRelationship	isStoredInAProductionPlant	Different parts are stored in a production Plant
ProductionData	ProductionPlanning	resultOf	resultOfProductionPlanning	The produced data are a result of the production planning
	ProductionPlant	normalRelationship	isReferredToAProductionPlant	The produced components are referred to a production plant
ProductionPlanning	ProductionPlant	normalRelationship	isReferredToAProductionPlant	The components to be produced are referred to a production plant
	Components	normalRelationship	plannedToProduce	The components are planned to be produced by a production plan
Shipping	ProductionPlant	normalRelationship	ShipsToProductionPlant	The shipping destiny is a specific plant

Concept	Related Concept	Relationship Type	Relationship Title	Relationship Description
ProductionPlant	Components	normalRelationship	isProducedAt	The production plant where a component is produced
	Inventory	normalRelationship	isStoredAt	The production plant where a part is stored
	ProductionData	normalRelationship	isRegisteredAt	The production plant where produced component are registered
	ProductionPlanning	normalRelationship	isPlannedFor	The production plant where components are scheduled to be produced
	Shipping	normalRelationship	isShippedTo	The production plant is the destiny of a shipping
Quality	Mapping	normalRelationship	SpecifiesQualityRepor	The mapping layout defines quality parameters of an operation
Traceability	Quality	normalRelationship	reportsQualityOfAUnit	The quality report refers to a specific unit (ithan assigned serial number) and operation

Table 3-3 provides the list of Properties and the related details that are considered in the first version of the XMANAI Data Model.

Table 3-3: Properties of XMANAI Data Model

Concept	Property Name	Property Description	Date Type
Market	MarketID	Unique Market identifier	Integer
	marketName	Name of the Market identified by MarketID	String
	marketAddress	Physical address of the Market being described	Addr/String
	location	Additional information about the Market location	String
	marketName	Name of the Market identified by MarketID	String
SaleOrder	customerID	Identifier of a Customer who requested a sales transaction	Integer
	orderID	Unique Identifier of a specific sales transaction	Integer
	sessionID	Identifier of the session the sales transaction was created	Integer
	Price	Cost of the product or products paid by the Customer to the Market	Double
	MarketID	Unique Market Identifier	Integer
	ChannelID	Unique Channel Identifier	Integer

Concept	Property Name	Property Description	Date Type
	Quantity	Number of pieces of ProductID purchased in the OrderID	Integer
	RequestedDeliveryDate	Delivery date requested by the customer	DateTime
	estimatedDeliveryDate	Date provided by the Market of the expected date the Product will arrive at the Customer	Date
	SaleStatus	Tracking of the state of the sales transaction, from pending payment transaction to shipping state, if it has been shipped, in transit or delivered	String
	orderDate	Date the ordered was placed on the Market by the Customer	Date
	ProductID	Identifier of the product or products that was ordered through the sale transaction	Integer
Customer	customerID	Unique identifier for the Customer Account or identification	Integer
	MarketID	Unique Market Identifier	Integer
	ChannelID	Unique Channel Identifier	Integer
	orderHistory	History of previous orders made by a Customer	List of Sales
	shippingAddress	Addresses Customer has requested deliveries to	List of Addresses
	email	Email contact the Customer has provided	String
	LogInInfo	Password defined by the Customer for their account	String
	name	Customer's name	String
Session	sessionID	Unique identifier for a session initiated by a Customer to a Market	Integer
	customerID	Identifier of the Customer who initiated the Session	Integer
	visitStartDate	Date the Session began	Date
	Device	Information about the device the Customer initiated the session in	String
	MarketID	Identifier of the Market the Customer is accessing to in the Session	Integer
	orderID	Identifier of the order or orders made by the Customer during the Session	Integer/List of integer
Product	ProductID	Unique identifier for a Product sold at a Market	Integer
	ProductInfoID	Identifier for extra information about a product, the extra information can be used for	Integer
	productDescription	Description of the product being sold	String
	SaleOrderId	Identifier for sales orders where this product was sold	Integer
ProductInfo	ProductInfoID	Unique identifier for extra product information	Integer

Concept	Property Name	Property Description	Date Type
	ProductID	Identifier that relates the information about the product to the product itself	Integer
	Size	Dimensions of the product or volume it occupies	Integer
	Price	Price the Product is being sold for	Integer
	Stock	Amount of product directly available for purchase	Integer
	EanCode	Product Bar Code	Integer
	Source	The supplier/the factory who provides the product in case of third sale	String
	GPH	Global Product Hierarch	
Traceability	OperationID	Id of the operation where the quality parameter is reported	Integer
	StationID	Id of the specific station within an operation. Used when a parallel process is running	Integer
	SerialNumber	Serial number of the component	String
	EngineSerialNumber	Serial number of the engine that the component belongs to	String
	ProcessTimeStamp	Time when a quality parameter is inserted	DateTime
Status	StatusID	Unique identifier for the status of a quality control check	Integer
	OperationID	Identifier for the operation a status report was made from	Integer
	StatusCode	Code the status verification returns for storage	Integer
Quality	QualityParameterID	Quality parameter to be measured	Integer
	QualityValue	Value of the specific quality parameter	Integer
	ProcessTimeStamp	Time when a quality parameter is inserted	DateTime
	SerialNumber	Serial number of the component	String
	OperationID	Id of the operation where the quality parameter is reported	Integer
Mapping	OperationID	Id of the operation where the quality parameter is reported	Integer
	QualityParameterID	Quality parameter to be measured	Integer
	DataDescription	Description of the quality parameter in a human understandable way	String
	LayoutID	Id of the layout used to link quality parameters to an operation	Integer

Concept	Property Name	Property Description	Date Type
Sensor	Deployment	Identifier of the company the plant/system/sensor belongs to.	String
	Deployment	Identifier of the production plant the system/sensor belongs to.	String
	System	Name and ID of the machinery	String
	Feature of Interest	Name of the system component/feature under observation	String
	Observable Property	Feature of Interest observable quality	String
	Result	Result of the sensor observation	Bouble
	date Time	Measurement time	Date Time
Group	GroupID	Id of the group	Integer
	GroupName	Name/Description of the group	String
	ParentGroupID	Id of the group parent	Integer
Asset	NodeID	Id of the Asset	Integer
	AssetName	Name of the Asset	String
	Alias	Description of the Asset	String
	ParentGroupID	ID of the Group that the Asset belongs to	Integer
	AreaPayPoint	Flag to indicate the Asset considered the output of the line	Integer
	LinePayPoint	Flag to indicate the asset considered the output of each group	Integer
	AssetType	Type of asset	String
	BottleneckPoint	Flag to indicate whether the asset is taken into account for analysis.	Integer
	DesignCycleTime	Design Cycle Time	Float
	JobsPerHour	Jobs Per Hour. Number of parts produced in 1 hour according to the DesignCycleTime	Float
	ParallelProcess	Name of the asset that runs in parallel in the line	String
	PartsPerCycle	Number of parts produced in a cycle	Integer
	RunAtRateJPH	Target JPH assigned to an asset (under JPH)	Float
	StdCycleTime	Cycle Time according to RunAtRateJPH	Float
Identifier	IdentifierID	Id of the identifier	Integer
	Category	Category of the identifier	String
	IdentifierName	Group inside a category	String
	NodeID	Id of the asset	Integer
	StartTime	Time at which a part is started to be produced	Timestamp
	EndTime	Time when a part is completed	Timestamp

Concept	Property Name	Property Description	Date Type
	Value	Code of the derivative that the produced part belongs to	Integer
	Property	Name of the derivative associated to the Value code	String
Events	EventVariableID	Id of the event and variable	Integer
	EventName	Group of the event	String
	VariableName	Type of event inside an event group	String
	NodeID	Id of the asset	Integer
	RecordTime	Time at which the event is registered	Timestamp
	Value	Cycle Tyme	Float
Incidents	IncidentID	Id of the incident	Integer
	Category	Category of the incident	String
	Description	Specific incident inside a category	String
	NodeID	Id of the asset	Integer
	StartTime	Start of the incident	Timestamp
	EndTime	End of the incident	Timestamp
	Duration	Difference between EndTime and StartTime	Float
Accumulator	AccumulatorID	Id of the accumulator	Integer
	Category	Category of the accumulator	String
	AccumulatorName	Specific accumulator inside a category	String
	NodeID	Id of the asset	Integer
	InsertTime	Time at which the accumulator is registered	Timestamp
	Value	Number of cycles counted by the accumulator between 2 InsertTimes	Integer
Shifts	ShiftID	Shift Identifier	Integer
	ShiftStartTime	Shift start date	DateTime
	ShiftEndTime	Shift end date	DateTime
	PeriodName	Period inside a shift identifier	String
	Duration	Period duration	integer
	IsProductive	Flag to mark a periodas a productive	Integer
3DScanner	Lateral Density	The density of the laser on the different sides of the surface	Float
	Direction density	The direction the density has during the measurement	Float
	Exposure time	The time the laser needs to make the measurement	Float
Measurement Position	X	The position of the data point in the X axis	Float
	Y	The position of the data point in Y axis	Float
	Z	The position of the data point in Z axis	Float

Concept	Property Name	Property Description	Date Type
DataPointCloud	DataPointsGroup	A group of data points	List of DataPoints
Surface direction	i	The i direction component of the measurement point	Float
	j	The j direction component of the measurement point	Float
	k	The k direction component of the measurement point	Float
ObjectDiameter	Diameter1	the diameter of the sphere measured using the 25 data points	Float
	Diameter2	The diameter of the sphere measured using the whole data point	Float
ClickstreamData	Property	The website	String
	VisitorID	Unique Visitor identifier	Integer
	SessionID	Unique Session Identifier	Integer
	VisitStartTime	Timestamp of the visit starting	Integer
	Date	Date of the visit starting	Integer
	TrafficSourceID	Unique Traffic Source Identifier	Integer
	DeviceID	Unique Device Identifier	Integer
	DimensionsID	Unique Dimensions Identifier	Integer
	HitsID	Unique Hits Identifier	Integer
TrafficSource	adwordsClickInfo.adGroupId	The Google ad-group ID	Integer
	adwordsClickInfo.campaignId	The Google Ads campaign ID.	String
Device	DeviceID	Unique Device identifier	Integer
	DeviceName	Name of the Device	String
	DeviceType	Type of the device (mobile phone, laptop, tablet)	String
	DeviceModel	Device Model	String
Hits	dataSource	The data source of a hit.	String
	hitNumber	The sequenced hit number. For the first hit of each session, this is set to 1	Integer
Component	PlantID	Plant identifier	String
	PartID	Part Identifier	String
	Status	Status of a specific component	String
	PartType	Type of the component	String
	PartDescription	Description of the component	String
Inventory	PlantID	Plant identifier	String
	WorkCenterID	Identifier of the area inside a plant where a part is	String
	PartID	Part Identifier	String

Concept	Property Name	Property Description	Date Type
ProductionData	PlantID	PlantIdentifier	String
	WorkCenterID	Identirfier of the area inside a plant where a part is	String
	RegistryDate	Date when an amount of produced parts is registered	DateTime
	PartID	Part Identifier	String
	ScheduledQuantity	Amount of parts expected to be produced	Integer
	ProducedQuantity	Amount of parts really produced	Integer
	ScrapQuantity	Amount of scrapped parts	Integer
ProductionPlanning	PlantID	Plant Identifier	String
	WorkCenterID	Identifier of the area inside a plant where a part is	String
	PartID	Part Identifier	String
	ScheduledDate	Date when an amount of parts are expected to be produced	DateTime
	ScheduledQuantity	Amount of parts expected to be produced	Integer
	PlanCreator	Flag to identify the creator of a production plan	String
Shipping	PlantID	Plant Identifier	String
	ConveranceNumber	Number of a conveyor	Integer
	DupId	Route number inside a cycle	Integer
	ContainersQuantity	Quantity of containers	Integer
	ConveryorSealId	Conveyor Container number	Integer
	ShippedDate	Date when a shipping starts	DateTime
	ExpectedArrival	Expected arrival date	DateTime
	ConveyanceStatus	Status of the conveyor	String
	TransportMode	Transport Mode (air, train, etc.)	String
	CarrierID	Carrier Identifier	String
	CarrierName	Name of the carrier	String
ProductionPlant	PlantID	Plant Identifier	Integer
	WorkCenterID	Identifier of the area inside a plant where a part is	Integer

3.5.3 Model lifecycle

In this section insights into the different states of the XMANAI Data Model lifecycle are provided. Besides the storage and usage, an evolution state has been created for the Data Model in which it is possible to execute transformation actions that will allow the model to be updated, completed, and improved to match the future needs of XMANAI users. The stages can be described as follows:

Data Model Storage: XMANAI Data Model is being stored as a Labelled Property Graph using Neo4j (https://neo4j.com/) in the XMANAI Storage.

Data Model Usage: All input data in the XMANAI need to be mapped to the Data Model. The XMANAI platform users can search over the different concepts and properties to find appropriate concepts and Properties for mapping.

Data Model Extension: XMANAI Data Model allows users to propose the creation of new entities (Concept, Property and Relationship) in the case of not finding the ones appropriated for their needs. The possible model extension proposal scenarios are as follows:

- For each input, one Concept must be selected by the user from the existing ones. If the user needs a specific concept that does not exist, he must request for officially adding the concept into the XMANAI Data Model.
- The user must map the fields of the input data to the properties of the selected Concept in the previous step. If the proper Property does not exist in the Data Model, the user can define the semantics manually and also send the proposal to the Data Model Admin in order to officially add it into the XMANAI Data Model.
- Data Model Relationships allows the user to access the Properties of the related Concepts, i.e., in the case of not finding suitable Properties in the selected Concept, he can use its relationships to reach the properties of related Concepts.

Data Model Maintenance: The XMANAI Data Model allows a set of transformation actions to be executed in order to expand the Model for the future needs. Note that the changes should not affect the pre-existing data assets. Any change request from the extension proposals can be accepted or rejected by the Data Model Administrator. He will have permissions to execute the following set of operations:

1. Add new Concept's category
1. Add new Concept
2. Add new Properties to an existing concept
3. Add new Relationships between two existing concepts
4. Update the title of an existing Concept
5. Update the description of an existing Concept
6. Update the title of an existing Property
7. Update the description of an existing Property
8. Update the title of an existing Relationship between 2 existing Concepts
9. Update the description of an existing Relationship between 2 existing Concepts
10. Delete an existing Concept (and all its connections, including Properties and Relationships with other Concepts)
11. Delete existing Properties
12. Delete existing Relationship between two Concepts

4 AI Bundles in XMANAI

4.1 View as a whole

Building upon the insights gained through the theoretical and technological landscape analysis presented in section 2 and the XMANAI data model development process and early outcomes presented in section 3, the current section will present the way XMANAI will implement the processes, mechanisms and tools through which the WP3 AI Bundles will be delivered.

In particular, as documented in D5.1, WP3 directly contributes to the provision of the following XMANAI services:

- **Data Manipulation services** where both involved sub-components, namely the Data Manipulation Engine and the Knowledge Graph Manager are implemented in the context of WP3 activities.
- **AI Model Lifecycle Management Services**, which involve three sub-components, two of them provided by WP3 (XAI Pipeline Manager and Model Engineering Engine) and the third (XAI Models Catalogue by WP4) has very strong connections with the first two.
- **AI Execution Services**, handled through the Execution & Orchestration Engine provided by WP3.
- **AI Insights Services,** which involve three sub-components whose functionalities are implemented within WP3, namely the XAI Pipeline Manager and the Model Engineering Engine mentioned above and the XAI Visualisation Engine.

It should be noted that the aforementioned sub-components correspond to the WP5 high-level architecture and their scope, positioning and interactions are documented in D5.1.

In the context of WP3, the relevant WP5 components are further elaborated, their expected functionalities are studied in more detail and a detailed WP3 architecture is drafted that encapsulates the insights from the thorough landscape review (presented in section 2), as well as more practical implementation-related aspects. This refined view, depicted in Figure 4-1, brings the WP5 architecture abstraction into a more concrete design of the components and the methods that will guide WP3 implementation activities.

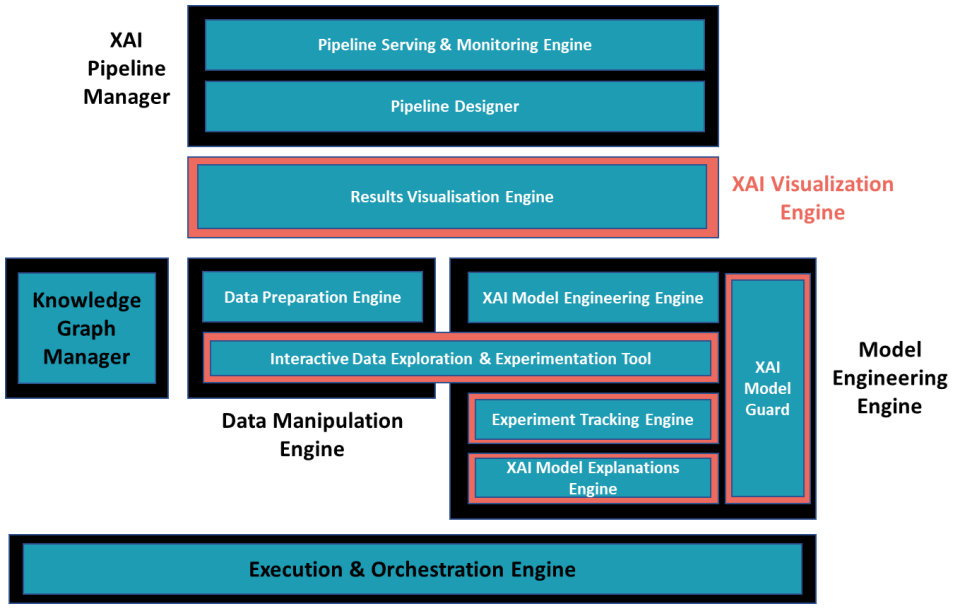

Figure 4-1: WP3 components & mapping to WP5 high-level architecture

As shown in Figure 4-1, two of the WP5 components, namely the **Knowledge Graph Manager** and the **Execution & Orchestration Engine**, have a one-to-one mapping with the corresponding WP3 sub-components and therefore their names remain unchanged. However, in the context of the more detailed WP3 architecture, the concrete functionalities that these sub-components will provide are refined and presented also in the current deliverable.

The remaining 4 high-level WP5 components, in a way in accordance with their positioning in and their interaction points within the AI pipeline implementation process, comprise more than one WP3 components, sometimes through many-to-many relationships:

The WP5 Data Manipulation Engine includes the **Data Preparation Engine**, whose role is fully revealed through its name (i.e. to prepare the data for running an analysis), and partially includes the **Interactive Data Exploration and Experimentation Tool**, which is shared between this high-level component and the WP5 Model Engineering Engine.

Although conceptually having a more targeted (i.e. WP3-level) sub-component being spread across two high-level components may seem counterintuitive, the reasons that led to this approach are directed by the state-of-the art insights. As explained also in the first consideration discussed in section 2.4 *("AI pipelines are not a collection/sequence of individual steps"*), experimentation is inherent in data science, both when it comes to data and to models. As a reminder from the state of the art insights, even in the context of MLOps and automation, notebooks are still considered valuable and established tools for data scientists and therefore an exception to the modularity paradigm in this case is part of the foreseen good practices. XMANAI aspires to have a strong research positioning in the XAI landscape and in particular in the manufacturing domain, therefore a sub-component providing (almost) full flexibility and freedom to data scientists to experiment with new technologies is foreseen to assist in this direction. Of course, this flexibility in experimentation brings limitations in terms of production deployment of the code generated through this process, as it will be explained in the corresponding section. As it can be easily understood, enabling experimentation only in terms of data preparation or model engineering, or attempting to decouple these processes in two experimentation-oriented sub-components would not be meaningful, hence the depicted sharing among the distinct WP5 components.

The WP5 Model Engineering Engine lies at the core of the the XMANAI AI pipelines and comprises in total five WP3 sub-components: (a) the partially included **Interactive Data Exploration and Manipulation Tool** described above, (b) the **XAI Model Engineering Engine** which provides the ML/DL model selection, configuration and training functionalities, (c) the **XAI Model Explanations Engine** which powers the explainability methods' configuration, application and results (explanations) generation, (d) the **Experiment Tracking Engine** that enables logging of experiments' metadata (performance metrics, corresponding hyperparameters and other artifacts) and performing comparisons and (e) the **XAI Model Guard** that brings safeguarding mechanisms for the security and integrity of the trained AI models.

The functionalities of the WP5 XAI Pipeline Manager are split in a more straightforward manner in two groups that correspond to an equal number of WP3 sub-components, i.e. (a) the **Pipeline Designer** that is in charge of the creation and management of the AI pipelines and of enabling and coordinating the collaboration of different stakeholders (business users, data scientists and data engineers) and team members across the various steps of the pipelines and (b) the **Pipeline Serving and Monitoring Engine** which is responsible for the deployment (with appropriate settings) of the pipelines that are suitable for usage in production and for monitoring their performance to provide insights regarding their operation.

