UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dottorato di ricerca in
Computer and Data Science for Technological and Social Innovation
Ciclo XXXVIII

(titolo tesi)
Multi-Agent Digital Twin for Monitoring, Prediction and Actuation in Water Quality Control

Candidato Alireza Rahimi

Relatore (Tutor): Prof. Giacomo Cabri

Eventuale Correlatore (Co-Tutor): Prof. Marko Bertogna

Coordinatore del Corso di Dottorato: Prof. Andrea Marongiu

Tesi di dottorato finanziata dall'Unione europea- Next Generation EU, Missione 4, componente 2 "Dalla Ricerca all'Impresa" - Investimento 3.3 "Introduzione di dottorati innovativi che rispondono ai fabbisogni di innovazione delle imprese e promuovono l'assunzione dei ricercatori dalle imprese".

Preface

This PhD dissertation is the result of my PhD program in PhD Program in Computer and Data Science for Technological and Social Innovation (CDS-TSI) at the department of Physics, Informatics and Mathematics (FIM), in the University of Modena and Reggio Emilia (UNIMORE), with the topic of Digital Twins. In this dissertation I have aggregated the results of many studies, observations and software usage and code development practices. I found the Digital Twins to be a very novel field of informatics technology, therefore I tried my best to seek an innovative approach of realizing my viewpoint on it, which yields innovation from design to implementation. There are three chapters in this work, first chapter includes the result of my studying of many topic papers, that resulted in 3 presented and published conference papers and unpublished work; These studies helped me obtain a benign vision of what Digital Twins are supposed to present, and therefore they provided me with the base knowledge of the topic. However, my emphasis on novelty conducted my novel works on a direction that yielded a completely novel realization of the Digital Twins, from definition, to implementation, under the architecture name I contrived as "VACCSMEDUS". About the substrate field which is the water 4.0 including water quality and SCADA systems of treatment plants, it was selected because of my inclinations towards sustainability and environmental matters, also my background in Physics was a decisive factor in such selection. I would like to initially sincerely thank very much my supervisor Professor Giacomo Cabri whose wise and kindly guidance and patience conducted my research activities in the best direction and helped me overcome challenges throughout the program. Further thanks will be presented in the acknowledgment section.

—Alireza Rahimi

Abstract

Digital Twins (DTs) have emerged as a foundational cyber-physical technology for Industry 4.0, promising enhanced simulation, monitoring, and control across complex systems. While their potential is widely recognized in fields like Model-Based Systems Engineering (MBSE), Product Lifecycle Management (PLM), Smart Manufacturing, and critical infrastructure such as power generation and water management, a significant gap persists between conceptual DT frameworks and their practical, interpretable, and resilient real-time implementation—particularly for continuous environmental monitoring.

This dissertation bridges this gap through a two-part structure and a transitional chapter in between. First, it establishes a comprehensive interdisciplinary foundation via a series of integrative surveys. These surveys map the incorporation of DTs into MBSE, PLM, predictive maintenance, and resilience engineering; their exploitation in Smart Manufacturing; their integration across thermal, nuclear, and renewable power plants; and their emerging role in the Water 4.0 paradigm. This synthesis reveals a common need for DTs that are not only predictive but also explainable, adaptive, and operational in real-time under constraints of irregular data. The second chapter is our breakthrough to into implementing the explainable AI in the field of water quality, an initial comparative study of machine learning algorithms for water quality prediction, evaluated using SHAP-based explainability, identified Random Forest as optimally balancing accuracy and interpretability. this work paved way for the third chapter which contains the main novel work of the thesis. The core original contribution of this research addresses these needs by pioneering a novel, complete-loop, end-to-end predictive Digital Twin architecture for real-time water quality monitoring and control. The research progresses from model selection to full system integration. The dissertation introduces VACCSMEDUS—a multi-agent Digital Twin architecture that embodies a paradigm shift from reactive monitoring to proactive, "Real-Time-Plus" monitoring and intelligence, in the sense that it provides predictive values and analysis of water quality sensors and proactive actuation of actuators, the complete smart loop of the Digital Twin, applicable also in other substrate fields given the field definitions and adjustments.

The VACCSMEDUS architecture's novelty is tripartite: (1) It introduces the Recent Real-Time Window (RRTW) concept, a self-calibrating temporal framework that fuses past, present, and predicted future data into a coherent "Plug-and-Play prediction" view, eliminating dependence on historical data archives. (2) It features a SHAP-MAE Engine that performs continuous, feature-specific model switching among a curated ensemble of regressors, optimizing both predictive performance and local explainability at every time cycle. (3) It is realized through a resilient multi-agent system (Collector, Counselor, Visualizer, Actor, Director, Utilitarian, Super-Director) that forms a closed cognitive loop for data acquisition, interpretable prediction, visualization, and autonomous control actions and actuation.

This work culminates in a functional Digital Twin that transforms supervisory dashboards into explainable, predictive partners. The architecture is designed for seamless integration into legacy SCADA systems, acting as an MTU plugin, alternative, or even a standalone control system for remote deployments. By ensuring predictive, interpretable, and confident management of water quality within health-based thresholds, this research provides a validated blueprint for next-generation cyber-physical systems, advancing the frontier of intelligent, resilient, and trustworthy water quality monitoring and treatment.

Contents

Introduction

The ongoing digital transformation of industry, often termed the Fourth Industrial Revolution, is fundamentally characterized by the cyber-physical integration of systems, where the virtual and physical worlds converge to enable unprecedented levels of automation, insight, and efficiency. At the forefront of this transformation is the Digital Twin, a dynamic virtual representation of a physical entity that is synchronized through a continuous flow of data. Born from the need for better product lifecycle management, the Digital Twin concept has evolved from a conceptual model into a powerful technological paradigm, fueled by advancements in the Internet of Things, artificial intelligence, big data analytics and sensors technologies. Its application now spans a vast spectrum of critical domains, from advanced manufacturing and energy production to the management of essential urban infrastructure like water systems. This proliferation underscores a universal industrial need to move beyond passive monitoring towards proactive, predictive, and fully intelligent system management.

The central aim of this thesis is to explore and advance the application of Digital Twin technology, with a specific focus on its role in creating intelligent, explainable, and resilient systems for environmental monitoring, in particular, water quality monitoring and treatment systems. The work is driven by the observation that while the theoretical benefits of Digital Twins are widely acknowledged—enabling predictive maintenance, enhancing resilience, and optimizing lifecycles—their practical implementation, particularly in complex, legacy-critical infrastructures like water networks, faces significant challenges. These challenges include integrating predictive intelligence in real-time, ensuring the trustworthiness of AI-driven decisions through explainability, and architecting systems that are both adaptable and ready for immediate deployment without reliance on extensive historical data, characteristic of our Digital Twin we have nicknamed as the "Plug-and-play" predictive mechanism. This research gap forms the core problem this thesis seeks to address.

To establish a solid foundation, the first chapter of this thesis presents a comprehensive review of the Digital Twin landscape. It systematically examines the integration of Digital Twins with established engineering disciplines such as Model-Based Systems Engineering, Product Life-cycle Management, and resilience engineering, highlighting their synergistic potential. It then surveys their transformative exploitation in Smart Manufacturing, their critical role in modernizing power generation as cyber-physical systems, and their emerging imperative within the Water 4.0 framework for sustainable water management. This review consolidates the state-of-the-art and clearly identifies the trajectory towards more active, predictive, and domain-specific Digital Twin implementations, which directly motivates the original research presented in the subsequent chapters.

The second chapter transitions from review to applied research, tackling the fundamental challenge of trust in AI-driven environmental prediction. It presents a comparative study of machine learning algorithms for predicting a Water Quality Index using real groundwater data. The chapter's primary contribution lies not only in evaluating predictive accuracy but in rigorously applying the SHAP frame-

work to provide consistent, model-agnostic explainability. The results delineate a clear trade-off between model complexity and interpretability, establishing Random Forest as a high-performing, yet explainable model, and demonstrating methods to open the "black box" of more opaque models like LSTM. This work provides the essential methodological groundwork on explainable AI that is crucial for operational trust, which is then scaled and automated in the final chapter.

The third and culminating chapter of this thesis introduces the novel VACCSMEDUS architecture, a complete and operational multi-agent Digital Twin system designed for real-time water quality supervision. This work represents the core novel contribution, integrating the concepts explored previously into a unified, proactive cyber-physical system. Its innovation is threefold. First, it proposes the Recent Real-Time Window paradigm, a new temporal framework that enables predictive, plug-and-play operation without dependency on historical data, synchronizing the Digital Twin with the immediate past and near future. Second, it introduces a SHAP-MAE Engine that automates explainable model selection in real-time, creating a self-correcting predictive core that continuously balances accuracy and interpretability. Third, it realizes these mechanisms within a robust multi-agent architecture named VACCSMEDUS named after the initials of its agents and AI engine (Visualizer, Actor, Collector, Counselor, SHAP-MAE Engine, Director, Utilitarian, Super-Director) that closes the loop from data acquisition to explainable prediction to visualization and potential actuation. Validated on a full suite of water quality features, this architecture demonstrates a decisive step from theoretical Digital Twin concepts towards a deployable, intelligent, and self-aware system for critical infrastructure. The architecture is also implementable on other substrate fields because it has a modular-design and embodies the general-purpose definitions of the Digital Twins with the complete task lists.

In summary, the thesis structure progresses from a broad scholarly foundation to focused methodological development, culminating in an integrated architectural innovation. Chapter 1 synthesizes the existing knowledge across key industrial and infrastructural domains. Chapter 2 is a transitional part and establishes and validates the explainable AI methodology critical for in the field of water quality prediction. Finally, Chapter 3 integrates these elements into the novel VACCSMEDUS Digital Twin, presenting a holistic solution to the challenges of real-time, explainable, and actionable environmental intelligence.

1. Reviews and surveys

Research on an emerging topic such as the Digital Twins demands exploration of the background of this new technology, and extensive studies on the existing notable works done in the field, so that the researcher acquires sufficient knowledge and insight necessary to conduct their own novel approach. Therefore, In the beginning of our research it was necessary to delve into the Model-Based Systems Engineering (MBSE) which the substrate field in which the concept of the Digital Twins is born, in particular because of its considerations of product life-cycle management and related topics, including smart manufacturing, hence the first part of this chapter presents the background of the Digital Twins concept and its formation and usages in the context of MBSE. Consequently, in the second part of this chapter, we present a a concise summary of the most relevant works to our generic work in the implementation of the Digital Twins in water 4.0, While our studies and observations were not limited to this field, but were expanded to almost all topics and substrate fields in which the Digital Twins were created.

1.1 Digital Twin background

Within the principal requirements of the Fourth Industrial Revolution or Industry 4.0 lies the interconnectivity and the integration of the virtual and physical environments. This capability is provided by the Digital Twin technology, being a real-time virtual representation of a physical entity of interest, which gets its status updated synchronously with the changes in the status and conditions of the physical system through sensors and communication means. The concept of Digital Twin was first introduced by Michael Grieves in 2002 at the University of Michigan [1], [2], and has been evolving since then with the emergence of Internet of Things (IoT), bringing new evolutionary advancements, capacities and horizons to a wide spectrum of industries such as engineering, manufacturing, construction, healthcare, agriculture, architecture, defence, oil and gas, etc. DT has the capacity of being a very powerful tool to realize the model-based systems engineering tools and techniques, and those of its focal and closely related concepts, disciplines and sub-disciplines such as product life-cycle management, predictive maintenance and resilience engineering within the context of the industries and especially manufacturing. DT can increase the efficiency of the frameworks due to it being a real-time replica of the physical twin, which can be an object, system or process. DT provides a wider range of accessibility to the components of a system and the system as a whole, rather than the digital models due to this real-time connectivity advantage, which can be bidirectionally influential through sensors and actuators.

In the following we principally aim at presenting a survey on the research done in the application of Digital Twins in systems engineering and its very focal and closely related sub-disciplines in a compact and all-inclusive manner which covers the principal concept of the DT implementation in an epistemological manner. It can serve as a concentrate of the theory and work done on the bridge between DT and MBSE, a starting point, or best defined as a “kick-off map” among these fields.

1.1.1 Digital Twin definition

As the name implies, a Digital Twin is the dynamically evolving virtual instance of a physical object, therefore there can exist Digital Twins of a product, factory, process or a business service. Considering the plethora of literature on the topic, a more appropriate definition of a Digital Twin has developed to outline the 'Different virtual representations at different stages of a products life-cycle' [2].

1.1.2 Digital twin in regards to CPS and IoT

A Cyber Physical System (CPS) is characterized by a physical asset and its Digital Twin. In contrast, a Digital Twin is limited to the digital model, not the twinning physical asset, though a Digital Twin cannot live without its twining asset in the physical space. In other words, Digital Twin represents the prerequisite for the development of a CPS [3]. IoT (Internet of Things) refers to connections between a network of physical assets through which data can flow between themselves. The connections are made possible by the secure implementation of computer networks, the Internet, and communication protocols. However, despite the connectivity, IoT does not include the idea of digital models in the cyberspace. The IoT is the infrastructure in the physical space for connecting physical assets. In fact Iot is a very important enabling technology exploited by the Digital Twins.

1.1.3 Model-Based Systems Engineering (MBSE)

MBSE is a model-centric (vs. document-based) approach providing a single point of truth which is reflected in a set of living artefacts [4]. The successful implementation of MBSE leads to lower cost, better quality and lower risk, resulted from the following benefits of the system model: early detection of errors and inconsistencies enabled by the ability perform analysis; use of modern modeling languages with clearer semantics that lower miscommunication; a single source of information ensures consistency between different stakeholders; automatic generation of up to date deliverables (e.g. documents); support of multiple views to address different stakeholders' needs using a single source of information; better management of complexity; early debugging and refinement of requirements, including behavioral ones, through simulation of state machines [4].

1.1.4 The general scheme of the application of Digital Twin in MBSE

Initially, it must be noted that DTs are not only exploitable once the physical system has been created. At the point DT has come to exist, it has already encompassed system design decisions and project errors, and there are limitations to optimization for performance and financial success. Therefore, maximum advantage must be extracted from the DT across the whole system life-cycle, i.e, having DT with optimal testing capabilities, integrated into the systems engineering framework and implemented on the design and production process. DT should encompass an instrumented testbed, that provides an environment enabling MBSE tools (e.g., system modeling and verification tools), and operational scenario simulations (e.g., discrete event simulations, agent-based simulations), to explore the behavior of virtual prototypes in a what-if simulation mode defined and controlled by the experimenter [5]. The MBSE tool suite includes system modeling methods, e.g. SysML models, Design Structure Matrix, process dependency structure matrix, probabilistic models such as Partially-Observable Markov Decision Process (POMDP),

discrete event simulation, agent-based simulation, model-based storytelling, an MBSE knowledge base that constitutes the authoritative sources of truth, and systems engineering life cycle process models. The virtual system model expands from lightweight models to full-up models. The lightweight models reflect simplified structure (e.g., simplified geometry) and simplified physics (e.g., reduced order models) to reduce computational load especially in upfront engineering activities through simulations of complex systems and system-of-systems (SoS) with high fidelity to the appropriate dimensions. These models can be shared within the organization and with the supplier network thereby increasing their understanding of the system being engineered. The DT concept also includes automated and manual processes in manufacturing environments. From the physical twin, performance, maintenance, and health data can be collected and supplied to the digital twin. This data includes operational environment characteristics, engine and battery status, and other such factors. An MBSE repository which also supports SE and data collection tools, shared by the digital twin and the physical twin, can support both entities. The MBSE models constitute the authoritative sources of truth. This configuration also ensures bidirectional communication between the digital twin and the physical twin [5].

Within the context of manufacturing from the viewpoint of systems engineering, the recent applications of DT in manufacturing has enabled the simulation, optimization and detailed visualization of the whole production system, from the manufacturing of single components to complete assembly [6]. In this regard, current studies focus at, e.g., production planning and control, maintenance, layout planning, logistics and product life-cycle [7].

1.1.5 Digital Twin application in product life-cycle management

Product life-cycle management (PLM) deals with every stage across a product's lifetime. it is an effective method for enhancing the market competitiveness of modern manufacturing industries [8]. PLM aims to integrate and manage product-related information in the whole life-cycle. It accomplishes data acquisition and management for product development, production, sales, use, maintenance, and scrapping. Thus, data sharing can be implemented throughout the enterprise's entire life-cycle to increase its market competitiveness [9] [10]. The industry will deal with the current challenges of shorter product lifecycle, highly customized products and stiff global competition [11]. There are three stages of product life cycle, beginning of life (BOL),the period between the conception of a product and its physical realization; middle of life (MOL), the duration of a product's use and maintenance, and end of life (EOL), the time during which a product is recycled or scrapped [12]. As defined by Geieves & Vickers in [2], a mirror or a "twin" is created, in a way that the virtual model mirrors (or twins) the real one, continuously throughout the entire lifecycle of the system, which contains four phases: creation, production, operation (sustainment/support), and disposal. Digital Twin has the capability of handling dynamic changes in product data and the mechanism for data sharing and collaboration at various stages. In the virtual space, data are collected, supplemented, managed, evolved, and updated. From product design to production, assembly, service, and recycling, data acquisition, unified management, and efficient supply are supported. Thus, the pressing needs of data association mapping, dynamic evolution, and real-time service in PLM can be met to realize iterative optimization and integrated product development.

One beneficial aspect of DT is collaboration. The digital twin will allow communication between customers and designers [13], but the main impact of a digital twin is performance.

It turns out that this performance aspect is more relevant during last stage of a product: use and maintenance. Among other things, it makes it possible to assess the current and future capabilities of a system during the phase [14]. The implementation of a Digital Twin no longer only concerns existing objects but also objects to be fabricated/constructed/produced. It can also be used to represent a production process or a connected factory [15].

1.1.6 Digital Twin life-cycles intertwinedness with product life-cycle

As a virtual product, the systems engineering and life-cycle concepts also apply to DT itself. By understanding the DT lifecycle in comparison to the product life-cycle, it is possible to understand the DT as a complex system and, thus to visualize the implementation steps towards a fully functional process DT in a systematic manner[**ref:SchÃijtzer2019**]. In the following, we expand the aforementioned stages of digital twin life-cycle:

- **Creation**: In the first phase, creation, the DT, starts to be conceived. The requirements are analyzed and defined, to set the characteristics, parameters and expected behaviors. Therefore, the system begins to take form in the virtual space as a DT Prototype (DTP), which is an initial virtual model of the system[**ref:SchÃijtzer2019**].
- **Production**: Once the characteristics and parameters are defined, simulations are initiated to assess the possible behaviors of the system before its physical implementation in the next phase, the production [2]. Such simulations assess the system behaviors in four categories: predicted desirable (PD), referring to behaviors the system should present according to design; predicted undesirable (PU), concerning behaviors not useful or negative to the system, but that were previously identified; unpredicted desirable (UD), referring to positive behaviors of the system that were not identified before; and unpredicted undesirable (UU), which refers to negative previously unidentified behaviors. It is desired that after running the simulations in iterative cycles, the number of UUs is significantly reduced. During the production phase, the full system is implemented. However, the flow goes opposite to the first phase. As the physical system is built with specific configurations, the need to reflect such configurations in the virtual space emerges. Hence, the system becomes an instance of the DT, creating a virtual representation of the physical system through data being sent to the virtual space. At the end of the second phase, the virtual and physical systems are functional and linked to each other [2]. Thus, the DT is implemented.
- **Operation**: In the next phase, the *operation*, also called the *support/sustain* phase, the predictions from the first phase are proved true or false, based on whether or not they are accurate enough, and limiting the UUs behaviors was utmostly performed by the development team, so that any emerging UU will be a minor problem. Since the data flow between physical and virtual systems is bidirectional, any change such as parts replacement, state change, and behaviors occur in both systems, in such as way that, while changes occur in the physical system data is captured in the virtual space and used to determine the exact configuration of each system at the moment of the change. The virtual information, on the other hand, can be used to predict performance and failures to the physical system. When aggregated, both sources of data and the resultant information can be employed in correlating specific state changes to future failures probability, that is, for predictive

maintenance [2], which represents a valued source for industry.

- **Disposal**: In the last phase, *disposal*, which is often ignored in the lifecycle according to Grieves and Vickers [2], there should be more recognition towards it, for being a source of mistake prevention for the system's next generation. Next generation systems usually have problems similar to their predecessors, and such issues can be avoided through using previous data and knowledge [2], [16]. In addition, Artificial Intelligence (AI) techniques may be employed for process optimization and prevention of new errors [17], [18]. Moreover, there is an environmental aspect to retaining information from previous systems generations, e.g., if material information is lost, the system may be improperly disposed of and cause environmental hazard [16].

1.1.7 Digital Twin and predictive maintenance

Predictive maintenance focuses on proactive methods to reduce cost and increase machine uptime [19]. Predictive maintenance aims to foresee when a component or system will no longer fulfill its function. This estimation is quantized through the Remaining Useful Life (RUL), a Health Indicator, or equipment effectiveness. Based on the RUL and other machine data, businesses can predict when to schedule maintenance for a component or system [20], [21]. In order to reduce maintenance cost and maintain machine uptime at the highest possible level, maintenance should be carried out in a proactive way. That means a transformation of maintenance strategy from the traditional fail-and-fix practices (diagnostics) to a predict-and-prevent methodology (prognostics) [22]. Prognostics and Health Management (PHM) is an engineering discipline, focused on predicting the time when a system or a component will no longer perform its intended function [20]. An estimation of this value is the Remaining Useful Life (RUL), which is an important concept in decision-making for contingency mitigation. Based on health assessment information, namely the RUL, as well as on additional information about multi-criteria mechanisms, it determines when a maintenance action should be executed [20]. RUL is the most necessary parameter, which should be predicted/estimated for the creation and execution of a maintenance plan [23]. Despite the fact that a number of predictive maintenance tools/platforms and methodologies have been focused on the RUL calculation, there are still gaps which are identified. More specifically, almost all the aforementioned approaches are based on the availability of past failure data of the manufacturing systems in order to calculate the RUL. It means that the RUL can not be calculated without historical data. Moreover, the already existing methodologies focus on machine-level RUL prediction and not in component-level. Due to the high complexity of the modern machines, the RUL calculation should take place in the component level. Another gap which is identified in the above tools and methodologies is that the models which are used for RUL prediction are not updated over the time. Most of them are static and can not be adapted online in case that real machine functionality is changed. Finally, a limitation of the already existing approaches is that the physics phenomena and their degradation profiles are not taken under consideration [24]. In [24], Aivaliotis et al. have proposed a methodology which focuses on the utilisation of the Digital Twin concept in manufacturing, aiming to estimate the RUL of each machine's component. As a simulation of the physics-based models, it refers to the solving of the inverse kinematics of the model by providing position signals as input and gets the calculated torque signals which are applied to each machine 's components as a simulation output. As in reality, the same tasks will be assigned to the simulation models aiming to use the simulation output for the RUL calculation.

The main objective of their research is the calculation of the remaining useful life of each machine in a production plant, based on the combined examination of the machines' controller data as well as the machines' physics-based simulation.

There are many examples of implementing Digital Twin for enhancement/advancement of predictive maintenance tools. Sresakoolchai and Kaewunruen's work [25], is the world's first to improve the railway infrastructure maintenance efficiency using deep reinforcement learning A2C integrated with digital twin based on track geometry parameters and track component defects.Rossini et al.[26] , developed a platform of IoT and Digital Twin-based predictive maintenance, REclaim oPtimization, and simuLatIon Cooperation in digitAl twin (REPLICA). The platform sits on top of two European research projects, RECLAIM and CPSwarm. They leveraged the knowledge from CPSwarm's optimization and simulation results and combined it with the Adaptive Sensorial Networks from RECLAIM. Wang et al. [27] used ANSYS, a finite element modeling software, to calculate the stress intensity factor under specific load conditions. This Digital Twin model was then applied for aircraft wing crack growth prediction. Rajesh et al. [28], created a Digital Twin of an automotive brake pad for predictive maintenance. They developed a 3D Computer Aided Design (CAD) model of the brake pad and used the Creo software to connect it to real-time data. They used the Creo Simulate software to generate additional data. This research aims to gather and synthesize the studies that focus on predictive maintenance using Digital Twins to pave the way for further research. van Dinter et al. [19] performed a systematic literature review (SLR) using an active learning tool conducted on published primary studies on predictive maintenance using Digital Twins, in which 42 primary studies have been analyzed.

1.1.8 Digital Twin applications in resilience engineering

In systems that provide critical services, resilience is characterized by four abilities: 1) to plan/prepare, 2) absorb, 3) recover from, and 4) adapt to both known and unknown threats [29]. As potential environmental and man-made risks contribute to unanticipated disruptions, the challenges of anticipating and characterizing potential vulnerabilities in complex infrastructure systems will compound. Digital twinning will promote a far more systematic understanding of system resilience and recovery opportunities even in instances of high uncertainty[30].

Jin et al. [30] have summarized the results of a workshop on the role of DT (alongside AI and edge computing) in the resiliene of cyber-physical spaces (CPS). They suggest, The DTs of individual buildings have the opportunity to integrate data and models from all domains (electric, water, cyber, occupation, etc.). This integration of otherwise "standalone" data sources enables the examination of correlations and causal relationships that currently appear independently in the individual networks. To accomplish this goal, data must be gathered from the individual domains, a data warehouse that joins the data from multiple domains constructed, and key indicators of resilience in each domain developed.

Cognitive DT(CDT), is a DT enriched with cognitive capabilities both internally and through cognitive-capable sensing [31]. Based on a data-driven approach, DTs have further evolved by incorporating fundamental aspects of cognition, such as attention (selective focus), perception (forming useful precepts from raw sensory data), memory (encoding and retrieval of knowledge), reasoning (drawing inferences from observations, beliefs, and models), learning (from experiences, observations, and teachers), problem-

solving (achieving goals) [32]. Eirinakis et al. [33] propose a conceptual framework for implementing the cognition for DTs of production processes towards achieving resilience in production. To that end, they have provided a mapping between the operational needs of five industrial cases (oil refineries, waste-to-fuel transformation, electronics' production for the automotive industry, steel processing and the textile industry) on decision support within their production operations with the corresponding cognitive capabilities of DTs and the suggested tools to materialize them, having considered a wide spectrum of production processes and corresponding events, from Liquefied Petroleum Gas (LPG) off-specs production recovery to production scheduling with predictive maintenance, and from taking into account disruptions caused by machine malfunctions or material failure to crane movement and to facilitating new urgent orders.

Ivanov [34] offers a generalized decision-making framework to use digital twins in resilience management, Supply chain (SC) stress-testing and resilience analysis and delineate how digital twins can contribute to theory development in SC resilience and viability.

1.1.9 Exploitation of Digital Twins in Smart Manufacturing

The advent of Industry 4.0 has propelled smart manufacturing into a new era of interconnectedness and data-driven decision-making, with the Digital Twin (DT) emerging as one of its most transformative enablers. A Digital Twin—a dynamic, virtual representation of a physical system or process synchronized through real-time data—promises unprecedented levels of visibility, simulation, and predictive control across the manufacturing lifecycle. This review critically examines the current state of Digital Twin implementation within smart manufacturing, analyzing the convergence of enabling technologies, practical deployment frameworks, and persistent challenges. By synthesizing recent advancements and real-world case studies, we aim to delineate the tangible benefits realized, identify critical gaps between theoretical potential and on-the-ground execution, and outline a forward-looking trajectory for achieving robust, scalable, and truly intelligent manufacturing ecosystems through Digital Twin technology [35].

1.1.10 Influential aspects of DT in manufacturing

Digital Twin can influence the future vision of Smart Manufacturing, in the following aspects [36]:

- **Digital Twin for Manufacturing Assets**: A manufacturing asset can be connected and abstracted to the cyberspace via its Digital Twin. Manufacturers can gain a clearer picture of real-world performance and operating conditions of a manufacturing asset via near real-time data captured from the asset and make proactive optimal operation decisions. With truthful information flowing from a manufacturing asset, manufacturers can improve their situational awareness and enhance operation resilience and flexibility, especially in the context of mass personalization [36].
- **Digital Twin for People**: Digital Twins can also connect workers at the shop floor. The representation of a person, including personal data like weight, health data, activity data, and emotional status can help to establish models to understand personal wellbeing and working conditions of humans in a factory. The understanding of human state at workforce can help design human-centered human-machine collaboration strategies to increase the physical and psychological health of workers, as well as achieving best production performance. Workers can also upskill themselves via ultra-realistic training programs which blend physical factory setups with virtual what-if scenarios.