Finally, the WP5 XAI Visualisation Engine, which in the context of the WP5 high-level architecture encapsulates the whole spectrum of visualisation functionalities, is actually mostly a perceived as a feature/ functionality of certain already discussed WP3 components, as denoted by the bright outline used in Figure 4-1. It is evident from numerous section 2 sub-sections and was also stressed in the

fifth consideration in section 2.4 *("Visualisation is not a step, it is an integral part of almost all steps of an AI pipeline"*) that in the AI pipelines development lifecycle (and even more in the XAI pipelines that are in scope here), visualisation, visual exploration and visual communication of results, explanations and overall findings is a constant need throughout the process. All stakeholders involved in an XMANAI AI pipeline will require visual support in the processes that are offered by the corresponding tools and willrequire different collaboration and interaction mechanisms, as they will each time focus on different aspects, i.e. on the data, the models, the experiments and the explanations.. In fact, in most cases the functionalities that are offered are so tightly connected to the relevant visualisation capabilities, that in the context of WP3 sub-components, visualisation becomes a feature of the particular sub-component. In more detail, visualisation in this context is enabled for the following already mentioned sub-components: (a) the **Interactive Exploration and Experimentation Tool** should naturally also allow its users to configure and use their selected visualisation libraries, which are expected in this context to be oriented towards scientific visualisation (e.g. statistical data visualisations), (b) the **XAI Model Explanations Engine** will utilise appropriate visual means to provide explanations to the requesting stakeholders - the nature of the visualisation here will probably differ according to the technical background of the requestor and the stage at which the request is performed (e.g. to comprehend a model's decision in production or to debug the model when still in the training phase), (c) the **Data Model Guard** may also utilise visualisations, depending on the nature of the model that is being examined and the format of the data used during its training, and (d) the **Experiment Tracking Engine**, which will offer targeted experiment comparison charts and visualisations to monitor model training performance, e.g. learning curves. Finally, there is an additional WP3 sub-component, namely the **Results Visualisation Engine**, whose main purpose is to allow the end results generated by the AI pipelines to be appropriately visualised and communicated to the business users and is therefore shown as the only WP3 sub-component that falls exclusively under the WP5 XAI Visualisation Engine scope.

It should be noted that hereinafter the term components will refer to the WP3 sub-components, unless otherwise specified.

Sections 4.2-4.12 present one by one the components and provide more details regarding their features, expected usage, positioning within the WP3 architecture and foreseen interactions and dependencies. Furthermore, indicative mockups of the core functionalities are provided when a user interface is foreseen.

Before proceeding to the individual components' descriptions, there are some WP3-level decisions regarding the methods that will be leveraged in XMANAI for the provision of its AI bundles that should be highlighted. These are briefly presented below and may also facilitate understanding of certain aspects in the components' design:

- **Explainability mechanisms**, both at the data and the model level, will be tightly integrated in various steps of the XMANAI AI pipelines and their results will be made available in different "flavours" to the stakeholders that need them. Data scientists will have different needs during model training and evaluation compared with business users trying to understand why a model made a certain prediction and what it would take for it to make a different one. Even when the same underlying need is shared (e.g. both technical and business users may need to comprehend what the important features that guide a model's decisions are and to what direction they steer them), but their different backgrounds should be considered when providing the respective explanation so that it can convey the right message to the users and they have the ability to interpret it correctly. As discussed in the state of the art section (and stressed in section 2.4 in particular), MLOps proposed practices do not yet reflect on how explainability can be properly integrated. XMANAI AI pipelines will adopt and adapt an MLOps setup that will support various explainability techniques across its steps, leveraging also the insights gained from the XAI landscape analysis provided in D1.1.

- XMANAI will provide a **collaborative environment** for data scientists, data engineers and business users to be able to work together to create, configure, train, evaluate and constantly monitor and refine AI pipelines in the manufacturing domain. Collaboration will be enabled across the complete flow of the AI lifecycle, thus contributing to reduce time in understanding data and identifying outliers, accelerate model validation and debugging (also with the help of AI explainability methods), timely detect data and model drift or other issues in production. XMANAI will work to deliver a solid collaborative experience around explainable AI models in manufacturing, not through a one-size-fit-all approach, but taking into consideration the background and needs of involved stakeholders in each step and also the processes to which they can/should contribute and the way this should be performed. It is also in this sense that in XMANAI AI bundles, the main discussed entity is the AI pipeline as a whole and not only a trained AI model.
- The benefits that can be gained from ensuring that data used as inputs to the AI pipelines conform to a **common data model** have been discussed in section 3, which also presented in detail the XMANAI approach. The XMANAI AI pipelines will leverage this targeted to the manufacturing domain data model across all the steps that this is possible, starting from fusing data explainability processes across the AI lifecycle. To provide some additional examples, the defined types and measurement units will help timely detect issues in the input data before these are propagated in training processes, the available semantics will be used to enable/disable feature creation and other data manipulation functionalities accordingly, the overall structure and relationships among data will make feature importance insights easier to comprehend. Integrating the data model in the AI lifecycle will also contribute towards collaboration, as all stakeholders will have a common view and understanding on the data and models that drive the AI pipelines.

4.2 Data Preparation Engine

4.2.1 Overview

As described in the XMANAI Deliverable D5.1, the Data Preparation Engine is the component responsible for providing the data processing functionalities which ensure that datasets retrieved by the Knowledge Graph Manager (described in section 4.3) can be appropriately filtered and transformed to be used as input in visualisations or in machine learning processes (e.g. as training data, as testing data or as actual data coming from the production environment to be used in decision making).

The component receives input in the form of a dataframe, i.e. two-dimensional tabular data, without precluding potentially nested structures (e.g. python dictionaries) in certain columns, depending on what is foreseen by the underlying data model and the mapping process that the original dataset has undergone (based on the processes explained in D2.1).

The Data Preparation Engine will allow the users to select and configure the desired data processing functionalities. Although an exhaustive list of functionalities cannot be provided, an initial set of foreseen actions that should be supported by the component is as follows:

- Aggregations and statistics computations
- Filters (static and dynamic based on other computations)
- Null values' imputation
- Mathematical operations
- Datetime features extraction (e.g. day of week, hour, etc.)
- Conditional creation of new columns
- Addition/deletion/duplication of columns
- Sorting
- Value replacement (upon condition)

- Merging of dataframes

The list of available functionalities will be continuously refined and extended to support the needs of the XMANAI users. In particular, apart from commonly used generic data preparation processes, more targeted tasks addressing manufacturing-related data processing are expected to be identified and supported as the project activities progress. The Data Preparation Engine is expected to be used by all XMANAI user roles (i.e. business users, data scientists and data engineers), and therefore the range of the provided functionalities as well as the flexibility and sophistication of the required configuration for each task will vary to ensure that the users' diverse needs are covered.

In a nutshell, the functionalities provided by the component are as follows:

- **DPE.1: Provision of configurable data preparation templates** for numerous data pre-processing and filtering functionalities. The templates will provide descriptions of the configuration parameters, as well as sensible default options when possible, to guide the user. The underlying data model will be leveraged to enable/disable options of the templates based on the data types and semantics and any other information available by the model.
- **DPE.2: Combination of data preparation templates** in a sequence of steps that need to be (and will be upon execution) applied to the input dataframe(s)
- **DPE.3: Transformation of the configured by the user data preparation templates into Python code** that can be executed by the Execution & Orchestration Engine described in Section 4.12. It should be noted that the execution code generated by this component is appropriate to be used in the production, i.e. can be part of the pipelines served in production – contrary to the execution code generated by the Interactive Data Exploration and Experimentation Tool, which is not optimized for production usage.
- **DPE.4: Summary statistics** (e.g. averages, minimum and maximum value, percentiles, number of unique values, percentage of missing data etc.) **and data subset preview** for selected columns of the dataframe that is returned as output of a specific data preparation action. Note that the output may in fact correspond to a sequence of executed data preparation tasks, but will always be linked to a specific (i.e. the last one applied) step of the data preparation process. The underlying XMANAI data model will be leveraged to increase comprehensibility of the presented results and allow users to interpret the computation outcomes, whenever possible.

It should be noted that even though the configured templates are translated into Python code through this component, the way it will be executed (e.g. resources to be used) and also certain decisions that may be linked to the underlying infrastructures or some higher-level user preferences depend on settings and configurations from other components, e.g. the Pipeline Designer and the Execution & Orchestration components, as it will be explained in the respective sections.

In this regard, the output of the component can be considered two-fold:

- The execution code that corresponds to the sequence of configured data preparation templates. When the respective data preparation processes are sent for execution, this code will be sent to the Execution & Orchestration component.
- A dataframe that holds the data generated by the execution of the configured data preparation templates. This dataframe can be used as input in further processing, either by the same component or by other WP3 components, e.g. the XAI Model Engineering Engine or the Results Visualization Engine.

4.2.2 Technology

Although the technology stack that will be utilized to provide the aforementioned functionalities is not yet finalized, based on the state of the art analysis presented in Section 2.3.1.2 and the

requirements that this component aspires to address, the following libraries and frameworks are currently under consideration for the development of this component:

- Pandas (https://pandas.pydata.org/) is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, which will be leveraged for the core data preparation functionalities of the component
- Pyspark (http://spark.apache.org/docs/latest/api/python/) is an interface for Apache Spark in Python and will be leveraged to support the big data processing needs of the component.
- Vue.js (https://vuejs.org/) will be considered for the development user interface
- S5 Analyze Platform, a proprietary analytics solution provided by the XMANAI technical partner Suite5, that offers different data preparation functionalities.

4.2.3 Mockups

This section provides mockups of the main functionalities that will be offered through the DPE user interface.

The Data Preparation Engine provides one main user interface, depicted in **Error! Reference source not found.**, through which the user can select and configure a series of data preparation steps and also review indicative results (e.g. sample data and computed statistics).

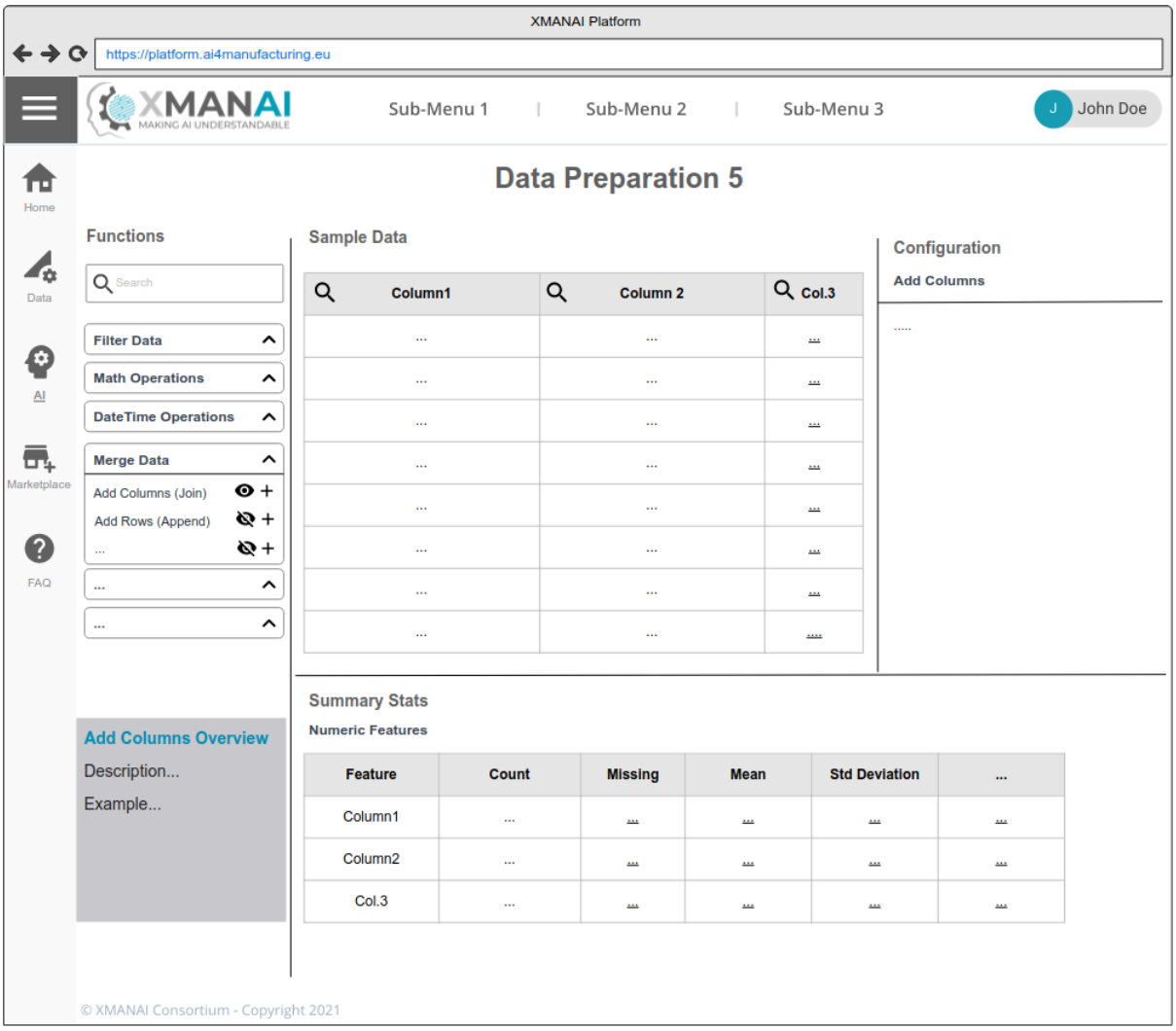

Figure 4-2: DPE – Steps configuration and results review

4.3 Knowledge Graph Manager

4.3.1 Overview

The Knowledge Graph Manager (KGM) helps to provide semantics and vocabularies with the purpose of increasing the explainability of the input data in the way of supporting a common Data Model for the XMANAI platform. This component visualizes the XMANAI Data Model, both in a tabular and graph view, so as to facilitate the mapping of the harvested data to the data model as well as the modification of the data structure itself if necessary. Process of mapping input data to the Data Model is described in more detail in the XMANAI Deliverable D2.1.

Since the ingested data into the XMANAI Platform (through the WP2 data collection activities) should have an appropriate association and definition of what the different concepts, properties and relationships between all of them are, this can be achieved by the user utilizing the functionalities provided by the Knowledge Graph Manager. It can be done through the Data Model defined entities or by the Data Model Maintenance feature that is intended to expand the Data Model if none of the available ones suit their needs.

The component should be able to provide the users with a way to request for adding new concepts, properties and relationships to the Knowledge Graph. In order to facilitate the use of the Data Model for the end-users, this component is considering the following actions which allow them to request for the enrichment of the Data Model:

- User requests for adding new entities to the data model
    - Adding new concept (propose description and categories)
    - Adding new properties to an existing concept
    - Adding relationship between two existing concepts

This components also allows the system admistrator (Admin) with special access to the XMANAI Data Model storage, to enrich and update the data model. The possible activites considered in this stage are as follows:

- Admin can view the new entity proposals
- Admin is able to add new entities to the data model
    - Adding new concept
    - Adding new properties to an existing concept
    - Adding relationship between two existing concepts
- Admin is able to edit an existing entity (concept, property, relationship) of the data model
- Admin is able to remove an existing entity (concept, property, relationship) of the data model

Another functionality for the KGM component is to restore data from the data storage and convert it to a data frame in order to be used by any relevant XMANAI components. The XMANAI ingested data through the data collection processes of WP2 that are either stored in a file or through an API must be mapped previously to the data model and stored as relational data (in WP2). The stored data in database can be used by the Knowledge Graph Manager to be extracted through a Python API to be developed as an assistance to the Knowledge Graph Manager, or be fed into an API made available by the same platform previously mentioned.

Based on what has been discussed, the main functionalities that are considered for the Knowledge Graph Manager component are as follows:

- **KGM1. Data Model visualization:** To visualise the XMANAI Data Model both in a tabular and graph view. This functionality allows the Data engineer and Business User to visualise the whole data model or step by step visualise the required categories information. It also helps and supports the UI for mapping the imported data to the XMANAI Data model.

- **KGM2. Data Model Lifecycle Manager:** To add, edit and remove Data Model entities including Concepts, Properties and Relationships. The request forms filled in by users will be available for revision by system administrators who can choose to add the new requested structures to the overall data model if the user chooses to request that the new structure is of benefit or can be shared with other users. The user also has the option to keep the requested addition to be made private to their own system instead. If the request is approved for public use, then the new relationship, concept or property will be available as an improvement on the data structure of the whole knowledge graph storage and recovery model.
- **KGM3. Data extraction from Database/datastore:** To extract the stored data from the database. Consequently, KGM3 is able to access and extract data from storage using SQL queries. The extracted data would be converted to the form of 2-dimensional Dataframe. This functionality provides back-end service for the other XMANAI components including Data Manipulation Engine and Model Engineering Engine for using extracted data to create AI pipelines.

Based on the utilized functionality of the component, the output of the component can be considered two categories:

- Cypher query output
- Data frame as the result of data extraction

Possible foreseeable limitations/Issues include the input data from WP2 to come in potentially different formats, since the different stakeholders might have data come in in different structures, making the extraction process potentially somewhat more complex.

4.3.2 Technology

This module is expected to utilize the following tools for the extraction, visualization and modifications of data for the knowledge Graph according to the brief analysis made so far:

- Neo4J(https://neo4j.com/) for Graph data Storage and Cypher query for data retrieval
- Pandas (https://pandas.pydata.org/) for its wide range of data manipulation functionalities
- Sqlalchemy (https://www.sqlalchemy.org/) python library to extract data from SQL files and databases
- Python (https://www.python.org/) frameworks to provide Rest services for interaction with other modules and other libraries for data manipulation depending on the data structures used for required data inputs and outputs. The exact frameworks will be finalised as the implementation activities progress.

4.3.3 Mockups

This section provides mockups of the main functionalities that will be offered through the KGM user interface.

The “Data Model Visualisation Dashboard” screen, depicted in Figure 4-3, shows the data model categories (on the right) that allow the user to select a concept category in the data model. The graph visualisation in the center of the screen represents the data model sub graph that is related to the selected category.

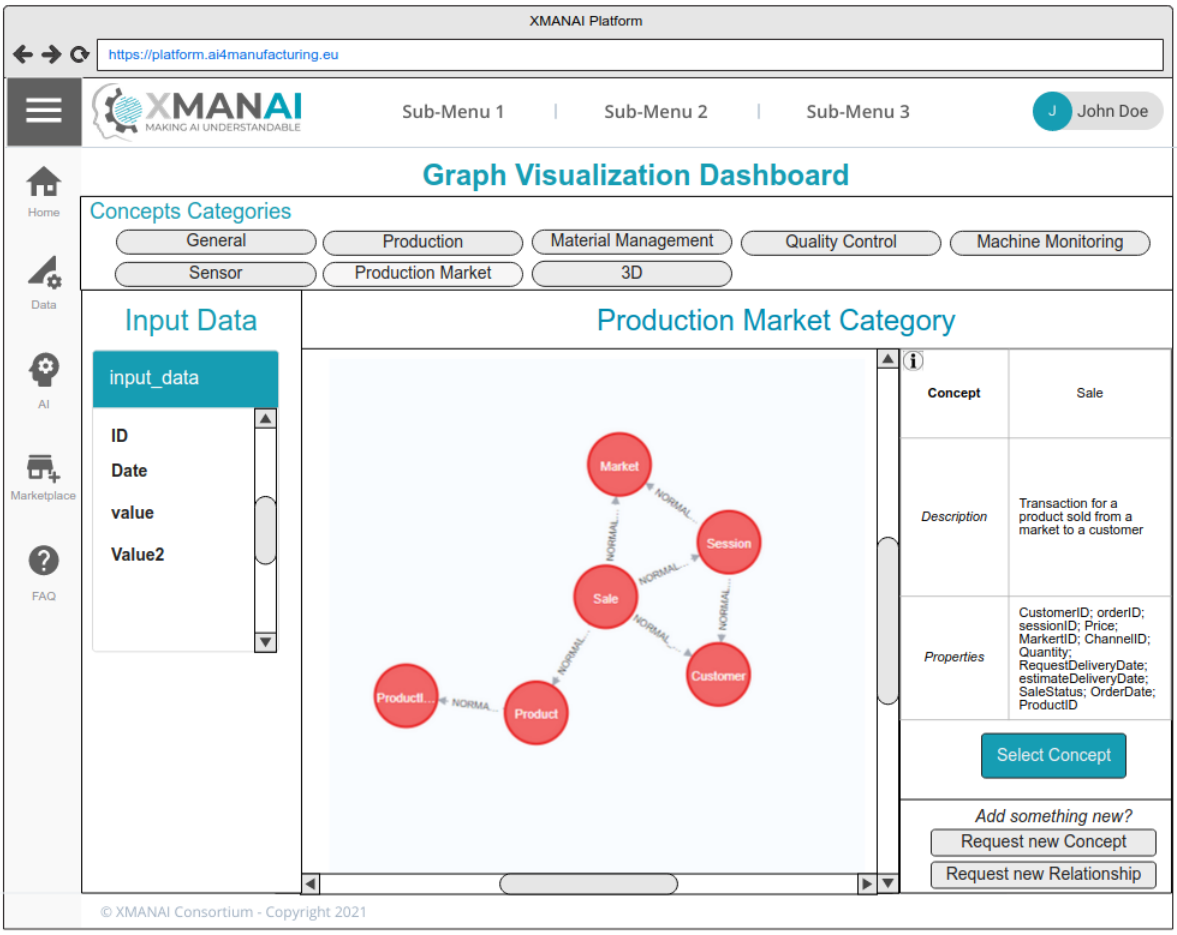

Figure 4-3: KGM - Data Model Visualisation Dashboard

Through the ”New Property Request Form”, depicted in Figure 4-4, the user will be able to define semantics for a property that is not available in the data model and propose to add it into the XMANAI data model at the same time.

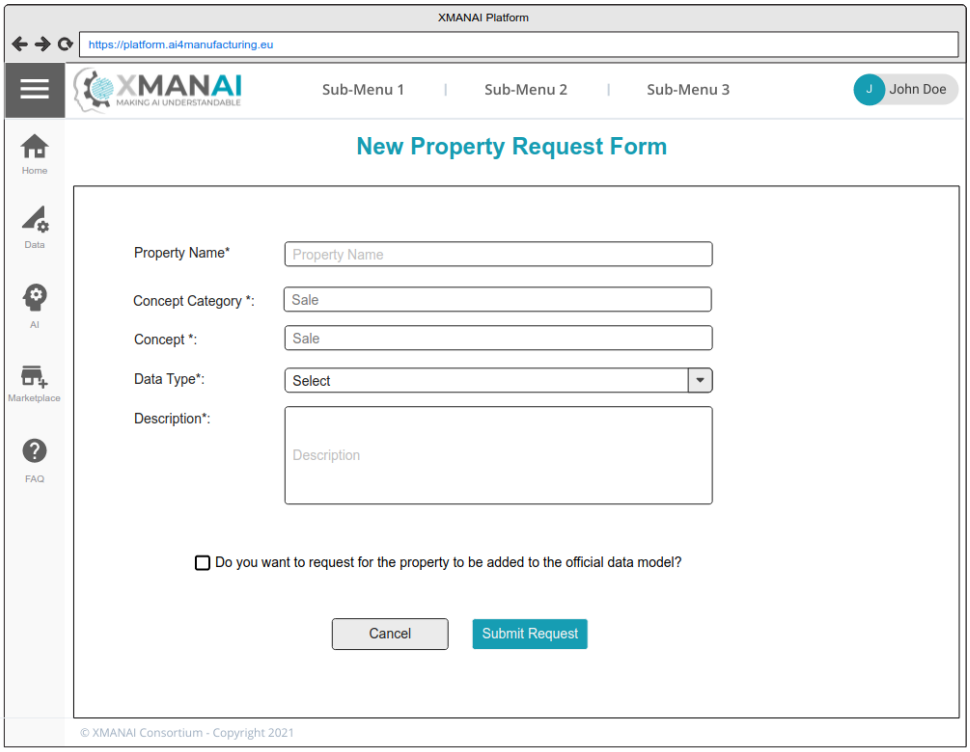

Figure 4-4: KGM – Property Request Form

Through the "Data Model Lifecycle Manager" screen (Figure 4-5), the data model administrator can review the list of data model entity (Concepts, Relationships, Properties) proposals that are created by XMANAI users.

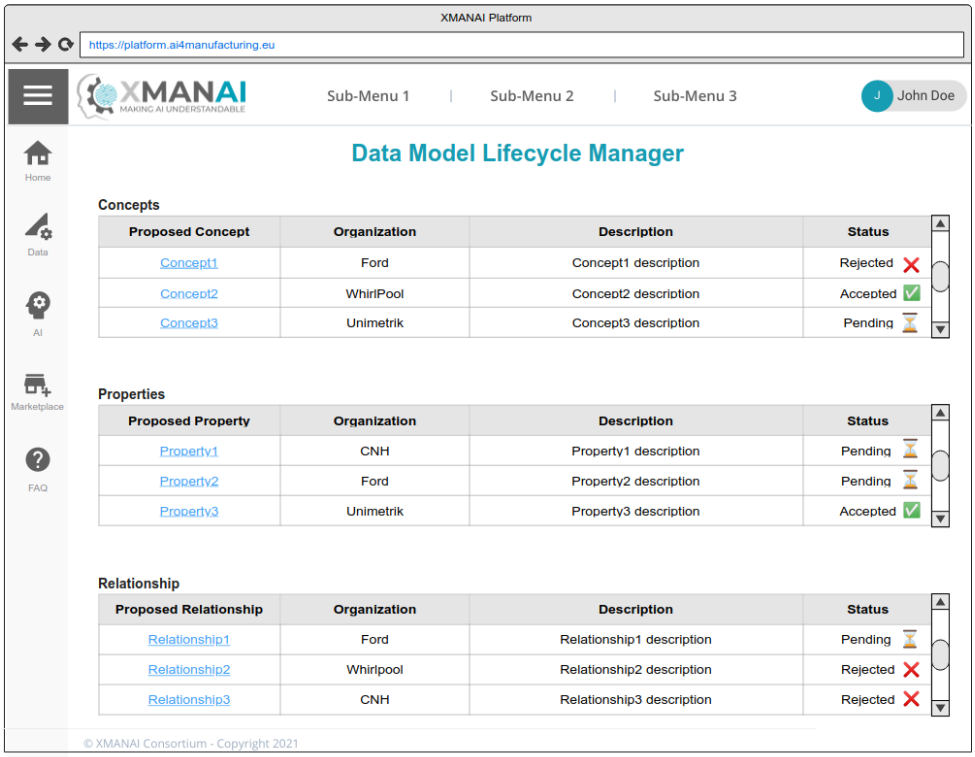

Figure 4-5: KGM – Data Model Lifecycle Manager

4.4 XAI Model Engineering Engine

4.4.1 Overview

The XAI Model Engineering Engine (XMEE) is the component responsible for managing all the operations regarding Artificial Intelligence models and their associated explainable tools. The operations performed by the XMEE will be invoked both on demand directly by the users (mainly scientists profiles) and in an indirect way as part of the XMANAI AI pipelines. In both cases, the XMEE component will need to communicate with the Models Catalogue to manage its contents.

The component allows the user to upload a new model to the XMANAI platform (which will be stored in the XAI Models Catalogue), select a baseline or trained model, choose the explainability tools available for a particular model (this functionality belongs to the XMXE component that is described in section 4.11 but the user may also access it through the XMEE component) and consider whether this model will be used to train a new one or will be used to add it to a deployment pipeline for inference. In case the user wants to train a model, the component will allow configuring the training aspects, such as the dataset to be used, the selection of hyperparameters or the validation strategy to be followed, in order to obtain the trained model.

The list of main actions to be performed using the XAI Model Engineering Engine is the following:

- Upload a new model to the XAI Models Catalogue.
- Select a baseline or trained model.
- Retrieve a model from the XAI Models Catalogue.
- Choose an explainability tool.
- Configure a model training session.

The XAI model engineering engine can be used by all XMANAI roles. Since the data scientist profile is the one most familiar with Machine Learning methods, it is expected to be the main user of this component. Although the following is a list of the different functionalities covered by this module, they will be continuously reviewed and updated throughout the project to meet the needs of all users. The functionalities are the following:

- **XMEE.1**. **Upload a new model to the XAI Model Catalogue**. This functionality allows new models to be included in the XAI Model Catalogue through a series of operations. First of all, it will be necessary to prepare a model package, which will include both the source code and the necessary package version file. This functionality also has to check that the package complies with the format required in the XMANAI platform. Once verified that the package is compliant with the catalogue specifications, it will be mandatory to fill in a form with the model name and its metadata to generate its associated metadata file, including the accessibility rights. Finally, after these steps, the model package and its metadata file will be uploaded to the platform.
- **XMEE.2**. **Access to the models catalogue**. To select a model and an explainability tool from the list, this functionality will allow the user to generate a list of ML/DL models, explainability tools and surrogate models (only if needed) sorted by model type (transparent, black-box and Graph ML) and by model category (classification, regression, clustering, etc.). This functionality also allows the user to retrieve the components desiderable to be used.
- **XMEE.3**. **Visualize model metadata**. Once the user has selected a model from the list, the metadata associated with that model will be displayed (providing a link to the full metadata associated with any data asset provided in the XAI Marketplace described in D5.1). This information will help the data scientist to ensure that the most appropriate algorithm is selected. The metadata will include information on the type and format of the input data, the parameters stored, the possibility of being re-trained, a list of configurable hyperparameters,

the validation metrics that apply to the model, performance statistics (for trained models only) such as inference time, a list of previous experiments (including in each one: the dataset used, the preprocessing applied, validation data such as segmentation or cross validation and validation metrics used and values obtained) and the list of python requirements needed to use the model locally.

- **XMEE.4**. **Support options to train from scratch or re-train a model**. In case of re-training, the necessary parameters from the previous experiment selected will be loaded.
- **XMEE.5**. **Configure training hyperparameters**. This functionality will allow the user to configure three types of hyperparameters: model hyperparameters, surrogate hyperparameters (only if required) and experiment hyperparameters. When a discrete hyperparameter is configured, the metadata will provide its possible values.
- **XMEE.6**. **Configure the validation strategy**. It will be possible to choose between segmenting the data into train/validation/test splits and configuring their ratios or selecting a cross-validation strategy and its number of folds.

The end result of the operations of this component will be a model trained to be integrated into different user-created pipelines.

The main dependencies of the XMEE are:

- Data Preparation Engine.
- Pipeline Designer.
- Experiment Tracking Engine.
- Pipeline Serving & Monitoring Engine.
- Results Visualization Engine.

4.4.2 Technology

Even though the main technologies of the XAI Model Engineering Engine component have not yet been selected, the possible technologies to be used in this component include:

- Pandas (https://pandas.pydata.org/) will be used to manipulate input dataframes.
- Numpy (https://numpy.org/) will support the required numerical operations.
- ML: scikit-learn (https://scikit-learn.org/stable/) will provide core ML functionalities as one of the most popular and mature machine learning libraries.
- DL: Tensorflow (https://www.tensorflow.org/) or Pytorch (https://pytorch.org/) will be leveraged for deep learning functionalities
- GUI: Vue.js (https://vuejs.org/) will be used for the development of the component's user interface.

4.4.3 Mockups

This section provides mockups of the main functionalities that will be offered through the XMEE user interface.

Through the "Model Selection" screen, depicted in Figure 4-6, the user will be able to select a model/experiment to be trained or to make inferences.

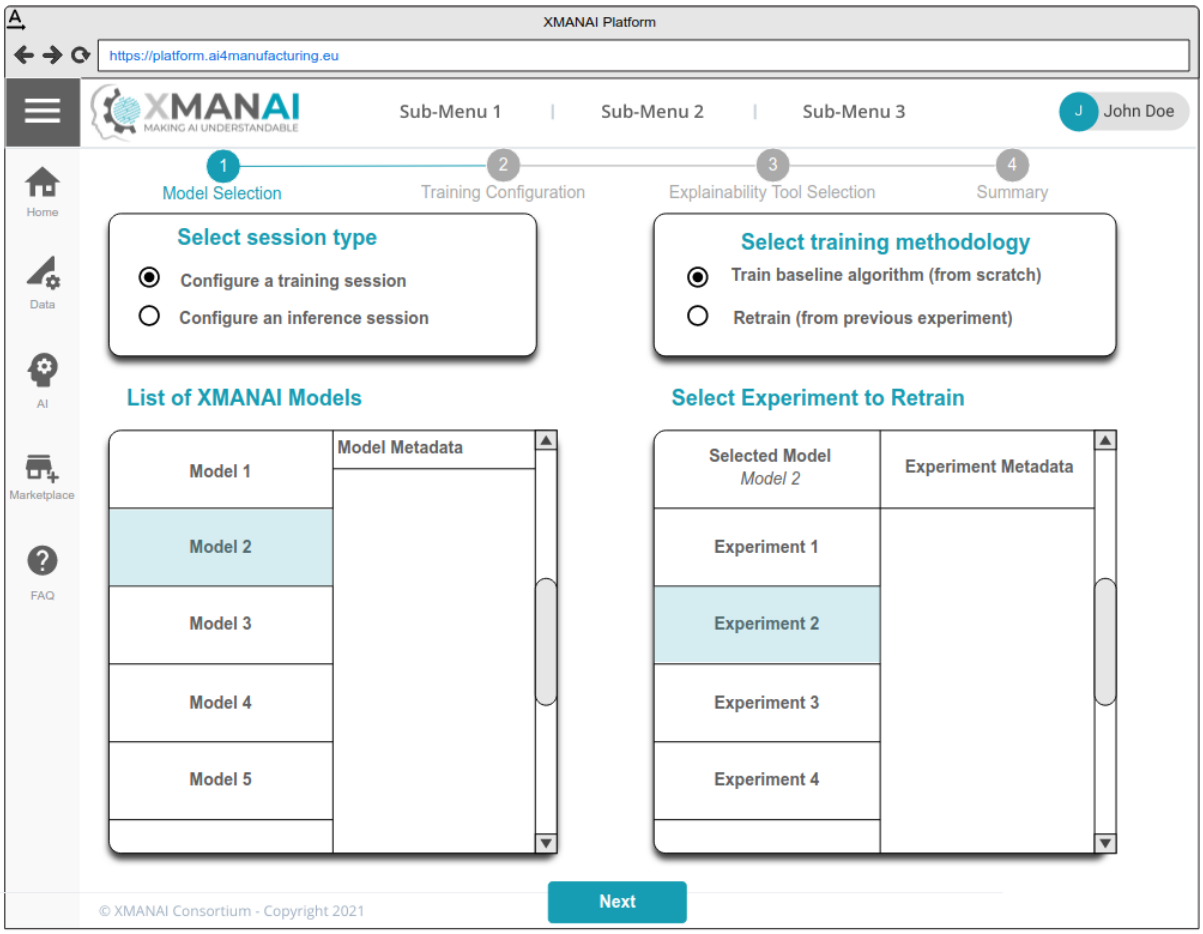

Figure 4-6: XMEE – Model Selection

The "Training configuration" screen depicted in Figure 4-7 is shown when the training functionality is selected. Through this screen, the user will be able to set the values of the hyperparameters of the model. Then, the user will be able to open the explainability tool selection interface (presented in section 4.11.3).

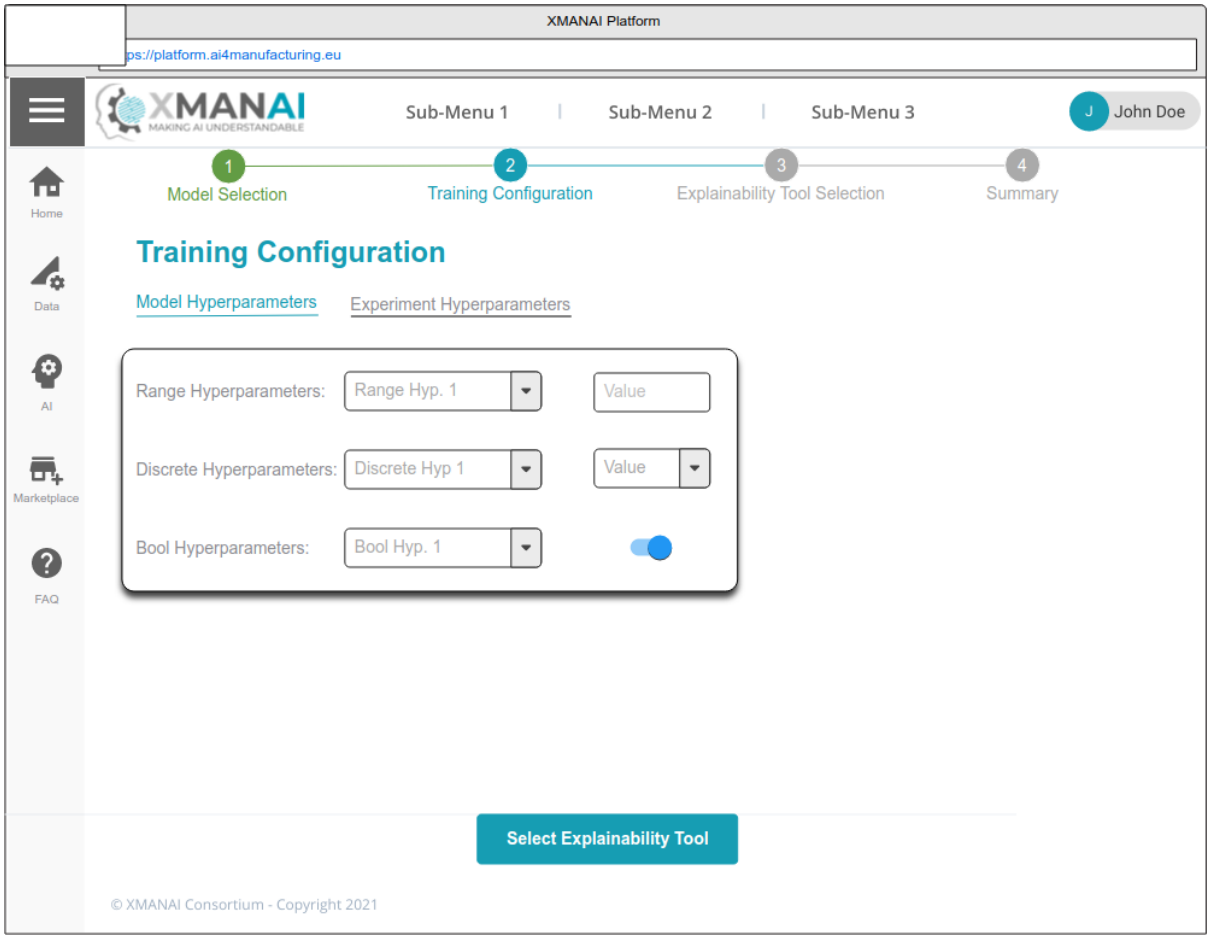

Figure 4-7: XMEE – Training Configuration

The "Summary" screen (Figure 4-8) shows a summary of the selected model, the explainability tool/model and their assigned hyperparameters, so that the user can check all the information.

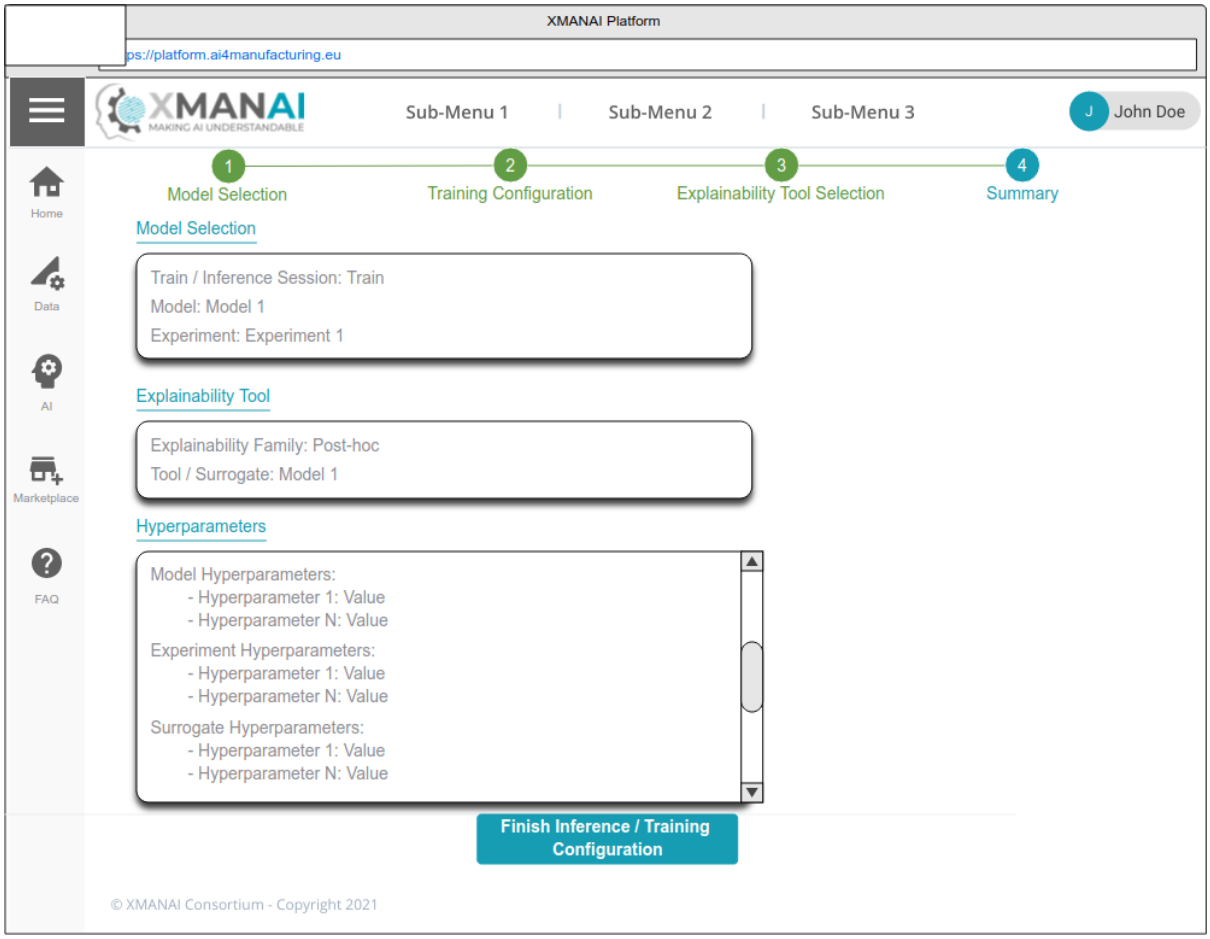

Figure 4-8: XMEE – Summary View

4.5 XAI Model Guard

4.5.1 Overview

The XAI Model Guard is the component responsible for providing the functionalities that safeguard the security and integrity aspects of the produced AI models in the XMANAI platform by employing a toolset of methods against adversarial attacks, as documented in the state-of-the-art analysis on XAI methods in deliverable D1.1. As Machine Learning has made tremendous advancements and its usage was significantly increased in the last decade, the produced AI models became an attractive target to adversaries that wish to manipulate them for malevolent purposes. The purpose of the XAI Model Guard is to formulate a security framework that is responsible for identifying potential adversarial attacks, assessing the risks associated with such attacks and proposing (and/or applying) corresponding risk mitigation measures.