The ability to set up personalized virtual training programs based on Digital Twins of workers and factories can lead to tremendous resource optimization and operational efficiency [36].

- **Digital Twin for Factories**: Digital Twins can also work for factories, making a replica of a live factory environment. Digital Twin and data-driven production operations can allow the establishment of a self-organizing factory environment with complete operational visibility and flexibility. Connectivity and data tracking throughout the complete manufacturing process enable factory operations to be transformed into data-driven evidence-based practices, offering the capabilities of tracing product fault sources, analyzing production efficient bottlenecks and predicting future resource requirements [36].
- **Digital Twin for Production Networks**: By connecting manufacturing assets, people and service via Digital Twin, every aspect of business can be virtually represented. Connecting distributed Digital Twins between companies will allow companies to build virtually connected production networks. Leveraging Big Data capabilities, this strategy provides unprecedented visibility into operation performance and creates the possibility of predicting future needs in a network of Digital Twins [36].

1.1.11 Principles of the Integration of DT in Smart Manufacturing

In the following we review the principal aspects and architecture of the integration of DT in smart manufacturing and the related sub-domains of Human-Robot Collaboration systems, assembly and quality control:

- **DT in Smart Manufacturing**

  With the development and evolution of digital technology, the use of digital technology to describe the essential factors in product manufacturing began with the use of simple coding and identification technology, and has developed to the digital twin technology of virtual reality interaction [37], [38], [39]. In the smart manufacturing system with industrial internet as the framework and platform, digital twin plays a key role throughout the whole process [37], [38]. Within a typical digital twin-based industrial information integration system in smart manufacturing supported by industry IoT, the digital virtual body mainly exists in cloud platform layer, and its control-oriented dimension model is arranged in edge layer to participate in real-time control. Industrial internet is composed of field layer, edge layer, platform layer and application layer. The application of digital twin is analyzed from these layers and the time dimension of design, production and operation and maintenance phases [40].

  In the design phase, the product design work is performed and the digital twin model of product design is created through collaborative design. After a series of simulation and optimization of kinematics, dynamics and other physical aspects, or technical services provided by a third party, the preliminary design and processing scheme are determined. Then, the virtual factory of manufacturing factory in cloud platform layer is used for virtual manufacturing to attain the simulation of the manufacturability. Now, the production phase can be triggered. At this point, the digital twin contains the physical characteristics of the product entity and all the information needed for manufacturing. The status quo key technologies supporting this phase are Model Based Design (MBD),

multi-physical property and multi-scale simulation, high fidelity modeling and model lightweight technology [40].

In the production phase, the production tasks are managed by the production management platform in the cloud platform, and the scheduling and control tasks with real-time requirements are handed over to the edge layer for management and control. Product manufacturing is an integration process of virtual and real. The digital twin of physical factory runs in the virtual space of cloud platform. The devices in the physical factory and IoT composed of sensors are located in the field layer, which exchange data with the edge layer through low delay networks such as time-sensitive network (TSN), 5G, etc. These multi-source data need different processing. On the one hand, some data directly interact with the model of the control dimension of digital twin, and get the predicted data. The optimal control of the manufacturing process is completed in the control cycle to realize the controlling of the real entity by the virtual model; On the other hand, the edge layer filters these data, transmits them to the platform layer, drives the virtual factory in the cloud platform to run synchronously (i.e. the interaction of virtual and real), and stores it in the big database, which provides data source for knowledge mining and carries out non real-time prediction and optimization of the manufacturing process. The key technologies in this phase are the real-time virtual and real fusion of multi-source sensor data, model-based control, etc [40].

In the operation and maintenance phase, the digital twin of the product is also provided when the product is provided by the manufacturer. The user can create and activate the virtual body of the product according to the digital twin template provided by the manufacturer in the virtual space of the industry internet. If it is a component, the simulation and optimization research of assembly process and assembly process can be started in virtual space; if it is a complete product, the simulation and optimization of the use environment and working process can be carried out, and the interaction of virtual and real can be achieved in the use process. Suppliers, technical service providers and users can obtain the status information of products on this cloud platform, so as to provide targeted technical services. In the operation and maintenance phases of the product, it is necessary to monitor the spatial position, external environment, use status and health status in real time, and establish a resume information database, which users can access and use through the application layer. The health status, function and performance of the product are analyzed and predicted by virtual body on the cloud platform, the problems are warned in advance, and the vivid visual means are provided to assist the rapid fault location and troubleshooting. In addition, in terms of operation training and guidance, digital twin can also provide more realistic effects with the help of the fusion technology of virtual and real. The key technologies in this phase are: the interaction and fusion of virtual and real, simulation, prediction, etc [40].

- **DT in Human-Robot Collaboration (HRC) systems**

  In an industrial production environment a notable proportion of work is attributed towards assembly operations [41]. Since the assembly tasks are often credited with handling difficult product geometries and require higher production flexibility they have traditionally been hard to automate [42]. With Lean automation, the concept of hybrid automation has emerged with balanced introduction of human flexibility and machine efficiency. However, the desired flexibility and human co-existence

lead to higher total complexity of the production system. The design, development, as well as operation - due to the frequent changes make it necessary to quickly validate the behavior of the system before it is put in the real world. DT is an emerging modelling and simulation technology that has been widely used in Human-Robot Collaboration (HRC), and aims to provide support for HRC in design, construction, and control. Within the operational hierarchy of a digital twin framework of an HRC system, all the objects of the physical system are synced with their digital shadows in virtual environment or, in other words, each element in virtual simulation is displaying the operating conditions of a connected physical object in the production system [43].

- **DT in Assembly and Quality Control**

  Typical complex product such as missiles, satellites and others, have characteristics of small batch production, long assembly cycle, complicated assembly data and interdisciplinary application [44], [45] .

  A digital twin approach for assembly process optimization is a useful mechanism to improve the quality of complex product assembly, because of its ability to provide a precise simulation result and avoid a lot of trial and error [46]. The deployment and implementation procedures for a data-driven and hardware-in-loop digital twin system can be divided into following steps [46]:

  1. **Build the Digital Entity of the Assembly Line**: The digital entity of the assembly line is the one to one corresponding virtual mapping in the digital world of the assembly line from the physical world. Building the digital entity of the assembly line is usually based on the history data and process knowledge provided by the knowledge base. It is worth noting that for an equipment or a product, its digital entity is a digital representation considering over its entire life cycle.
  2. **Real-time Online Sensing in Multi-Source Heterogeneous Environment**: To guarantee the real-time consistency between the physical assembly line and its digital entity, the multi-source heterogeneous environment data sensing technology is very important by transferring the real-time data from the physical world to the digital word. Typically, the data sensing technology in the multi-source heterogeneous environment should solve the problems of hardware equipment deployment and management, data collection and multi-source heterogeneous data processing.
  3. **Real-time Simulation of Equipment and Assembly Process**: With the help of the multi-source heterogeneous environment data sensing technology, the real-time data from the physical world can be analyzed and utilized to realize the real-time simulation of equipment and the assembly process. It should be noted that, the history data and process knowledge stored at the knowledge base should be accessible to train a more precise simulation mode and provide reference benchmarks to the real-time simulation process.
  4. **Realizing the Intelligent Production Scheduling under Uncertainty Conditions**: Based on the analysis from the realtime simulation of equipment and assembly process in the digital world, the intelligent production scheduling under uncertainty conditions can be realized. The intelligent production scheduling technology should include functions such as

the generation and simulation of the work plan, predictions and analysis based on data and models, decision-making and optimization of assembly process.

5. **Dynamically Adjusting the Assembly Process**: The ultimate aim of the digital twin system is to dynamically adjust the assembly process, although the analyses on the real-time simulations have provided many insights to the factory workers. In reality, this step is usually added manual actions considering the production safety.

1.2 Implementation of the Digital Twin in Water 4.0

In the second part of this chapter, we provide a specific overview of the implementations of the Digital Twins in the field of smart water industry or water 4.0, because our own work which will be introduced in chapter 3 is in the same category of Digital Twins implementations. Here in this chapter we firstly present the doamin-specific definition of the Digital Twins in this particular field and explain its foundations, consequently we present the notable works done in the same field accompanied by direct comparative analyses that could be drawn between these works and ours.

1.2.1 Domain-specific definition and foundations

The International Water Association's (IWA) report on digital water [47], emphasized the importance of digitalization of the water infrastructures as an imperative necessity despite the challenges due to aging infrastructure, inadequate investment, changing climate and demographics. To this purpose, the Digital Twin (DT) technology has become a point of realization and aggregation of technologies such as Internet of Things, advanced sensors, artificial intelligence, 3D simulation methods, cloud storage, etc.

A definition of DT was released during the first SWAN DT H2O Workshop as: "*A Digital Twin can be defined as an actively integrated, accurate digital representation of our physical assets, systems, and treatment processes with a constant stream of data pairing from the physical twin for continuous calibration. It will unlock value by enabling improved insights that support better decisions, leading to better outcomes in the physical world.*" [48].

Being existent from the design phase of a physical twin or being mounted as a component onto a legacy infrastructure, DT has the capacity to offer real-time and predictive information throughout the life-cycle of the physical system entwined with. Water treatment plants being complex infrastructures with interconnected devices, distribution and sewer networks, anthropized and natural watercourses, possess great capacity to exploit the DT, considering issues such as water-loss and the need for sustainable distribution operations towards water 4.0. In the following we present a review of the implementation of the Digital Twins in the water sector [49].

Digital twins can be leveraged to manage and operate all types of water-related infrastructure, including pump stations, pipe networks, storage tanks, and treatment facilities [50]. DT moves a step beyond the SCADA, providing dynamic operation of simulations which are fed the (previously unused) data available at many facilities on a day-to-day basis [51]. A smart water network can be realized by supplying advanced telemetry to a control water-energy management supporting a set of conditioning factors, including the following [52]:

1. Flow management: reduction in water losses through a detection and communication system that intelligently supervises sensors, telemetry, and actuators to regulate water pressure and flow in critical network points.
2. Water and energy monitoring: a monitoring system transmits the pertinent data to a data acquisition system, control, and management hub.
3. Water-grid control: a remote-control platform, which uses big data analytics, empowers the water–energy network manager to make the system progressively more efficient with real-time control and data-driven decisions.

The framework for the DT of a hydro-graphical networks is based on a Water Distribution System (WDS) structure on which a hydro-logical model can be associated. The design of the DT requires the hydraulic model of the hydro-graphical system, identification of the dynamics of the controlled devices, and calibration of the hydraulic model [53].

DTs within the smart water grids, integrate virtual engineering models with city-scale reality models and Geographic Information System (GIS) data. The DT is then connected to a software such as Matlab or to a computer code and programs that implement the expert rules or the predictive and adaptive control strategies toward the objective of a platform for designing the new control techniques and management strategies adjustable and testable by the DT [52]. DT development requires continuous adjustments and learning techniques supported by large field data stored in big-data platforms with its main components being GIS, sensors, data acquisition (SCADA), and smart metering computerized maintenance management system (CMMS) [52]

1.2.2 Notable works and comparative analysis

In the following we present some of the most notable works done in this field and the direct comparative analysis between these works and ours. It may be noted that the comparisons between these works cannot be completely head-to-head in terms of characteristics, because of different usages, and the fact that our DT is its first example and completely new in terms of its design mechanisms and purposes.

Here the case of Gothenburg from the project "Future City" is introduced. The problem intended to be tackled was to prevent 3 billion liters per year of untreated wastewater being discharged to the environment, as a consequence of utilities regularly experiencing high flows in the sewer collection systems leading to spills from combined sewer overflows (CSOs) caused by heavy rainfalls. An operational DT approach was envisioned as a decision support system with online flow predictions and suggestions for control strategies. Part of the last stage of the project focuses on implementation of full model predictive control (MPC) [54]. The main part of the DT consists of a model of the sewer system built in MIKE URBAN[1] using several different modules, including a dynamic model for the hydraulics of the pipes and tunnels, conceptual hydraulic models to describe sub-catchments, optimisation modules for real-time control, and modules for handling of precipitation forecasts and rain gauge data quality control. The model has been calibrated manually for many of the sub-catchments in the network where rain gauges and flow measurements are available, as well as for the hydraulic model of the main tunnel system. The

[1] https://www.mikepoweredbydhi.com/products/mike-urban

catchment model is re-calibrated manually to achieve better flow prediction as more sensors are added to the system over time, thus providing greater spatial resolution in the system for model calibration. The physical infrastructure connected to the DT consists mainly of flow and water level sensors in the central part of the sewer system and at the WRRF that provide online, real-time updates of these measurements, as well as status from actuators (pumps, valves and gates) in the system [55]. CWRP DT was created and envisioned as an advisory tool without direct control capabilities which can further be equipped with[56]. It has automated data inputs directly from both the SCADA system and the laboratory information management system (LIMS), as well as auto-calibration and soft sensor capabilities. Simulated scenarios to test and calibrate strategies to enhance the plant's water quality as well as optimization of the plant's energy and chemical consumption are among the objectives expected from this model [55]. The DT includes dynamic semi-mechanistic models of the CWRP whole plant hydraulics, controls, and processes. Data-driven influent predictions are used as part of the DT functionality for predicting plant performance up to five days into the future. The DT inputs from SCADA are the various flows measured in the facility, the online primary effluent ammonia values, air rates to the various bioreactor zones and other relevant operational setpoints. From the data, in combination with the LIMS data, a dynamic raw sewage influent file is generated, thus creating a soft sensor of the actual influent to the facility. The initial calibration of the DT was done manually; control and hydraulics calibrations were based on actual measurements. The process calibration was first done in a steady state, then dynamic calibration was done based upon the first six months of a full data-set. The DT also included limited auto-calibration while running. This was accomplished as measurements became available where, for example, primary effluent laboratory total suspended solids (TSS) and chemical oxygen demand (COD) data were used to calibrate both the primary clarifier TSS removal and the soluble COD/COD fraction in the raw sewage based on the most recent performance data [55]. In regards to the comparative analysis we can deduce, it must be firstly noted that the goals of their DT and ours are different, since theirs is designed to control effluent water from being discharged, and our DT is designed to control the influent water, in fact to make sure the water incoming for use or treatment possesses the minimum healthy qualities. Considering this basic difference, we can compare the two DTs in terms of their realization of DT goals, or in relation with SCADA which both DTs are connected to, or even in terms of technical design, performance etc. One important advantage of our DT is that, while this operational CWRP DT demonstrate the value of hydraulic and process models for decision support, they remain fundamentally advisory—providing forecasts and recommendations without closing the control loop, somthing that our DT does with its actuation commands. In our system the commands are human-readable json files transmitted that can be both advisory to the operators, and fully automated given the correct mapping into the PLCs of the SCADA system, and since there is no universal SCADA code for a particular action such as closing a valve for example, that depends on the target SCADA which can be different case-by-case, so our DT can be considered to be a universally connectable entity, of course it course serve in the SCADA Master Termial Unit as a plugin, or just as an alternative for the MTU software based on the needs of the treatment facility. As noted, these two DTs are different in purpose, their DT is focussed on flow control so they need a complete hydraulic model, while for our DT that is not needed, because the communications between our DT and the Remote Terminal Units (RTUs) will suffice for all operations for automatization of SCADA, and that can be as flexible as can be imagined, because it can be unlimitedly instantiated (of course there is a limit to the computational capacity of the host system), and if for any reason, an

operational hydraulic source or instrumentation is out of service, its DT in our ensemble of DTs will be turned off, the same will be for the communication with its RTU, and so all automated operations with it will be shut down. The connection with SCADA, in fact its RTUs as the source of input data is one of the common points of these two DTs, but different goals make also great distinctions between the type of data each use from their respective SCADA. While their system is focused on plant hydraulics and receives its chemical information from LIMS, our DT is completely independent from out-of-SCADA software or entities for chemical information, due to its extended covering of the most important water quality parameters in real-time, i.e. making the most out of one of the most important DT-enabling technologies which is sensors technology. This quality makes our DT operable in all sorts of advanced or less advanced, equipped or less equipped circumstances in different parts of the world with any level of instrumentation and infrastructural availability, i.e. our DT is more universally deployable. Among all differences, the most notable advantage of our DT is its unique innovative data and timing agenda. The CWRP DT, for its first time calibration needs the historical data of six months, and then, its future range of prediction is five days. Our DT does not need historical data at all, it merely relies on its innovative mechanism of Recent Real-Time Data (RRTD) of its Recent Real-Time Window (RRTW), and opens the future window or releases the prediction horizon temporally 20 percent of the length of the RRTW, and most importantly it is completely temporal-agnostic which makes it flexible about the rate or absence of transmission rate of incoming data, meaning any disordered flow regime of incoming data can be handled by our DT and generate a reliable future prediction. Therefore, it works with all ranges of average intervals of receiving input from milliseconds to days. To make the comparison with CWRP DT more clear, assuming the chemical data of the same six months of the target site (which is considered to be an scarce input transmission temporal regime for our DT) would be applied to our DT under those input circumstances, then our DT's prediction horizon will not be 5 days which is provided by CWRP DT, it will be 20 percent of the six month, which is 36 days, and every 36 days after that first 6 months another prediction horizon of 36 days will be released, if the same input regime holds, and the cycle continues, or changes with the change of input transmission temporal regime. Also in the case that we have all those six months data ready, our DT can still use these historical data, if the data is be converted to CSV format and be fed to our DT with adjustable intervals of 1 second (or more), so that it could traverse all that data very quickly and provide predictions. Moreover, another comparison is that their reliance on manual calibration makes their DT less of a fully automated DT, while our DT is fully automated from the very beginning of its deployment, and as defined it is "plug and play".

The work of Ramos et. al [52] is a DT applied to all subsystems using the EPANET hydraulic simulator. It is calibrated, compiled in a single model, and then applied to a water distribution network supplied by the Gaula's tank. All the location information and characteristics of physical elements used in elaborating the hydraulic model, such as the pipes' site and geometry, valves, and tanks, as well as the information related to existing altimetry and cartography, were used in GIS format. The DT-based model considers the volume of real losses separately from apparent losses and demands through emitter coefficients. With this approach, the DT has different variables associated with each node. This process results in the first group of values being applied to the emitter coefficients of the hydraulic model, and those values generate an output flow from the reservoir. The domestic consumption and apparent losses are considered constant values, thus, obtaining the real loss flow is possible. The value readjusts during the learning process which

is completed when the difference between the municipal real losses and the DT model reaches 0.01 L/s. The DT model was validated through a comparison made between the simulated pressure results with the real measurements observed at 52 points of the WDN, and based on the DT several analyses were performed to identify opportunities for improving the pressure values of the WDN[52]. As far as a comparative analysis could be made, again the design goals and implementations of this DT and ours is very different, and since their DT is based on hydraulic models, there will be the same comparisons made above as the case of CWRP DT. Meanwhile their DT is more of a geographically-mapped DT, while ours is SCADA-based.

The work of Bonilla et. al [57] is mainly focused on networks with reduced size in the context of a limited budget, proposing methods for creating a hydraulic model of a real WDS, with the assistance of a low-cost technology, in their case drones. The results obtained in the hydraulic simulations of the created model are compared with pressure measurement data at different points in the network. Their methodology is comprised of the following steps [57]:

1. collecting and analyzing the existing data related to the land registry of the network infrastructure such as WDS and urban layouts, satellite images, tacit information obtained from field visits, Subsequently a preliminary digitization of the WDS using a computer-aided design (CAD) tool, or geographic information system (GIS). Documentary records from governmental water companies such as WDS layouts, urban layouts (roads, infrastructure, land use), updated (and freely accessible) satellite images of the study site used to identify new neighborhoods, tacit information obtained from field visits accompanied by water utilities experts to identify the visible components of the system (valves, reservoirs, pumping stations, hydrants, and sensors and their physical properties). Subsequently a preliminary digitization of the WDS using a computer-aided design (CAD) tool, or geographic information system (GIS).
2. conceptualization of the basic hydraulic model to understand the main elements of the WDS (e.g. main pipes) and their functioning, such as the direction of the flows and storage volumes,
3. Photogrammetric images obtained using a drone, which helps create the city's digital elevation model. This digital elevation model is used to obtain the network elevation for both pre-existing and new expansion areas that comprise the city
4. Analysis using existing user consumption data are used in this step to analyze the base demands of the model. The hydraulic model is built using hydraulic computer engines (in this case the EPANET 2.2 software [58])
5. The hydraulic analysis of the model and the in-field pressure measurements of the water utilities, used to compare the obtained results (basic network calibration).
6. a preliminary calculation of the non-revenue water index.

Also in this case, the DT can be considering more of geographically mapped for the purpose of controlling physical presence of water, it makes use of advanced imagery systems, while our DT is SCADA-based and exploits the sensors technology for the purpose of obtaining automated high-quality water and feeding it to the treatment plant to be automatically refined through actuation which closes the loop in our DT.

Here we present some similar works altogether and will make the comparison afterwareds. GBRA invested

in a DT of its Raw Water Pump Station #2, which indicates when the screens are fouling, allowing operators' reaction and cleaning of them before they are damaged by the differential pressure, and real-time financial metrics for the replacement of worn-out pumps and other equipment. It allowed GBRA to operate smoothly through the "Texas Freeze" in February 2021 by giving operators the ability to use model-based control when sensors froze and traditional proportional, integral, derivative control failed[59]. MWRD invested in DT to resolve several challenges with its water pumping facilities (chronic failures of seals and bearings due to droughts several times per year). Their DT is used like a car's tachometer such that the operators now know how to adjust the speed of the pumps when lake levels change and that improved the meantime between failures, also used as a training tool for the new staff. Murfreesboro also commissioned a DT of its river raw water pump station, using which operators recognize when the pumps are clogging, and they cycle pumps on and off until the DT indicates that the material has either passed through the pump or dislodged back into the wet well. They also commissioned DTs of the micro-filtration membrane permeate and high-service pumping systems, which identified that the existing pumps were undersized [59]. MSUD employed DT in the operation and maintenance (O&M) of its facilities, used to keep the pumps operating within their PORs on the basis of varying conditions of peak summer demands to lower winter demands. During the 2021 Texas Freeze and the unexpected cutoff of power for utilities, since instrumentation systems and the DT were powered by battery backups, operations staff saw which tanks were dropping most quickly and prioritized getting generators to those sites [59]. Lakewood's DT is commissioned at each of the seven pumping facilities to handle each pump's POR, efficiency and productivity from wear and tear. Lakewood now prioritizes pump replacement using data-driven analysis. By linking the DTs, Lakewood has a unified specific energy map of the most efficient combinations of pumps across all of the facilities. This system allows operators to simply set a desired flow, and the DT determines the most efficient combinations of pumps and speeds to deliver that flow from the various facilities into the pressure plane [59]. The DT of Chattanooga Citico Pump Station—when it was created to determine the appropriate size for the new pumps based on the full-scale hydraulic data because past episodes of reduced capacity, excessive energy use, and too-frequent maintenance The DT indicated that the hydraulic system curve was higher at times than expected, but that this additional head was caused by surcharging from the downstream interceptor. This situation has been put forth by operations staff, but previous models of the interceptor could never replicate the real-world results. The DT confirmed the size of the pumps, but diagnosed the wear of two. More importantly, the DT showed that excessive plugging by debris, such as flushable wipes, was robbing the station of needed capacity and spiking energy demand [59]. All these DTs represent excellent examples of acquiring the benefit of O&M from the DT technology, and they take into consideration the physical aspect of the physical twin, therefore these works are of a completely different orientation of purpose, so comparing them to our work would not be grounded in reasonable terms, however we try a different approach in the comparative analysis in these cases. We believe that these DTs could be enhanced using our temporal framework of RRTW, and competitive AI-engine mechanism SHAP-MAE Engine, even though they have exploited sufficient frameworks and mechanisms that have yielded them good results, i.e. we believe that our tools can take the operational reflectiveness and predictive mechanisms of those DTs to a next level, making them more responsive. Of course, we are speaking about the tools' framework and not the details, meaning that the parameters of their concern which are mostly of physical identity can replace the chemical parameters in our framework, and since the statistical and AI foundations hold in general, the overall performance will

be more advanced.

At this point we present a number of works directly related to our work. In the context of water quality, there is an increasing need for DTs that can monitor, predict, and communicate water quality dynamics by combining contaminant fate and transport models, online sensor data assimilation, and real-time visualization and response capabilities [60]. Early warning systems for chemical spills are also a major concern for water managers [61], also the DT models enable mitigation of water quality problems through active control of hydraulic infrastructure [60]. Pasika and Gandla [62] proposed a monitoring system which consists of a number of sensors which are interfaced with the Micro-controller Unit (MCU) used to measure several quality parameters like turbidity, pH value, water level in the tank, dampness of the adjoining environment and temperature of the water; additional processing is carried out on PC [63]. Mukta et. al developed an IoT based Smart Water Quality Monitoring (SWQM) system which helps in incessant measurement of quality of water on the basis of four different parameters of water quality i.e., pH, temperature, turbidity, electrical conductivity, and many other sensors including humidity sensors. All the sensors are connected a core controller which controls the operation, gets data from sensors, and compares it with that of the standard values and sends the values through wireless modules [64]. The work of Kim and Bartos [60] is a DT model named "Pipedream". The governing equation for the Pipedream-WQ solver is the unsteady advection–reaction–diffusion (ARD) equation, and the water quality model is implemented in the Python programming language. The model may be initialized within a server environment, allowing the model to interface with external software like APIs and visualization tools by HTTP request. The work of Mutri et. al [65] is a monitoring system capable of providing water quality in a river utilizing the technology of Internet of Things utilizing communication of LPWAN LoRa. A microcontroller called ATmega328P-AU which is linked with various sensors for capturing predetermined variables such as pH and turbidity air, and is Programmed using Arduino IDE. All measurement data are sent to a namely Antares as a cloud service to store data and then display it on an Android-based smartphone. The application is programmed in Java and XML. The work of Qiu et. al [66] is a novel digital twin lake framework (DTLF) aimed to integrate, represent and analyze multi-source monitoring data on Harmful Algal Blooms (HABs, to prevent and control them) and improve water quality. Based on the constructed DTLF, HAB information was acquired with satellite remote-sensing, land-based video monitoring and in situ monitoring was integrated and expressed in 3d visaulization. The vido-based real-time monitoring of HABs is performed using Canvas API [67] which is executed periodically with browser-based front ends and the period can be set as needed. The DTLF employs U-Net [68], a segmentation model of digital images which can be trained with a small dataset, for HAB identification, and also Dynamically calculate the results of HAB monitoring. The work of Bogdan et. al [69] is a DT which offers a prototype implementation of an IoT water-testing solution for observing potable sources. The DT is controlled and managed from a mobile application connected to an Arduino UNO board with communication via a Bluetooth-module and submersible sensors (temperature, pH, TDS, turbidity). The sensors values are updated onto and read from a Firebase database consisting of JSON files for any source of interest which are five. Considering the above-mentioned examples, we can draw the main comparison which is the lack of predicitve option in them in comparison with our work. In fact they are merely focused on the matter of realtime monitoring, which our system also covers, while also providing prediction. They do not exploit AI while we use extensive usage of AI, and

it is one central aspect of our design. Moreover, these works present some connectivity solutions such as bluetooth, which is in fact acceptable solutions in small-scale which is good for experimentation purposes, while we offer MQTT connectivity which can be used in wide-scale and wide-range, and can be considered more practical and industrially implementable. The same comparison could be drawn with their use of electronic tools such as Arduino, while our DT can connect to a SCADA system, and that option makes it more deployable and more general purpose which has advantage over case-study-specific works.