In principle, the AI models are trained utilising datasets which are considered as representative and trustworthy for the selected problem in question, hence these datasets facilitate the modelling and training of effective and efficient ML models for the specific problem. Adversarial attacks target this process and the two primary adversarial attacks are as follows (Ilmoi, 2019):

- **Pre-training (poisoning) attacks** in which adversaries attempt to inject malicious data in the training dataset in order to poison it and as result to decrease the classification accuracy of the classifier. These attacks can be also classified based on the adversarial access level which

can be scaled from the logic corruption (the highest level) that disrupt the learning logic to the data manipulation of the training data, the data injection into the training data and finally the transfer learning (the weakest level) that eliminates the pre-existing training with a new one.

- **Post-training (evasion) attacks** in which adversaries attempt to mislead a trained classifier in order to perform a mis-classification of a sample towards a malevolent intent. These attacks constitute the most practical and the most used ones when targeting deployed ML models.

Depending on the type of the adversarial attack, different mitigation countermeasures can be employed. However, these countermeasures are heavily dependent on the nature of the input or training data, the selected algorithm for the ML process as well as the environment of the executed ML process. Another important aspect is the level of information that is accessible to the attacker. In white box attacks, the attacker has access to the information and parameters of the ML model under attack, while in the black box attacks this information is not accessible, hence the attacker tries to replicate the ML model and apply the adversarial data on the original model.

The XAI Model Guard will comprise a security framework that incorporates a set of techniques and methods which act as risk assessment and mitigation countermeasures against adversarial attacks. While several countermeasures can be encountered in the literature, the candidate countermeasures are heavily influenced by the parameters of the attack as described above. For the pre-training (poisoning) attacks the most common type of countermeasures is the outlier or anomaly detection techniques which are applied on the training data to detect poisoned data. Another commonly used approach is the micromodels defence that utilises classifiers on non-overlapping epochs of the training data in order to evaluate it using a majority voting approach. Finally, another common type of defence is the evaluation of the model's accuracy level through a sandbox run with new training data against the previous accuracy level with the existing training data. With regards to the post-training (evasion) attacks the most common type of countermeasures is the adversarial training (Ilmoi, 2019) where the training data are augmented with adversarial examples on purpose, so as to increase the model's robustness against adversarial perturbation. However, even these defence attacks are not a panacea. For example, extensive adversarial training could result in the decrease of the model's accuracy since the corresponding classifiers might be altered by the adversarial examples while also the approach of denoising ensembles requires extensive knowledge of the data else it could lead to computationally intensive solutions.

On top of the aforementioned defence techniques commonly encountered in the literature, the XAI Model Guard may also integrate additional defence and security measures towards safeguarding the integrity of the produced ML models in the XMANAI platform, for example through the provision of an integrity and security control mechanism which ensures that the ML models have not been altered or corrupted with the use of checksums.

It should be noted that an exhaustive list of supported countermeasures cannot be provided at the moment as this will be continuously refined and extended once the details of the applied ML techniques and ML models are clearly defined, as part of the WP4 activities.

The main functionalities offered by the XAI Model Guard component are as follows:

- **MG.01: Support identification of, and mitigation countermeasures against adversarial attacks.** This component will incorporate several techniques and methods which can be leveraged by the XMANAI users as defence against pre-training (poisoning) attacks and post-training (evasion) attacks depending on the designed ML process.
- **MG.02: Safeguard the integrity of the stored ML models** with the offering of a sophisticated mechanism based on checksum that detects any possible alteration or corruption of the stored ML models.

4.5.2 Technology

Although the exact technology stack has not been finalised yet, it is foreseen that for the implementation of the XAI Model Guard multiple technologies will be effectively combined and integrated for the realisation of the described functionalities. To this end, the Python programming language will be utilised and certain libraries and frameworks are currently under consideration:

- Pandas (https://pandas.pydata.org/) is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- Fast API (https://fastapi.tiangolo.com/) is powerful and easy to use web frameworks for the implementation of RESTful APIs in Python.
- NumPy (https://numpy.org/) is the de-facto choice for numerical computations in Python offering an extensive set of functions and ease of use.
- PyTorch (https://pytorch.org/) is a well-established library for machine learning specialised in deep learning built in top of Tensorflow.
- Scikit-learn (https://scikit-learn.org/stable/) is a dominant machine learning library for the Python programming language offering an extensive library of ML algorithms implementations.

4.5.3 Mockups

As explained in section 4.5.1, the functionalities provided by the Model Guard for safeguarding the security and integrity aspects of the produced AI models in the XMANAI platform are directly related to the utilised datasets and the use cases in which these datasets are exploited. As a consequence, the possible attacks and candidate mitigation countermeasures are also dependent on the utilised datasets.

Hence, at this point the user interface aspects of the Model Guard cannot be clearly defined and the efforts will be focused on the backend functionalities of the Model Guard. Nevertheless, an indicative user interface is presented in Figure 4-9 however it is expected that this user interface is bound to change as the project evolves and additional information is defined with regards to the applied ML techniques and ML models as part of the WP4 performed activities.

In detail, Figure 4-9 presents how the user can perform an adversarial attack risk assessment by selecting the parameters of the assessment, such as a specific ML model from the list of ML models that has legitimate access, the type of adversarial attack that will be assessed, the adversarial input data that will be used in the assessment and a set of parameters that fine-tune the assessment. As a result, the user is presented with the adversarial attack risk assessment results in the form of charts.

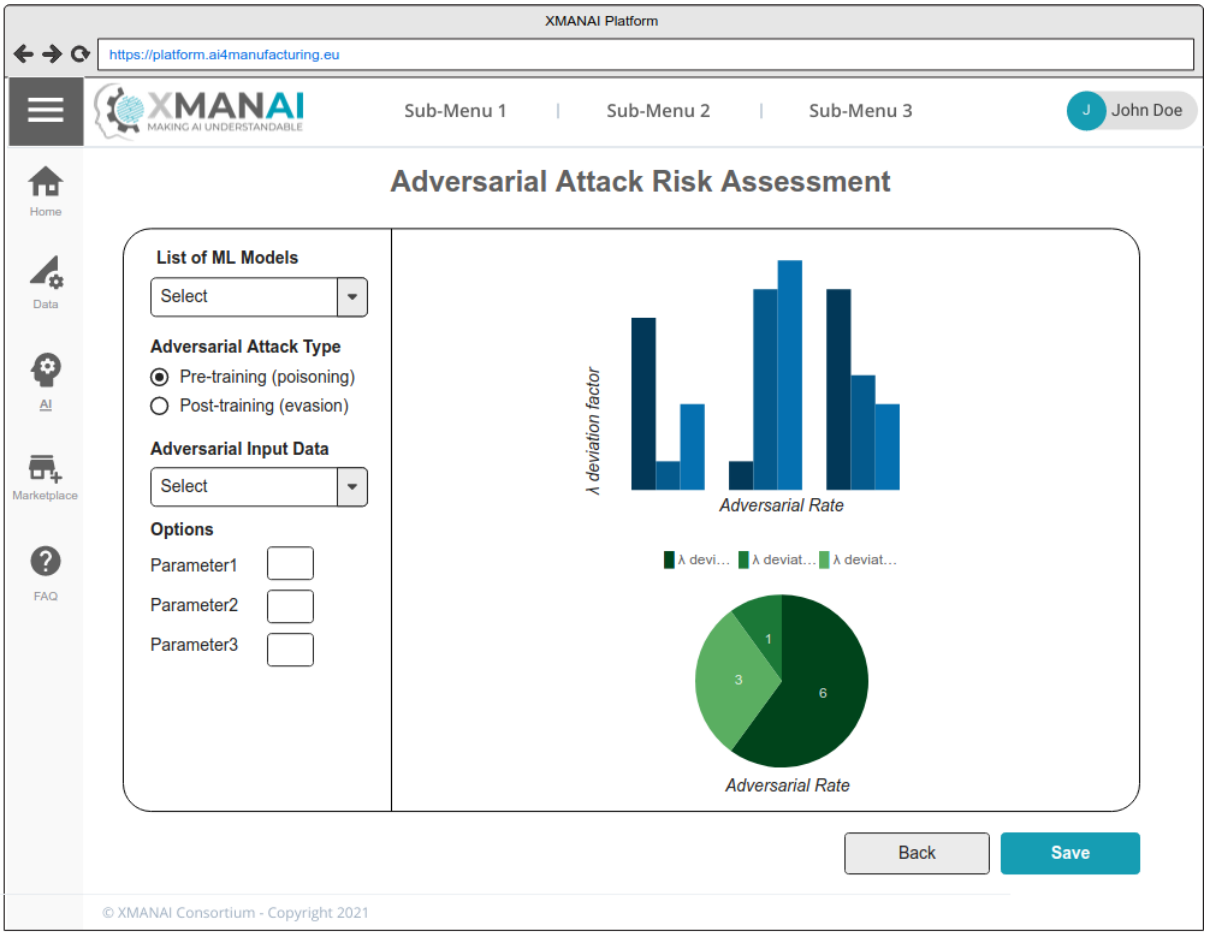

Figure 4-9: MG – Indicative View

4.6 Interactive Data Exploration & Experimentation Tool

4.6.1 Overview

The Interactive Data Exploration and Experimentation tool is suitable for the XMANAI user who wants to explore the available dataset and experiment with Machine Learning and Explainability techniques on it. Although the component can be used by all XMANAI roles (i.e., business users, data scientists and data engineers), it is principally tailored for the data scientist/engineer since it will provide low-level experimentation functionalities.

This component gives the user the freedom to design, experiment and evaluate any approach they consider applicable. The user will use it to gain intuition about the data and the models and decide which technique to use in each step, e.g., preprocessing, ML model, explainability model. For this reason, it will provide a wide range of tools for delivering the proper level of flexibility.

Since the component is used for experimentation, it will provide flexibility in collecting the input data. The user should be able to use both high-level tools described in D2.1 and D3.1 (e.g. the knowledge graph manager) and low-level queries to the XMANAI database (e.g., SQL queries, neo4j queries). Although not restricted to it, in most cases, the input will be in the form of a dataframe.

The Interactive Data Exploration part will allow the user to explore the properties of the dataset. The provided functionalities are divided into two categories; the predefined interactive tools and the defined-by-the-user functionalities. The former category includes a list of predefined exploration

tools, making some assumptions about the nature of the data. For example, in the case of tabular data, we may provide interactive tools with the statistics of each feature. The second category offers the user a list of appropriate Data Exploration packages for writing code and extracting the information they want.

In broad terms, the user should be able to perform the following actions as part of the Data Exploration Tool:

- Check the summary statistics of each feature (e.g., mean, median, max, min, standard deviation)
- Visualize the distribution of each feature
- Check the correlations between pairs of features
- Visualize pairwise plots between the features
- Apply a dimensionality reduction technique (e.g., SVD, Isomap)

The same requirements shape the functionalities of the Experimentation part of the tool, which will provide to the user the autonomy to experiment and research with any ML or explainability technique. Hence, the component will support several environments with appropriate ML and explainability packages installed, including the algorithms developed by the XMANAI project.

In general terms, the user should be able to perform the following actions as part of the Data Exploration Tool:

- Train and evaluate a custom ML algorithm using a general ML python package (e.g., Scikit-Learn)
- Use, i.e. train and evaluate, an ML algorithm already developed by the XMANAI application as part of some other component, for example 4.4 Model Engineering engine
- Apply an explanation technique using an available explainable AI package (e.g., Interpret ML)
- Apply an explanation technique developed by the XMANAI application as part of some other component, for example 4.11 Model Explanations Engine

It is crucial to clarify the restrictions that should be possed on the user. Since the component will provide low-level execution capabilities, it is essential to restrict the available actions, to avoid security issues. For example, the user should be able to access only the datasets available from his organization. Moreover, in terms of low-level queries, the user should be able to fetch a dataset but not alter or delete it. In general, there is a wide range of security aspects that should be handled appropriately in collaboration with the Identity and Authorisation Manager (described in D5.1 and D2.1).

In a nutshell, the functionalities provided by the component

- **IDEE.1 Provide a notebook with a preconfigured environment and specific restrictions** so that the user can apply low-level experimentation writing code
- **IDEE.2 Selecting data from XMANAI's datasets**, using either a high-level tool described in D2.1 and D3.1 (e.g. the knowledge graph manager) or low-level queries to the XMANAI Secure Datastore (e.g., SQL queries).
- **IDEE.3 Data Exploration tools for the available dataset**.
- **IDEE.4 Machine Learning and Explainability packages for experimenting** on the available dataset.

The functionalities described above will be executed using the computational resources provided by other components (e.g., the Execution and Orchestration component). Hence, it will interact with them to ask for these resources, for example, a python kernel with a preconfigured environment.

4.6.2 Technology

Although the final choice regarding the appropriate tools for providing the aforementioned functionalities cannot be finalized yet, we may describe a set of possible solutions that are currently considered and examined:

- Jupyter notebook (https://jupyter-notebook.readthedocs.io/en/stable/) is a powerful, expressive and robust tool widely used by data scientists for experimentation.
- JupyterLab (https://jupyterlab.readthedocs.io/en/stable/) is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab can configure and arrange the user interface to support various workflows in data science, scientific computing, and machine learning. Finally, there is a broad range of open-source plugins that integrate well with JupyterLab.
- Jupyter Hub (https://github.com/jupyterhub/jupyterhub) is a multi-user version of the notebook designed for companies, classrooms, etc. It gives users access to computational environments and resources without burdening the users with installation and maintenance tasks. Users can get their work done in their workspaces on shared resources that system administrators can manage efficiently.
- Jupyter Widgets (https://jupyter.org/widgets) enable interactive data visualization in the Jupyter notebooks. They can be used to create preconfigured interactive environments where the user may apply steps without writing code.
- Apache Zeppelin (https://zeppelin.apache.org/) is a web-based notebook that enables data-driven, interactive data analytics and collaborative documents with SQL, Scala, Python, R, and more. Zeppelin has multi-user support.

4.6.3 Mockups

This section provides mockups of the main functionalities that will be offered through the IDEE user interface.

Figure 4-10 shows the notebooks overview screen, through which the users can review, browse through and access stored notebooks from previous data exploration and experimentation processes.

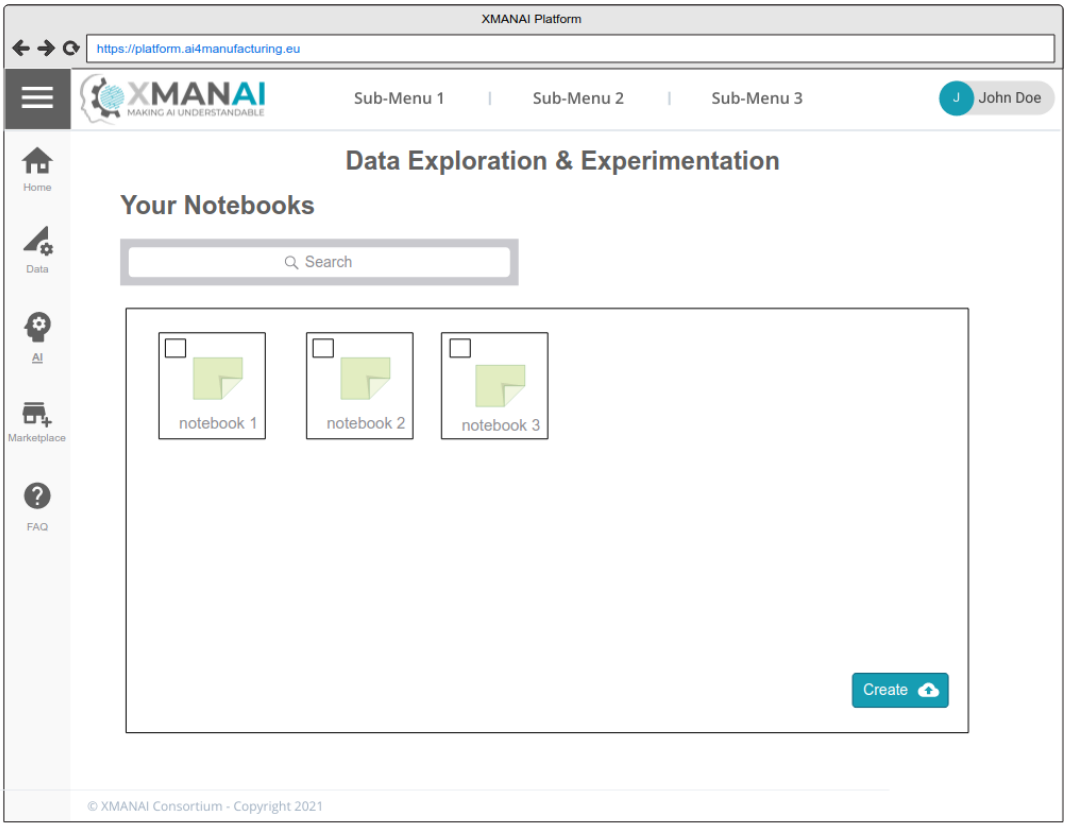

Figure 4-10: IDEE - Notebooks Overview

Figure 4-11 illustrates how the users will be able to create a new notebook, by selecting the appropriate preconfigured environment (Figure 4-12) or template (Figure 4-13)