We now move to the the overview of the works in which AI has been implemented. In the work of Makumbura et al. [70], the authors combine machine learning (ML) models–Random Forest (RF), LightGBM, and XGBoost–with Explainable AI (XAI) methods to improve both the accuracy and interpretability of water quality assessment. They use the Weighted Arithmetic Water Quality Index (WAWQI) [71], [72] as the target variable, calculated from multiple physical and chemical parameters. Notably, they apply SHAP to interpret feature contributions, revealing that COD (Chemical Oxygen Demand) and BOD (Biological Oxygen Demand) were the most influential features. The study shows XGBoost achieved the highest predictive performance, but importantly, it promotes transparency and trust in black-box ML models using SHAP, helping domain experts better understand predictions. In the work of Sen et al. [73], the authors propose an explainable deep learning framework by integrating both SHAP and LIME into LSTM-based models for time series forecasting. They demonstrate this on financial data, specifically NASDAQ stock prices, where the LSTM model outperformed ARIMA, SARIMA, GRU, and RNN in accuracy and directional forecasting. SHAP was used to provide global and local insights into how features (like lagged price or volume) influenced predictions, while LIME offered localized, instance-based interpretations. This dual-explanation approach aimed to bridge the gap between high-performance black-box models and transparent decision-making. However, their focus was domain-specific (financial data), and the methodology, while elegant, is not evaluated in the environmental or water domain. In the work of Kanagarathinam et al. [74], a comprehensive machine learning (ML) framework is proposed for classifying water quality using a large dataset of 7,996 water samples with 20 chemical, biological, and radiological features. The study evaluates 14 different ML algorithms, including advanced classifiers like LightGBM, XGBoost, and CatBoost. Techniques like Yeo–Johnson transformation, Principal Component Analysis (PCA), and SMOTE are used for preprocessing, feature selection, and class balancing respectively. The best performance was achieved by XGBoost (96.31% accuracy), with CatBoost showing notable improvement after SMOTE. This study is notable for its thorough benchmarking and inclusion of multiple preprocessing strategies. In the work of Nishat et al. [75], a comprehensive comparison of fourteen machine learning algorithms was conducted to predict the Water Quality Index (WQI) of four highly polluted rivers in Dhaka, Bangladesh. The study employed a variety of models including ANN, Random Forest, SVM, XGBoost, Ridge Regression, and others, assessing their performance using metrics such as RMSE, MAE, and R^2. Their results demonstrated the superiority of ANN and Random Forest in capturing complex pollution patterns, with ANN achieving an MAE of 1.24 and an R^2 of 0.97. The study further integrated explainable AI techniques, including SHAP and LIME, to interpret model predictions and emphasize the influence of key features such as Fe and DO. Their focus on urban waterways and the inclusion of a wide algorithmic range provide a strong benchmark for WQI modeling efforts. In the work of Kundu et al. [76], the authors applied machine learning and explainable AI (XAI) techniques to assess water quality in nine major Indian rivers. Using models such as Random Forest, XGBoost, and SVR,

they predicted key water quality indicators and employed SHAP to analyze feature influence. Dissolved Oxygen (DO) was consistently the most interpretable and accurate target parameter across rivers. In the work of Ngwenya et al. [77], the authors reviewed machine learning applications in ambient water quality monitoring, emphasizing predictive modeling of key parameters like pH, dissolved oxygen, and turbidity. While they surveyed popular algorithms such as Support Vector Machines, Random Forest, and Artificial Neural Networks, they noted a major gap: most models lacked explainability and failed to adopt tools like SHAP for transparent interpretation. In the work of Abdelhedi et al. [78], several machine learning algorithms–including Random Forest, SVM, XGBoost, and MLP Regressor–were evaluated for predicting water quality parameters such as pH, salinity, and groundwater depth in the Chebika region of Tunisia. Their approach relied solely on spatial features (latitude, longitude, and altitude), demonstrating the feasibility of low-input models for hydro-environmental predictions. In the work of Afan et al. [79], a comparative evaluation was performed using four neural network models (MLP, RBF, GRNN, and MLP-PCA) to predict water quality indicators in wastewater treatment plants. Their benchmarking focused on improving prediction accuracy through a multi-step model structure that incorporates intermediate settler outputs. The comparisons we can draw between these works and ours, is that while the main similarity is the usage of AI and XAI exists, the method in which they are implemented are different, particularly because of their dependence of historical data, which our DT is not reliant on, and it could be only considered a bonus. Our system works completely in a "plug and play" mode as soon as it receives real-time data. Some of these works are not complete-loop DTs, and only present comparative results between different AI models, without actuatory mechanisms. Some of them are limited to only a few AI models while we have a complete inventory of 24 ML models. Most importantly in our work we present a competitive model selection mechansim that makes our DT self-corrective and auto-enhanced with the passing of time. Another important difference is the inclusiveness of our water quality parameters, which are 12 obtained from sensors in real-time. Overall, we can again draw this conclusion that these works are more of a case-study and experimentation oriented works, and they lack the general deployability that our DT possesses. In fact, these are works that provide good information on the usage of AI and XAI on historical data and there are absent characteristics of DT in them in contrary to our work.

Considering these presented works and beyond, in regards to the implementations of the DTs in water industry, in particular about the water quality and treatment, there are still gaps to be filled by further advancements. One such advancement could be the aggregation of hydraulic models, historical and real-time data of water quality parameters, and the inclusion of weather condition parameters. As researchers have noted, there is currently "no holistic view of hydrological cycles and their interactions, whether this be rainfall or groundwater, and the different scales at which they operate". Integrated water models that combine these elements are "essential tools to understand the impacts of climate change," yet water quantity and quality continue to be modeled separately despite being fundamentally interconnected . The practical difficulty is that hydraulic models alone "are unable to account for the spatiotemporal dynamics of customer demand," while concurrently "water quality models are not equipped to simulate the fate and transport of emerging" . Recent efforts to address RDII (Rainfall Derived Inflow and Infiltration) demonstrate the complexity, as researchers working on real-time wastewater modeling have found that even small rain events can "elevate average flows up to 20% for up to a week before flows return to their normal average daily flow," with effects worsening when ground conditions are already saturated . This

indicates that weather parameters are not merely supplementary but essential for accurate prediction [80].

The complete ensemble of factors contributing to the water quality from the natural source to the treatment plant and within—including hydraulic behavior, water chemistry, weather patterns, and even seasonal biological variations—could be given as input to integrated models, and that would increase the fidelity of simulations and the precision of predictions . However, this is precisely where a huge complication lies within this field. The way to overcome this complication is the availability of interconnected and relevant data, yet such sufficient integrated data remains scarce. The literature confirms our assessments that "the broader adoption of machine learning-enabled digital twins faces several barriers, including limited data availability, technical integration challenges, and organizational and human resource constraints" [81]. Furthermore, water data are "disparate, held in many places, and sometimes behind barriers," making data linkage itself a primary statistical challenge [80] . The water sector remains "underserved" compared to domains like air quality, energy, and mobility, with current digital landscapes lacking "adaptive, future-proof pathways capable of integrating diverse data sources into a cohesive framework" [82]. Even when data exists, researchers attempting to combine data-driven models with process expertise have found that achieving both accuracy and the ability to capture variability simultaneously remains challenging [83] . Thus, the fact that there exists not such sufficient interconnected data—spanning the entire source-to-treatment continuum with weather context—remains a fundamental obstacle to realizing the full potential of digital twins for water quality management, and therefore this particular all-inclusive approach marks the main future challenges in this field, concerning also the potential of our work to be expanded into such scale.

2. A Comparative Study of Machine Learning Algorithms for Water Quality Prediction Using SHAP-based Explainability

In this chapter we present our first step from studying Digital Twins to actually building one. This chapter represents our breakthrough into using explainable AI for water quality prediction. The importance of this step cannot be overstated. Artificial intelligence is one of the core enabling technologies of any Digital Twin—without it, a Digital Twin can only show what is happening now, not what will happen next. So before we could build a complete Digital Twin for water quality, we needed to understand how AI could be used in this context, and more importantly, how we could make it trustworthy.

This led us to explore explainable AI, or XAI. The idea is simple but powerful. When a model makes a prediction about water quality, we need to understand what drove that prediction. We need to know which factors played a role and how they influenced the result. Without this understanding, operators cannot trust the system, and they will not act on its predictions. So our goal became clear: we needed to find a way to make AI predictions not only accurate, but also understandable.

To do this, we designed a comparative study. We took a real dataset of groundwater quality from the Emilia-Romagna region in Italy and used it to test six different machine learning models. Some of these models are simple and easy to interpret, like linear regression. Others are more complex and powerful, like random forests. We wanted to see how they performed side by side, not just in terms of accuracy, but also in terms of how well we could explain their predictions using SHAP (SHapley Additive exPlanations), a method that shows how each feature contributes to a model's output.

What we found was a clear pattern. As models become more powerful and flexible, they tend to become harder to understand. But some models, particularly random forests, offered a good balance between the two. They were accurate enough to be useful, while still allowing us to see which features mattered most. This was an important insight. It told us that we did not have to choose between accuracy and explainability, we could have both, if we chose the right model and the right approach.

This study was not just an experiment. It was the foundation for everything that follows in this thesis. The results gave us the confidence to move forward and the knowledge to design the SHAP-MAE Engine, which sits at the heart of our VACCSMEDUS architecture. By starting with AI, and by focusing on explainability from the very beginning, we ensured that the Digital Twin we built would be not only intelligent, but also transparent and trustworthy.

2.1 Methodology and Dataset preparation

The dataset used in this study was prepared from groundwater data recorded by ARPAE in the Emilia Romagna region of Italy. We selected six relevant and complete features from the ARPAE data[1]: Temperature, pH, Electrical Conductivity, Dissolved Oxygen, Hardness, and Chlorides. Among these, three parameters,Dissolved Oxygen, pH, and Temperature—correspond directly to those used in the NSF-WQI formulation and were assigned their standard weights of 0.17, 0.11, and 0.10 respectively [84].

Electrical Conductivity was employed as a proxy for Total Dissolved Solids (TDS), reflecting their strong correlation in representing ionic content, and thus was assigned a weight of 0.07. Hardness was treated similarly to turbidity in terms of impact on water usability and aesthetics, warranting a weight of 0.08. Lastly, Chlorides were considered analogous to nitrates—both serving as contamination indicators from sources like seawater intrusion and agricultural runoff—adopting the nitrate weight of 0.10. The final weights were normalized to sum to one as follows: Dissolved Oxygen (0.270), pH (0.175), Temperature (0.159), Electrical Conductivity (0.111), Hardness (0.127), and Chlorides (0.159).

To ensure balanced contribution of all parameters, the raw feature values were first normalized using a standard scaling approach (*StandardScaler*). Unlike the official NSF-WQI method, which relies on empirically derived sub-index scores, our approach computes the Water Quality Index (WQI) as a weighted sum of these normalized parameters:

$$\text{WQI} = \sum_{i=1}^{n} w_i \cdot q_i \tag{2.1}$$

where w_i denotes the weight assigned to the i^{th} water quality parameter, q_i represents the normalized value of that parameter, and n is the total number of parameters considered. This formulation facilitates consistent model training and interpretability.

The dataset was split into 80% training, 10% validation, and 10% testing subsets using a fixed random seed to guarantee reproducibility. For temporal models such as LSTM, input data was reshaped accordingly to meet sequence input requirements.

We trained six regression models:

- **Linear Regression (LR)** and **Ridge Regression**: used as interpretable baselines.
- **K-Nearest Neighbors (KNN)**: implemented with $k = 5$ and Euclidean distance.
- **Support Vector Regressor (SVR)**: configured with an RBF kernel and tuned regularization parameters.
- **Random Forest Regressor (RF)**: used 100 estimators with default depth and feature subsampling.

[1] https://dati.arpae.it/dataset/rete-regionale-per-la-qualita-ambientale-acque-sotterranee-dati-2018

- **Long Short-Term Memory (LSTM)**: a two-layer deep network with dropout regularization, trained for 20 epochs using the Adam optimizer and mean squared error loss.

SHAP (SHapley Additive exPlanations) [85] is a widely used method in the field of Explainable AI (xAI) that helps interpret how machine learning models make predictions. It is grounded in cooperative game theory, where each input feature is treated as a "player" contributing to the final prediction. By estimating each feature's marginal contribution across different input combinations, SHAP assigns a consistent and mathematically sound importance value to every feature. Unlike traditional feature importance methods, SHAP is model-agnostic and supports both global and local interpretability, making it especially useful in domains where transparency is critical.

We applied SHAP to all models to understand how individual features influenced their predictions. For tree-based models like Random Forest, we used `TreeExplainer`, while for models such as SVR, Ridge, and KNN, we employed `KernelExplainer` with a 100-sample background for computational efficiency. Even though linear models are inherently interpretable, we included SHAP to maintain a consistent explanation framework across all models.

Since SHAP does not support LSTM models directly, we followed a commonly adopted practice in explainable AI: training an interpretable surrogate model to approximate the LSTM's predictions. This enables SHAP analysis through the surrogate, while preserving consistency in the interpretability framework originally proposed by Lundberg and Lee [**b13**]. Surrogate models are used to help explain how complex models arrive at their predictions by approximating their behavior with simpler, more interpretable alternatives.

2.2 Results

This subsection presents a detailed evaluation of six machine learning models in terms of both predictive accuracy and explainability. We report traditional performance metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE), and apply SHAP analysis to understand each model's internal logic with respect to feature importance.

Interpreting SHAP Summary Plots: In each SHAP summary plot, the horizontal axis represents the SHAP value—the contribution of a feature to the model's output. A wider horizontal spread implies greater influence of the feature across the dataset. Dots are colored by actual feature value (e.g., red for high pH, blue for low). SHAP provides a model-agnostic approach to interpretability, revealing how models attribute predictions to input features.

- **LSTM Model Performance and Explainability**:

  The Long Short-Term Memory (LSTM) model achieved a Test MAE of 0.0468 and Test Loss of 0.0057. This performance suggests the LSTM was moderately successful in capturing the underlying temporal patterns in the water quality data, although it did not outperform the simpler models in accuracy. Given that LSTM networks are designed for time series data, the model's ability to model sequential dependencies may have been underutilized due to the relatively shallow temporal window or lack of strong autocorrelated structure in the features.

To enable explainability, a Random Forest surrogate model was trained on the LSTM's predictions. The SHAP summary plot from this surrogate, shown in Figure 1, reveals that *Hardness*, *pH*, and *Temperature* were the most influential features. This pattern suggests that the LSTM prioritized physiochemical attributes associated with long-term water balance, rather than more transient or threshold-sensitive indicators like *Electrical conductivity*. This divergence may be due to the LSTM's capacity to capture interactions among slow-changing features more effectively than features with sharp decision boundaries.

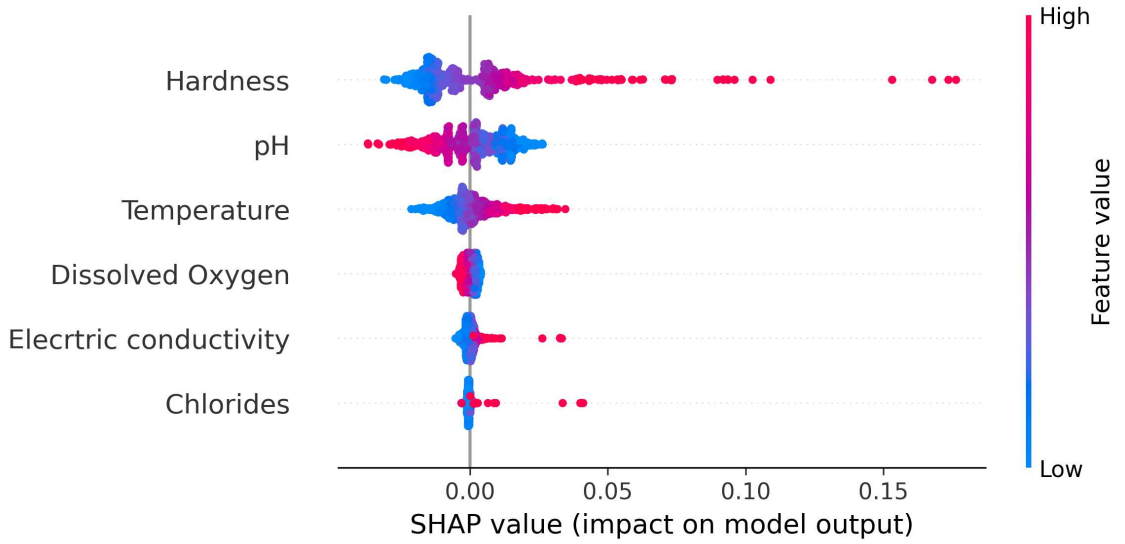

Figure 1: SHAP summary plot for LSTM model using Random Forest surrogate.

- **Random Forest Regression**:

  The Random Forest model yielded the best overall performance, with MAE = 0.0108 and MSE = 0.0016. Its ensemble architecture allows it to capture complex feature interactions and non-linear dependencies, which likely contributed to its superior predictive accuracy. SHAP analysis using `TreeExplainer` (Figure 2) showed that *Electrical conductivity* was the dominant feature, followed by *Hardness* and *Chlorides*.

  The prominence of *Electrical conductivity* indicates that the model learned to associate ionic concentrations in water with overall quality. The wide SHAP spread for conductivity and hardness suggests a highly variable but consistent relationship: higher conductivity generally increased the predicted WQI, likely reflecting mineral-rich but uncontaminated water. Random Forest's strength lies in identifying these localized patterns, and its robustness to outliers helps maintain performance across variable regimes.

- **Support Vector Regression (SVR)**:

  The SVR model delivered the weakest performance, with MAE = 0.3986 and MSE = 0.1644. Despite using a radial basis function (RBF) kernel designed to capture non-linear relationships, SVR failed to generalize effectively to this dataset. The likely cause is the high feature dimensionality relative to the model's effective capacity and sensitivity to hyperparameter tuning, such as the penalty term and kernel width.

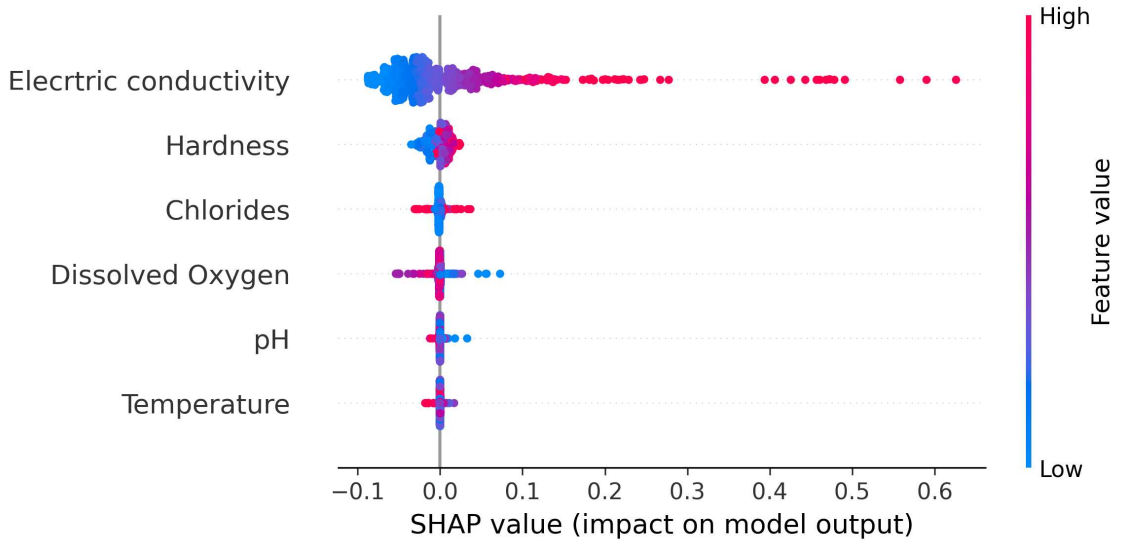

Figure 2: SHAP summary plot for Random Forest Regressor.

SHAP analysis using `KernelExplainer` confirmed this weakness (Figure 3): all features showed near-zero SHAP values, indicating no consistent signal was learned. This flat SHAP distribution reflects the model's inability to attribute changes in the output to variations in the input space. In practical terms, SVR's predictions fluctuated unpredictably, lacking coherent interpretability or stability.

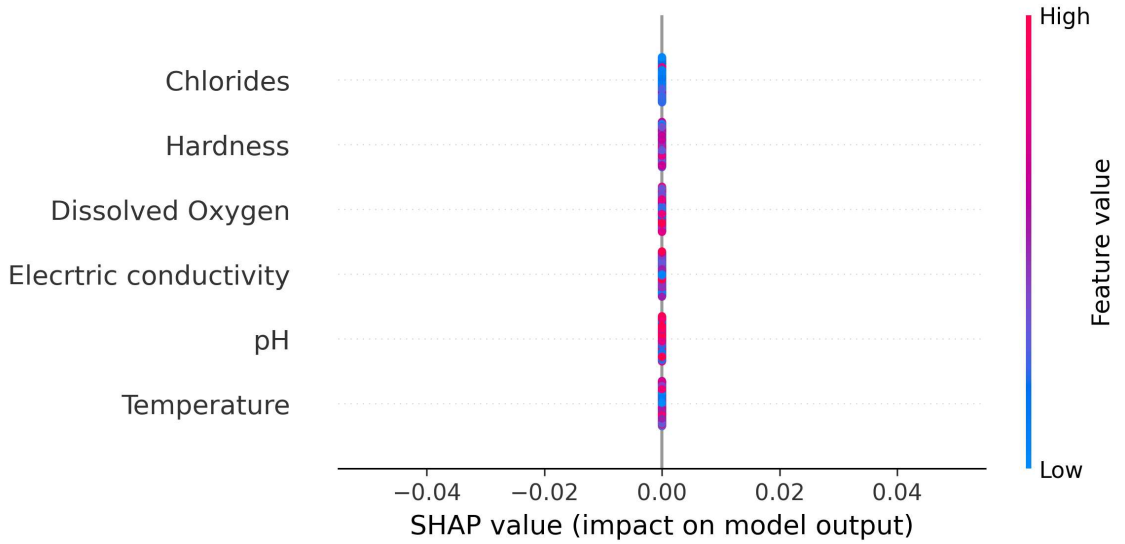

Figure 3: SHAP summary plot for SVR.

- **Linear and Ridge Regression**:

  Linear Regression achieved MAE = 0.0294 and MSE = 0.0046, while Ridge Regression had MAE = 0.0473 and MSE = 0.0060. Although simpler in structure, these models demonstrated relatively competitive performance, particularly for early-stage deployments or constrained environments where explainability and simplicity are paramount.

  The SHAP plots in Figures 4 and 5 revealed that *Electrical conductivity*, *pH*, and *Hardness* were

the top influencers. The absence of deep nonlinear structures allowed for stable, direct attribution. Ridge's regularization introduced slight smoothing, downweighting the impact of *Electrical conductivity* compared to Linear Regression. This likely reduced overfitting but also marginally decreased performance. Nevertheless, both models provide clear interpretability with respect to how each feature shifts the prediction, making them ideal in policy-driven settings.

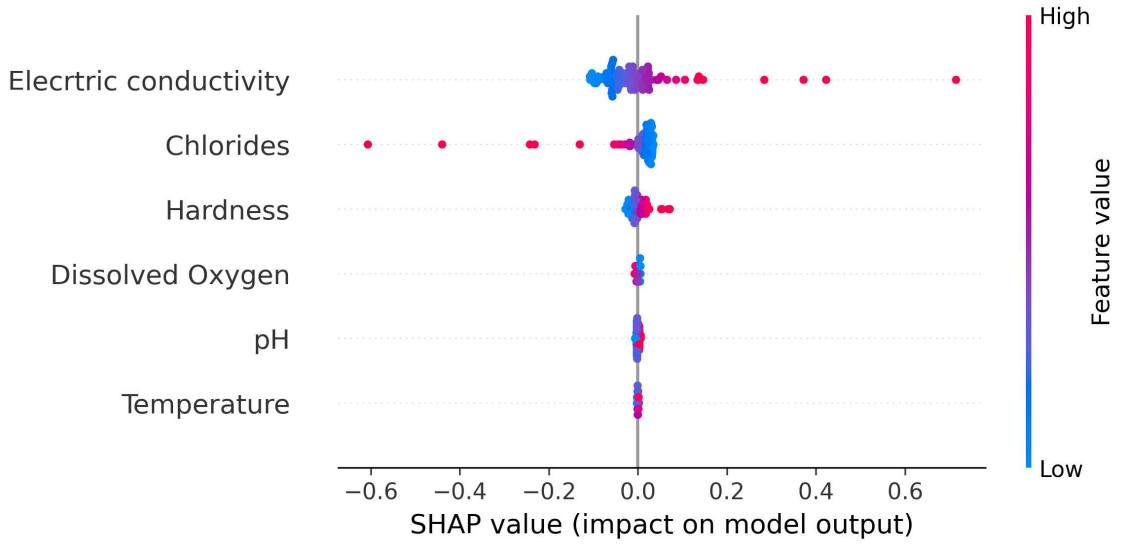

Figure 4: SHAP summary plot for Linear Regression.

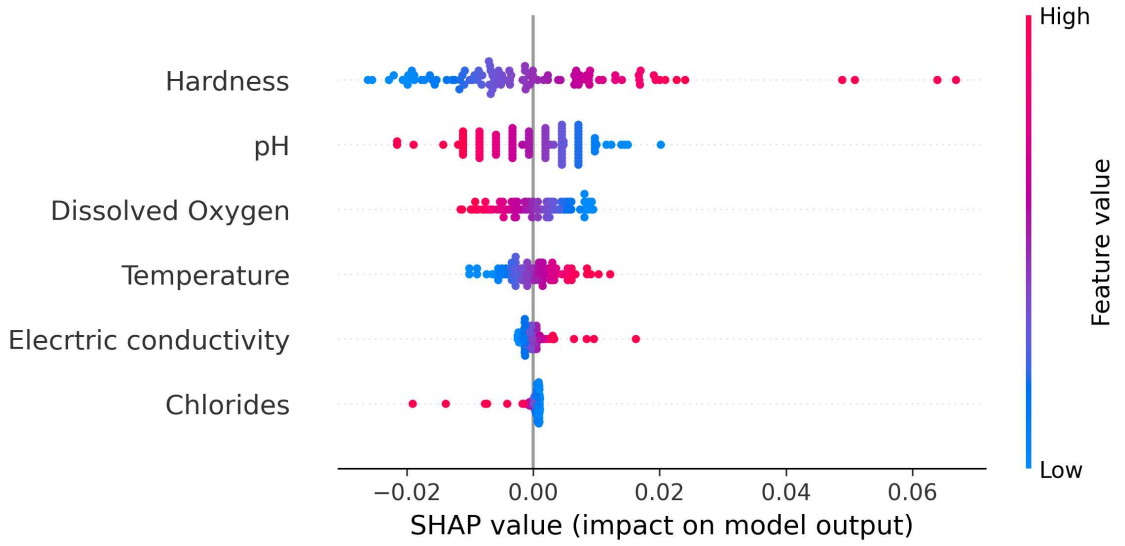

Figure 5: SHAP summary plot for Ridge Regression.

- **K-Nearest Neighbors (KNN)**:

  KNN achieved MAE = 0.0395 and MSE = 0.0044, outperforming both linear models and closely approaching LSTM. As an instance-based learner, KNN does not form global rules but instead relies on proximity to known samples. This local reasoning makes it resilient to noise but potentially weak in sparse regions of the feature space.

  SHAP analysis using `KernelExplainer` (Figure 6) indicated that *Hardness*, *Temperature*, and *pH*

were the dominant drivers. This profile closely mirrors that of the LSTM model, reinforcing the notion that both models respond primarily to underlying water chemistry rather than conductivity-based thresholds. The smooth SHAP gradients reflect how small changes in input values can subtly shift the prediction in KNN.

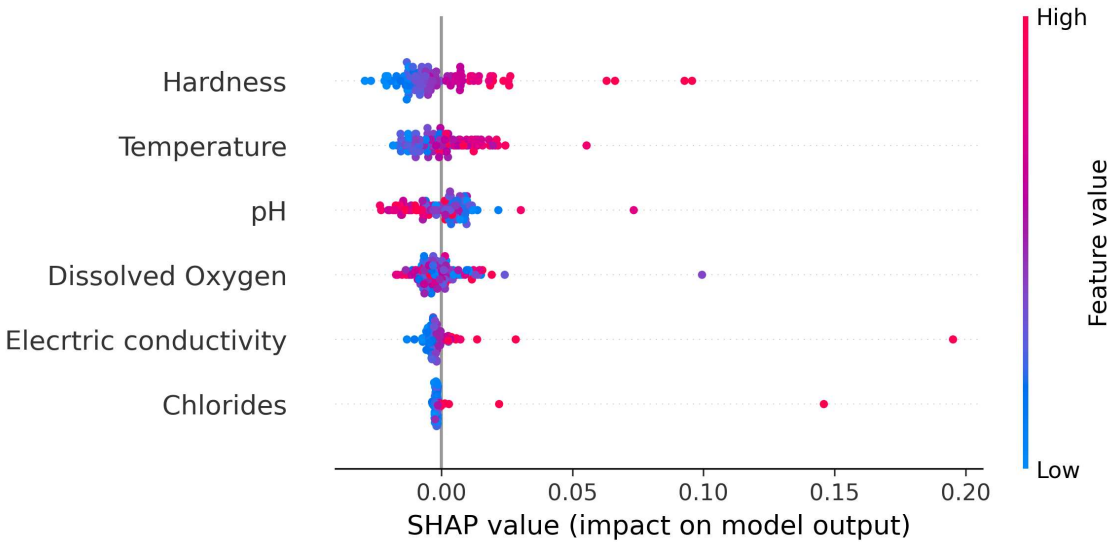

Figure 6: SHAP summary plot for KNN Regressor.

Table 1 summarizes the performance metrics and dominant features of all models. Among them, **Random Forest** stood out with the lowest MAE (0.0108) and MSE (0.0016), clearly outperforming the others in predictive accuracy. Its SHAP analysis showed a strong dependency on *Electrical conductivity*, followed by *Hardness* and *Chlorides*, indicating that it effectively captured threshold-based changes in water composition. This makes it highly suitable for precision-critical deployments where feature interactions are complex.

LSTM, though not the most accurate (MAE: 0.0468), offered a biologically coherent attribution pattern via surrogate SHAP analysis, emphasizing *Hardness*, *pH*, and *Temperature*. Its architecture likely captured subtle temporal dependencies or interactions, making it promising in contexts with streaming or time-sequenced water telemetry.

KNN performed comparably to LSTM (MAE: 0.0395) and shared a similar SHAP fingerprint, indicating reliance on smooth local chemistry variations. As a memory-based learner, it is well-suited for cases where water behavior follows distinct local clusters, and its SHAP results confirmed its meaningful reliance on relevant physicochemical markers.

Linear and **Ridge Regression** showed competitive results given their simplicity. Linear Regression (MAE: 0.0294) attributed high importance to *Electrical conductivity*, whereas Ridge (MAE: 0.0473) showed a more balanced SHAP distribution due to regularization. While not optimal in accuracy, both models offer direct, stable, and transparent feature attributions—valuable for regulatory or stakeholder-facing environments where interpretability is essential.

In stark contrast, **SVR** was the weakest model with MAE = 0.3986 and nearly flat SHAP values, indicating a failure to model meaningful input-output relationships. This suggests poor kernel parameter

tuning or an inherent mismatch with the data structure.

Table 1: Comparative Results and Explainability Summary

Model	MAE	MSE	Top Features	Interpretability Notes
LSTM (surrogate)	0.0468	0.0057	Hardness, pH, Temperature	SHAP via surrogate RF
Random Forest	0.0108	0.0016	Electrical conductivity, Hardness, Chlorides	High precision, clear SHAP attribution
SVR	0.3986	0.1644	None	SHAP flat — no signal learned
Linear Regression	0.0294	0.0046	Electrical conductivity, pH, Hardness	Transparent, balanced SHAP
Ridge Regression	0.0473	0.0060	Hardness, pH, Temperature	Regularized, conductivity suppressed
KNN Regressor	0.0395	0.0044	Hardness, Temperature, pH	Local logic, aligns with LSTM

Overall, Random Forest offers the best overall performance between accuracy and explainability. Its reliance on Electrical conductivity as the dominant feature demonstrated its sensitivity to ionic composition, aligning with known chemical drivers of water quality. In contrast, LSTM and KNN, although moderately accurate, exhibited similar SHAP attribution patterns, prioritizing Hardness, pH, and Temperature—factors often associated with gradual environmental variation. These models appear better suited for real-time or streaming scenarios where smoother transitions and localized changes dominate. Surprisingly, SVR failed to produce meaningful results, with both performance metrics and SHAP values indicating it could not extract useful patterns. This suggests poor kernel generalization and a lack of compatibility with the dataset's structure. Linear and Ridge Regression, while simpler, performed competitively and provided fully interpretable SHAP profiles—making them attractive options when transparency, auditability, and regulatory communication are essential. The radar chart in Figure 7 supports this narrative, clearly showing the divergence of feature importance across model types.

Overall, our findings underscore a spectrum of trade-off between accuracy and explainability, shown in Figure 8. Speaking about model characteristics being taken into account, apart from SVR, which performed poorly in this experiment, providing neither accuracy nor meaningful explainability due to its kernel-induced feature transformation that obscures input-output relationships, we have linear regression as the least explainable of the performing models with good accuracy. On the other side, we have Ridge regression, while structurally identical to linear regression, introduces a regularization term that shrinks coefficients toward zero. This penalty slightly reduces variance at the cost of introducing bias, marginally decreasing accuracy while enhancing explainability by preventing any single feature from dominating the model through excessive coefficient magnitudes. The comparison of these two models alone reveals the existence of this tradeoff, but the positions of the remaining models affirm it more broadly.

Moving from left to right along the explainability axis, we observe a clear pattern which is that, as models become more explainable, their accuracy tends to decrease. Random Forest sits at the leftmost end. It is the least explainable of the four but achieves the highest accuracy. This is because Random Forest combines many individual decision trees, each looking at different subsets of the data and features, and then averages their predictions. This ensemble approach captures complex patterns that single models might miss, but understanding exactly why it makes a particular prediction is difficult. We see the final result but not the detailed reasoning of every tree.

Moving rightward, KNN offers greater explainability. For any prediction made by KNN, we can simply look at the actual data points that were most similar to the new input and see what their values were. This makes each prediction understandable in a concrete way. However, this approach is less powerful at

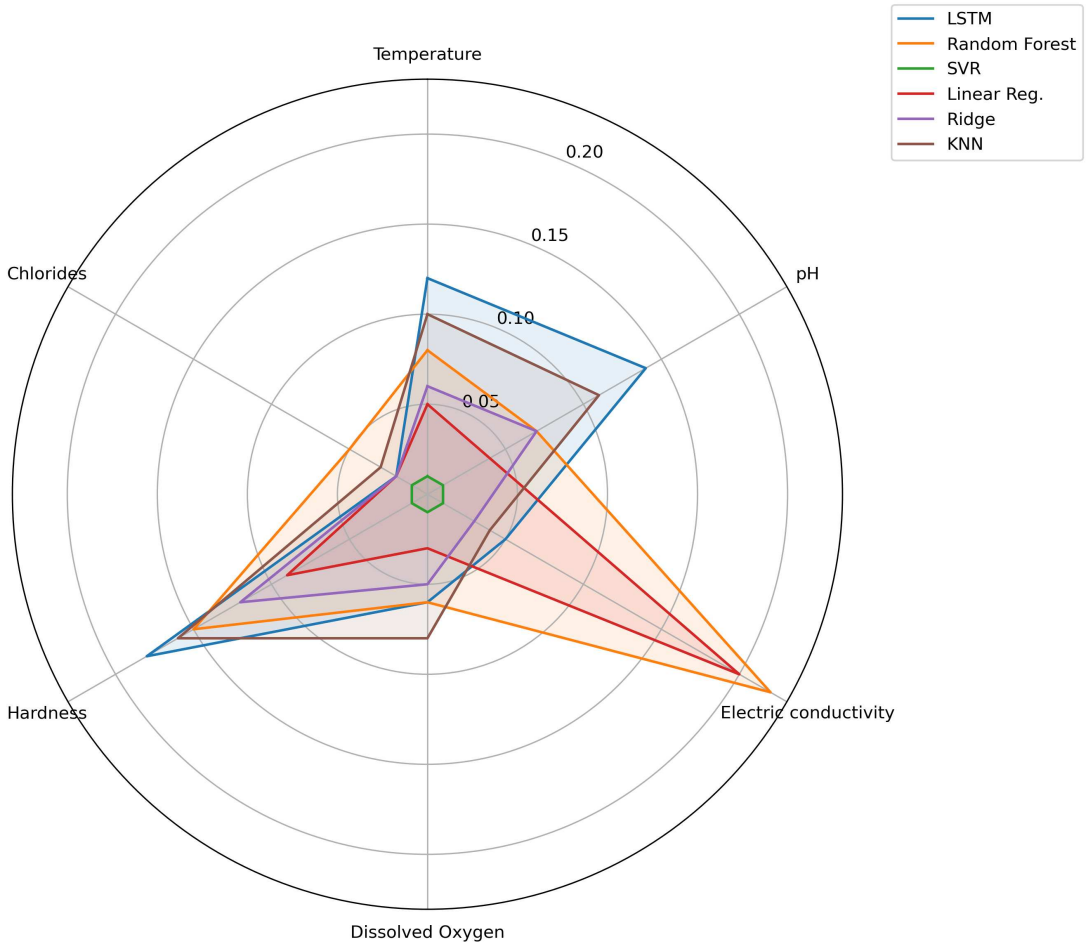

Figure 7: Radar chart comparing mean absolute SHAP values across all models and features. Each axis represents a water quality parameter, and each colored polygon traces the feature importance profile of a model.

capturing general patterns in the data, which explains its lower accuracy compared to Random Forest. Next comes LSTM, which we found to be even more explainable than KNN.

An LSTM processes data step by step over time, and we can observe how its internal memory evolves as new information arrives. This gives us a window into its reasoning process that is not available with Random Forest. Yet despite this transparency, its accuracy is lower—likely because training such networks requires large amounts of data, and our water quality time series may not be long enough for it to reach its full potential.

Ridge Regression stands at the rightmost end. It is the most explainable model of all. Every prediction is simply a weighted sum of the input features, with the weights (coefficients) clearly showing how much each feature contributes. We can write down the exact formula. But this simplicity is also its limitation which is that, the linear models cannot capture the complex relationships in the data that other models can, which is why its accuracy is the lowest among these four. Thus, the ordering from Random Forest

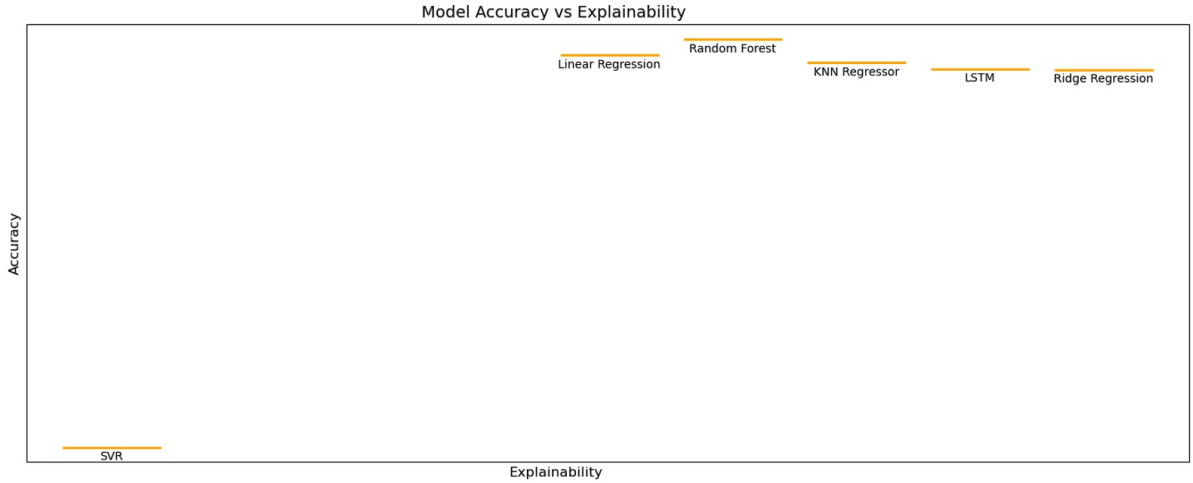

Figure 8: Radar chart comparing mean absolute SHAP values across all models and features. Each axis represents a water quality parameter, and each colored polygon traces the feature importance profile of a model.

through KNN and LSTM to Ridge Regression reflects a fundamental reality, which is that the more we force a model to be transparent and understandable, the more we limit its ability to learn complicated patterns from the data.

3. VACCSMEDUS: A Multi-Agent Predictive Digital Twin Architecture and implementation

In the final chapter , we present the most novel part of the thesis, VACCSMEDUS, which is a multi-agent predictive digital twin architecture of water SCADA RTUs and MTU-plugin-or-Alternative system, with SHAP-Guided Model Switching and the Recent Real-Time Concept. Our very own definition of the Digital Twins is a suitable point to begin with: "A Digital Twin is the interactive digital replica of assets and processes of a physical system or entity, and beyond it, being equipped with predictive mechanisms, it is the projection of the physical system in the future timeline, which brings about many benefits of such capability, among them the predictive-maintenance and foresight action abililty. On the other hand, the DT documents the activities path of a system that can be used for real-time-plus monitoring and analytics, in that sense it has active-archiving capabilities."

In fact the Digital Twins are rapidly evolving from passive monitoring frameworks into active, self-aware systems capable of anticipating future states of complex environments. In the field of water quality management, where early detection of contamination and system degradation is vital, prediction, explainability, and actuation must coexist in real-time.

The architecture is also implementable to create Digital Twins of other substrate fields because it has modular-design and embodies the general-purpose definitions of the Digital Twins with the complete task lists. However, in its details it is specifically designed for continuous water-quality supervision, forecasting and actuation. The system integrates four collaborative agents–the Collector, the Counselor, the Actor, and the Visualizer, which take use of a supporting agent named as the Utilitarian and other micro-agents, under the orchestration of the Director and Super-Director layer, forming a distributed yet synchronized ecosystem that transforms raw sensor streams into foresight, reasoning, and control.

The core intelligence capacity of our DT is realized in a novel mechanism named ***SHAP-MAE-Engine***, which inside its analytical processes lies the a hybrid explainability–performance mechanism that simultaneously interprets, selects, predicts, and validates models in real-time. It couples Shapley-based feature importance with rolling Mean Absolute Error feedback which will be explained later, creating a self-correcting loop in which model choice is not static but adaptively earned through performance competition. This mechanism transforms model selection from a manual design step into an ongoing evidence-driven process that continuously balances interpretability and accuracy.

Another central novelty of this work is the introduction of the **Recent Real-Time Window (RRTW)** and its associated **Recent Real-Time Data (RRTD)** paradigm. In fact, We have contrived a new concept regarding the categorization of systems, ***Recent-Real-Time Systems*** as an iconoclastic approach opposed to the bipolar distinction of real-time systems, and historical systems. Our digital-twin instead of *what is happening now* emphasizes *what has just happened*, and on the other hand it does not require historical data, hence our digital twin is a ***Plug-and-Play predictive RRT system***. Unlike traditional

monitoring systems that visualize instantaneous readings, the RRTW maintains a continuously sliding temporal frame that represents the immediate past and the near future simultaneously. This calibrated time axis, derived from the Average Transmission Delay (ATD) of incoming packets, allows the Digital Twin to operate in a temporally self-synchronized manner even under irregular or asynchronous telemetry. By coupling this dynamic timeline with predictive horizons, the platform achieves a seamless transition from "what just happened" to "what will happen," defining a new category, which is the *real-time-plus* monitoring.

The third contribution lies in the **multi-agent design** of the Digital Twin itself. Each agent operates as a specialized, autonomous component: the Collector provides normalized telemetry and WQI from RTUs (Remote Terminal Units); the Counselor performs SHAP-and-MAE-guided prediction and model switching; the Visualizer aligns real and predicted curves over the calibrated time axis; and the Actor executes rule-based control actions. The Director arranges and keeps the interactions among agents robust as a DT unit. The Utilitarian supporting agent provides the unified numerical tools backbone, feature ranges, normalization/denormalization, etc, on which all other agents depend. And finally Super-Director instantiates and manages the DTs start and stop and provides them with MQTT channel information. Together they form a closed cognitive loop that continuously learns, explains, validates, and acts. We have coined the acronym for this architecture as VACCSMEDUS (which stands for Visualizer, Actor, Collector, Counselor; SME stands for SHAP-MAE-Engine, then D stands for the Director, U for utiltarian, and S for Super-Director).

Together, the SHAP–MAE Engine, the Recent Real-Time Window paradigm, and the multi-agent modular design of VACCSMEDUS, establish a predictive twin that is not merely reactive, but self-interpreting and proactive. It does not only predict future values, but understands the cause of its own predictions, adjusts its reasoning dynamically, and translates insight into immediate operational decisions. This union of explainability, adaptability, and system integration represents a decisive step toward truly autonomous digital twins for water quality management and, more broadly, for environmental cyber-physical intelligence.

Having reviewed many related works and market software, we came to the conclusion that the Digital Twin as a pioneering technology of the era must be created from scratch by its developers for two reasons. The first reason is that relying on external providers of DT imposes highly expensive costs to the receiver of the DT service, and yet it is not fully compatible with the requirements of the system. The second reason is also related to the first one, in the sense that the market commercial software are very general-purposely designed, and not specifically tailored to the requirements of a target usage, they condone the details that are crucial to the performance of the system, some of them rely on code generators that make the system very heavy in terms of computational cost and black-box algorithms which cannot be adjustable to the best needs of the system, so we avoided all those pre-prepared instruments. We followed a classical coding approach in which we used strong and reliable programming languages, Java, and Python. The backbone and structure of our system, the agents are written in Java. As for Python, we used it to create a complete inventory of the Artificial Intelligence tools, including SHAP, and Machine-Learning models. The excellent performance of our DT system proved this approach to be correct.

We have studied the structure and network of the SCADA system [86] and the fundamnetals of the mulit-

agents [87]. We also studied the substrate field of water quality and treatment [88], to gain sufficient information on the requirements of such systems. Our VACCSMEDUS digital twin can be integrated within legacy SCADA systems of water quality assessment systems, or water-treatment plants, on the MTUs side, or as an intermediary between the MTUs and RTUs. However, given the vast and inclusive capabilities of our DT, it can even replace an MTU unit, given the necessary wiring to the PLCs. In this way our DT can fully control the pumps in a predictive way , warn and stop its flow. For example when a quality feature like Arsenic is above the acceptable threshold, our DT can provide the water consumers of that well with confidence that because of it, the water they obtain from that well has drinking-level purity, and in fact they can see the dashboard and logs and know exactly the amount of each 12 features we provided. 12 very critical features are more than acceptable considering that the most famous work in this field, Brown et. al's NSF-WQI, contains 9 features [84].

Based on the foundational content of chapter 1, a set of functional requirements can be defined for any Digital Twin intended to operate in this domain. The system must provide continuous real-time monitoring of water quality parameters, as the dynamic nature of water sources demands constant vigilance rather than periodic sampling. It must be capable of generating predictions about future water quality states, enabling proactive rather than reactive responses to potential contamination events. The system should integrate with existing SCADA infrastructure commonly found in treatment facilities, working alongside or within current operational frameworks. The architecture should accommodate the heterogeneity of field instrumentation, recognizing that different sites may have different sensor configurations and communication protocols. Finally, the system must be deployable across a range of facilities with varying levels of existing infrastructure, from fully instrumented modern plants to more resource-constrained environments. These requirements establish the operational context that our Digital Twin for water quality control addresses.

3.1 Data preparation

In constructing the Water Quality Index (WQI) for the Emilia–Romagna dataset[1], we preserved the original weight distribution proposed by Brown et al. [84]for the features that remained available, namely Dissolved Oxygen (0.17), pH (0.11), Temperature (0.10), Nitrates (0.10), and Orthophosphates (0.10, replacing Phosphates in the original formulation). For parameters absent from the Brown model or unavailable in our records, their collective influence was redistributed among variables exhibiting similar physicochemical behavior. The weight of Turbidity was assigned to Total Organic Carbon (TOC), reflecting their shared sensitivity to optical and organic load, while the weight of Total Solids was distributed across Electrical Conductivity (0.07), Hardness (0.08), and Chlorides (0.05), which together represent the mineral and ionic content of water. To extend the index toward contemporary health-related monitoring, additional weight was given to Fluoride (0.07) and Arsenic (0.15), the latter acting as a gatekeeper parameter that can nullify the overall WQI in case of exceedance, in accordance with an argument from the "Do we dare" paper from Brown et. al [84] which states "if any toxic exceeds its assigned upper limit, the water quality index will register as zero". The resulting ten-feature model preserves the structural balance

[1] https://dati.arpae.it/dataset/rete-regionale-per-la-qualita-ambientale-acque-sotterranee-dati-2018

of the Brown index while adapting it to regional water chemistry and modern contaminant priorities:

$$\sum_{i=1}^{10} w_i = 1.$$

Each selected parameter was normalized to the interval $[0, 1]$ according to its desirable concentration range, and then converted into a non-dimensional sub-index q_i describing the quality contribution of that feature. For variables whose higher values indicate better water quality, such as Dissolved Oxygen, the normalized value itself was used ($q_{DO} = n_{DO}$). For parameters where increasing concentration implies degradation (e.g., Temperature, Electrical Conductivity, Hardness, Nitrates, Orthophosphates, Chlorides, and Fluoride), the quality response was defined inversely as $q_i = 1 - n_i$. The pH sub-index was treated symmetrically around neutrality, $q_{pH} = 1-2|n_{pH}-0.5|$, ensuring that both acidic and alkaline deviations reduce quality. Arsenic was modeled with a linear decline from $q_{As} = 1$ at 1 µg/L to $q_{As} = 0$ at 10 µg/L, beyond which the entire index is set to zero.

The overall Water Quality Index (WQI) was then computed as a weighted sum of the sub-indices:

$$\text{WQI} = 100 \times \sum_{i=1}^{10} w_i\, q_i, \qquad 0 \le \text{WQI} \le 100, \tag{3.1}$$

with WQI = 0 enforced whenever the arsenic concentration exceeds 10 µg/L.

3.2 System Structure

This Digital Twin system is built upon the principle of task segmentation. Initially, we analyzed the general tasks and functional requirements that any Digital Twin should fulfill—independently of its domain. This top-down analysis included core abilities such as data collection, data processing, feedback generation, predictive inference, visualization of the physical entity's critical variables, and connectivity. Enabling technologies such as artificial intelligence, sensor networks, and communication protocols were also considered. From this analysis, a natural mapping emerged between each required Digital Twin function and an autonomous software entity capable of executing it, leading to the adoption of a modular approach. In this approach, each functional concern is encapsulated within a dedicated agent. Life-cycle management issues are also taken into consideration and relevant agents have been designed to handle such this level of operations. We tailored this multi-agent architecture to obtain a direct modular realization of the decomposed functionality of the Digital Twin concept. This architecture is composed of three groups of agents:

1. Directive Agents
2. Collaborative Agents

3. Micro and Supporting Agents

Directive Agents: These agents include the Super-Director and the Director. The Super-Director sits at the top of the hierarchy and orchestrates the creation and management of multiple Digital Twin instances (RTUs). It supervises their start and stop operations, ensures process isolation at the Java-thread level, and maintains session organization through timestamped run directories. The Director, instantiated within each DT, acts as the local orchestrator of each DT. It initializes and interconnects all internal agents, manages MQTT connectivity, coordinates clean shutdowns, and provides local health and control endpoints.

Collaborative Agents: These agents perform the operational intelligence of the DT and include the Collector, the Counselor, the Actor, and the Visualizer. As their names suggest:

The Collector acquires telemetry data from sensors located at remote Terminal Units (RTUs) near groundwater sources. Communication relies on the MQTT protocol, handled internally by a sub-agent called the Connector.

The Counselor acts as the analytical core of the system, hosting the AI-driven predictive models. It employs a hybrid SHAP-MAE Engine to perform model selection and prediction using machine learning methods.

The Actor forms the actuation layer. It continuously interacts with both the Collector and the Counselor—waiting for confirmation from the Collector before executing actions. Based on rules defined per feature, it can issue standby warnings derived from Counselor's predictions or decisive actuation commands (e.g., `CLOSE-PUMP`) when Collector's real values confirm a breach.

The Visualizer provides the system's user interface. Each collaborative agent has its dedicated dashboard tab within the Visualizer for configuration and monitoring. It also includes analytical views: one for real-time plots of feature values and errors, and another that exposes Counselor's internal model selection trace and computational process.

3.2.1 Inter-Agent Communication and Data Flow

The system operates through a coordinated set of agents. The Super-Director provides the Swing-based interface and acts as the global MQTT client. The Director manages one DT-RTU per process and launches its local agents. The Collector runs in **REMOTE_RTU** mode to receive real telemetry via MQTT. The Counselor performs SHAP-based model selection and ten-step prediction using the embedded Python bridge. The Visualizer renders real-time plots and maintains run logs. The Actor executes rule-based evaluations and publishes MQTT commands and warnings. Supporting elements include the *global* configuration holder, the *WqiSpec* that fixes features and weights for the run, and the Python *AIengine.py* that provides the model registry and prediction routines.

Each DT-RTU communicates within its own runtime namespace,

```
BASE = dt/run<SESSION_STAMP>/<RTU_ID>/
```

ensuring complete isolation between runs and units. All MQTT topics are defined relative to this prefix. During execution, the director announces its state on `BASE/state` with retained payloads (`STARTING`,

RUNNING, STOPPED) and emits a 1 Hz heartbeat on BASE/hb. The Super-Director subscribes to these topics to track DT-RTU health and can issue termination commands via BASE/ctrl/stop. The Collector follows the same pattern: it reports on collector/state and collector/hb, listens on collector/ingest for telemetry packets, and receives start or stop instructions on collector/ctrl. Telemetry arrives as JSON containing a sequential index and arrays of normalized or raw feature values. The actor publishes its operational streams on actor/tasks, actor/log, and actor/warn, carrying OPC-UA–style actions, runtime breadcrumbs, and predictive alerts. All MQTT clients inherit broker information from the director at startup, guaranteeing coherent communication across the DT. Within each DT-RTU process, the Director orchestrates all in-memory links. The Collector provides its latest raw and normalized data to the Counselor, which computes SHAP-weighted model selection and ten-time-step forecasts. The Visualizer consumes both real and predicted values for plotting and logging. The Actor continuously evaluates the incoming measurements and predictions against its rules and publishes MQTT alerts or OPC-UA tasks when thresholds are crossed. At run start, the Collector freezes the chosen feature order, builds a *WqiSpec* via utilitarian.buildWqiSpec(selected), and exposes it to the other agents. The Director initializes counselor.beginRun(WqiSpec) and actor.beginRun(), and drives actor.cycle() every 200,ms while emitting its own heartbeat.

During runtime, the Collector updates its latest snapshot (lastRaw, lastNorm, tickIndex) from MQTT ingestion and republishes its state. The Counselor observes the normalized stream, selects the best model using SHAP, and generates ten-step predictions through the Python *AIengine*. These predictions flow to the Visualizer, while the Actor evaluates both real and forecasted values to publish alerts or control tasks. The Director maintains its supervisory topics throughout, and the Super-Director oversees all DT-RTUs and may issue stop commands.

Initialization finalizes the feature selectyion and the associated *WqiSpec*, which governs all WQI computations in both recent and prediction windows. The shared global class provides a lightweight configuration hub for consistent runtime parameters. Each DT-RTU maintains its own run directory: the *counselor* writes CSV logs containing windows, SHAP values, and MAE metrics, while the *actor* records its outputs in actor.jsonl and actor_config.json, ensuring full reproducibility of the execution.

The overall setup of the VACCSMEDUS digital twin in the cyber-physical space with the flow of data and processing externally via MQTT communication, and internally between agents is shown in Figure 9.