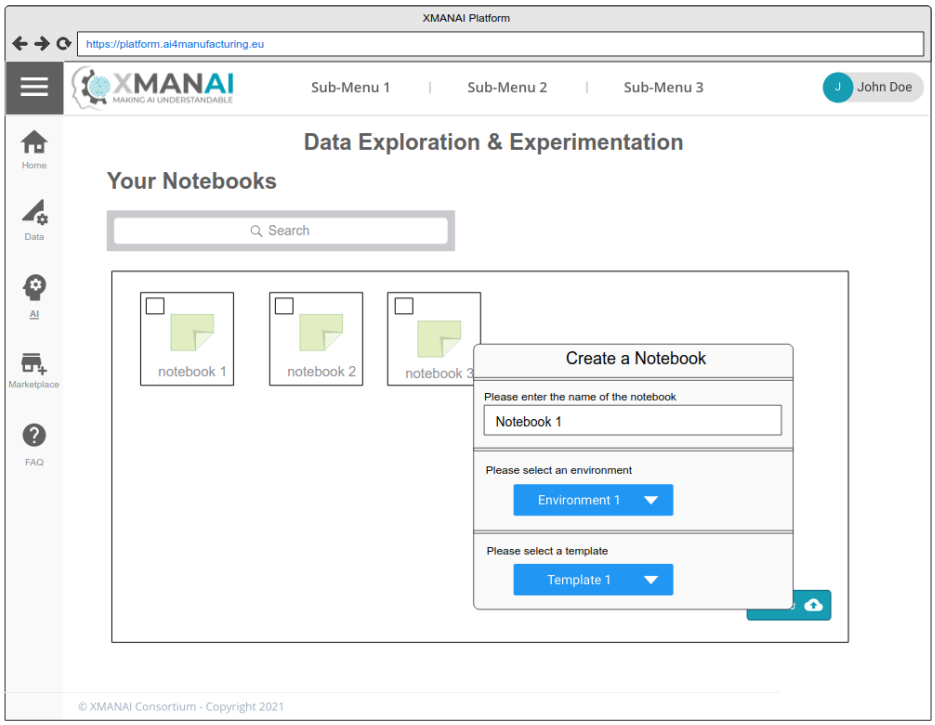

Figure 4-11: IDEE – Notebook Creation

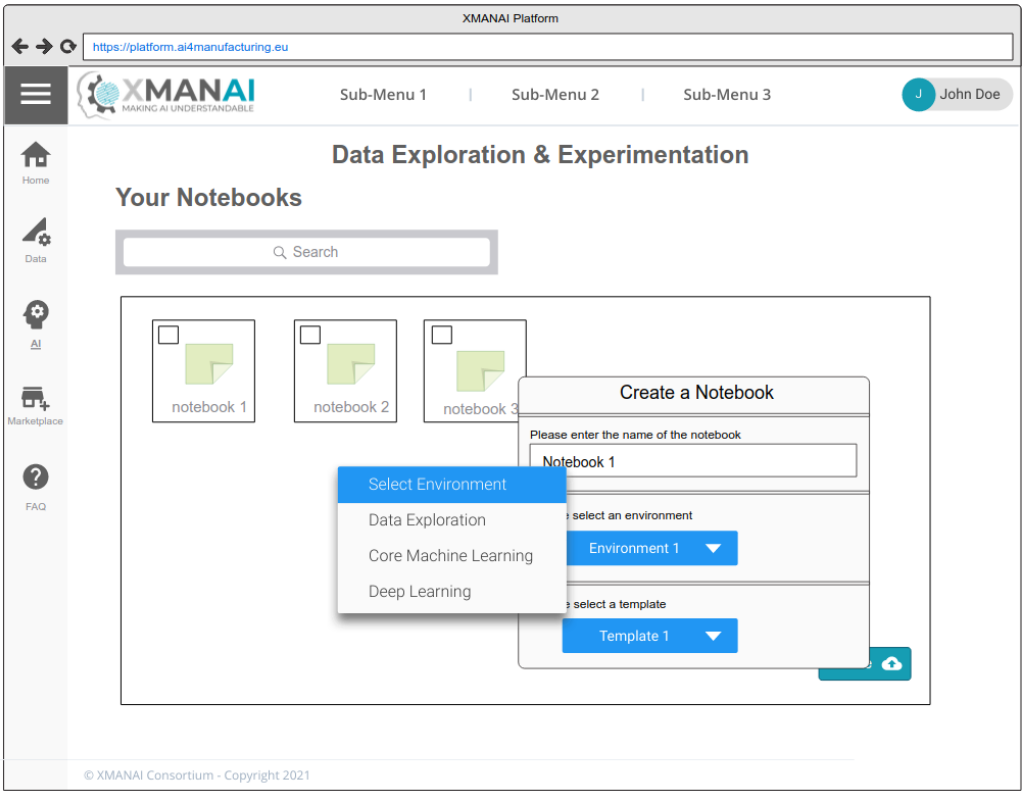

Figure 4-12: IDEE – Environment Selection

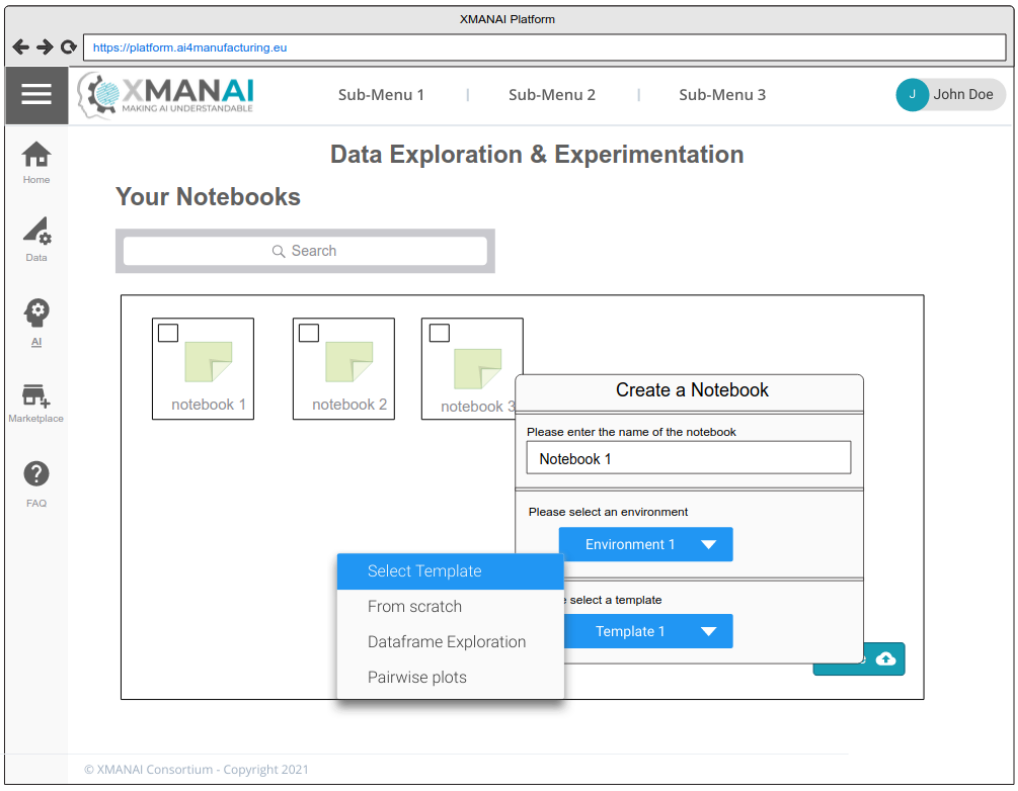

Figure 4-13: IDEE – Template Selection

After the users have selected or created a new notebook, they will be sent to the typical interface of the Jupyter Notebook, shown in Figure 4-14.

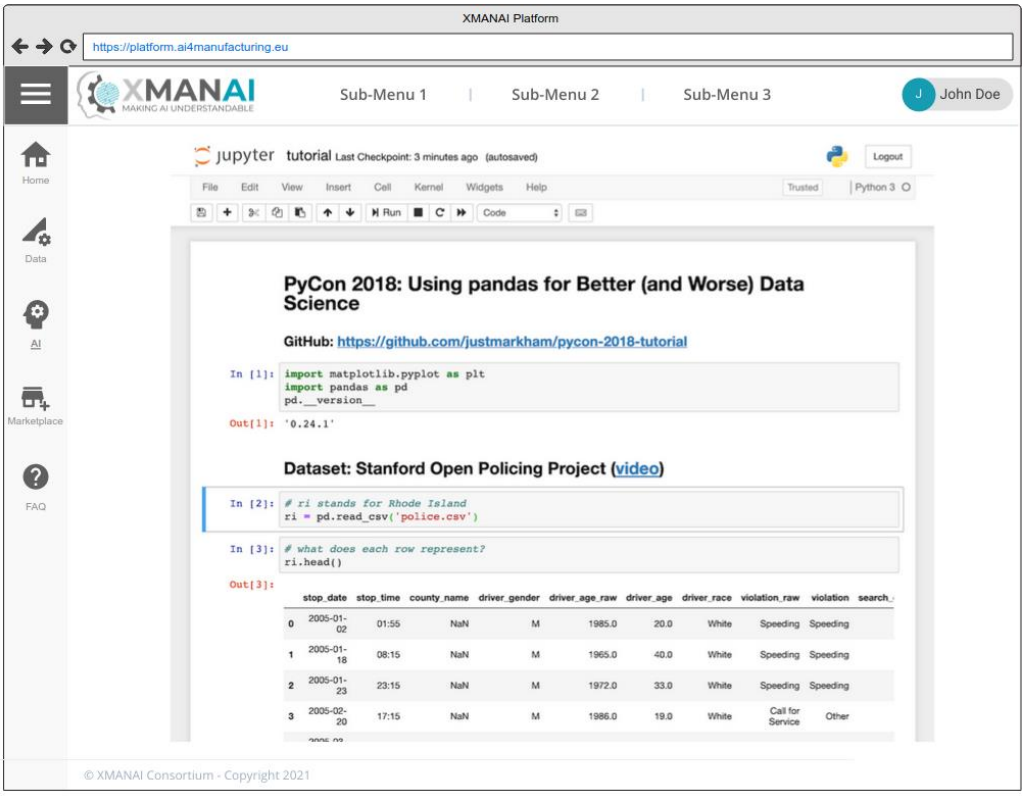

Figure 4-14: IDEE – Notebook Interface

4.7 Pipeline Designer

4.7.1 Overview

The Pipeline Designer is the component responsible for the end to end creation and management of the XMANAI AI pipelines, i.e. for bringing together the various parts of the AI lifecycle, from data exploration and understanding, to data pre-processing and feature extraction, to training and evaluating machine learning models and applying explainability methods, to deploying AI processes in production environments.

In a nutshell, the Pipeline Designer offers the following functionalities:

- **PD.1: AI pipelines design** which mainly involves the selection of the types of data processing and analytics tasks (flows) that should be configured, combined and executed as part of a pipeline. As explained in Section 2, the experimentation and development process followed in data-driven and AI enabled tasks is not sequential but instead includes numerous feedback loops, and involved tasks may be required to be executed in each iteration zero, one or multiple times. The component allows users to create new pipelines in a flexible way that fits the specific goal at hand and the needs of the particular user, enabling the selection of processes offered by other components. The Pipeline Designer can be used to create both simple processes (e.g. a single sequence of steps configured through the Data Preparation Engine) nd more complex ones (e.g., a pipeline that combines data preparation steps with ML model application steps, generation of instance level explanations and result provision as part of a pre-configured chart - thus involving the Data Preparation Engine, the XAI Model Engineering Engine, the XAI Models Explanation Engine and the Results Visualization Engine).
- **PD.2: AI pipelines CRUD operations** which include all processes around creating, retrieving, editing and deleting pipelines. It should be noted that due to the complexity of the pipelines and the different pipeline flavours that may be created depending on the types of tasks that

are included (explained in PD.1 above), the component needs to ensure consistency across all affected components and handle all artifacts that are related to a specific pipeline, e.g. resulting datasets, saved ML models, configured visualizations etc. Also, not all artifacts connected to a pipeline may be possible to be deleted upon its deletion. Furthermore, as part of pipeline creation, the component allows users to configure the settings of the underlying types of flows that are part of the pipeline, in case such settings are exposed and available to be defined.

The Pipeline Designer acts as a coordination mechanism and a common entry point to the various processes enabled by the WP3 components and also ensures the validity and consistency of the pipelines that are eligible to be used in production by the Pipeline Serving and Monitoring Engine.

It should be mentioned that although the Pipeline Designer acts as an entry point to various underlying functionalities (offered by different WP3 components) that target needs of all XMANAI user roles, it is primarily addressed to the Data Scientists and Data Engineers: The first can be considered as the core users, as they are the ones that will explore and prepare data and then apply ML models and explainability techniques until they reach a solution that solves the underlying manufacturing problem/challenge. They will also be able to connect various flows depending on the approach, e.g. in case of data preparation, feature extraction and model application have been developed as separate workflows that should be brought together. Once an end-to-end flow has been configured, if production deployment is foreseen, the Data Engineers will be able to change the deployment settings accordingly, or improve certain parts of the processing to increase computational efficiency.

4.7.2 Technology

Although the technology stack that will be utilized to provide the aforementioned functionalities is not yet finalized, taking into consideration the requirements that this component aspires to address, it is expected that the Pipeline Designer will leverage the S5 Analyze Platform, a proprietary analytics solution provided by the XMANAI technical partner Suite5, that offers pipeline design functionalitie.

4.7.3 Mockups

This section provides mockups of the main functionalities that will be offered through the PD user interface.

The first view (Figure 4-15) presents the list of AI pipelines that are accessible by the users (configured by themselves or a team member) and the information that is available for each one, as well as the actions that can be performed, which are actually links to the underlying components' interfaces.

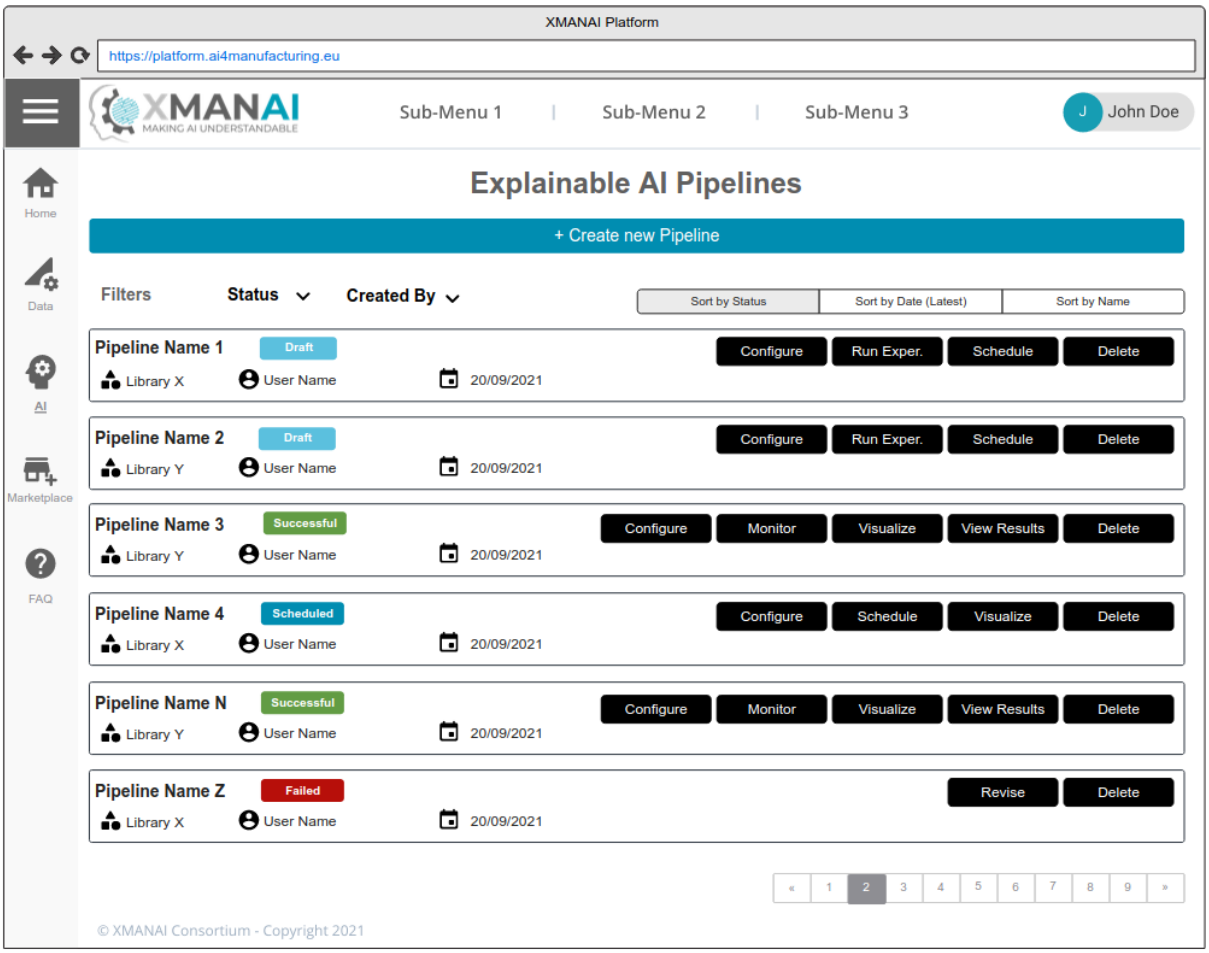

Figure 4-15: PD – Pipelines List

Figure 4-16 shows the way users can create and edit XAI pipelines in a graphical manner.

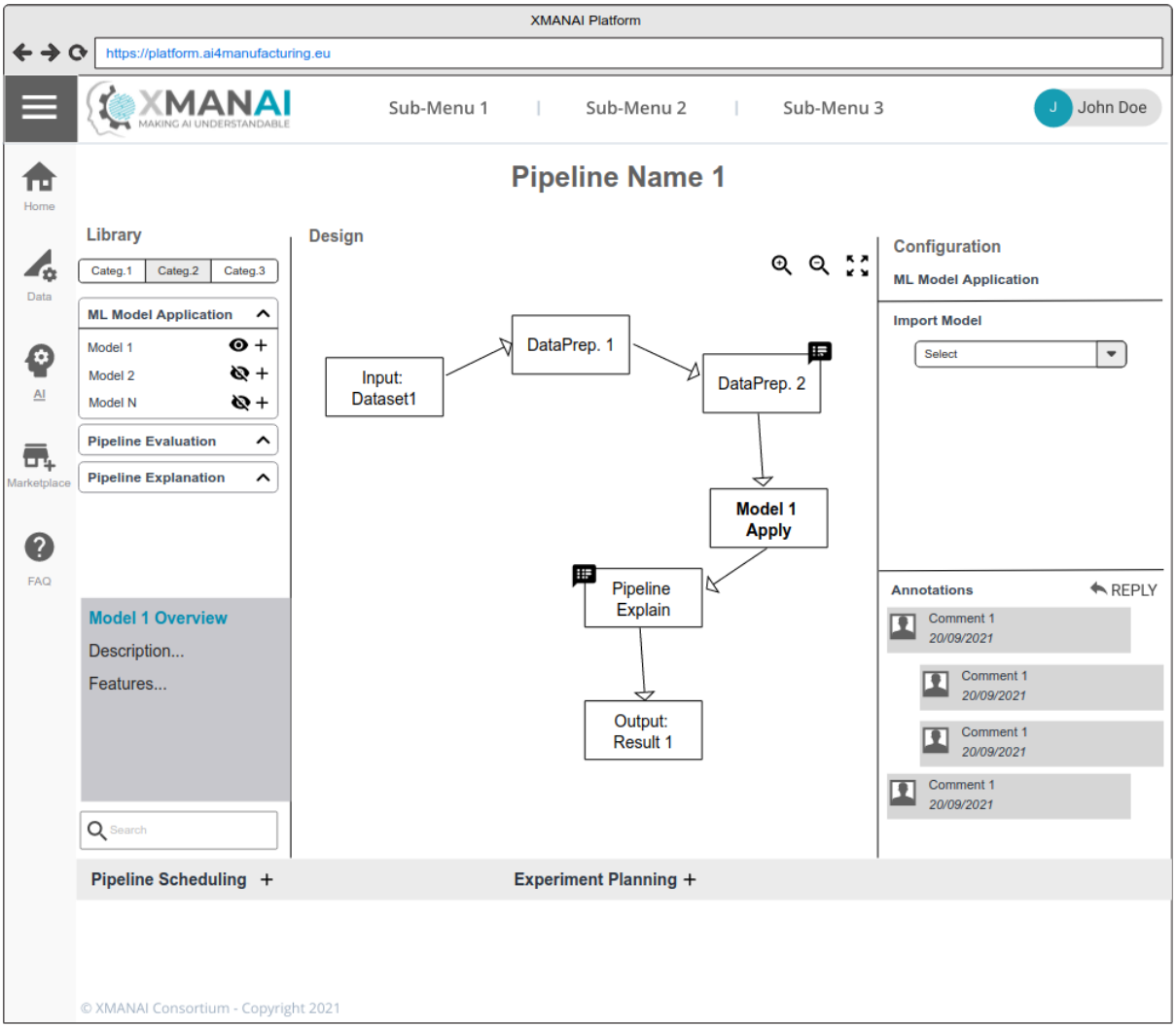

Figure 4-16: PD – Pipeline Creation

4.8 Experiment tracking engine

4.8.1 Overview

The Experiment Tracking Engine is responsible for recording experiments' information which run in the XMANAI platform. The term "experiment" refers to any model training process with a specific configuration of training data, metrics and hyperparameters. The specific engine makes sure that all this information (metadata) gathered from different experiments is stored in one place (database) easily accessed by the users.

The Experiment Tracking Engine mainly addresses the needs of data scientists that, during the model development phase, try various model configurations in order to find the one with the best model performance. The Experiment Tracking Engine saves the different model parameters and the performance metrics, allowing the users to go through the saved metadata and find the preferred ones, eliminating the risk of losing any sets of parameters.

The inputs to the engine are dictionaries of parameters and metrics that the users decide to save per experiment. It works closely with the XAI Model Engineering Engine as the inputs will be provided by the configurations and the training of the models which will be realized through the functionalities of

XAI Model Engineering Engine. It is also connected to the XAI Visualization Engine as the users will be offered an interface to navigate through the recorded experiments and visualize their metadata.

The Experiment Tracking Engine will allow the users to inspect the information of all the executed experiments (metrics, parameters, artifacts), visualize them for more intuitive understanding, compare the outcomes of different experiments and select the experiment that best addresses their needs.