3.2.2 Directive agents

3.2.2.1 Super-Director (Session Orchestrator)

The Super_Director is the top-level orchestrator responsible for launching, supervising, and terminating multiple RTU processes. It provides a Swing-based console and tracks each RTU's published state and 1 Hz hb heartbeat. At startup, it creates a session root directory logs/run<SESSION_STAMP>/, and each RTU launch produces a subdirectory <RTU_ID>_run<START_STAMP>/ that is passed to the child DTSCADA.director as both its working and logging folder. All downstream agents (collector, counselor, visualizer, actor) write their outputs inside this per-run directory.

The Super_Director establishes a Paho MQTT client with automatic reconnection and subscribes to

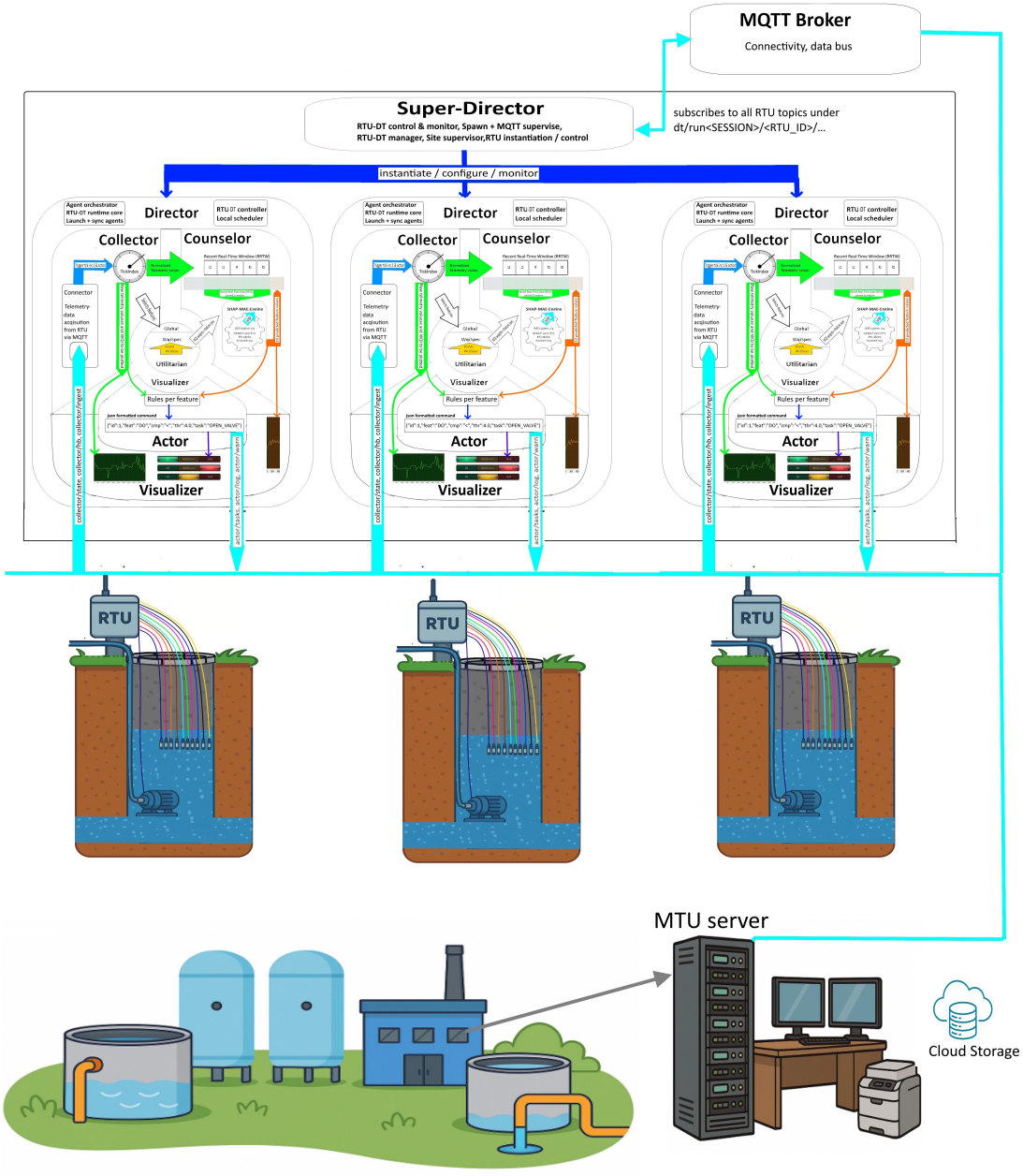

Figure 9: The complete structure of the Digital Twin system in the cyber-physical system, including internal inter-agent communication, dataflow and cycle, the MQTT communications channel, and the schematic of the relationship of the whole Digital Twin and Physical twin.

`dt/ run<SESSION_STAMP>/+/(state|hb)`. Incoming `state` messages update the cached last state, and incoming `hb` messages refresh the last-seen heartbeat timestamp. A scheduled polling task every 1.5 s evaluates liveness using the exact logic implemented in `pollHealthTick()`: an RTU is classified as `RUNNING` if its heartbeat is newer than three seconds or if its retained `state` equals `RUNNING`; otherwise, if the OS process is alive but the heartbeat is stale, it remains in `STARTING`; and once the Java process terminates, it is marked as `STOPPED`. This provides a consistent and immediate supervisory view even when reconnecting mid-session.

In addition to state and heartbeat monitoring, the `super_director` subscribes to `health/snapshot` and `health/alert` topics under the same session namespace. Each snapshot is a compact JSON object containing fields such as `freeze`, `tickGapMs`, `expectedPeriodMs`, `heapUsedMB`, `heapMaxMB`, `runFreeGB`, and `ts_ms`. These are parsed through regular-expression extractors into a lightweight `Health` structure and stored per each DT of RTU (DT-RTU). The tiles incorporate this information directly: the health line displayed on each tile is constructed exactly from these values, and a freeze or alert condition triggers a temporary border flash as implemented in `flashAlert()` and `shouldFlash()`.

Stopping a DT-RTU is performed by publishing the payload `"now"` to `dt/ run<SESSION_STAMP>/<RTU_-ID>/ctrl/stop` and terminating the OS process after a brief delay if it does not exit on its own. Starting a DT-RTU allocates a new per-run directory, computes its MQTT base topic `dt/run<SESSION>/<RTU_-ID>/`, and launches a child JVM with the arguments `-rtuId`, `-workDir`, `-logDir`, `-runStamp`, `-titleSuffix`, `-aiEnginePath`, `-mqttBroker`, and `-mqttBaseTopic`. The `RTUProc` structure maintains the process handle, run directory, timestamps, and state for each managed DT-RTU.

The user interface arranges each DT-RTU as a draggable tile displaying the identifier, current state, MQTT prefix, most recent run directory, the health line, and action buttons for starting, stopping, restarting, and opening the run directory. All updates triggered by MQTT callbacks or polling are dispatched through the Swing event-dispatch thread to ensure consistency. Tile placement is fully interactive, and the supervisory view is continuously updated without blocking or interfering with the RTU execution.

In summary, the Super-Director defines the session namespace, spawns and supervises DT-RTU-level Director processes, evaluates their liveness via heartbeats and retained state, incorporates lightweight health snapshots, and exposes all lifecycle operations to the operator. All internal communication among the Collector, the Counselor, the Visualizer, and the Actor occurs inside each DT-RTU process, while the Super-Director remains the global coordinator. Figure 10 illustrates the UI console of the Super-Director.

3.2.2.2 Director

Each DT-RTU is governed by its own Director process, responsible for instantiating all subordinate agents, establishing MQTT communication, driving internal scheduling, and maintaining the operational state of the digital twin throughout the run. At startup, the Director receives its configuration through command-line arguments, including the DT-RTU identifier, run-specific working and logging directories, the session-derived MQTT base topic, and the path to the embedded Python engine. These parameters are propagated into system properties so that all agents share the same run context.

The Director constructs the DT-RTU in a strict order. A fresh Collector is created first, followed by the Visualizer that consumes its telemetry. The Counselor is then initialized with references to both the Collector and Visualizer, and finally the Actor is created. Each agent receives the correct run directory and timestamp. The Actor is additionally configured with the MQTT broker and base topic, after which its rule engine is armed. The Counselor is also started immediately by providing it with the Collector-defined *WqiSpec*, ensuring that prediction and WQI computation are aligned with the active feature bundle.

Once the agents are initialized, the director establishes full MQTT connectivity. It connects to the broker

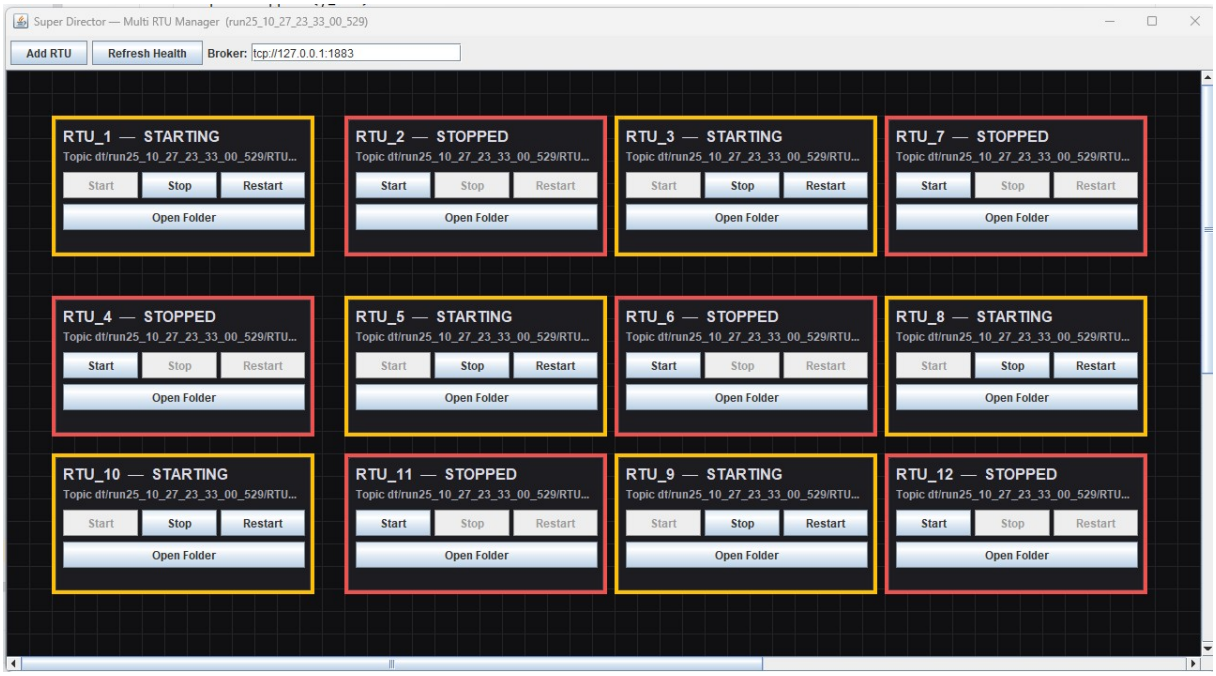

Figure 10: The UI of the Super-Director, capable of instantiating and managing the Digital Twin units, also passes the MQTT broker URL to them.

using an auto-reconnecting Paho client, subscribes to the `ctrl/stop` topic for graceful termination, and publishes its own retained lifecycle state. After announcing `STARTING` and completing initialization, the Director publishes `RUNNING` and keeps this state retained so the Super-Director can discover DT-RTUs instantly upon subscription. In parallel, the director transmits a 1,Hz heartbeat on `BASE/hb`, carrying the current epoch time. All MQTT messages use QoS,1 and rely on memory persistence, allowing the system to remain lightweight yet resilient.

Beyond basic state management, the Director also implements the DT-RTU health telemetry that the Super-Director displays. A separate single-threaded sampler runs every second and produces a JSON snapshot on `BASE/health/snapshot`. This snapshot contains timing drift information by comparing the collector's latest tick index to wall-clock time, an estimate of the expected tick period, JVM heap usage, available disk space for the run directory, MQTT connectivity status, and identifying metadata such as the DT-RTU ID and run stamp. The sampler also detects severe timing freezes—cases where no new collector ticks arrive for an extended interval—and in such situations emits a rising-edge alert on `BASE/health/alert`. These health messages are non-retained by design so that only live conditions are visible to the supervisor.

Two internal schedules keep the DT-RTU active regardless of UI activity: a dedicated actor ticker fires every 200,ms and invokes `actor.cycle()` to evaluate rules and dispatch MQTT tasks or warnings, while the heartbeat publisher runs once per second. This design ensures that actuation remains operational even if the Visualizer interface is idle, minimized, or under heavy load.

Shutdown is handled through a JVM shutdown hook. When the DT-RTU is terminated, whether via the `ctrl/stop` MQTT command or local exit, the director publishes a retained `STOPPED` state, disconnects from MQTT, ends the actor and counselor runs, stops the collector, and shuts down all schedulers. This grants that every run produces a consistent and complete termination sequence, and that the supervisor

immediately perceives the DT-RTU as offline.

While the director includes an optional lightweight HTTP interface exposing `/healthz` and `/stop`, this facility remains disabled by default and is no longer part of the operational path, having been superseded entirely by the MQTT-based supervision model.

In summary, the Director establishes the DT-RTU as an autonomous, fully instrumented digital-twin instance. It launches and binds all agents, manages MQTT presence and control channels, publishes continuous heartbeats and health snapshots, drives prediction-driven actuation, and grants a clean and observable shutdown sequence. This makes each DT-RTU a self-contained digital twin whose status, timing, and internal health can be monitored remotely and continuously by the Super-Director.

3.2.2.3 Recent Real-Time Window Platform

Classical monitoring frameworks and even other systems in general which could be categorized under the umbrella of *Real-Time Systems* focus on *what is happening now*; they display instantaneous sensor values with no regard for temporal continuity. On the other hand model training needs a prerequisite of historical data, so we can assume that the systems that exploit historical data to train their models can be called *Historical Systems*. We have contrived a new concept regarding the categorization of systems, ***Recent-Real-Time Systems*** as an iconoclastic approach opposed to the bipolar distinction of real-time systems, and historical systems. Our digital-twin instead of *what is happening now* emphasizes *what has just happened*, and on the other hand it does not require historical data, hence our digital twin is a ***Plug-and-Play predictive RRT system***. We exploit small, continuously sliding segments of recent history rather than a frozen current state. The visual reference point "now" defines the rightmost edge of the middle part of the chart and advances each time a new telemetry packet arrives, in a seismographic visual manner. This choice turns the temporal axis into a living timeline, where the observer perceives short-term system memory rather than discrete readings. Such formulation captures not only raw values but also their evolution across the immediate past, forming the foundation for coupling prediction segments ahead of real time.

In **Connector mode**, no assumptions are made about fixed data rhythms. Each DT-RTU may publish irregularly, intermittently, or in bursts, and yet the visualization remains coherent. This is achieved through an internal calibration process that measures the *Average Transmission Delay (ATD)* between received packets.

Let the i-th packet arrive at time t_i (seconds). During the first calibration stage, when 50 packets have been received, the visualizer computes:

$$\mathrm{ATD}_1 = \frac{t_{50} - t_0}{50}.$$

Figure 11: The calibrated plot according to the calculated ATD, the Recenet Real-Time Window is in the middle with green background, also the real curves are green, the future window has orange background and the predictions appear there in orange curves. Also because of the seismograpic behavior of the plots (tick-driven shift) curves end up in the historical window in the left with gray background

This value defines three temporal compartments:

$$T_{\text{recent}} = 50 \times \text{ATD}_1,$$

$$T_{\text{future}} = 10 \times \text{ATD}_1 \times \kappa,$$

$$T_{\text{hist}} = T_{\text{future}},$$

where $\kappa = 1.15$ is a small cushion factor ensuring that the predicted horizon does not visually collide with the real-time edge.

The **historical window**, having the same length as the first future window, extends the operator's visual awareness backward in time and provides a smoother temporal context for interpreting the latest changes.

After initialization, recalibration occurs every 10 new arrivals to maintain smooth adaptation:

$$\text{ATD}_m = 0.6\,\text{ATD}_{m-1} + 0.4\,\frac{t_N - t_{N-10}}{10}.$$

This exponentially smoothed ATD drives continuous adjustment of T_{recent} and T_{future}, so the visual timeline always reflects the true transmission tempo, even under non-periodic or jittered input.

Hence, the platform transitions from a timestamp-driven interface to a **self-synchronizing temporal environment** capable of matching any RTU's communication rhythm–including asynchronous, event-triggered, or disrupted sampling.

Once the recent window reaches equilibrium, the future window is activated. Every prediction cycle initiated by the Counselor generates a 10-step horizon extending $10 \times \text{ATD}$ seconds beyond "now". These

predicted values are plotted as a continuous curve immediately following the real-time trace, forming the system's *real-time-plus* capability:

$$\text{Monitoring:} \quad \underbrace{\text{past}}_{\text{historical}} + \underbrace{\text{present}}_{\text{recent real-time}} + \underbrace{\text{future}}_{\text{predicted horizon}} .$$

This unified continuum enables operators not only to see the system's most recent behavior but also to anticipate its short-term trajectory. The transition from monitoring "what just happened" to monitoring "what will happen" converts the platform from a reactive dashboard into a proactive and predictive Digital Twin, a bridge between observation and foresight. Figure 11 shows the window platform, the *Recent Real-Time Window* is shown in green background.

To bootstrap and validate inter-agent communication, a **CSV-reader mode** was implemented in the Collector. This mode replays historical datasets at configurable polling intervals of 1 s, 2 s, or 3 s, emulating real-time operation deterministically. Although not adaptive, it allowed the team to test synchronization, prediction logging, and MAE visualization with repeatable timing. Once the agent framework achieved stability, this deterministic environment was replaced by the adaptive ATD-calibrated Connector mode–transforming the Digital Twin from a scheduled simulator into a live elastic monitor capable of assimilating data from heterogeneous RTUs with arbitrary transmission frequencies.

3.2.3 Micro and supporting Agents

3.2.3.1 Micro-Agent: `global`

The `global` micro-agent is the simplest and most shared component in the entire system. It serves as the run-wide configuration nucleus, maintaining a tiny set of synchronized parameters that every agent depends on. All higher agents consult it to understand the temporal scale of the run, the list of active features, and whether a data stream is currently in progress.

Its necessity comes from the distributed nature of the Digital Twin: the Collector, the Counselor, the Visualizer, and the Actor each operate in different threads or separate processes, yet all of them must agree on the same basic truths. The polling interval defines the rhythm of the run; the ordered list of features must remain identical among agents; and the run-state flag must be visible to all components at all times. Centralizing these values in one minimal class prevents configuration mismatches and eliminates the need for heavier synchronization protocols. Declaring the fields as `volatile` ensures that updates made by one agent are immediately seen by all others.

In practice, the `global` agent provides a small shared memory space that the rest of the system reads from throughout a run. When a run begins, the run-state flag is set to true; when it ends, it is reset. Agents needing timing simply read the polling interval. Agents responsible for initializing features set the active feature list once before the run starts, after which it remains frozen for the entire duration of the session. This unobtrusive pattern keeps the system coherent and coordinated without drawing attention to itself.

Although nearly invisible in runtime logs, `global` silently holds the fundamental assumptions that make synchronized multithreaded operation possible. It carries the minimal set of universal truths such as feature scope, and run-state that bind all agents into a single coherent Digital Twin.

3.2.3.2 Micro-Agent: `WqiSpec`

`WqiSpec` is the micro-agent that formalizes the definition of the Water Quality Index (WQI) for a given run. It encapsulates the ordered set of features used in the computation and the exact numerical weights assigned to those features, both of which are frozen at construction time. The feature list becomes an unmodifiable sequence, and the weight vector is defensively copied to prevent any modification during execution. Because of this immutability, every agent interprets "WQI" in precisely the same way throughout the run.

The specification also exposes explicit regulatory thresholds, such as the upper limit for arsenic concentration. These remain as clear constants rather than being embedded deep within computational logic, making the numerical assumptions of the system both transparent and auditable. Each instance additionally computes a stable diagnostic hash, its `version()`, derived from the ordered feature list and the weight array. This identifier allows logs, agents, and external tools to verify that a particular run used a consistent and well-defined WQI formulation.

From a computational standpoint, `WqiSpec` provides the strict blueprint for how the WQI is produced. All agents rely on the same index alignment: the i-th element of the weight vector corresponds exactly to the i-th feature in the frozen feature list. Given a normalized feature vector

$$\mathbf{x} = (x_1, x_2, \ldots, x_n),$$

and the specification-defined weight vector

$$\mathbf{w} = (w_1, w_2, \ldots, w_n),$$

the WQI score is computed consistently across the system using the weighted-sum rule

$$\mathrm{WQI} = \sum_{i=1}^{n} w_i \, s_i(x_i),$$

where $s_i(\cdot)$ denotes the sub-index associated with the i-th feature. The weights are assumed to satisfy $\sum_i w_i \approx 1$, since normalization is performed upstream during WQI initialization.

This formulation is deliberately simple. Each term is transparent, and the contribution of each feature is governed entirely by its corresponding weight. The immutability of `WqiSpec` guarantees that this

Table 2: Adopted upper bounds for the 12 features used by the `utilitarian` agent

Feature	Max Value
Temperature (°C)	100.0
pH	14.0
Electrical Conductivity (µS/cm)	50000.0
Dissolved Oxygen (mg/L)	15.0
Hardness (mg/L as $CaCO_3$)	5000.0
Chlorides (mg/L)	20000.0
Total Organic Carbon (mg/L)	200.0
Fluoride (µg/L)	2000.0
Orthophosphate (mg/L as PO_4)	2.0
Nitrate (mg/L as NO_3)	3000.0
Ammonium Ion (µg/L as NH_4)	400000.0
Arsenic (µg/L)	300.0

computation is performed identically by the Collector, the Counselor, and the Visualizer.

In operational terms, `WqiSpec` functions as the immutable anchor of the water-quality computation layer. Once constructed at run start—based on the selected features and their redistributed weights—it defines the exact semantics of WQI for the entire session. No agent may reorder features or adjust weights thereafter. This ensures auditability, numerical consistency, and full reproducibility across runs, providing a single point of truth for how water quality is quantified within the Digital Twin.

3.2.3.3 Supporting Agent: `Utilitarian`

Before introducing the Utilitarian support agent, it is necessary to clarify the numerical envelopes used for feature normalization and WQI computation. To avoid ad-hoc assumptions, all upper bounds were aligned with globally recognized drinking-water and groundwater quality references, including the WHO Guidelines for Drinking-Water Quality [89], the EPA National Primary Drinking Water Regulations [90], the European Drinking Water Directive (EU 2020/2184) [91], UNESCO–IHP groundwater assessments [92], and the FAO AQUASTAT global water-quality compendium [93]. These documents collectively span regulatory thresholds, natural extremes, and documented outliers across diverse hydro-geological contexts. Since none of these parameters admit physically meaningful negative values, all minimum bounds are fixed at zero by design, ensuring a uniform and stable normalization baseline. Because our system must support heterogeneous RTUs and unpredictable field conditions, each upper bound was rounded slightly upward beyond the highest reported values in the literature, providing conservative headroom that prevents premature clipping and maintains scaling smoothness during SHAP-guided prediction cycles. Table 2 reports the final adopted maxima for all twelve features used by the Utilitarian.

`utilitarian` serves as the shared numerical toolbox of the system. It provides the canonical min–max anchors for every feature, implements normalization and denormalization routines, stores the reference WQI weight map for the full 12-feature scheme, constructs run-frozen WQI specifications, computes WQI from normalized inputs, assembles recent-window tensors for the Counselor–Visualizer pipeline, fills WQI horizons for prediction boxes, and exposes thin, safe wrappers for calling the Python `AIengine` through

JEP. In practice, it is the numerical bedrock that keeps all agents aligned.

The agent maintains a single source of truth for each feature's physical range. Normalization is implemented through the usual affine mapping

$$n = \frac{x - \min}{\max - \min},$$

followed by strict clamping into $[0, 1]$ to protect against out-of-range or corrupted sensor values. Denormalization simply reverses this mapping,

$$x = \min + n(\max - \min),$$

again with safe clamping. Feature-specific helpers (such as arsenic in μg/L, nitrates in mg/L as NO_3^-, etc.) wrap this logic with the correct physical domains.

A reference weight map for the full 12-feature WQI is defined internally and sums to unity. When a user selects a subset $\mathcal{F}$ of features for a run, the total weight of the unselected features is redistributed equally across all selected features. If w_i is the reference weight of feature i, and M is the total "missing mass" excluded by the current selection, then each selected feature receives an additional

$$\Delta = \frac{M}{|\mathcal{F}|}.$$

The resulting redistributed weights are passed to the constructor of a run-frozen `WqiSpec`, guaranteeing that all agents share exactly the same features, ordering, and weights throughout the run.

WQI computation based on a given specification follows a consistent structure. Let $\mathbf{n} = (n_1, \ldots, n_F)$ be the normalized values of the selected features, aligned with the `spec.features` order, and let $\mathbf{w} = (w_1, \ldots, w_F)$ be the redistributed weight vector stored in the spec. The WQI is computed through a feature-specific sub-index:

$$\mathrm{WQI} = \sum_{i=1}^{F} w_i \, s_i(n_i),$$

where s_i is defined by three rules: dissolved oxygen contributes linearly ($s_{\mathrm{DO}}(n) = n$), pH follows a soft bell curve peaking at neutral

$$s_{\mathrm{pH}}(n) = \max(0,\ 1 - 2|n - 0.5|)\,,$$

and all remaining features decrease linearly with their normalized value ($s_i(n) = 1-n$). An explicit safety rule is built in: if arsenic is part of the selected features and its denormalized concentration exceeds the fixed upper limit of $10\,\mu g\,L^{-1}$, the WQI immediately collapses to zero.

For the Counselor's temporal logic, the agent assembles $(F+1)\times 50$ tensors representing the concatenation of the five historical sub-bundles (L2, L1, M, R1, R2), with the final row reserved for WQI computed column-wise using the specification. Similarly, prediction boxes of dimension $(F+1) \times H$ are filled by applying the same WQI rule to each of the H horizon steps, ensuring that predicted WQI is always derived from the same spec as the real-time values.

Finally, the Utilitarian encapsulates the Python bridge. The wrapper `withInterpreter` ensures that the shared interpreter is initialized before use, executes the requested task inside the Python executor, and returns the result while propagating failures in a controlled manner. Two higher-level routines rely on this: one computes SHAP values from in-memory arrays, and the other generates batched multi-model predictions for all selected features. Both ensure a strict and safe boundary between Java and Python, while preserving numerical fidelity in both directions.

Taken together, Utilitarian provides the common arithmetic foundation of the Digital Twin. Every agent that touches numbers, the Collector, the Counselor, the Visualizer, and the Actor, ultimately depends on its domain scales, its normalization logic, its WQI definition, and its array-construction helpers. It is the shared substrate that enforces coherence, stability, and reproducibility across the entire system.

3.2.4 Collaborative agents

3.2.4.1 Agent Collector

The Collector Agent is the sensory inlet of the Digital Twin. It receives, preprocesses, and publishes the raw and normalized stream of environmental telemetry that becomes the factual substrate of the system. Its task is to translate the external world of sensors into a coherent, internally usable numeric reality. During early development, when the higher agents were still being engineered, a CSV-based reader mode was added so that telemetry could be replayed at controllable polling intervals. This allowed the us to design, debug, and refine the remaining agents with stable, repeatable inputs. In the final architecture, however, the Collector operates almost entirely in its remote arrival-driven configuration, through its Connector sub-agent which communicates over MQTT channels. The CSV mode remains as a supportive development tool; the Connector mode is the operational mode of the system.

A central design decision was to make the Collector fully *interval-agnostic*. SCADA installations differ radically in how often RTUs emit telemetry, sometimes every few seconds, sometimes every hour, occasionally in irregular bursts. The Collector never assumes a fixed cadence. Every inbound telemetry event, whether periodic or chaotic, is treated as a ***Tick***. This tick acts as the system's internal notion of time, akin to a clock edge in a CPU. It drives all sub-bundle shifts, window updates, prediction rollouts, logging operations, and state transitions across all agents. In a multi-RTU deployment, each Digital Twin instance maintains its own independent `tickIndex`, making the entire architecture *tick-driven*.

Although the two operational modes differ in how ticks are generated, they share the same computation pipeline. At the beginning of a run, the Collector freezes the active feature list $\mathcal{F} = \{f_1, \ldots, f_F\}$ either

from system configuration (CSV mode) or from the first received packet (Connector mode). It constructs a WQI specification tailored to this feature set, with redistributed weights and a fixed arsenic safety threshold. It also constructs a timing description, which the Counselor uses to interpret historical and future windows. It then allocates two synchronized state vectors,

$$lastRaw = (f_1, \ldots, f_F, WQI_{0\text{–}100}),$$
$$lastNorm = (n_1, \ldots, n_F, WQI_{0\text{–}1}),$$

which will be updated on every tick and made visible system-wide.

In CSV mode a scheduled thread reads one row per polling interval, parses the feature values, normalizes each value using the domain anchors in the Utilitarian, computes the WQI, assembles the snapshots, writes the arrival timestamp, and increments the tick index. When the end of the file is reached, the Collector rewinds to the first data row and continues indefinitely, enabling unlimited regression-testing and reproducibility during development.

In Connector mode, the Collector listens to a telemetry topic where RTUs publish JSON payloads. Each payload, regardless of its timing, constitutes a tick. If the features are not yet frozen, the Collector infers the order from the inbound packet or applies the expected order supplied by the UI, depending on configuration. It then normalizes each raw value using the feature-specific anchors.

3.2.4.2 Agent Counselor

The Counselor serves as the analytical core of the Digital Twin. It is responsible for interpreting the most recent telemetry, performing predictive reasoning through embedded Python models, and guiding the visual layer of the system. Unlike passive agents, the Counselor is alive as a **state machine**, cycling through well-defined operational phases. Each phase corresponds to a distinct temporal buffer that accumulates, interprets, or projects data. Its function is to continuously maintain equilibrium between what has occurred recently (recent bundles), and what will occur (predicted horizon). The Counselor stands between two major streams: the Collector, from which it receives normalized data in real time, and the Visualizer, to which it publishes prediction segments, SHAP evaluations, and model-assignment decisions. In doing so, it embodies the cognitive loop of the digital twin: perception, reasoning, and foresight. In the following we delve into the design and procedures of the Counselor.

3.2.4.2.1 State Machine Design: Internally,considering the recent real-time window, the Counselor evolves through a deterministic sequence of phases, each corresponding to the progressive filling of a ten-sample sub-bundle per feature. At startup, the system is empty and begins accumulating real, normalized telemetry in the first sub-bundle "leftmost" (L2). Each sub-bundle holds ten samples per feature. Once a sub-bundle becomes full, the Counselor computes its SHAP signature for that ten-sample block and then advances to the next sub-bundle. After the fifth sub-bundle "rightmost" (R2) is filled, the state machine transitions into *STEADY_RIGHTMOST*.

In steady state, the machine reuses the SHAP signatures already stored in the four older sub-bundles

(L2, L1, M(Middle), R1). For every new batch of ten truth samples in the rightmost sub-bundle (R2), it computes SHAP only for this newest block, and then derives a single "global" SHAP vector by applying a weighted aggregation across all five sub-bundles. This aggregated SHAP vector drives the per-feature model selection, after which the system rolls the sub-bundles leftward, performs a prediction, and opens a new future horizon.

We can express the control logic as reported in Algorithm 1.

Algorithm 1 Counselor State Machine for SHAP–Guided Prediction

```
state ← PRIMING_LEFTMOST
while run_active do
    row ← Collector.lastNorm()
    append row to current_bundle(state)
    if bundle_full(state) then
        // Compute SHAP only for the bundle that just filled
        compute_SHAP(current_bundle(state))
        if state == PRIMING_RIGHTMOST then
            state ← STEADY_RIGHTMOST
        else
            advance_state()
        end if
    end if
    if state == STEADY_RIGHTMOST and bundle_full(R2) then
        // L2, L1, M, R1 already have SHAP signatures
        // R2 has just received its SHAP signature
        shap_global ← weighted_SHAP(L2, L1, M, R1, R2)
        performPredictionCycle(shap_global)
    end if
end while
```

3.2.4.2.2 Matrix Structure and Flow of Data: Each sub-bundle $\mathcal{B}^{(k)}$ is a rectangular matrix of size $(2F + 2) \times 11$, where F denotes the number of selected features and the extra two rows correspond to the real and predicted (calculated from the formula) WQI. The ten main columns $(0 \dots 9)$ represent the prediction horizon of ten ticks, while the last column (10) stores auxiliary values (SHAP or MAE, depending on phase). Each bundle therefore encapsulates both real and predicted telemetry, as well as per-feature statistics used for model evaluation and ranking.

During priming phase, only real normalized telemetry is available. Each sub-bundle gradually fills only its upper half with real feature values and their calculated WQI. Column 10 holds SHAP importance values obtained from Python after the tenth column is filled.

$$\mathcal{B}^{(k)}_{\text{priming}} = \left[\begin{array}{cccc|c} x_{1,1} & x_{1,2} & \cdots & x_{1,10} & \text{shap}_1 \\ x_{2,1} & x_{2,2} & \cdots & x_{2,10} & \text{shap}_2 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{F,1} & x_{F,2} & \cdots & x_{F,10} & \text{shap}_F \\ \hline wqi_1 & wqi_2 & \cdots & wqi_{10} & - \\ \hline & & \cdots & & \\ & & \cdots & & \\ \vdots & \vdots & & \vdots & \vdots \\ & & \cdots & & \\ \hline & & \cdots & & \end{array}\right]$$

Here, each $x_{f,h}$ is the normalized real value of feature f at offset step (tick of the sub-bundle) h, and wqi_h is the computed Water Quality Index at the same tick.

Once the Counselor transitions into the steady phase, it generates a ten-step *prediction box* $\mathcal{P}$ at every segment cycle (each 10 ticks), which holds the predicted feature trajectories and their projected WQI:

$$\mathcal{P} = \begin{bmatrix} \hat{x}_{1,1} & \hat{x}_{1,2} & \cdots & \hat{x}_{1,10} \\ \hat{x}_{2,1} & \hat{x}_{2,2} & \cdots & \hat{x}_{2,10} \\ \vdots & \vdots & & \vdots \\ \hat{x}_{F,1} & \hat{x}_{F,2} & \cdots & \hat{x}_{F,10} \\ \widehat{wqi}_1 & \widehat{wqi}_2 & \cdots & \widehat{wqi}_{10} \end{bmatrix}$$

The prediction box is then written in the bottom half of the current rightmost bundle $\mathcal{B}^{(R_2)}$ when the new segment is published.

When the corresponding ground-truth values arrive for the predicted horizon, the rightmost bundle becomes complete and the MAEs are stamped into column 11 (array column with index 10). The resulting matrix $\mathcal{B}^{(R_2)}_{\text{final}}$ thus holds both the observed and predicted trajectories, along with per-feature and WQI error metrics:

$$\mathcal{B}^{(R_2)}_{\text{final}} = \left[\begin{array}{ccccc} x_{1,1} & x_{1,2} & \dots & x_{1,10} & \text{shap}_1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{F,1} & x_{F,2} & \dots & x_{F,10} & \text{shap}_F \\ wqi_1 & wqi_2 & \dots & wqi_{10} & \text{mae}_{\text{mean}} \\ \hline \hat{x}_{1,1} & \hat{x}_{1,2} & \dots & \hat{x}_{1,10} & \text{mae}_1 \\ \vdots & \vdots & & \vdots & \vdots \\ \hat{x}_{F,1} & \hat{x}_{F,2} & \dots & \hat{x}_{F,10} & \text{mae}_F \\ \widehat{wqi}_1 & \widehat{wqi}_2 & \dots & \widehat{wqi}_{10} & \text{mae}_{\text{wqi}} \end{array}\right]$$

In this structure:

- The top block $(0 \dots F)$ stores real values and SHAP scores, and the lowest element in the top half stores the mean of all MAEs.
- The bottom block $(F+1 \dots 2F+1)$ stores predicted values and MAEs.
- The last column (index 10) serves as a stamp field—SHAP in the upper block, MAE in the lower block.

At the end of each prediction cycle (10 ticks), all sub-bundles roll one step to the left. This leftward shift ensures that the Counselor always maintains a five-sub-bundle temporal context (L_2, L_1, M, R_1, R_2), which comprises the recent-real-time bundle (or window), while continually refreshing the rightmost bundle with the latest horizon predictions. A schematic view of this steady-right evolution is shown below in Figure 12.

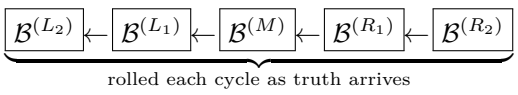

Figure 12: Sub-Bundle rolling scheme

This rolling mechanism fuses real, predicted, and validated data into a continuously learning pipeline, where each sub-bundle is a self-contained memory cell of the Counselor's steady-state cognition.

3.2.4.2.3 AIengine.py: The Embedded Python Intelligence Layer: Java alone, though structurally reliable, lacks the deep ecosystem of modern machine-learning libraries; Python, conversely, provides a mature body of regressors, explainers, and optimizers. The `AIengine.py` module therefore acts as an embedded analytical coprocessor inside the JVM through JEP, forming the Python side of the SHAP–MAE Engine. Its invocation does not start an interpreter from scratch: a single Python environ-

ment is initialized at run start, held in memory, and reused indefinitely. As a result, all SHAP analyses and forecasting operations execute in-place with no serialization overhead and no inter-process latency, allowing each predictive cycle to complete in milliseconds.

Through this integration, the Counselor gains direct in-memory access to Python's scientific stack–NumPy, scikit-learn, and model explainers–while remaining entirely within Java's runtime and scheduling. The AIengine unifies the two ecosystems as a *stateless, callable module* exposing a minimal API that the Counselor invokes synchronously via `utilitarian.withInterpreter()`, ensuring that every Python task runs inside a dedicated thread and returns deterministically to Java.

During operation, the Counselor continuously hands the AIengine normalized telemetry matrices. In the priming phase, each ten-sample sub-bundle is sent solely for the computation of local SHAP scores, enabling the explanatory structure required before forecasting begins. Once the system transitions to steady operation, the Counselor transmits two data slices at every prediction cycle: (i) the newest ten-sample sub-bundle for SHAP computation, and (ii) the most recent fifty samples of each feature for forecasting. The AIengine responds with both the SHAP importance vector for the new sub-bundle and the predicted ten-step horizon for each feature.

A unified model registry inside the AIengine allows the Counselor to request any predictor by name. Each model is instantiated only when needed (lazy construction), fitted on the forty overlapping training pairs extracted from the most recent fifty samples of the feature, and then used to generate a ten-step recursive forecast through a sliding-window autoregressive routine. No model persists across cycles; every learning episode is local, lightweight, and derived entirely from the feature's immediate history.

Through this architecture, `AIengine.py` serves as the embedded numerical core of the Counselor–executing SHAP analyses, performing autoregressive forecasting, and enabling the model-selection mechanism, while remaining tightly synchronized with Java through a deterministic ten-tick handshake. The result is a hybrid intelligence layer in which Java provides structural control and timing, and Python provides fast, domain-agnostic learning and explainability.

For explainability over a ten-sample subbundle, the engine trains a light `ExtraTreesRegressor` on

$$X \in \mathbb{R}^{10\times F} \quad \text{(ten time steps, } F \text{ features)},$$
$$y \in \mathbb{R}^{10} \qquad \text{(ten WQI values)}.$$

then computes SHAP values with `shap.Explainer`. If $\mathbf{S} \in \mathbb{R}^{10\times F}$ denotes the SHAP matrix, the engine returns mean absolute contributions per feature:

$$\bar{\mathbf{s}} \in \mathbb{R}^{F}, \qquad \bar{s}_j \;=\; \frac{1}{10}\sum_{t=1}^{10}\left|S_{t,j}\right|.$$

These $\bar{s}_j$ are stamped into the sub-bundle's last column and consumed by the Counselor for feature ranking and model assignment. These scores indicate how strongly each feature contributed to the WQI pattern inside that ten-sample window.

Five consecutive subbundles $(L2, L1, M, R1, R2)$ are processed in this way. Their SHAP scores are then combined using fixed weights $1, 2, 3, 4, 5$ to form a single smoothed importance vector $S_i^{(w)}$:

$$S_i^{(w)} \;=\; \frac{1}{15}\Big(1\,S_i^{L2} + 2\,S_i^{L1} + 3\,S_i^{M} + 4\,S_i^{R1} + 5\,S_i^{R2}\Big), i = 1, \ldots, F.$$

This weighted vector provides the current ranking of feature contributions and hence their relative importance. At the very beginning of the run, the user has already chosen and ranked the models, so the ranked features pair with the ranked models for the first prediction round.

From that point onward, the feature–ranking mechanism remains entirely SHAP-driven, but the model ranking becomes performance-driven: after every prediction cycle (10 ticks), the system evaluates how well each model performed.

At the end of each prediction cycle, the Counselor receives the real values corresponding to the ten predictions just made. For each feature f (with $f = 1, \ldots, F$), the Mean Absolute Error over the 10-step horizon is computed exactly as in the implementation:

$$\mathrm{MAE}_f = \frac{1}{10}\sum_{j=1}^{10} |\, y_{f,j} - \hat{y}_{f,j} \,|\,,$$

where j indexes the ten forecasted time steps, $y_{f,j}$ is the true value of feature f at step j, and $\hat{y}_{f,j}$ is the predicted value returned by the AIengine.

To keep a running estimate of how reliable each model has been historically, the system updates that model's error using an exponential moving average (EWMA):

$$\mathrm{MAE}_m^{\mathrm{new}} = (1-\alpha)\,\mathrm{MAE}_m^{\mathrm{old}} + \alpha\,\mathrm{MAE}_f,$$

where $\alpha \in (0, 1)$ controls how fast the recent error influences the model's long-term score. The rolling MAEs are used only to maintain a ranking of the models. In each cycle, the ranked models are paired with the ranked features (from SHAP), so that the best-performing model is assigned to the most important feature, the second-best model to the second feature, and so on. The whole process can be summarized in Algorithm 2.

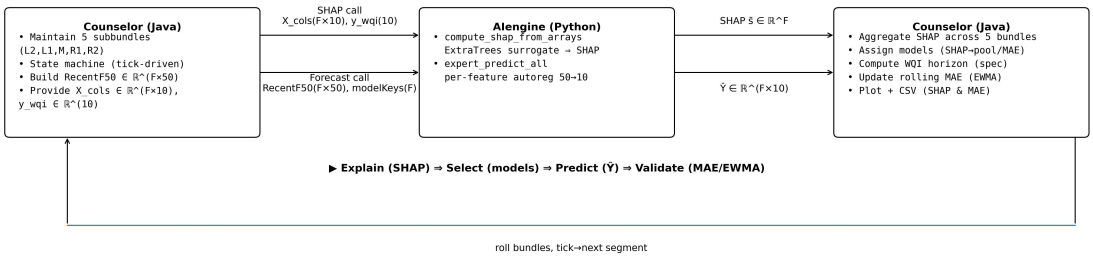

Figure 13: Cyclic interaction between the Counselor and the AIengine. At each cycle, the Counselor sends feature recent-realtime sub-bundles (segments) for SHAP analysis; once five sub-bundles are available, they are assembled into a 50-sample bundle for forecasting. The AIengine returns SHAP scores and predicted horizons, and the Counselor updates model selection, WQI, and rolling MAE before advancing to the next cycle.

Algorithm 2 Cycle of the SHAP–MAE Engine (sub-bundle driven)

For each tick:
if a sub-bundle of 10 samples has just completed **then**
    Explain: compute SHAP on the new 10-sample sub-bundle
    store the SHAP scores in its SHAP column
    if five sub-bundles are now present $(L2, L1, M, R1, R2)$ **then**
        Aggregate: compute weighted SHAP importance $S^{(w)}$ over the five sub-bundles
        Select Models: rank features by $S^{(w)}$ and models by rolling MAE; pair ranked features with ranked models
        Predict: $\hat{Y} \leftarrow$ AIengine.expert_predict_all$(X_{F\times 50}, \text{models})$
        compute WQI$(\hat{Y})$ over the 10-step horizon
        publish the prediction segment
    end if
end if
if a post-prediction sub-bundle of 10 new truth samples completes **then**
    Validate: compute per-feature MAE over the horizon
    update each model's rolling MAE using EWMA: $\text{MAE}_m^{\text{new}} = (1-\alpha)\text{MAE}_m^{\text{old}} + \alpha\text{MAE}_f$
end if

This cyclical pipeline is self-correcting: the stronger the model's historical accuracy, the more it dominates the next prediction round. Simultaneously, SHAP ensures that dominance remains explainable rather than arbitrary. Figure 13 is a schematic explaining the process.

For each feature, the engine keeps the most recent fifty normalized samples:

$$\mathbf{s} = [s_1, s_2, \ldots, s_{50}] \in \mathbb{R}^{50}.$$

From this short history, the regressor learns how to predict the next value from the previous ten. In other words, it tries to approximate a function

$$M : [s_t, s_{t+1}, \ldots, s_{t+9}] \mapsto s_{t+10}.$$

To do this, forty overlapping training pairs are extracted from the fifty samples. Each pair uses ten consecutive inputs and the immediately following sample as the target. The design and target matrices are therefore:

$$X \in \mathbb{R}^{40\times 10}, \quad X_{i,\cdot} = [s_i, s_{i+1}, \ldots, s_{i+9}],$$
$$y \in \mathbb{R}^{40}, \qquad y_i = s_{i+10}, \qquad i = 1, \ldots, 40.$$

The last window starts at s_{40} and ends at s_{49}, predicting s_{50}; any later window would require data that does not yet exist. Hence, the most recent fifty points yield exactly forty valid input–output pairs for local training.

After fitting the model M on (X, y), the next ten values are generated recursively:

$$\begin{aligned}
\hat{s}_{51} &= M([\, s_{41},\, s_{42},\, \ldots,\, s_{50}\,]),\\
\hat{s}_{52} &= M([\, s_{42},\, s_{43},\, \ldots,\, s_{50},\, \hat{s}_{51}\,]),\\
\hat{s}_{53} &= M([\, s_{43},\, s_{44},\, \ldots,\, s_{50},\, \hat{s}_{51},\, \hat{s}_{52}\,]),\\
&\vdots\\
\hat{s}_{60} &= M([\, s_{50},\, \hat{s}_{51},\, \hat{s}_{52},\, \ldots,\, \hat{s}_{59}\,]).
\end{aligned}$$

Thus, the model repeatedly observes a ten-point sliding window, learns the short-term pattern inside it, and extends the feature's trajectory by forecasting the next ten steps one by one. Algorithm 3 explains the process in pseudocode.

Algorithm 3 Recursive 10-step autoregression

function PREDICT10_DISPATCH(*model_name*, *recent_50*)
  $M \leftarrow$ `factory(model_name)` ▷ lazy construction
  return `_predict10_autoreg`(*recent_50*, M)
end function
function _PREDICT10_AUTOREG(*recent_50*, M)
  Build (X, y) from *recent_50* with lag $L = 10$ ▷ $X \in \mathbb{R}^{40\times 10}$, $y \in \mathbb{R}^{40}$
  Fit M on (X, y)
  $window \leftarrow [s_{40}, \ldots, s_{49}]$
  for $k = 1$ **to** 10 **do**
    $\hat{s} \leftarrow M(window)$
    append $\hat{s}$; $window \leftarrow window[2..10] \,\|\, \hat{s}$
  end for
  return $[\hat{s}_{50}, \ldots, \hat{s}_{59}]$
end function

We employ a collection of 24 predictive ML models, organized into coherent methodological families. This structure enables a fair comparison of models, provides fine-grained control over their behavior, and facilitates performance analysis across heterogeneous operating regimes.

The model families and their constituents are as follows:

Linear Models

- **Linear Regression** [94]

  Linear regression models the conditional mean of a continuous target variable as a linear combination of the inputs. The model assumes a Gaussian likelihood:

$$p(y \mid x, \theta) = \mathcal{N}\big(y \mid w^\top x,\, \sigma^2\big)\,,$$

  where w is the vector of regression coefficients and σ^2 is the observation noise variance. This formulation implies that the response y follows a normal distribution centered around the linear prediction $w^\top x$, with constant variance σ^2 across all observations. Predictions are made by projecting the input x onto w, with deviations treated as normally distributed residuals.

  The model remains linear in parameters even with a nonlinear feature map $\phi(x)$, preserving tractability while accommodating mild nonlinearities through basis expansions. Parameters are estimated by maximizing the likelihood, which is equivalent to minimizing the sum of squared residuals due to the Gaussian assumption. This solution has a geometric interpretation: the fitted values correspond to the orthogonal projection of the response vector onto the subspace spanned by the inputs, ensuring the residuals are orthogonal to the column space of the design matrix.

  We implement the model using `sklearn.linear_model.LinearRegression`, refitting on the most recent data window $X_{\text{train}} \in \mathbb{R}^{40\times 10}$ at each update. This ensures coefficients reflect short-term temporal structure while maintaining computational efficiency through the normal equations solution.

- **Ridge Regression** [94]

  Ridge regression stabilizes ordinary least squares by adding an ℓ_2 penalty, which corresponds to maximum a posteriori (MAP) estimation with a zero-mean Gaussian prior on the weights. The objective function combines data fidelity with regularization:

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}\big(y_i - (w_0 + w^\top x_i)\big)^2 + \lambda\|w\|_2^2,$$

  where $\lambda \geq 0$ controls the shrinkage strength. The ℓ_2 penalty $\|w\|_2^2 = \sum_{j=1}^{p} w_j^2$ shrinks coefficients toward zero, reducing model variance at the cost of introducing bias. The closed-form solution demonstrates the regularization effect:

$$\hat{w}_{\text{ridge}} = (\lambda I + X^\top X)^{-1} X^\top y.$$

The addition of λI to the Gram matrix $X^\top X$ improves conditioning, making the solution more stable when features are correlated or when $N \approx p$. This is particularly beneficial for our short, autoregressive windows where the 40×10 design matrix may be ill-conditioned.

We use `Ridge(alpha=1.0)` with regularization strength $\lambda = 1.0$, providing substantial shrinkage to stabilize estimates while preserving predictive performance on our temporal sequences.

- **Lasso Regression** [94]

  Lasso regression employs an ℓ_1 penalty $\|w\|_1 = \sum_{j=1}^{p} |w_j|$, equivalent to MAP estimation under a Laplace prior $p(w_j) \propto e^{-\lambda|w_j|}$. The objective function:

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} \big(y_i - (w_0 + w^\top x_i)\big)^2 + \lambda \|w\|_1$$

  encourages sparsity by driving some coefficients to exactly zero, performing embedded feature selection. Unlike ridge regression's smooth shrinkage, the geometry of the ℓ_1 ball creates corners at the axes where coefficients can be exactly zero. The solution involves soft-thresholding:

$$\hat{w}_j = \text{sign}(\hat{w}_j^{\text{OLS}})(|\hat{w}_j^{\text{OLS}}| - \lambda)_+,$$

  where $(x)_+ = \max(0, x)$, setting coefficients with insufficient magnitude to zero.

  We implement `Lasso(alpha=0.001)` with mild regularization strength $\lambda = 0.001$, allowing the model to drop weak lag dependencies in our 40-sample windows while maintaining stability through coordinate descent optimization.

- **Elastic Net** [94]

  Elastic Net combines ℓ_1 and ℓ_2 penalties to overcome limitations of pure Lasso, particularly when predictors are correlated or when $p > N$. The objective function:

$$J(w) = \|y - Xw\|^2 + \lambda_2 \|w\|_2^2 + \lambda_1 \|w\|_1$$

  provides the sparsity of Lasso while maintaining the stability of Ridge. For $\lambda_2 > 0$, the criterion is strictly convex, yielding a unique solution that encourages grouping of correlated variables—when predictors are correlated, Elastic Net tends to include or exclude them together, unlike Lasso which may arbitrarily select one.

  The hybrid penalty structure performs double shrinkage, which can be corrected through parameter rescaling, but the core benefit remains: balanced regularization that avoids Lasso's instability while

retaining feature selection capabilities.

We use `ElasticNet(alpha=0.001, l1_ratio=0.5)` where `alpha` $= \lambda_1 + \lambda_2$ controls total regularization and `l1_ratio` $= \lambda_1/(\lambda_1 + \lambda_2) = 0.5$ specifies equal Lasso-Ridge mixing, providing balanced regularization for our multivariate time series.

- **Bayesian Ridge Regression** [94]

  Bayesian linear regression treats weights w as random variables rather than fixed parameters. With Gaussian likelihood $p(y \mid X, w, \sigma^2) = \mathcal{N}(y \mid Xw, \sigma^2 I)$ and Gaussian prior $p(w) = \mathcal{N}(w \mid w_0, V_0)$, the posterior remains Gaussian due to conjugacy:

$$p(w \mid X, y, \sigma^2) = \mathcal{N}(w \mid w_N, V_N),$$

  where

$$V_N^{-1} = V_0^{-1} + \frac{1}{\sigma^2} X^T X, \quad w_N = V_N \left(V_0^{-1} w_0 + \frac{1}{\sigma^2} X^T y \right).$$

  The posterior mean w_N balances prior beliefs with data evidence, while the covariance V_N quantifies parameter uncertainty. The predictive distribution integrates over both parameter and observation uncertainty:

$$p(y_* \mid x_*, D) = \mathcal{N}\left(x_*^T w_N, \sigma^2 + x_*^T V_N x_* \right),$$

  producing calibrated uncertainty estimates that widen for extrapolations.

  We use `BayesianRidge()` which places spherical Gaussian priors and learns hyperparameters via evidence maximization, providing automatic regularization adapted to our short windows while quantifying prediction uncertainty.

- **Huber Regression** [95]

  Huber regression robustifies least squares by using a loss function that transitions from quadratic to linear beyond a threshold k:

$$L(r) = \begin{cases} \frac{1}{2} r^2, & |r| \le k, \\ k|r| - \frac{1}{2} k^2, & |r| > k, \end{cases}$$

  where $r = y - \hat{y}$ is the residual. This hybrid loss maintains the efficiency of least squares for small errors while reducing the influence of outliers through linear penalization of large errors. The

transition point k controls the trade-off between efficiency and robustness, with $k \approx 1.35$ providing 95% efficiency for Gaussian errors.

The Huber loss is convex and differentiable, enabling efficient optimization while offering robustness to heavy-tailed error distributions that may occur in sensor measurements or transient anomalies.

We implement `HuberRegressor(epsilon=1.35, alpha=1e-4)` with default threshold $k = 1.35$ and mild L2 regularization $\alpha = 10^{-4}$, providing robustness to outliers in our streaming SCADA data while maintaining stability.

- **ARD Regression** [94]

  Automatic Relevance Determination (ARD) implements Bayesian variable selection through a hierarchical prior structure. Each coefficient receives an individual precision parameter:

$$p(w) = \mathcal{N}(w \mid 0, A^{-1}), \quad A = \mathrm{diag}(\alpha_1, \ldots, \alpha_D),$$

  where each α_j controls the prior precision for w_j. The marginal likelihood after integrating out w is:

$$p(y \mid X, \alpha, \beta) = \mathcal{N}\left(y \mid 0, \beta^{-1} I + X A^{-1} X^T\right),$$

  and maximizing this marginal likelihood (evidence maximization) drives $\alpha_j \to \infty$ for irrelevant features, effectively forcing $w_j \approx 0$. The posterior over weights is Gaussian with mean:

$$\mu = \beta \Sigma X^T y, \quad \Sigma^{-1} = \beta X^T X + A,$$

  where sparsity emerges through the precision parameters rather than explicit ℓ_1 penalization.

  We use `ARDRegression` with uninformative Gamma hyperpriors set via `alpha_1=1e-6, alpha_2=1e-6, lambda_1=1e-6, lambda_2=1e-6`, allowing automatic feature relevance determination during fitting without strong prior assumptions.