In brief, the functionalities of the Experiment Tracking Engine are as follows:

- **ETE.1: Recording of parameters, metrics and artifacts** using the appropriate functions provided by the APIs of the selected library. The parameters and the metrics will be stored in a dedicated database which will be accessed by the users through a user interface. Along with the parameters, the library provides functions for storing artifacts (output files of any type e.g. images, parquet or pickle files) which will also be accessible by the users through the same interface.
- **ETE.2: Visualization of the metrics and parameters** through the dedicated user interface. Data scientists will be able to select an experiment through a list of all executed experiments and visualize the corresponding parameters and metrics along with any other stored information. According to the type of the model, the visualizations could indicatively be learning curves or histograms.
- **ETE.3: Experiment comparison**. The users will be able to select a set of experiments and compare their performances using either tables of parameters and metrics for all the experiments or visualizations such as scatter plots, contour plots or parallel coordinates plots for a more intuitive insight on the experiments' outcomes.
- **ETE.4: Retrieval of the best parameters** through the user interface or the library's APIs. The interface provides the functionality of filtering the experiments based on conditions laid down by the users. At the same time, the engine's APIs shall provide functions that can be used to fetch the parameters of the best performing model in the form of a pandas series allowing the users to select any of these 2 approaches.

It should also be noted that the model registry (described in D5.1 and developed in WP4) works closely with the Experiment Tracking Engine. The users could find the best performing experiment through the recorded experiments and instead of selecting the parameters used for training, they could select the trained model directly from the model registry.

4.8.2 Technology

State of the art libraries will be used for the implementation of the Experiment Tracking Engine to ensure that the aforementioned functionalities will be provided. Specifically, the ML-Flow (https://mlflow.org/) library and its provided APIs will be leveraged for logging the experiments' metadata and MongoDB as the database for storing this information.

4.8.3 Mockups

This section provides mockups of the main functionalities that will be offered through the ETE user interface.

Figure 4-17 depicts an indicative visualisation of a comparison between different experiments, along with the available accompanying information for each experiment. The user will be able to search for and select the experiments to be compared, as well as the important metrics and parameters across which the comparison will be performed.

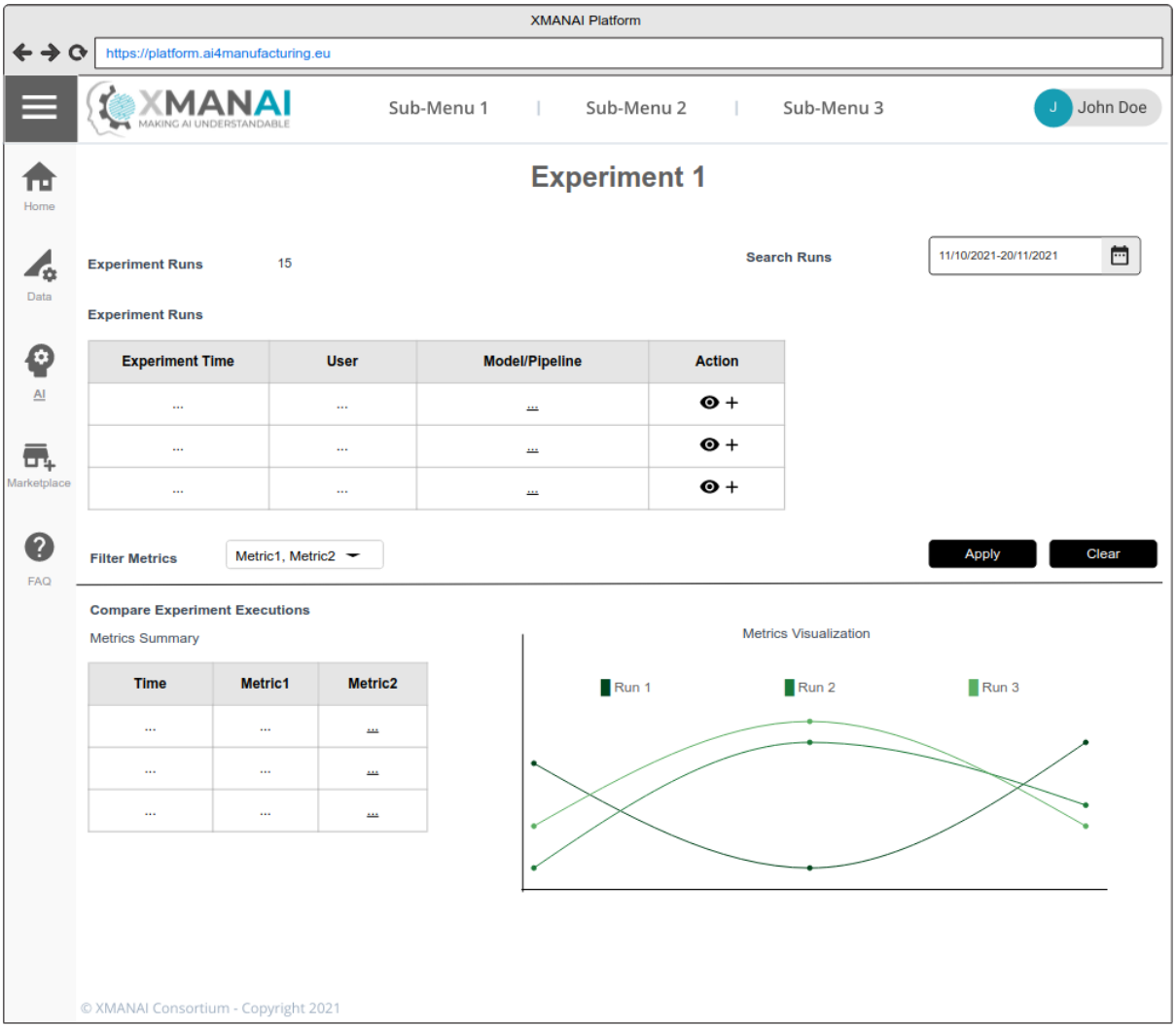

Figure 4-17: ETE – Experiments' Comparison

As depicted in Figure 4-18, more details for each experiment will be also made available through the "experiment details" view of the Experiment Tracking Engine.

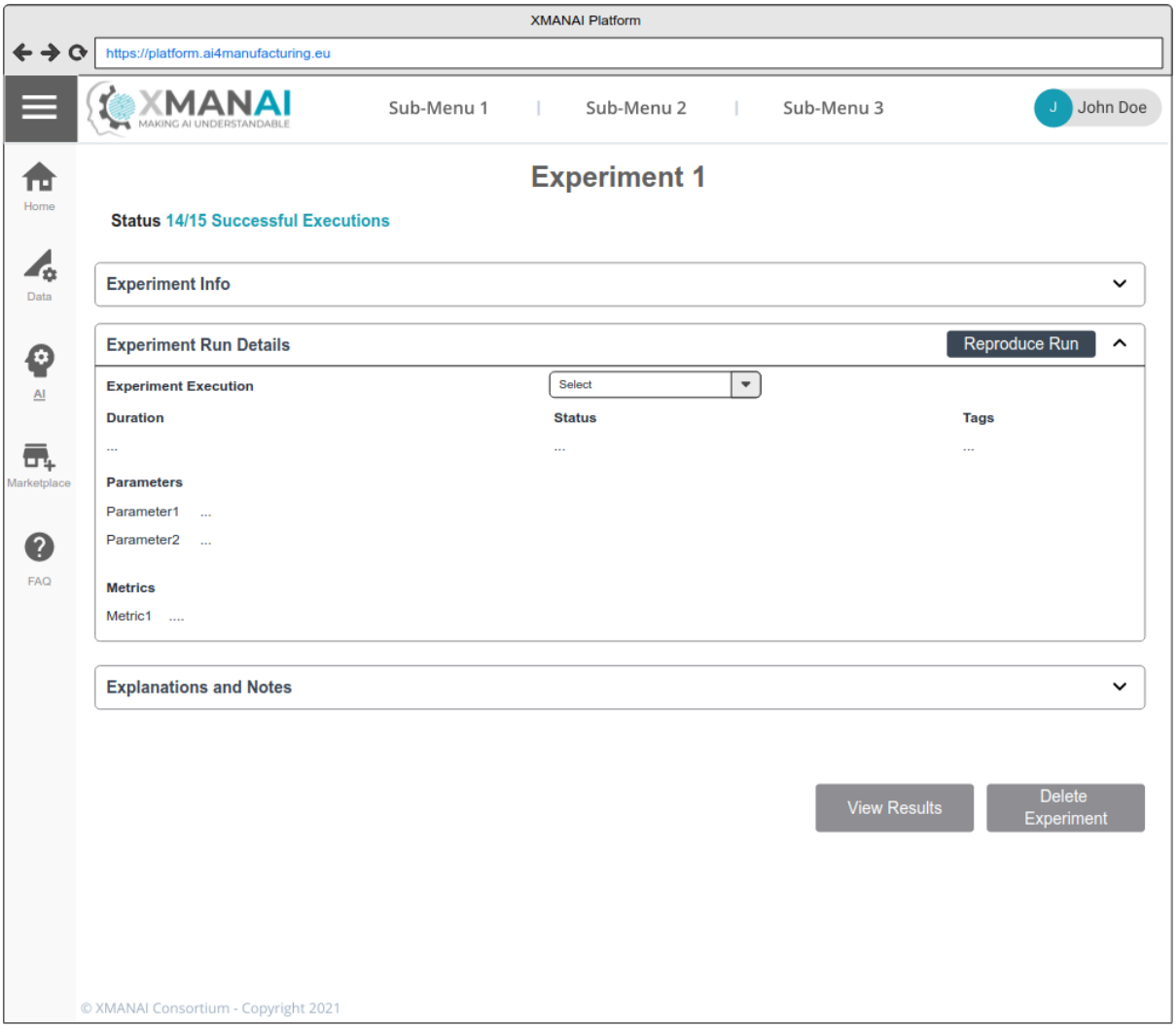

Figure 4-18: ETE - Experiment Details

4.9 Pipeline Serving & Monitoring Engine

4.9.1 Overview

As described in XMANAI Deliverable D5.1, the Pipeline Serving & Monitoring Engine (PSME) is part of the XAI Pipeline Manager and it is responsible for the managing and monitoring of Explainable AI Pipelines. This component serves two main purposes for the XMANAI Platform and can be found below.

- **PSME.1: Pipeline managing**, which comprises all the activities related to scheduling activities for execution and takes into account a few key aspects needed for the pipeline in order to be effective:
    - *Execution Location*: the environment in which the pipeline will be executed. In can be referred to the centralized cloud provided by the XMANAI Platform or a private cloud instance.
    - *Execution Modality*: refers to the activities needed for scheduling the execution that will be automatically triggered for certain pipeline types, and deferred for others. In the former case the user responsible for the pipeline serving must specify the time

instant at which the execution should start, as well as the execution pattern (e.g., once every hour, twice a month, …) In the latter case, instead, the user must specify the set of constraints, requirements or circumstances under which the triggering occurs.

- **PSME.2: Data monitoring,** which can be defined as the ongoing process of measuring the data's fitness for use. Monitoring a pipeline execution process is a crucial task since potential issues with operations in the earlier stages can be propagate through the pipeline and the negative effects can cascade. If an execution block is tainted, everything that follows will be too. The PSME component can achieve data monitoring by presenting the user with a custom dashboard that displays all the metrics needed in order to evaluate the pipeline behavior over time. The visualization dashboard also warns the user about the presence of one or multiple tasks that failed during the process (if any), along with a brief description of the root source that may have caused the issue to occur and a more detailed execution log that displays step-by-step the trace of execution until the failure point. This component will be crucial in identifying and understanding why and how a failure occurred, allowing for an easier recovery and reduced disservices.

The first input for the PSME component can be denoted as the architectural skeleton of the pipeline that needs to be executed, along with all the necessary steps composing the pipeline structure. Another necessary input is the Execution Location of choice, whether it is a local on-premise environment or the XMANAI Platform. Finally, this component requires an Execution Modality to be decided in order to appropriately schedule the pipeline.

The expected output for this component is a complete pipeline architecture, ready to be ingested by the Execution & Orchestration component and executed according to its predefined Execution Modalities on the established Execution Location.

4.9.2 Technology

Apache Airflow is the selected component that will employ the following technologies and tools in order to exploit the expected functions.

Airflow is an open source Python-based platform to programmatically author, schedule and monitor workflows through a robust and modern web interface. Further information can be found in the Airflow Docs: https://airflow.apache.org/docs/apache-airflow/stable/index.html

4.9.3 Mockups

This section provides mockups of the main functionalities that will be offered through the PSME user interface.

As explained, scheduling the pipelines' execution is one of the functionalities offered by this component. Figure 4-19 shows the user interface through which the user can configure the scheduling time of a batch pipeline. The user can select both the scheduling and when to start the scheduled execution.

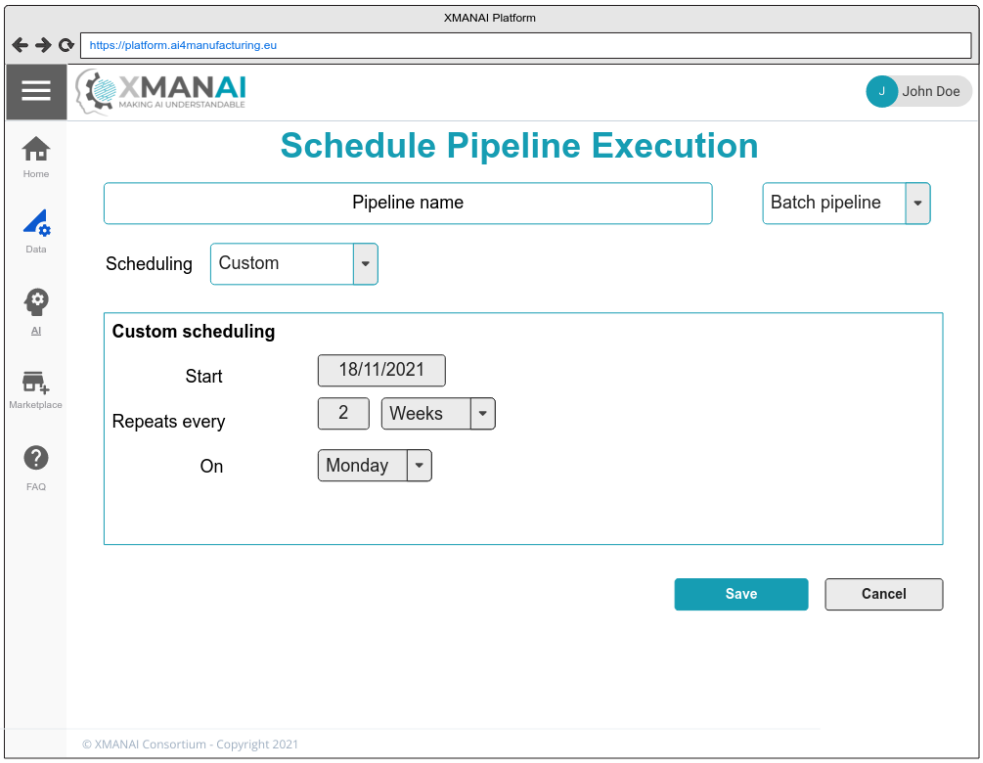

Figure 4-19: PSME - Pipeline Scheduling

Figure 4-20 shows the dashboard from which the user can monitor the execution of all pipelines related to the projects they have access to.

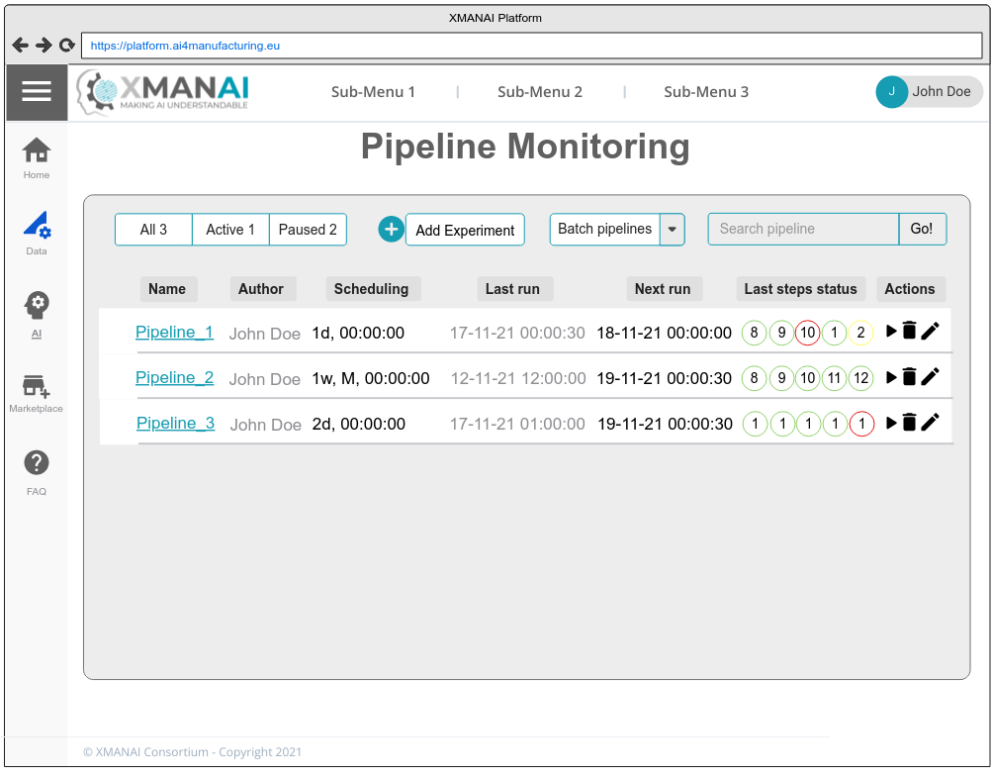

Figure 4-20: PSME - Pipelines Dashboard

4.10 Results Visualization Engine

4.10.1 Overview

The Results Visualisation Engine undertakes the responsibility of providing the visualisation capabilities of XMANAI with regards to the visual representation of the results generated from the performed data analysis, as provided by the Execution & Orchestration component. Hence, the Results Visualisation Engine offers the intuitive and easy-to-use interface through which the XMANAI users are able to create the visual representation of the produced results in a variety of visualisation types spanning from simple static charts to more advanced multilevel and interactive charts. In this sense, the Results Visualisation Engine provides the means to XMANAI users to leverage the results of the data analysis in order to gain a more comprehensive picture of the produced results, extract valuable information, discover trends and unveil patterns.

The list of supported visualisation types, constituting the results visualisation suite, will include the most prominent visualisation types having in mind the expected manufacturing-related analysis that will be performed and will be continuously extended with new and more-targeted types that will be identified based XMANAI stakeholders' needs as the project evolves. The component receives as input the produced results from the execution of an XAI pipeline, prepares internally the retrieved data into the most suitable form based on the selected visualisation type and renders the customised visualisation to the user.

In particular, the results visualisation process will follow a three-step process where the user is guided to provide the required information on each step towards the creation of the desired visualisation. At first, the user is presented with the list of results that the user has legitimate access to, upon consulting the Policy Engine and the Contract Manager, which are documented in deliverable D2.1, in order to select the desired data analysis result that will be visualised. Upon the selection of the desired result, the user is presented with the list of available visualisation types in order to select the desired visualisation type. Once the visualisation type is selected, the user is prompted to select the columns or entities of the result that will be utilised in the visualisation creation, as well as to set the corresponding parameters depending on the selected visualisation type, in order to customise the visualisation based on the user's needs. Once all input is set, the selected visualisation is generated and presented to the user. Depending on the nature of the selected visualisation type, different options are presented to the user in order to interact with the produced visualisation, to save it for later use or to export it for usage outside of the platform.

The main functionalities offered by the component are as follows:

- **RVE.1: Support of a guided and effortless user-interface in the form of a stepwise wizard for the creation of the desired visualisation** in which the user is able to select the data analysis result that will be visualised from the ones that has legitimate access, the desired visualisation type and its parameters that will customise the selected visualisation type.
- **RVE.2: A comprehensive visualisation suite** with the support of multiple visualisation types through a large number of built-in charts which can be customised and parameterized based on the nature of the produced results as well as the needs of the user.
- **RVE.3: Storage and exporting of the visualisation result** upon its creation. The generated visualisation can be saved for later use or it can be exported in various formats, such as JPEG or PDF in the form of a report, in order to be utilised outside of the platform.

4.10.2 Technology

The realisation of the aforementioned functionalities of the Results Visualisation Engine requires the combination of technologies that will facilitate the addressing of the respective requirements in an efficient and effective manner, as well as the implementation of both the backend-oriented and frontend-oriented functionalities that should be smoothly integrated. Although the exact technology stack has not been finalised yet, the state of the art analysis performed on Section 2.3.2 provided useful insights on the candidate libraries and frameworks that constitute the suitable candidates for the implementation of the Results Visualisation Engine. Hence, for the backend-oriented functionalities, Java 11 and the powerful Java-based Spring Boot framework will be utilised to provide the aspired functionalities. When it comes to the frontend-orient functionalities their development will be based on one of the libraries (or combination of them) as listed below:

- Apache Echarts (https://echarts.apache.org/en/index.html) is a powerful interactive charting and visualisation library with an advanced rendering engine which is compatible with most widely used browser and mobile devices. It provides an extended list of visualisation types with responsive design which are highly customizable, support of multiple data formats and easy integration.
- D3.js (https://d3js.org/) is a well-established visualisation library that supports multiple rendering technologies such as SVG, Canvas and HTML. It supports a large variety of visualization and interaction techniques that can be utilised or combined to provide the aspired visualisation results through multiple visualisation types.
- Plotly (https://plotly.com/javascript/) is a high-level declarative charting library offering an extensive set of chart types. It supports multilevel and interactive charts which are highly customisable while also supporting extensive integration capabilities.
- Google Charts (https://developers.google.com/chart) is a powerful charting library suitable for interactive charts spanning from simple charts to more complex charts which are highly customisable and easily embeded with the use of simple Javascript.
- Leaflet.js (https://leafletjs.com/) is a well-established charting library specialised in the rendering of interactive maps which can be extended with multiple plugins and include multilevel information on top of maps.

4.10.3 Mockups

This section provides mockups of the main functionalities that will be offered through the RVE user interface. As explained in section 4.10.1, results visualisation process is a three-step process.

In the first step, the user selects the desired result that will be visualised and its metadata are displayed, as depicted in Figure 4-21.

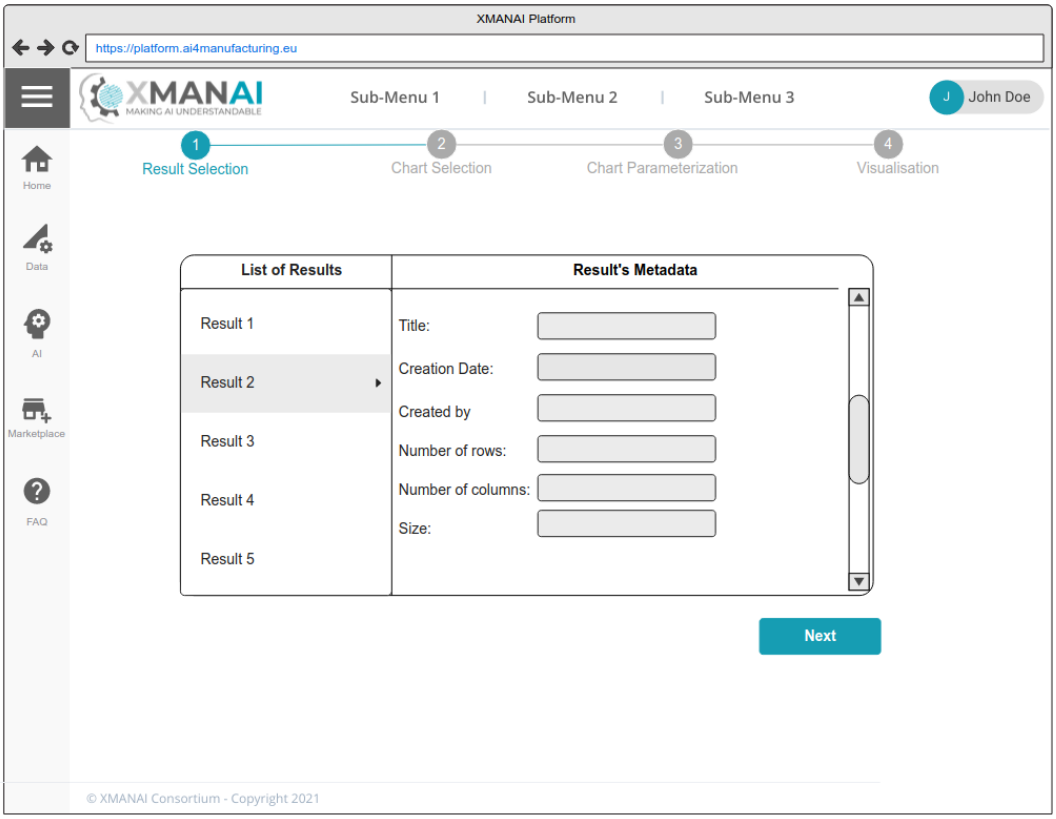

Figure 4-21: RVE - Results Selection

In the second step, the user selects the desired chart type from the list of available charts (Figure 4-22).

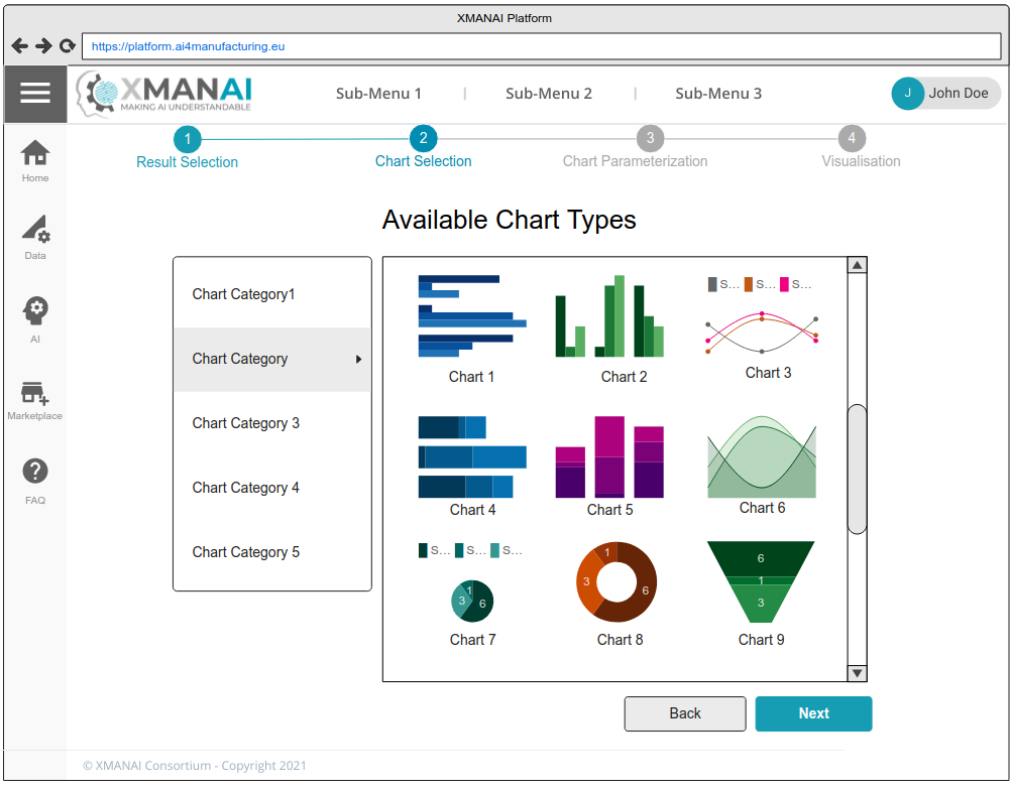

Figure 4-22: RVE - Chart Selection

Upon the selection of the chart type, the user is requested to select the columns of the results that will be used as the input parameters for the chart generation (Figure 4-23).

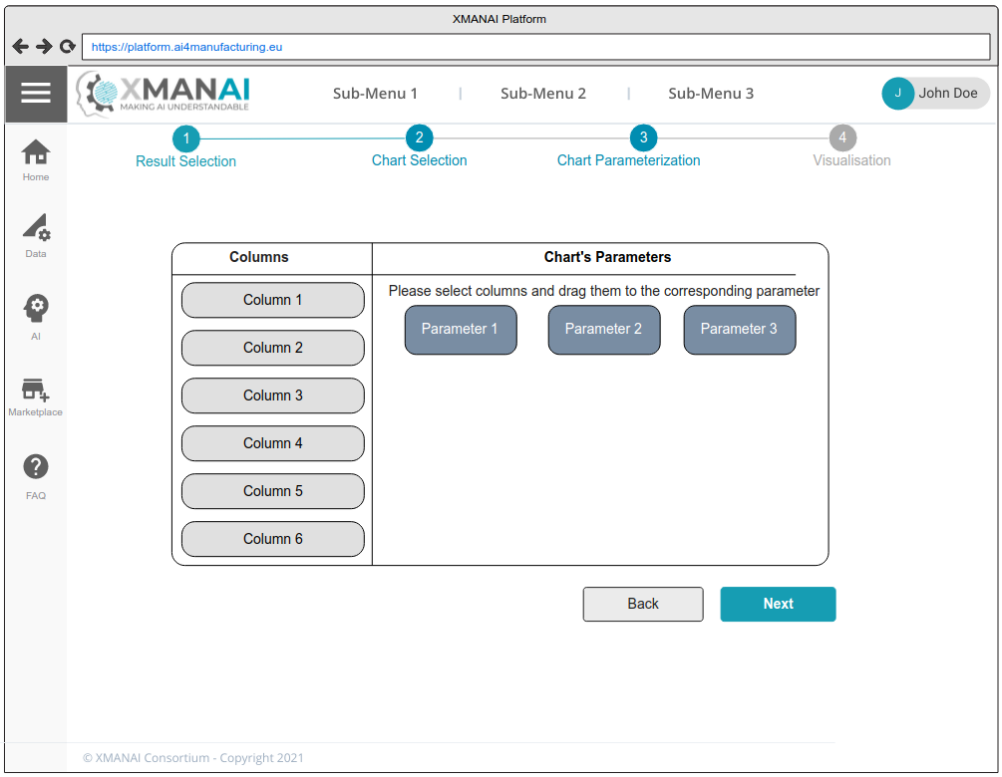

Figure 4-23: RVE - Chart Parameterisation

Finally, the chart is generated and presented to the user along with a set of parameters that allow the dynamic interaction with the generated chart (Figure 4-24).

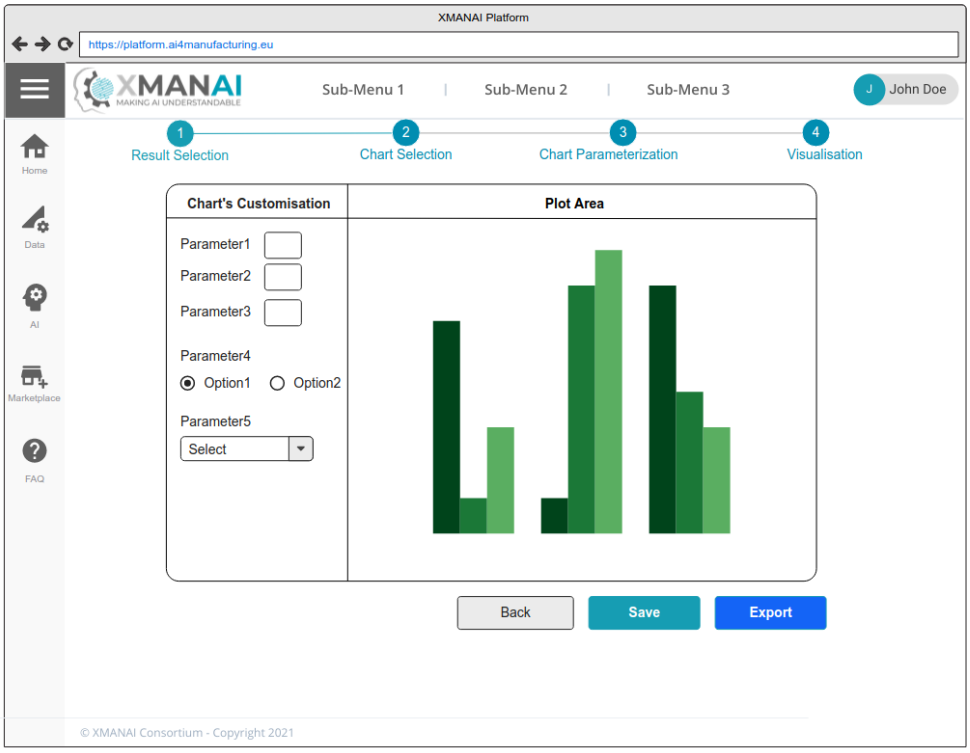

Figure 4-24: RVE - Chart Generation

4.11 XAI Model Explanations Engine

4.11.1 Overview

The XAI Models Explanations Engine (XMXE) is the component responsible for managing the explanatory information that is generated by the explainability tools along with the predictions of ML/DL models. The XMXE component is mainly configured during the training or inference configuration session of the XMEE component by the user, which decides a specific set of parameters (such as the type of explainability tool that will run along the predictive model), and included as an independent block during the execution of a pipeline within XMANAI.

The XMXE component is intended for both business users and data scientists, and therefore it will have to adapt their different needs:

- The Data Scientists using the XMANAI platform will need to design and fine tune ML/DL models to achieve the most accurate prediction results on an industrial scenario. This is usually performed through an iterative process where the data scientist studies the performance of the model with regard to the training dataset and reinforces the learning of difficult cases that were not learned properly in previous iterations. In this sense, achieving certain transparency of the designed model, in the form of explainability and understandability hints during the training phase, will assist the data scientists to make better model design decisions.
- The Business Users using the XMANAI platform will run trained ML/DL to detect, solve and optimize specific industrial processes. In this sense, having additional explanatory information regarding the predictions will help them to detect and understand the influence of the inputs on a specific prediction, leading to the discovery of potential root causes. This information, along with their expert knowledge will eventually help them to make better operational decisions.

Next, we describe the main functionalities of the XAI Models Explanations Engine are the following:

- **XMXE.1**. **Selection from different explainability techniques**. For each model, there is a list of explainability techniques that can be applied to it. This functionality will take care of listing these explainability methods (retrieving this information from the XAI Model Catalogue) once a model is chosen. This list can be filtered by end-user profile into data scientist methods, business user methods or all of them.
- **XMXE.2**. **Visualize explainability technique metadata**. For a specific explainability tool, its associated metadata will be displayed to help the users to know whether this tool is suitable to meet their explanatory needs. This metadata will include information on the type of explanation it will provide. Thus, it can provide explanations about the behaviour of the model (e.g. in decision trees) or explanations about the predictions made by the model, both local and global explanations.
- **XMXE.3**. **Run the explanation technique when it is needed**. This functionality will generate suitable explanations and store metadata on how these explanations should be displayed.

The output of this component will be the explanations per se. These explanations will be presented in different formats depending on the algorithms applied. Thus, it may generate the explanation as:

- Charts on the relevance or influence of input features on the outcome. These can be presented as bar or line charts.
- Visualization of pixels or regions that influence the system output.
- Tables for counterfactual explanations.

The main dependencies of the XMXE are:

- XAI Model Engineering Engine
- Data Preparation Engine.
- Pipeline Designer.
- Experiment Tracking Engine.
- Pipeline Serving & Monitoring Engine.
- Results Visualization Engine.

4.11.2 Technology

Although the main technologies of the XAI Model Explanations Engine component have not yet been selected, the possible technologies to be used in this component include:

- Alibi Explain - https://github.com/SeldonIO/alibi
- SHAP implementation - https://github.com/slundberg/shap
- LIME method implementation - https://github.com/marcotcr/lime
- GraphLIME - https://github.com/WilliamCCHuang/GraphLIME
- Captum - https://github.com/pytorch/captum
- DeepExplain - https://github.com/marcoancona/DeepExplain
- What-if-tool - https://pair-code.github.io/what-if-tool/

All the above tools are presented in D1.1, apart from the **GraphLIME** (https://github.com/WilliamCCHuang/GraphLIME under MIT license). This tool is an adaptation of LIME for GNNs. It employs the nonlinear interpretable model Hilbert-Schmit Independence Criterion (HSIC) Lasso. This implementation is based on PyTorch Geometric.

4.11.3 Mockups

This section provides mockups of the main functionalities that will be offered through the XMXE user interface.

In the “Explanation Model Selection & Configuration” screen, depicted in Figure 4-25, the user will be able to select a the explainability type (post-hoc/surrogate) and then the explainability tool/model. In the case of the user selecting a surrogate model, a window will be appear to set the values of the hyperparameters of the model.

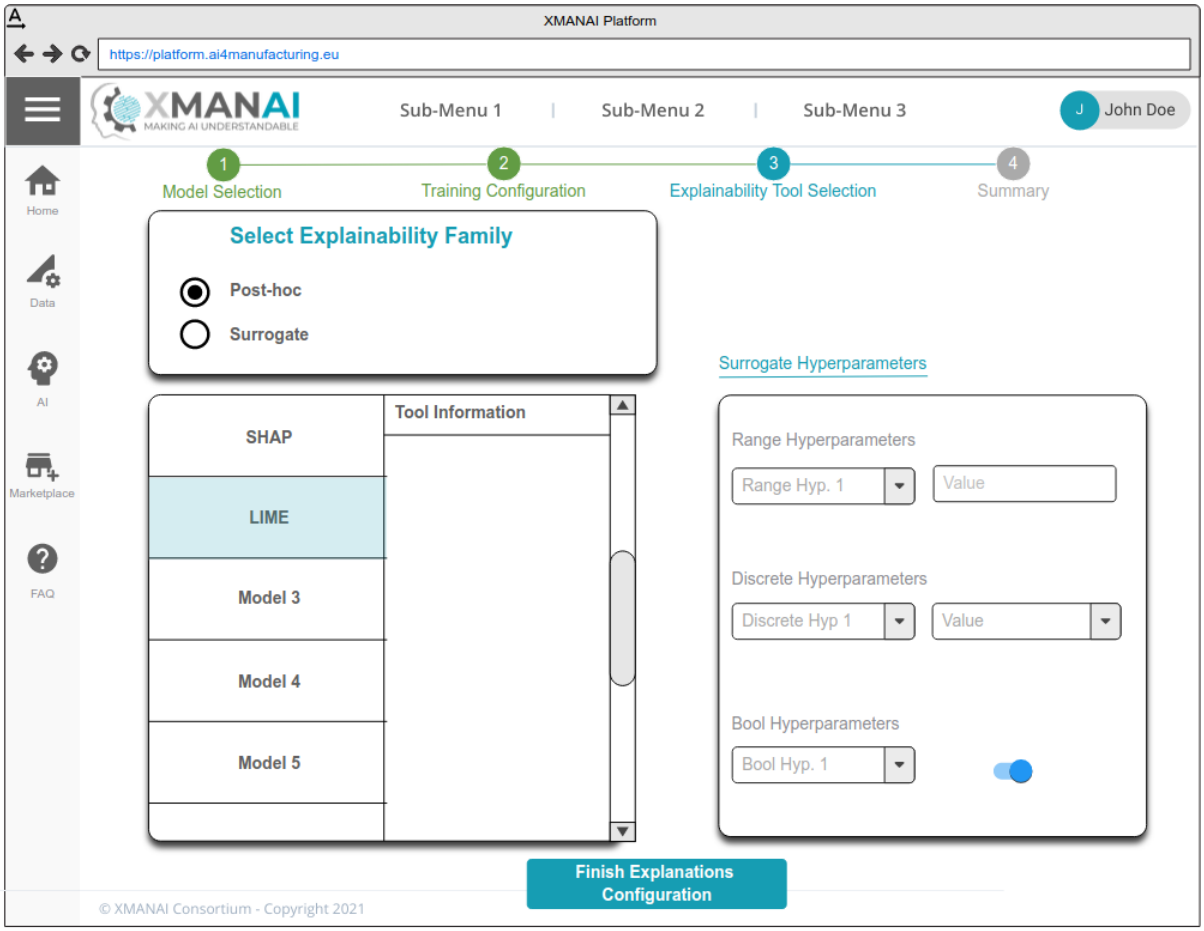

Figure 4-25: XMEE – Explanation Model Selection & Configuration

Depending on the selected model and its configuration, which in turn also depend on the user intended to request and view the respective explanation, appropriate explanations in diverse formats (e.g. visualisation, text) will be generated. Figure 4-26 shows an indicative example of an explanation provided through a SHAP visualisation.