- **Linear Support Vector Regression (LinearSVR)** [94]

  LinearSVR uses an ε-insensitive loss that ignores errors smaller than ε:

$$L_\varepsilon(y, \hat{y}) = \max(0, |y - \hat{y}| - \varepsilon),$$

  focusing optimization on substantial prediction errors. The complete objective combines this loss

with L2 regularization:

$$J = C \sum_{i=1}^{N} L_\varepsilon(y_i, \hat{y}_i) + \tfrac{1}{2}\|w\|^2,$$

where C controls the trade-off between margin size and training errors. The ε-insensitive zone creates a tube around the predictions where errors are not penalized, yielding sparse solutions where only points outside the tube (support vectors) influence the model.

This formulation provides robustness to small fluctuations while focusing model capacity on capturing significant patterns, particularly beneficial for noisy temporal data.

We implement `LinearSVR(epsilon=0.01, C=1.0, max_iter=2000)` with narrow insensitive zone $\varepsilon = 0.01$ and balanced regularization $C = 1.0$, making it suitable for capturing dominant trends in our short windows.

- **Stochastic Gradient Descent (SGD)** [94]

  Stochastic Gradient Descent approximates the true gradient using individual examples, making it suitable for online learning and small-sample settings. The parameter update at iteration k follows:

$$w_{k+1} = w_k - \eta_k \left(w_k^\top x_k - y_k\right) x_k,$$

  where η_k is the learning rate and (x_k, y_k) is a single training example. This stochastic approximation provides computational efficiency and inherent regularization through the noise in gradient estimates, often improving generalization compared to batch gradient descent.

  The learning rate schedule crucially affects convergence: diminishing rates ensure convergence, while adaptive methods maintain progress in flat regions. The noise in gradient estimates helps escape shallow local minima, particularly beneficial for non-convex problems.

  We use `SGDRegressor(loss="squared_error", penalty="l2", alpha=5e-4, learning_rate="adaptive", eta0=0.01, max_iter=500)` with squared loss, L2 regularization strength $\lambda = 0.0005$, and adaptive learning rate starting at $\eta_0 = 0.01$, providing efficient optimization for our 40-sample windows.

Kernel and Distance-Based Models

- **Support Vector Machines (SVM)** [**ref:Hastie2009**] A powerful kernel-based method for both classification and regression that finds optimal separating boundaries by maximizing margins between classes. For regression (SVR), the model minimizes the regularized loss function:

$$H(\beta, \beta_0) = \sum_{i=1}^{N} V_\epsilon(y_i - f(x_i)) + \frac{\lambda}{2}\|\beta\|^2$$

where the ϵ-insensitive loss function is defined as:

$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon \\ |r| - \epsilon & \text{otherwise} \end{cases}$$

The solution takes the form:

$$\hat{f}(x) = \sum_{i=1}^{N} (\hat{\alpha}_i^* - \hat{\alpha}_i) K(x, x_i) + \beta_0$$

where only a subset of coefficients are non-zero, defining the support vectors. In our implementation, key parameters include: `kernel="rbf"` specifying the radial basis function kernel $K(x, x') = \exp(-\gamma \|x - x'\|^2)$, `C=1.0` controlling regularization strength (inverse of λ), `epsilon=0.01` setting the margin of tolerance where no penalty is applied, and `gamma="scale"` automatically setting kernel bandwidth as $1/(n\_features \cdot \text{var}(X))$. The method effectively handles nonlinear relationships through kernel functions while maintaining robustness to outliers.

- **k-Nearest Neighbors (KNN) Regression** [94]

  KNN is a nonparametric method that predicts using local averaging of the k nearest training examples. The distance-weighted prediction:

$$\hat{y}(x) = \frac{\sum_{i \in \mathcal{N}_k(x)} d(x_i, x)^{-1} y_i}{\sum_{i \in \mathcal{N}_k(x)} d(x_i, x)^{-1}}, \quad d(x_i, x) = \|x_i - x\|_2$$

  assigns higher influence to closer neighbors, providing a smooth local approximation. The Euclidean distance $\|x_i - x\|_2$ measures similarity in the input space, with closer points presumed to have similar responses.

  As a lazy learner, KNN defers all computation to prediction time, making it highly adaptive to local data structure without explicit model assumptions. The choice of k controls the bias-variance tradeoff, with smaller k yielding more flexible fits and larger k providing smoothing.

  We use `KNeighborsRegressor(n_neighbors=5, weights="distance", p=2)` with $k = 5$ neighbors, inverse-distance weighting, and Euclidean distance, creating an adaptive local smoother for our temporal patterns.

- **Kernel Ridge Regression (KRR)**

  KRR combines Ridge regularization with the kernel trick, performing linear regression in a high-

dimensional feature space implicitly defined by the kernel. The dual solution:

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i \kappa(x_i, x)$$

shows that predictions are kernel expansions, with coefficients $\alpha = (K + \lambda I)^{-1} y$ obtained by solving a regularized linear system. The RBF kernel $\kappa(x, x') = \exp(-\gamma \|x - x'\|^2)$ enables modeling of nonlinear relationships while the Ridge penalty λ controls complexity by shrinking coefficients in the feature space.

Unlike SVR, KRR uses all training examples in predictions but benefits from the same kernel-based nonlinear modeling capabilities. The method provides smooth interpolations and naturally handles complex regression surfaces.

We implement `KernelRidge(kernel="rbf", alpha=1.0, gamma=0.2)` with RBF kernel bandwidth $\gamma = 0.2$ and regularization $\lambda = 1.0$, packaged in a standardization pipeline to ensure proper kernel scaling.

- **Gaussian Process Regression (GPR)** [94]

  Gaussian Processes define a distribution over functions, specified by a mean function $m(x)$ and covariance kernel $\kappa(x, x')$:

$$f(x) \sim \mathcal{GP}(m(x), \kappa(x, x')).$$

  For any finite set of inputs, the function values follow a multivariate Gaussian:

$$p(f \mid X) = \mathcal{N}(f \mid \mu, K), \quad K_{ij} = \kappa(x_i, x_j).$$

  With Gaussian observations $y = f(x) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma_y^2)$, the predictive distribution at new points is Gaussian:

$$p(f_* \mid x_*, X, y) = \mathcal{N}\Big(f_* \,\Big|\, k_*^\top (K + \sigma_y^2 I)^{-1} y,\ k_{**} - k_*^\top (K + \sigma_y^2 I)^{-1} k_*\Big).$$

  where $k_* = [\kappa(x_*, x_1), \ldots, \kappa(x_*, x_N)]$ and $k_{**} = \kappa(x_*, x_*)$. The mean provides the prediction while the variance quantifies uncertainty, naturally increasing away from observed data.

  The RBF kernel $\kappa(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2} \|x - x'\|^2\right)$ with length scale ℓ controls the smoothness of functions in the prior.

We implement `GaussianProcessRegressor` with composite kernel `RBF(length_scale=1.0)` + `WhiteKernel(noise_level=1e-3)`, capturing both the signal correlation and observation noise in our temporal sequences.

Bagging Ensembles

- **Bagging (Bootstrap Aggregating)** [**ref:Hastie2009**]

  Bagging reduces variance by averaging predictions across multiple bootstrap samples of the training data. Given a base estimator $\hat{f}(x)$ trained on data $Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, bagging generates B bootstrap samples Z^{*b} and computes the aggregated prediction:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x).$$

  This Monte Carlo estimate approximates the true bagging estimate $\mathbb{E}_{\hat{P}}[\hat{f}^*(x)]$, where $\hat{P}$ is the empirical distribution placing probability $1/N$ on each data point. Bagging is particularly effective for nonlinear or adaptive estimators like decision trees, where each bootstrap tree typically involves different features and terminal nodes. For classification, the bagged classifier selects the class with the most votes from the B trees, while for regression it averages the numerical predictions.

  We implement `BaggingRegressor` with `n_estimators=60` bootstrap samples, using `DecisionTreeRegressor(max_depth=4, min_samples_leaf=2)` as the base estimator to control individual tree complexity. The ensemble provides variance reduction through bootstrap aggregation while maintaining the interpretability benefits of tree-based models.

Tree-based models

- **Decision Trees (CART)** [94] Decision trees represent a fundamental approach to nonparametric regression and classification by recursively partitioning the feature space into rectangular regions. The model operates as an adaptive basis function expansion where each basis function corresponds to an indicator function for a particular region of the input space. The prediction function takes the form:

$$f(\mathbf{x}) = \sum_{m=1}^{M} w_m \mathbb{I}(\mathbf{x} \in R_m)$$

  where R_m denotes the m-th region defined by a sequence of axis-aligned splits, and w_m represents the constant prediction within that region (typically the mean response for regression or the majority class for classification).

  Tree construction employs a greedy, top-down algorithm that recursively partitions the data. At each node, the algorithm selects the splitting variable j and threshold t that maximize the purity of the resulting partitions. For classification tasks, common impurity measures include the Gini

index:

$$\text{Gini} = 1 - \sum_{c=1}^{C} \hat{\pi}_c^2$$

and cross-entropy, while regression trees minimize the residual sum of squares. The recursive partitioning continues until meeting predefined stopping criteria, such as maximum tree depth or minimum samples per node.

The model's flexibility necessitates careful regularization to prevent overfitting. Key hyperparameters include `max_depth`, which constrains the tree's complexity; `min_samples_leaf`, which ensures sufficient data in terminal nodes for stable estimates; and `random_state` for reproducibility. While decision trees offer intuitive interpretability and handle heterogeneous data types naturally, their piecewise constant structure and greedy construction algorithm make them susceptible to high variance, motivating the development of ensemble methods that aggregate multiple trees.

- **Random Forests** [94] An ensemble method designed to address the high variance of individual decision trees by combining multiple de-correlated trees. The core idea is to build a collection of trees, where each tree f_m is trained on a bootstrap sample of the original data (bagging) and uses a random subset of features at each split. The final prediction is obtained by averaging the outputs of all trees in the ensemble:

$$f(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} f_m(\mathbf{x})$$

This dual randomization—of training data and features—reduces variance and decorrelates the individual trees, leading to significantly improved generalization over a single tree. Key implementation parameters include `n_estimators=60` controlling the number of trees in the ensemble, `max_depth=4` and `min_samples_leaf=2` regulating individual tree complexity, `bootstrap=True` enabling bagging, and `n_jobs=1` specifying parallel processing. While the ensemble loses the straightforward interpretability of a single decision tree, it typically achieves state-of-the-art predictive performance and is robust to overfitting.

- **Extra-Trees** [**ref:Geurts2006**] An ensemble method that builds upon the random forest framework by introducing additional randomization during tree construction. The key innovation lies in how splits are determined: while random forests find the optimal split point for randomly selected features, Extra-Trees selects split points completely at random for each candidate feature. This approach, combined with using the entire dataset rather than bootstrap samples, creates more diverse trees and reduces computational complexity. The prediction is obtained by aggregating outputs from all trees in the ensemble. Key parameters include `n_estimators=80` controlling the ensemble size, `max_depth=4` constraining tree complexity, and `max_features` (defaulting to all features) determining feature subset size at each split. The increased randomization provides stronger variance reduction while maintaining computational efficiency through simplified split selection.

Boosting-type Ensembles

- **AdaBoost (Adaptive Boosting)** [94] An ensemble method that combines multiple weak learners (typically decision trees with depth 1, called *stumps*) through a weighted majority vote. The algorithm operates by iteratively reweighting training examples, focusing more on previously misclassified instances. At each iteration m, AdaBoost minimizes the exponential loss:

$$L_m(\phi) = \sum_{i=1}^{N} w_{i,m} \exp(-\beta \tilde{y}_i \phi(x_i))$$

where $w_{i,m} = \exp(-\tilde{y}_i f_{m-1}(x_i))$ are the current weights. The optimal weak learner ϕ_m is selected by minimizing the weighted error rate $\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(\tilde{y}_i \neq \phi_m(x_i))}{\sum_{i=1}^{N} w_{i,m}}$, and its weight is computed as $\beta_m = \frac{1}{2} \log \frac{1-\text{err}_m}{\text{err}_m}$. The model update is $f_m(x) = f_{m-1}(x) + \beta_m \phi_m(x)$, with weights updated via $w_{i,m+1} = w_{i,m} \exp(2\beta_m I(\tilde{y}_i \neq \phi_m(x_i)))$. In our implementation, key parameters include `n_estimators=120` controlling the number of boosting stages, `learning_rate=0.1` scaling the contribution of each weak learner (equivalent to $\nu\beta_m$ where ν is the learning rate), and `random_state=0` ensuring reproducibility.

- **Gradient Boosting Regression Trees (GBRT)** [**ref:Hastie2009**] A powerful ensemble method that builds regression trees sequentially through gradient descent optimization. The algorithm minimizes a differentiable loss function by iteratively adding weak learners that correct the errors of the current ensemble. The objective function takes the form:

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i))$$

where $f(x)$ is constrained to be a sum of regression trees. At each iteration m, the algorithm proceeds in three key steps: first, it computes the negative gradient (pseudo-residuals) of the loss function with respect to the current predictions:

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

For squared error loss, these reduce to ordinary residuals $y_i - f_{m-1}(x_i)$. Second, a regression tree is fitted to these pseudo-residuals, partitioning the input space into terminal regions R_{jm}. Finally,

the model is updated by adding the tree predictions scaled by an optimal step size:

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$$

where γ_{jm} minimizes the loss within each region. In our implementation, `n_estimators=120` specifies the total number of boosting stages (iterations M), controlling the ensemble size and model complexity. The `max_depth=3` parameter constrains the size of individual trees (J_m), preventing overfitting by limiting tree complexity and ensuring weak learners. The `random_state=0` ensures reproducible tree construction across runs. This sequential refinement process enables GBRT to capture complex patterns while maintaining robustness through controlled tree growth.

- **Histogram-Based Gradient Boosting Tree (HGBT) [96]**: We implement a Histogram-Based Gradient Boosting Tree (HGBT) following the scikit-learn implementation . This method represents an efficient variant of gradient boosting that leverages histogram-based algorithms for split finding, offering substantial speed advantages over traditional gradient boosting, particularly for large datasets. The core algorithm operates by discretizing continuous feature values into histogram bins, which dramatically reduces the computational complexity of finding optimal splits from $O(n\_features \times n\_samples)$ to $O(n\_features \times n\_bins)$ where $n\_bins \ll n\_samples$. During training, gradient statistics are accumulated in these histogram bins, and optimal split points are determined by scanning through bins rather than sorting individual feature values. Our implementation is configured with max_iter=120 boosting iterations, a learning rate of 0.1 to shrink tree contributions and prevent overfitting, max_depth=3 to control individual tree complexity, max_leaf_nodes=15 to limit tree size, min_samples_leaf=2 to ensure sufficient data in terminal nodes, and random_state=0 for reproducible results. The histogram-based approach provides natural regularization through binning while maintaining competitive predictive performance with significantly reduced training time.
- **XGBoost (eXtreme Gradient Boosting) [97]**: We employ XGBoost, a scalable tree boosting system known for its computational efficiency and state-of-the-art performance on various machine learning tasks. XGBoost implements gradient boosting with several key innovations including a regularized learning objective to prevent overfitting, handling of sparse data through sparsity-aware split finding, and weighted quantile sketch for approximate tree learning. The model builds an ensemble of trees where the prediction is the sum of outputs from K additive functions:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\mathbf{x}_i)$$

with each f_k representing a regression tree. Our configuration uses the histogram-based tree method (`tree_method="hist"`) for efficient split finding, with `n_estimators=80` trees in the ensemble,

`max_depth=3` to control individual tree complexity, and `learning_rate=0.1` to shrink tree contributions. We apply L2 regularization through `reg_lambda=1.0` and employ stochastic gradient boosting via `subsample=0.9` for instance sampling and `colsample_bytree=0.9` for feature sampling per tree. The model is configured for regression tasks using `objective="reg:squarederror"` and utilizes a single job with `random_state=0` for reproducible results.

- **LightGBM:** [98] We employ LightGBM, a highly efficient gradient boosting framework that introduces two novel techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). Unlike conventional gradient boosting implementations that scan all data instances for split finding, LightGBM utilizes histogram-based algorithms that group continuous feature values into discrete bins. This approach significantly reduces computational requirements by working with binned data rather than individual data points. The GOSS technique focuses computational resources on data instances with large gradients, which contribute more to model improvement, while randomly sampling from instances with smaller gradients. The EFB technique combines features that rarely have non-zero values simultaneously, effectively reducing the number of features without substantial information loss. Our configuration uses 120 boosting iterations with a learning rate of 0.05 and unlimited tree depth through leaf-wise growth. We apply instance sampling at 90% and feature sampling at 90% per tree, with no L2 regularization. The implementation employs column-wise histogram building for optimization and ensures reproducible results through fixed random seeding.

- **CatBoost:** [99] We employ CatBoost, a gradient boosting framework that introduces ordered boosting and ordered target statistics to address prediction shift in conventional gradient boosting. Unlike other implementations that suffer from target leakage during gradient estimation and categorical feature encoding, CatBoost utilizes permutation-driven approaches that create unbiased estimates. The ordered boosting mechanism trains on different data subsets using a time-series inspired approach where each training example's gradient is computed using a model that hasn't seen that example. For categorical features, CatBoost calculates target statistics using only the "historical" data from random permutations, preventing information leakage. Our configuration uses 200 boosting iterations with a learning rate of 0.1 and tree depth of 4 to balance model complexity and generalization. We employ RMSE loss for regression tasks, disable verbose logging for clean execution, and ensure reproducibility through fixed random seeding. The implementation runs on CPU with single-threading to prevent resource contention and disables file writing for optimized performance in constrained environments.

Neural Networks

- **Multi-Layer Perceptron (MLP)** [94]

  Multi-Layer Perceptrons are feedforward neural networks that learn nonlinear mappings through layered functional transformations. For regression with two hidden layers, the model takes the form:

$$p(y|x,\theta) = \mathcal{N}(y|w^T z(x), \sigma^2),$$

$$z(x) = g(Vx) = [g(v_1^T x), \ldots, g(v_H^T x)]$$

where g is a nonlinear activation function, $z(x)$ represents the hidden layer as a deterministic function of the input, H is the number of hidden units, V is the weight matrix from inputs to hidden nodes, and w is the weight vector from hidden nodes to the output. The nonlinearity of g is essential—without it, the model collapses into a linear regression $y = w^T(Vx)$. MLPs are universal approximators capable of modeling any sufficiently smooth function given enough hidden units.

We implement `MLPRegressor` with architecture `hidden_layer_sizes=(32,16)` creating two hidden layers of 32 and 16 neurons respectively, using ReLU activation for nonlinear transformation. The model employs the Adam optimizer with learning rate 0.001 and includes early stopping with patience of 10 iterations and tolerance 10^{-3} to prevent overfitting. Training uses mini-batches of size 32 with maximum 300 epochs, providing efficient nonlinear modeling of complex temporal dependencies while maintaining computational tractability.

Choice of ML over deep learning. Deep architectures with millions of parameters are empirically powerful when trained on very large datasets, but in the Digital Twin each feature's training set within a prediction cycle is intentionally small, and models are refit frequently. In this regime, classical machine-learning regressors—regularized linear models, kernel methods, Gaussian processes, and tree ensembles—offer a better trade-off: they converge quickly, remain stable under repeated retraining, and their inductive biases are well documented in the literature. The `AIengine` therefore favors a broad but *shallow* inventory of such models, leaving deep learning to future extensions where longer historical archives and larger cross-site datasets become available.

Lazy instantiation and registry. In implementation, each model is produced by a small factory()) and registered in a dictionary `PREDICTORS` : key $\mapsto$ factory. The Counselor requests models by string keys, and a new estimator is instantiated only when needed for a feature. This keeps Python object creation and heavy imports off the critical prediction path, while still exposing the full methodological catalogue whenever the SHAP–MAE engine decides a model is worth testing on a given feature.

3.2.4.3 Agent Actor

The Actor is the operational executor of the Digital Twin and the component that closes the twinning loop. While the Collector ingests and normalizes live telemetry, the Counselor predicts feature trajectories, and the Visualizer exposes them to the operator, the Actor is responsible for turning these analytical states into concrete field actions or supervisory alerts. It operates in a fully tick-driven pattern synchronized with the Collector: at each tick, the Visualizer calls `actor.cycle()`, which reads the current tick index, the last raw telemetry vector from the Collector and, when available, the most recent prediction box from the Counselor. All outgoing messages are published via MQTT under an RTU-specific base topic and are simultaneously appended to `actor.jsonl` in the per-run log directory, ensuring that every actuation decision leaves a durable trace.

At each tick, the Actor receives the latest raw telemetry vector `lastRaw` from the Collector and, when the prediction pipeline is active, the predictive box `predictionbox` and timeline index `ti` from the Counselor.

It iterates over the fixed feature list `global.FEATURES`, aligning each feature name with its position in `lastRaw`, and transparently handles configurations where the raw vector contains an additional WQI tail element by simply ignoring the last position and evaluating only the physical features. For each feature, a corresponding `FeatureRule` object encodes its rule key, units, comparator (greater-than or less-than), enabled flag, threshold, and target OPC–UA `NodeId`. The Actor maintains a small runtime record for each feature containing its last state, last emission timestamp (kept for compatibility), and most recent raw value; this state is later exposed in the Visualizer as a tri-light bar showing the instantaneous supervision status of every monitored variable. Even features that do not trigger direct actuation still participate in the evaluation process, so the supervisory picture remains coherent across the entire feature bundle.

The decision logic for each feature is best understood as a tri-state model. A feature is considered normal when its real-time measured value is on the safe side of its configured threshold. It enters the alarm state when the real-time measured value crosses the threshold in the configured direction, as determined by a simple comparison between the current raw value and the threshold in the `evaluate()` method. In parallel, an additional stand-by alert condition or warning is derived from the Counselor's predictions: the Actor projects each normalized prediction point back to raw units using the same denormalization functions as the rest of the system (e.g., `denormTemp`, `denormEC`, `denormAs_u`), scans up to a configurable prediction horizon, and checks whether any future predicted value would breach the rule. If such a predicted breach is detected, an early standby warning is produced, indicating that the system is on a trajectory towards violation even if the current real value is still within bounds. In this way, the Actor fuses real-time measurement and near-term forecast into a single operational decision per feature and per tick.

The tri-state regime is reflected both in the messages and in the user interface. Each feature's state is mapped to the Visualizer's tri-light bar: green for normal, orange whenever a prediction-based standby warning is active, and red whenever the measured value is in direct breach. Only one light is logically active per feature at a time, giving the operator an immediate visual snapshot of which parameters are healthy, which are under predicted risk, and which are in critical violation requiring attention. Figure 14 shows the dedicated Actor tab: on the top, tri-light bars summarize per-feature status, while in the lower area the outgoing JSON payloads are listed as a chronological log, mirroring the content of `actor.jsonl`.

The mapping from rule keys to actions is kept deliberately simple and transparent. For each feature, the Actor stores a `FeatureRule` that binds a human-readable rule key to a semantic action and to a concrete OPC–UA `NodeId`. For example, `TEMP_HIGH` corresponds to initiating cooling or otherwise mitigating high temperature, `DO_LOW` corresponds to increasing aeration when dissolved oxygen falls too low, and `AS_HIGH` instructs the system to isolate a contaminated source by closing the relevant pump. Other rules such as `COND_HIGH`, `TOC_HIGH`, or `NH4_HIGH` primarily raise alarms or drive process adjustments, but in all cases the Actor's responsibility is to convert the abstract notion of a threshold breach into a concrete Boolean command targeted at a specific `NodeId`. Table 3 summarizes the default per-feature rules and their intended actions; these defaults are encoded directly in the Java implementation by constructing `FeatureRule` objects with appropriate thresholds, units, and OPC–UA identifiers.

From a messaging perspective, the Actor is intentionally minimal. For each RED event, it constructs a JSON payload that includes an identifier `id`, an ISO-formatted timestamp `ts_iso`, the feature name

`feature`, the rule key `rule`, the current severity (typically "HIGH"), the measured value `value`, the units `units`, and an `opc_ua` section containing the list of commands. Each command is currently a Boolean write instruction to a configured `NodeId`, with `action="write"`, `datatype="Boolean"`, and `value=true`. These payloads are serialized via Jackson and published over MQTT to an RTU-specific topic tree: the director or Super-Director sets a base topic such as `dt/run<SESSION>/<RTU_k>/`, and the Actor publishes RED tasks on `<base>actor/tasks`. In addition, a compact breadcrumb log with only the task identifier, feature, and rule may be sent to `<base>actor/log`, offering a lightweight stream suitable for dashboards or monitoring tools. For ORANGE events, a prediction-based standby JSON payload is built with similar metadata but without an `opc_ua` section; instead, it includes the predicted breach value, the prediction time index at which the breach is expected, and an expiry index. These standby messages are published on `<base>actor/warn` and also appended to `actor.jsonl`, so the entire evolution of both warnings and commands is preserved.

Internally, the Actor keeps a small outbox of recent tasks. Every alert emission constructs a `TaskRecord` object that records the identifier, tick, feature, rule, severity, value, units, node identifier, target MQTT topic, and serialized JSON payload, together with simple status fields for dry-run or error codes. These records are retained in a bounded deque with a default capacity of 200 entries, exposed via a snapshot method that the Visualizer uses to populate the on-screen log. In addition to the streaming `actor.jsonl` file, a structured configuration snapshot is maintained as `actor_config.json`, which holds the trigger policy, the entire set of feature rules, and the MQTT broker, base topic, and authentication parameters. Both files are written in a per-run log directory shared with the rest of the DT-RTU agents, so that all decisions made by the Actor can be replayed or audited after the fact.

The decision logic that underlies Actor's behavior is summarized in Algorithm 4, which expresses the per-tick loop over features, the derivation of the feature state from real and predicted values, and the conditional construction and emission of JSON payloads.

Algorithm 4 Per-tick Rule Evaluation and MQTT Emission (Actor Cycle)

```
for each tick t do
    for each feature f do
        obtain x_f^real and x_f^pred
        determine state_f ∈ {NORMAL, STANDBY, ALERT}
        if emitCondition(state_f) then
            build JSON payload
            publish via MQTT and append to file
        end if
    end for
end for
```

From a system perspective, the Actor is the component that makes the digital twin operational rather than purely observational. The Collector and Counselor build a high-fidelity, time-synchronized view of the process state and its near-future evolution; the Visualizer translates that view into human-readable plots, normalized indices, and intuitive tri-light indicators; and the Actor is the mechanism that binds this analytical layer back to the physical plant by emitting OPC–UA-oriented tasks in real time under an RTU-specific MQTT namespace. Because the Actor shares the same tick index as the other agents and writes into the same per-run directory, each emitted task or warning can be aligned exactly with the

Table 3: Per-feature rules and actions implemented by the Actor Agent. Rule keys and thresholds match the defaults encoded in the Java implementation.

Feature	Rule Key	Action	Cmp./Thr.
Temperature	TEMP_HIGH	Cooling	>30 °C
pH	PH_HIGH	Chemical dosing	>9.0
DO	DO_LOW	Aeration	<5.0 mg/L
EC	COND_HIGH	Conductivity alarm	>2000 µS/cm
Hardness	HARDNESS_HIGH	Softener control	>180 mg/L
TOC	TOC_HIGH	Organic alarm	>2.0 mg/L
Chlorides	CL_HIGH	Salt purge	>250 mg/L
Fluorides	F_HIGH	Fluoride alarm	>2.0 mg/L
Orthophosphates	PO4_HIGH	Nutrient alarm	>0.10 mg/L
Nitrates	NO3_HIGH	Nitrate alarm	>10.0 mg/L
Ammonium Ion	NH4_HIGH	Ammonia alarm	>0.50 mg/L
Arsenic	AS_HIGH	Close pump	>10 µg/L

underlying telemetry and predictions that motivated it. In practice, the additional latency introduced by the Actor is negligible: constructing the payload, serializing it to JSON, and publishing to MQTT typically remains well below a few milliseconds per tick, and in dry-run mode these operations are confined to local file writes and in-memory bookkeeping.