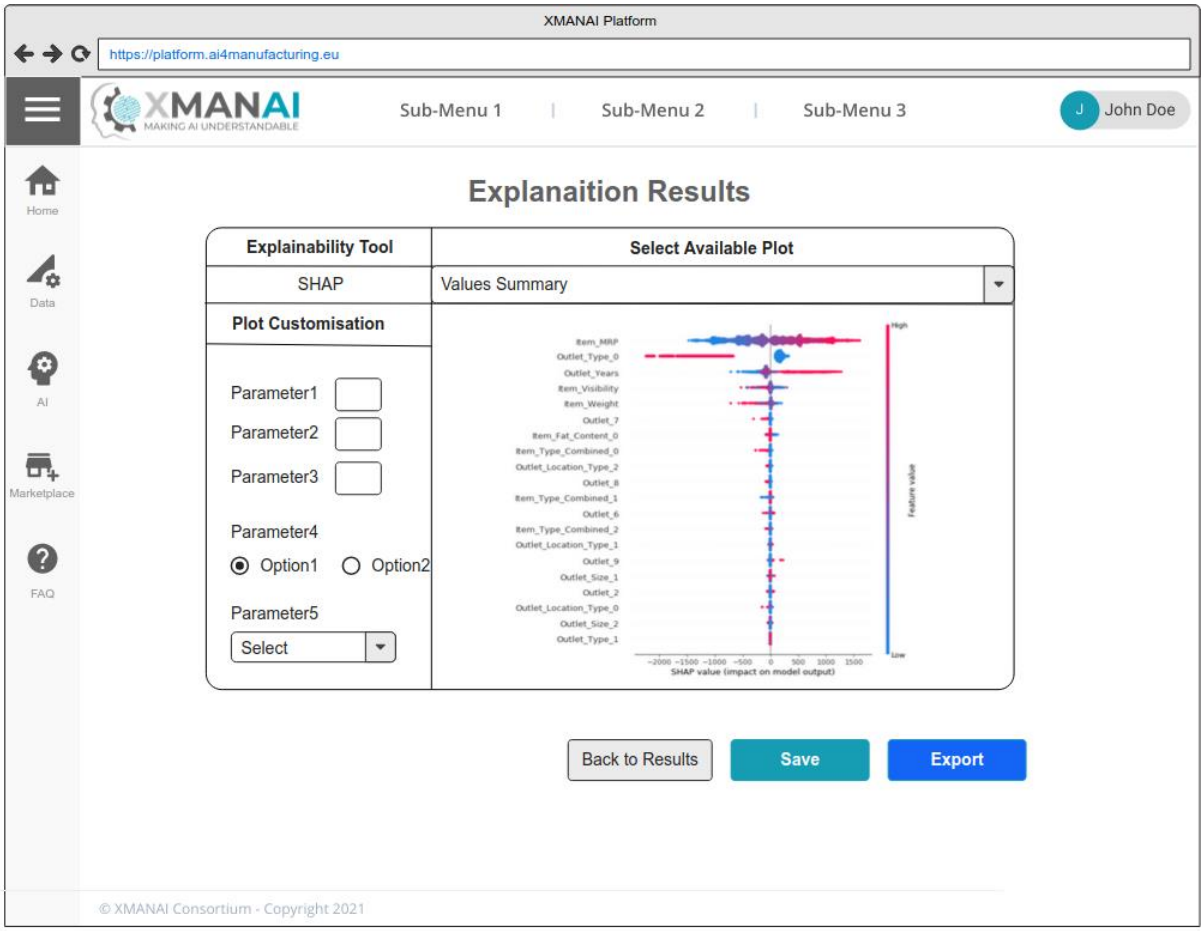

Figure 4-26: XMEE – Indicative Explanation Visualisation

4.12 Execution & Orchestration Engine

4.12.1Overview

The Execution & Orchestration Engine (EOE) component is responsible for the execution of the fully configured XAI Pipelines created by the XAI Pipeline Manager beforehand. As explained in section 4.1, this component has a one to one mapping to the respective component of the WP5 high-level architecture. Due to the nature of the Execution & Orchestration Engine, no further refinement in its features can be provided in this phase. To avoid duplication, more information on the component's scope and features can be found in D5.1.

The following section offers some information regarding the technologies that will be leveraged for the development of the component, which are not provided in D5.1.

4.12.2Technology

The EOE component will consider building on the following tools:

- *Apache Airflow (https://airflow.apache.org/)*: an open source Python-based platform to programmaticlally author, schedule and orchestrate workflows through a robust and modern web interface.
- *Apache StreamPipes (https://streampipes.apache.org/)*: is an industrial IoT toolbox that enables users to connect, analyze and explore real-time data streams.

The first component is used in scenarios for which a batch execution is required (e.g., when the involved data sources consist of stable long-term memory instances such as a Database), while the second component is used for real-time use cases in which data streams are involved (e.g., activities related to sensor logging). This division allows the EOE component to adapt the execution environment to the use case it is presented with and to execute the requested pipeline architecture in the appropriate way.

4.12.3 Mockups

This component does not offer a User Interface.

5 Conclusions and Next Steps

The current deliverable D3.1 “AI Bundles Methods and System Designs” documented the results that have been achieved during the first iteration of activities performed within all WP3 tasks towards the realisation of the XMANAI Artificial Intelligence bundles and algorithms’ lifecycle management. The results presented in the current deliverable and the activities through which they were achieved, built upon the insights presented in D1.1 “State of the Art Review in XMANAI Research Domains”, D1.2 “XMANAI Concept Detailing, Initial Requirements, Usage Scenarios and Draft MVP” and (indirectly) D6.1 “Demonstrators Requirements” while being developed in parallel with D5.1 “System Architecture, Bundles Placement Plan and APIs Design” and D2.1 “Asset Management Bundles Methods and System Designs”.

In this context the deliverable first presented insights gained through the state of the art analysis of the areas that were identified as the drivers of XMANAI AI bundles development and of the way the complete AI algorithm lifecycle will be handled. The analysis spans in two directions: (1) the methods, processes and practices of the aforementioned areas, mainly focusing on providing the scientific and theoretical foundations for the AI bundles, and (2) the technical assets, i.e. tools and technologies, that can be either directly leveraged within the WP3 implementation activities or indirectly in order to gain a deeper understanding of prevalent technologies, their strengths, limitations and faced challenges, and use these insights to properly design the WP3 architecture and anticipate issues that shall eventually come up during the WP3 implementation activities.

In this context and building upon the insights gained through D1.1, the current deliverable explored mainly the following areas (although additional ones were explored when needed to complement the view, either from a methodological or technical perspective):

- **Data manipulation processes and technologies in the context of AI pipelines**, including any data preprocessing task (curation, scaling, feature extraction, etc.) needed to bring the raw input data to a format appropriate for insights extraction, through visualisations and/or machine learning models. Data versioning technologies were also presented.
- **Visualisation and visual exploration processes, spanning data and model visualisations**, were studied to gain insights on how different functionalities can be leveraged across the various steps of AI pipelines development and monitoring, depending on the problem at hand and the stakeholder (data scientist, data engineer, business user) who creates or requires the respective visualisation.
- **MLOps (Machine Learning Operations)**, which offer good practices for building, deploying and managing Machine Learning (ML) systems, taking into consideration their differences from traditional (non-ML) development projects. The MLOps lifecycle and principles were studied and presented, along with the way they complement, automate and support the data scientists, data engineers but also business users throughout the numerous diverse steps of AI pipelines. State of the art tools that enable or facilitate the application of MLOps principles, including feature stores, experiment trackers and ML automation systems were explored.

Identified advantages, potential benefits or limitations and challenges associated with each of the studied domains (processes and technologies) were highlighted to assist in the XMANAI design decisions. Additionally, six broader considerations summarizing important outcomes of the landscape analysis were discussed and were later on leveraged in the XMANAI AI bundles design.

The deliverable then presented the activities related to the design and development of the XMANAI data model, which will be leveraged across all steps of the AI pipelines, starting from data ingestion (which is covered in D2.1) to enhanced data monitoring and increased data explainability in production deployments. The methodology that was followed to elicit concrete needs for and insights on knowledge representation in the manufacturing domain in general and in particular for the project’s demonstrators, was then presented. In particular, the deliverable described how

demonstrators' data, as well as a broader study performed on existing models and standards of the manufacturing domain, helped in the development of the first release of the XMANAI data model, which defines 40 concepts, almost 200 properties and more than 110 relationships among concepts. Furthermore, the foreseen model extension and maintenance functionalities were presented which will guarantee that new needs will be possible to be accommodated and the XMANAI data model will evolve and grow to support the data representation needs of the manufacturing domain.

The deliverable finally presented the WP3 architecture and its connection to the more high-level WP5 architecture that is documented in D5.1, presented the WP3 sub-components and explained what each brings in the realisation of the XMANAI collaborative environment for managing the lifecycle of XAI pipelines aiming to bring new insights in the manufacturing decision making and operations. The presented architecture comprises **11 components**, namely: (1) the Knowledge Graph Manager, (2) the Data Preparation Engine, (3) the XAI Model Engineering Engine, (4) the XAI Model Guard, (5) the Interactive Data Exploration and Experimentation Tool, (6) the Pipeline Designer, (7) the Experiment Tracking Engine, (8) the Pipeline Serving and Monitoring Engine, (9) the Results Visualisation Engine, (10) the XAI Models Explanation Engine, (11) the Execution and Orchestration Engine.

For each of the components, the deliverable provided the main functionalities and supported methods, their positioning within the WP3 architecture and their interactions with other components, as well as the technologies considered for their implementation. Indicative low-fidelity mockups of the core screens of the components that offer a user interface were finally presented.

The purpose of this deliverable was to provide the draft design of the XMANAI AI bundles, in particular the processes that will be supported, the design of the components that will be implemented to support these processes, the foreseen high-level user interfaces and the rationale underpinning the relevant design decisions. The next step will be the implementation of the aforementioned components and their functionalities, towards the realisation of the XMANAI AI bundles, whose first release is due on M18 and will be documented in Deliverable D3.2 "XMANAI AI Bundles – First Release".

References

Agrawal, S. and Agrawal, J., 2015. Survey on anomaly detection using data mining techniques. Procedia Computer Science, 60, pp.708-713.

Ali, M.M., Rai, R., Otte, J.N. and Smith, B., 2019. A product life cycle ontology for additive manufacturing. Computers in Industry, 105, pp.191-203.

Alla, S. and Adari, S.K.,2021, Beginning MLOps with MLFlow, Apress, Berkeley, CA

Andres, B., Poler, R. and Sanchis, R., 2021. A data model for collaborative manufacturing environments. Computers in Industry, 126, p.103398.

Apley, D. W. a. J. Z., 2017. Visualizing the effects of predictor variables in black box supervised learning models. s.l., s.n.

Arrieta, 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities. s.l., s.n.

Arrikto (2021), MLOps Explained, Available at: https://www.arrikto.com/mlops-explained/ (Accessed: 19 November 2021)

Available at: https://towardsdatascience.com/ml-ops-challenges-solutions-and-future-trends-d2e59b74dc6b

Been Kim, R. K. a. S. K., 2016. Examples are not Enough, Learn to Criticize. Criticism for Interpretability.. s.l., s.n.

Berti-Equille, L., 2019, May. Learn2clean: Optimizing the sequence of tasks for web data preparation. In The World Wide Web Conference (pp. 2580-2586).

Breck, E., Cai, S., Nielsen, E., Salib, M. and Sculley, D., 2017, December. The ML test score: A rubric for ML production readiness and technical debt reduction. In 2017 IEEE International Conference on Big Data (Big Data) (pp. 1123-1132). IEEE.

Brownlee, J., 2020. Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python. Machine Learning Mastery.

Brutzman, D. and Flotyński, J., 2020, November. X3D Ontology for Querying 3D Models on the Semantic Web. In The 25th International Conference on 3D Web Technology (pp. 1-6).

Cheatham, M. & Cox, M., 2005. AI workflow management in a collaborative environment. s.l., IEEE, pp. 160-166.

Google (2021) MLOps: Continuous delivery and automation pipelines in machine learning. Available at: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning.

Giustozzi, F., Saunier, J. and Zanni-Merk, C., 2018. Context modeling for industry 4.0: An ontology-based proposal. Procedia Computer Science, 126, pp.675-684.

Haviv, Y., 2020. ML Ops Challenges, Solutions and Future Trends. [Online]

He, L. and Jiang, P., 2019. Manufacturing knowledge graph: a connectivism to answer production problems query with knowledge reuse. IEEE Access, 7, pp.101231-101244.

Hsu, C.W., Chang, C.C. and Lin, C.J., 2003. A practical guide to support vector classification.

Ilmoi, (2019). Poisoning attacks on Machine Learning. Available at: https://towardsdatascience.com/poisoning-attacks-on-machine-learning-1ff247c254db (Accessed: 19 November 2021)

Izenman, A., 2008. Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning. s.l.:Springer.

Kaggle (2021). A Guide to Handling Missing values in Python. Available at: https://www.kaggle.com/parulpandey/a-guide-to-handling-missing-values-in-python (Accessed: 19 November 2021)

Klaise, J., Van Looveren, A., Cox, C., Vacanti, G. and Coca, A., 2020. Monitoring and explainability of models in production. arXiv preprint arXiv:2007.06299.

L.J.P. van der Maaten, G. H., 2008. Visualizing High-Dimensional Data Using t-SNE. s.l., Journal of Machine Learning Research.

Landset, S., Khoshgoftaar, T.M., Richter, A.N. and Hasanin, T., 2015. A survey of open source tools for machine learning with big data in the Hadoop ecosystem. Journal of Big Data, 2(1), pp.1-36.

Leila Arras, G. M. K.-R. M. W. S., 2017. Explaining Recurrent Neural Network Predictions in Sentiment Analysis. s.l., s.n.

Lemaignan, S., Siadat, A., Dantan, J.Y. and Semenenko, A., 2006, June. MASON: A proposal for an ontology of manufacturing domain. In IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06) (pp. 195-200). IEEE.

Lundberg, S. M. a. L. S.-I., 2017. A Unified Approach to Interpreting Model Predictions. s.l., Curran Associates, Inc..

Marciszewski J. (2018) Using R to Visualize Google BigQuery Export Schemas. Available at: https://www.cardinalpath.com/blog/using-r-visualize-google-bigquery-export-schemas (Accessed: 19 November 2021)

McInnes, L. H. J., 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. s.l., s.n.

Molnar C. (2021). Interpretable machine learning. Available at: https://christophm.github.io/interpretable-ml-book/index.html (Accessed: 19 November 2021)

Muralidhar, N., Muthiah, S., Butler, P., Jain, M., Yu, Y., Burne, K., Li, W., Jones, D., Arunachalam, P., McCormick, H.S. and Ramakrishnan, N., 2021. Using AntiPatterns to avoid MLOps Mistakes. arXiv preprint arXiv:2107.00079.

Onose, E. (2021), Explainability and Auditability in ML: Definitions, Techniques, and Tools, Available at: https://neptune.ai/blog/explainability-auditability-ml-definitions-techniques-tools (Accessed: 19 November 2021)

Panetto, H., Dassisti, M. and Tursi, A., 2012. ONTO-PDM: Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment. Advanced Engineering Informatics, 26(2), pp.334-348.

Rachuri, S., Han, Y.H., Feng, S.C., Wang, F., Sriram, R.D., Lyons, K.W. and Roy, U., 2003, July. Object-oriented representation of electro-mechanical assemblies using UML. In Proceedings of the IEEE International Symposium onAssembly and Task Planning, 2003. (pp. 228-234). IEEE.

Salama, K. Kazmierczak, K. Schut, D. S.,2021. Practitioners guide to MLOps: A framework for continuous delivery and automation of machine learning, Available at: https://services.google.com/fh/files/misc/practitioners_guide_to_mlops_whitepaper.pdf (Accessed: 19 November 2021)

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V. and Young, M., 2014. Machine learning: The high interest credit card of technical debt.

Šormaz, D. and Sarkar, A., 2019. SIMPM–Upper-level ontology for manufacturing process plan network generation. Robotics and Computer-Integrated Manufacturing, 55, pp.183-198.

Terkaj, W., Pedrielli, G. and Sacco, M., 2012, July. Virtual factory data model. In Proceedings of the workshop on ontology and semantic web for manufacturing, Graz, Austria (pp. 29-43).

Thudumu, S., Branch, P., Jin, J. and Singh, J.J., 2020. A comprehensive survey of anomaly detection techniques for high dimensional big data. Journal of Big Data, 7(1), pp.1-30.

TowardsDataScience (2021) ML Ops Challenges, Solutions and Future Trends. Available at: https://towardsdatascience.com/ml-ops-challenges-solutions-and-future-trends-d2e59b74dc6b . (Accessed: 19 November 2021)

Turck, M., 2021 Red Hot: The 2021 Machine Learning, AI and Data (MAD) Landscape, Available at: https://mattturck.com/data2021/ (Accessed: 19 November 2021)

Usman, Z., Young, R.I.M., Chungoora, N., Palmer, C., Case, K. and Harding, J., 2011, March. A manufacturing core concepts ontology for product lifecycle interoperability. In International IFIP Working Conference on Enterprise Interoperability (pp. 5-18). Springer, Berlin, Heidelberg.

Valohai. 2021, Practical MLOPS: How to get ready for Production Models

Venkatesh, B. and Anuradha, J., 2019. A review of feature selection and its methods. Cybernetics and Information Technologies, 19(1), pp.3-26.

Visengeriyeva L., Kammer A., Bär I., Kniesz A., and Plöd M. (2021) MLOps Principles. Available at: https://ml-ops.org/content/mlops-principles (Accessed: 19 November 2021)

XMANAI Deliverable D1.1 “State-of-the Art Review in XMANAI Research Domains”, 2021.

XMANAI Deliverable D1.2 “XMANAI Concept Detailing, Initial Requirements, Usage Scenarios and Draft MVP”, 2021.

XMANAI Deliverable D2.1 “Asset Management Bundles Methods and System Designs”, 2021.

XMANAI Deliverable D5.1 “System Architecture, Bundles Placement Plan and APIs Design [M12] The first version of the system architecture, the bundles design, the APIs”, 2021.

XMANAI Deliverable D6.1 “Demonstrators Requirements”, 2021.

List of Acronyms/Abbreviations

Acronym/ Abbreviation	Description
AI	Artificial Intelligence
AMO	Additive Manufacturing Ontology
ANOVA	Analysis Of Variance
API	Application Programming Interface
AR	Augmented Reality
AWS	Amazon Web Services
BSD	Berkeley Source Distribution
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CD	Continuous Delivery
CI	Continuous Integration
CLI	Command-line Interface
CNTK	Microsoft Cognitive Toolkit
CPU	Central Processing Unit
CRUD	Create Read Upldate Delete
CT	Continuous Training
DAG	Directed Acyclic Graph
DL	Deep Learning
DoA	Description of Action
DPE	Data Preparation Engine
DVC	Data Version Control
EDA	Exploratory Data Analysis
EDS	Electronic Data Sheet
EKS	Elastic Kubernetes Service
EOE	Execution & Orchestration Engine
ERP	Enterprise Resource Planning
ETE	Experiment Tracking Engine
FOL	First Order Logic
GCS	Google Cloud Services
GNN	Graph Neural Networks

Acronym/ Abbreviation	Description
GPH	Global Product Hierarchy
GPU	Graphics Processing Unit
GRAD-CAM	Gradient-weighted Class Activation Mapping
GRPC	Google Remote Procedure Call
HDFS	Hadoop Distributed File System
HTML	Hypertext Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identity
IDEE	Interactive Data Exploration & Experimentation Tool
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
JPH	Jobs per Hour
JSON-LD	JavaScript Object Notation for Linked Data
KGE	Knowledge Graph Embedding
KGM	Knowledge Graph Manager
k-NN	k Nearest Neighbours
KPI	Key Performance Indicators
LDA	Linear Dimensionality Reduction
LIBSVM	Library for Support Vector Machines
LIME	Local Interpretable Model-agnostic Explanations
LOCF	Last Observation Carried Forward
MDS	Multidimensional Scaling
MG	Model Guard
MK	Manufacturing Knowledge
ML	Machine Learning
MPP	Manufacturing Process Planning
MR	Mixed Reality
MVP	Minimum Viable Product
NLP	Natural Language Processing
NOCB	Next Observation Carried Backward

Acronym/ Abbreviation	Description
OGC	Open Geospatial Consortium
OMG	Object Management Group
OWL	Web Ontology Language
PCA	Principal Component Analysis
PD	Pipeline Designer
PDES	Product Data Exchange Specification
PDM	Product Data Management
PLC	Product LifeCycle
PP	Production Problems
PSF	LICENSE
PSME	Pipeline Serving & Monitoring Engine
RAD	Rapid Application Development
RDB	Relational Database
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	REpresentational State Transfer
RML	RDF Mapping language
RVE	Results Visualisation Engine
SDAI	standard Data Access Interface
SDK	Software Development Kit
SFTP	Secure File Transfer Protocol
SHAP	SHapley Additive exPlanations
SIMPM	Semantically Integrated Manufacturing Planning Model
SKOS	Simple Knowledge Organization System
SNN	Semantic Sensor Ontology
SOM	Self-Organizing Map
SPARQL	Simple Protocol and RDF Query Language
SQL	Structured Query Language
SSH	Secure Shell
STEP	STandard for the Exchange of Product model data
SVD	Singular Value Decomposition

Acronym/ Abbreviation	Description
SVG	Scalable Vector Graphics
SWRL	Semantic Web Rule Language
TCP	Transmission Control Protocol
TF	Tensorflow
TPE	Tree-Structured Parzen Estimator
t-SNE	t-distributed stochastic neighbor embedding
UI	User Interface
UMAP	Uniform Manifold Approximation and Projection
VFDM	irtual Factory Data Model
VR	Virtual Reality
WP	Work Package
XAI	Explainable Ariftificial Intelligence
XMEE	XAI Model Engineering Engine
XML	eXtensible Markup Language
XMXE	XAI Models Explanations Engine
YARN	Yet Another Resource Negotiator