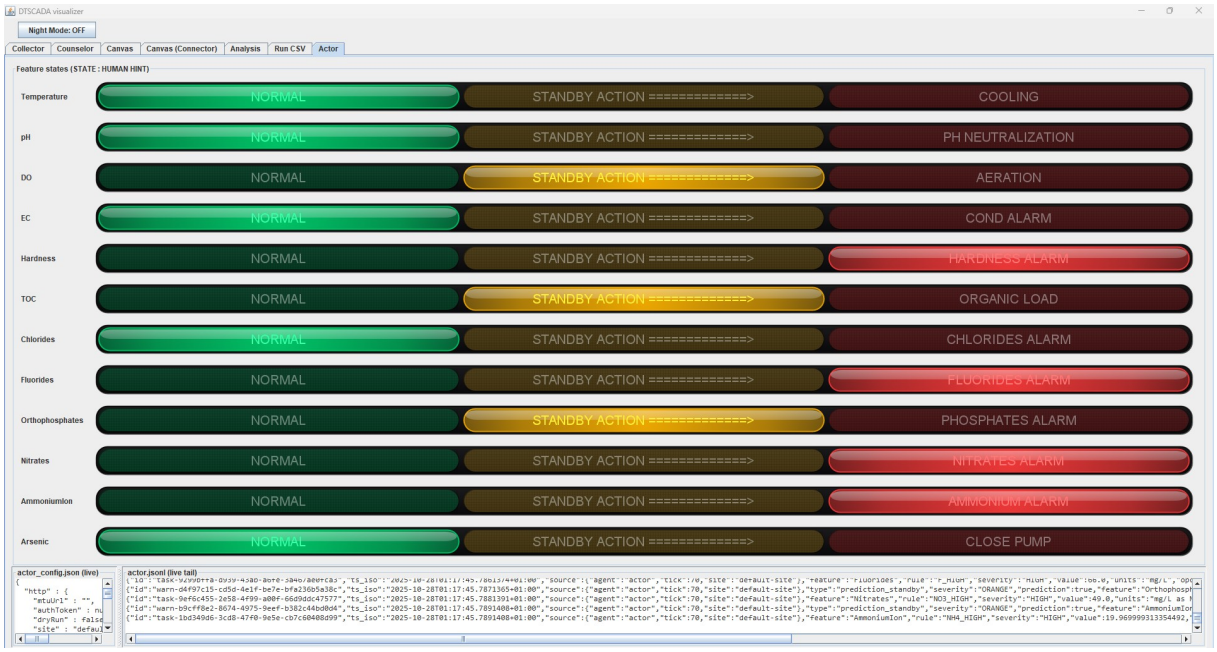

Figure 14: Agent Actor's visual tab in which the commands are both shown in the log area in JSON format and visually summarized using tri-color lights per feature.

—

3.2.4.4 Agent Visualizer

The Visualizer agent functions as the collaborative hub and synchronization orchestrator of the VACCSMEDUS architecture, providing both real-time monitoring and historical archiving while maintaining

temporal coherence across the entire digital twin ecosystem. It does not generate or analyze data autonomously, but instead synchronizes, visualizes, and archives the live interaction among the Collector,the Counselor, and the Actor agents within a unified temporal framework that adapts dynamically to the data source characteristics.

The Visualizer receives raw and normalized telemetry from the Collector at each tick, with incoming normalized values being dynamically denormalized using feature-specific scaling functions (utilitarian.denormTemp(), utilitarian.denormPH(), etc.) before plotting and archival in the run's visualizer_log_<stamp>.csv. Each new telemetry frame is aligned within the current real-time window, the dynamically calibrated windows to preserve temporal consistency across plots and logs, while maintaining backward compatibility through the prLog buffer that stores up to 400 PredRealRow objects for immediate analysis panel display before CSV persistence.

Through the plotpredictionboxVisualizerFromCounselor() function, the Visualizer receives 10-step prediction boxes containing float[][] arrays where rows correspond to monitored features and the final row represents WQI predictions. These predictions are plotted as future trajectories using distinct orange-colored series and merged with real data once ground truth arrives, triggering automatic completion of pending PredRealRow entries in both in-memory buffers and CSV files. The Visualizer also displays corresponding model performance metrics which are the denormalized MAE values calculated through denormMaeByNames() and updates model-selection indicators in real time, reflecting the current model chosen by the Counselor.

The Visualizer maintains a comprehensive error tracking system through dual CSV outputs: errors_-<stamp> .csv records denormalized MAE values for all features plus WQI, while visualizer_log_-<stamp>.csv maintains synchronized sequences of predicted and actual values with timestamp alignment. These files are automatically created with appropriate headers when a new run begins, with the Visualizer managing write pointers (errorsRowsWritten, visualizerRowsWritten) to prevent duplication and ensure sequential logging. The system employs auto-reload timers (csvReloadTimer, vizLogReloadTimer) that periodically refresh the CSV tab display and update the Analysis panel with the most recent prediction-reality comparisons.

A 500ms polling loop (startActorLightsRefresh()) retrieves the Actor's LightRow snapshots and updates tri-state indicator lamps. Each feature lamp displays both operational state and human-readable hints (STATE : HUMAN HINT) such as "PH HIGH — DOSE ACID" or "CL HIGH — BLEND/RO", with red labels dynamically mapped through RED_LABEL_BY_FEATURE including specialized responses like "CLOSE PUMP" for Arsenic alerts and "START AERATION" for Dissolved Oxygen violations. The Visualizer also provides live tail viewing of actor configuration and JSONL logs through actorFilesTimer, which monitors file modification times and appends new content without loading entire files into memory.

Through its Counselor tab interface, the Visualizer provides interactive model management capabilities including initial feature-to-model mapping through ordered selection lists, model registry browsing with double-click addition, and ranking visualization through custom-drawn panels (FeatureRankPanel, ModelRankPanel, MappingMatrixPanel) that display SHAP scores and rolling MAE metrics. The Initiate button validates selection counts against global.FEATURES size before committing mappings to the Counselor, while maintaining backward compatibility through featureToModel maps and automatic

synchronization with Collector feature selections.

The Visualizer's significance extends beyond graphical display to encompass state management, timing coordination, archival integrity, and user interaction across multiple operational contexts, maintaining a synchronized and traceable flow of information throughout the entire DT runtime.

3.3 Results and discussion

With twelve monitored features and twenty-four candidate machine-learning models, the number of possible model-feature configurations becomes extremely large; thus, assessing every combination would be practically unbounded. For this reason, we designed a performance experiment creating four simultaneous DTs each representing a simulated RTU (in Python using jupyter notebook), each with an initial transmission interval set at 4 seconds transmitting the downloaded data from different starting points, and each in *full-feature* mode, where all features are concurrently predicted within a single competitive framework. In the counselor tab, we selected a combination of the models to be implemented for that DT, considering having 24 models, in this manner, the top half, the bottom half, the one in the middle starting from the first model, and finally the one in the middle starting from the second model:

1. DT-RTU1 models: *adaboost*, *ard*, *bagging*, *bayesianridge*,*catboost*, *dtree*, *enet*, *extratrees*, *gbrt*, *gpr*, *hgbt*, *huber*
2. DT-RTU2 models: *knn*, *krr*, *lasso*, *lightgbm*, *linearsvr*, *linreg*, *mlp*, *rforest*, *ridge*, *sgd*, *svm*, *xgboost*
3. DT-RTU3 models: *adaboost*, *bagging*, *catboost*, *enet*, *gbrt*, *hgbt*, *knn*, *lasso*, *linearsvr*, *mlp*, *ridge*, *svm*
4. DT-RTU4 models: *ard*, *bayesianridge*, *dtree*, *extratrees*, *gpr*, *huber*, *linreg*, *rforest*, *sgd*, *xgboost*

In this way of selection, each model is selected two times for two of the DTs each., Each ATD ≈ 4.23. They were started and stopped almost at the same time, and 124 rounds of predictions was selected as the basis of comparison.Figure 15 presents the normalized MAE evolution for all monitored features and the Water Quality Index (WQI) across the four RTUs. Despite minor oscillations, most of the features maintain low MAE values throughout the experiment, indicating consistent predictive stability of the SHAP-guided model selection mechanism. Sporadic peaks correspond to transient regime transitions or short-term disturbances in the telemetry, confirming that the system dynamically adapts its model assignment without sustained error propagation across DT-RTUs. Another notable observation from this diagram is the synchronization mechanism which is inherent in the VACCSMEDUS architecture, validated through the simultaneous operation of multiple Digital Twin instances. All instances maintained coherent tick-aligned operation throughout the 124 prediction rounds, demonstrating that the Recent Real-Time Window (RRTW) framework successfully decouples each Digital Twin from fixed timing assumption, because also the MQTT connectivity added further undetermined delay to each transmission. The consistent MAE patterns across all four RTUs further confirm that the SHAP-MAE Engine's performance is independent of the underlying transmission schedule, validating that synchronization within each DT is achieved through the tick-driven design rather than external timing coordination. This tick-agnostic behavior is precisely what enables plug-and-play deployment across heterogeneous RTUs with arbitrary transmission patterns.

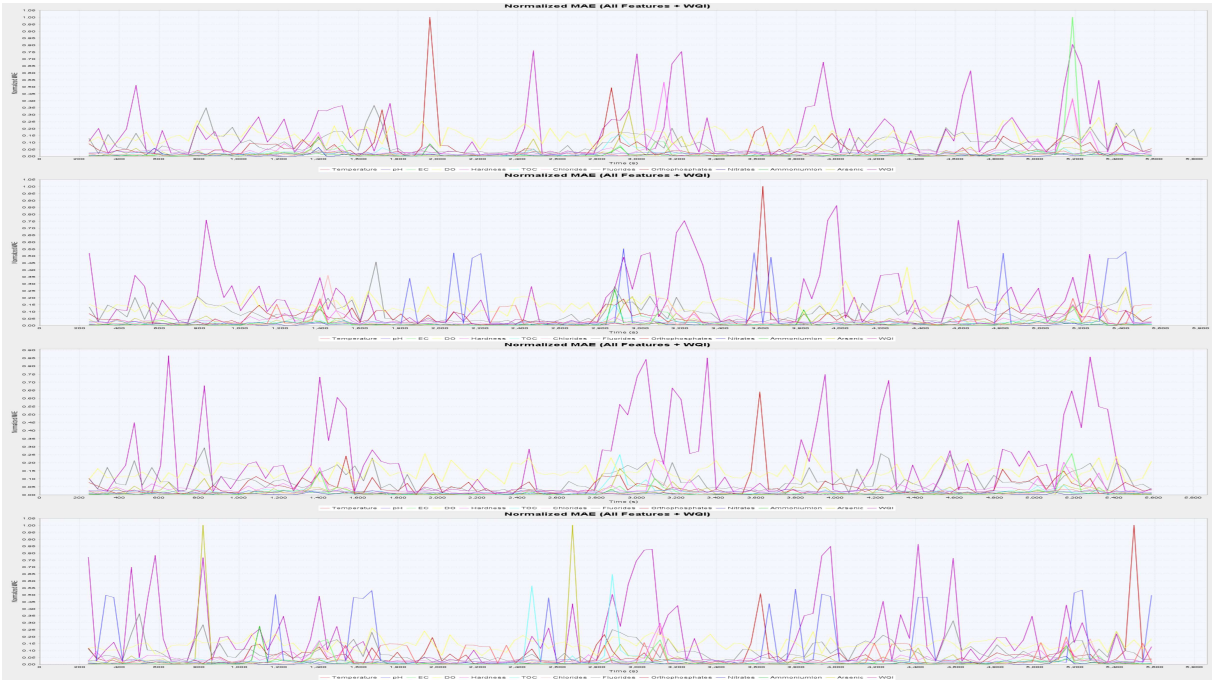

Figure 15: Normalized Mean Absolute Error (MAE) trends for all 12 features and WQI across the four RTUs during the complete experiment. Each panel corresponds to one RTU (top to bottom: RTU 1–RTU 4), showing real-time MAE evolution over all prediction rounds. The curves remain mostly stable with occasional spikes reflecting transient regime shifts or abrupt water-quality fluctuations.

We have another way of demonstrating the efficiency of our system with another type of diagram, since our system is based on the prediction of features through switching models we present a diagram in Figure 16, which shows the complete schematic of what has been performed in the system, in the sense that the higher the plot of a feature is, the higher was its overall SHAP scores throughout the whole run across 4 DT-RTUs. Also the x-axis order of the model denotes their overall performance throughout the whole run of 124 rounds of predictions across all RTUs, meaning that we now have the criteria upon which we can base the selection of the best models for future runs, 12 from left to right and ranked as the Figure 17 shows the overall performance in this criteria.

In order to validate and confirm the ranking or selection of models, meaning selecting 12 best out of 24 in the beginning, we ran another simultaneous series of Digital Twins with the initial selection of: *huber*, *lightgmb*, *ridge*, *rforest*, *knn*, *linearsvr*, *sgd*, *adaboost*, *ard*, *lasso*, *bagging*, and *gbrt* for all 4 DTs with the same ATD ≈ 4.23, Considering the same number of the rounds of prediction, 124. The results in the logs, in particular the counselor csv log of each DT, gave us complete MAEs for all features and WQI, as a calculated variable, therefore we compared the average of the first series of the WQI_MAEs, to the second series (with the top performing selection of models in the first series), the results confirm the competitive mechanism is meaningful and directly lead to the selection of best models, as shown in Figure 18 in the radar chart, the MAEs of the second series are lower than the first series of runs, confirming the validity of the competitive SHAP_MAE_Engine mechanism.

In order to obtain another validation of the architecture, especially about the mechanism of model selection which is done through the SHAP-MAE Engine, we processed our counselor log files of the first series of DT-RTUs and the second series (totaling 8), and produced a schematic diagram shown in

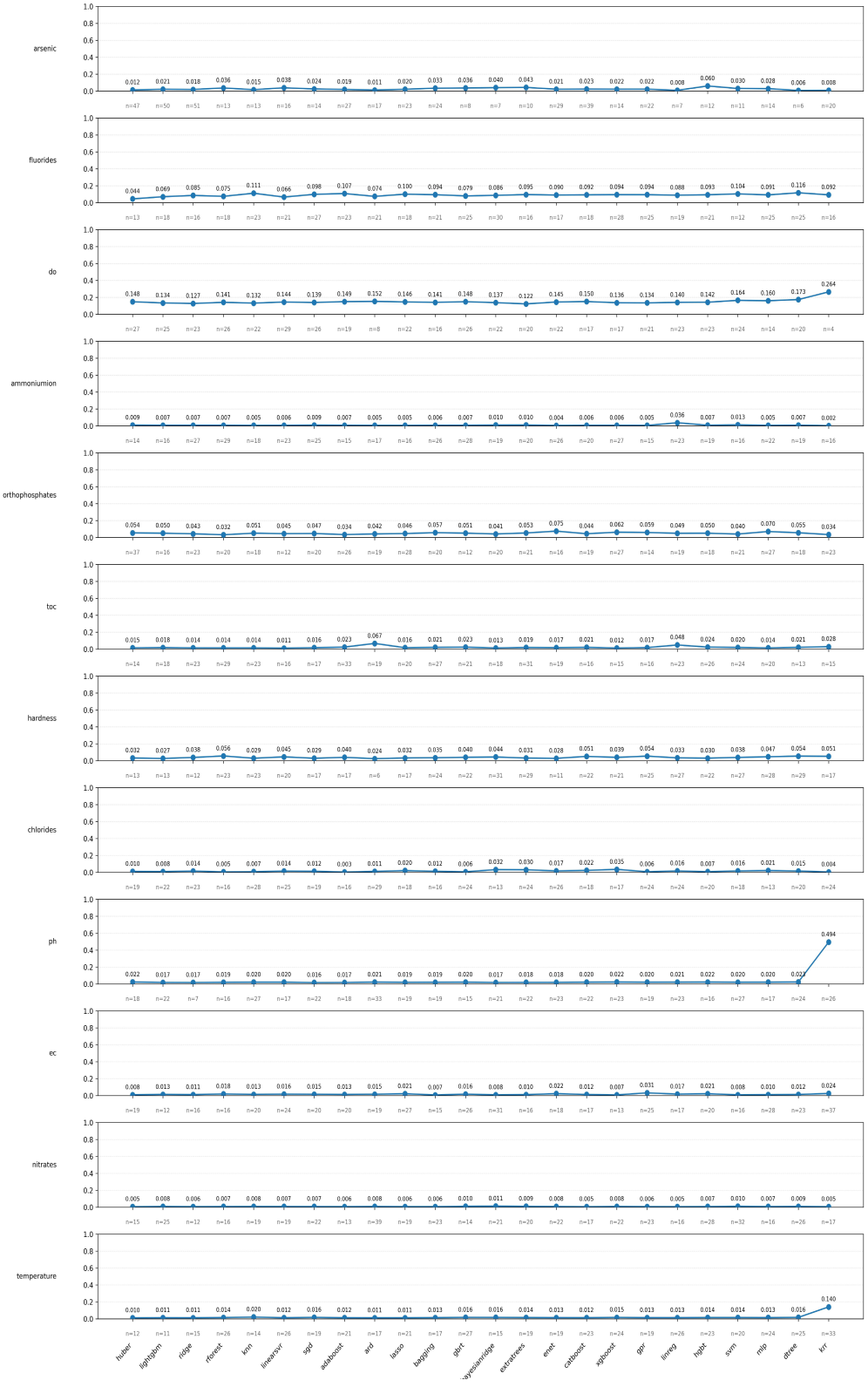

Figure 16: Feature-wise mean absolute error (MAE) of each predictive model, ordered by feature importance (top to bottom). Each tier corresponds to a single feature, with blue markers representing the mean MAE of each model for that feature. Values above the points show the averaged MAE, while the counts below the x-axis (denoted as n) indicate how many times that model was selected across all RTU runs for the given feature.

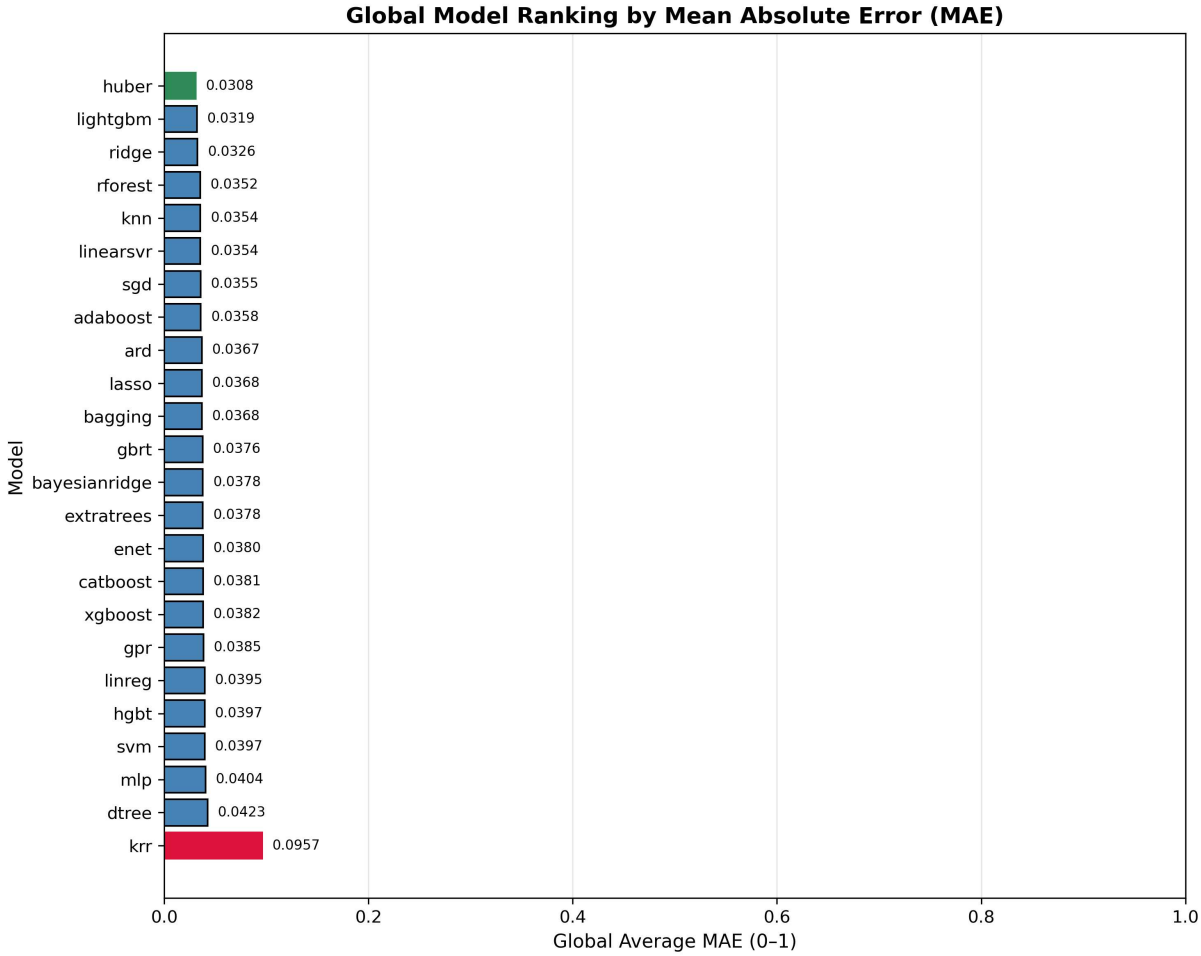

Figure 17: Normalized Mean Absolute Error (MAE)

Figure 19 which needs some explanations to be properly understood. We computed in each log, per-feature averages of SHAP scores, selected model ranking (shown in Figure 17 , assigning rankings from 1 to 24 from top to the bottom), and the MAEs. In Figure 19 for each of these feature (which are ordered from top to bottom according to the feature importance shown in Figure 16) 8 rectangles (first four representing the first series, and second four representing the second series of DT-RTUs in numeric order) with the color representing the average SHAP score ranking is shown (please see the figure legend), inside each rectangle there is a circle with its color representing the average model ranking. Both rankings are either rounded or truncated (based on the fraction being more or less than 0.5). The position of the rectangle represents the average MAE of that feature for each DT-RTU. This schematic is quite explanatory in the sense that, we can see the very presence of all characteristics of SHAP-MAI Engine, i.e. its explanatory process, and allocation of the best model to the most important features. For example we can see that in the case of Arsenic the most important feature in terms of its SHAP score, all of its visuals show the darkest blues and greens i.e. are in the range of highest SHAP scores, and best ranking models. Moving to the second important feature we can see that the visuals turn more into less important with its visuals turning into lighter blue and green with some dark yellow circles. This shift in the visuals is constant as we move to further less important features, until we arrive to the feature with most bright reddish visual which is temperature. We consider this diagram as a "path-oriented" heatmap, or "progressive explanatory localized heatmap" if we can assign it such names. In any case,

considering each feature, comparing the visuals of its 8 DT-RTUs' visuals, we can see that the first 4 with models that are not completely top 12, have a higher MAE, considering the position of rectangles, and the second 4 are mostly of lower MAE, we can see that the visuals represent the best performing models in their defined color have lower height (if the diagram page is seen horizontal, or more on the left if we maintain the page orientation).

Another observation to be noted in this diagram, is the overall decrease of MAEs when we move from most important features to the less important, although not a one-to-one behavior, but there is a regime of decline. That is another affirmation of how explainability should be used in such systems with multi-feature nature, in which each feature's data regime is very different. For example Arsenic's data was comprised of mostly very low amounts with rare but sudden momentarily peaks with large value, that leads to immediate drop of water quality to zero and therefore stopping the inflow of water, that is health-critical, and explainability covers that need to discover and prevent those situations, the system's perceptive and then predictive ability is agile enough to trigger that health-critical actuation. This shows a high level of intra-agent arrangement within the multi-agent architecture, especially in regards to the synchronization between the Collector, the Counselor and the Actor. Despite the existence of these large peaks that arrive completely improvised, with no pattern or regime, the DT's VACCSMEDUS architecture is capable of predicting them with a good level of reliability, which is shown in the very low level of Arsenic's MAE, this could be the most important service that this DT provides, guaranteeing that the contaminant water does not find its way into the treatment plant, or in any other case such as small scale water extractions. Then we look at Fluorides and DO, these features' data had a similar variation to the Arsenic, but with more and even more peaks respectively, therefore we see that for Fluorides, MAE is higher than Arsenic, and DO's MAE is higher than that of Fluorides. As we said, on this diagram there is a regime of decline as we move forward from most important features, i.e. with the highest average SHAP scores, to the ones with less SHAP scores, especially if we group the features in 4, each group containing 3 features, this declining MAE regime is more apparent. This affirms the necessity of XAI in allocating the best performing models to the most important features. It does not mean that the features with lesser SHAP scores will receive less performant service, because this diagram shows completely that this is not the case, the very low MAEs of the features with lesser SHAP scores shows that the allocation of the system's services was enough for them. In fact it leads us to a very straightforward conclusion, that our DT has a very optimized service allocation mechanism based on the need of each feature, i.e. the supply meets the demand. That is the direct result of the model selection mechanism, and we can claim that "utmost optimization of services" is one of the characteristics of this architecture. Speaking about the computational costs, apparently this optimization reduces the it to minimum, and therefore even within the context of energy optimization which is not a direct criteria of this research, our optimized DT holds on the best standards. This matter is indirectly related to our work, because of the PLM and O&M matters which are foundational to the DT technology, especially when it comes to interaction with physical devices of the treatment plant, SCADA systems and PLC, it is very important that the DT is optimized intrinsically and internally, so that its external communications and commands that even actuate instrumentation bring forward energy-optimized operations.

The experimental validation confirms that the VACCSMEDUS architecture presents accurate solutions for water quality monitoring and treatment. The integrated SHAP-MAE Engine introduces a continuous,

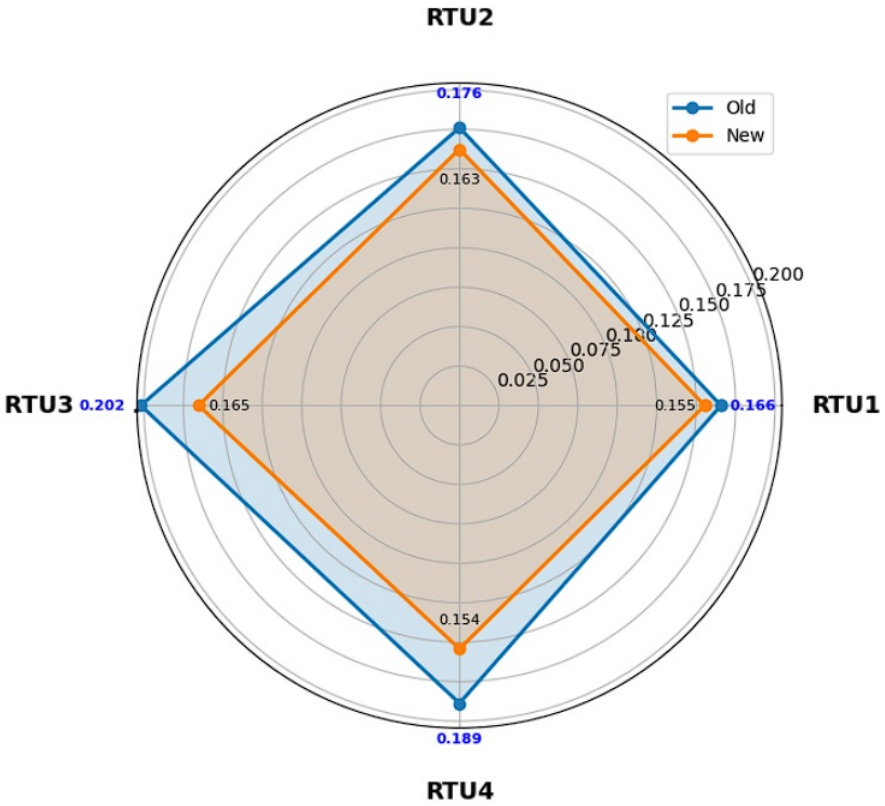

Figure 18: Comparison of the first series of runs,first or old in blue with 4 different selections as mentioned in the text, vs the second or new series of runs with equal top 12 performing models of the first series, the lower errors confirm the validity of the competitive SHAP_MAE_Engine mechanism

performance-driven model-switching strategy that yields precise predictions. The Digital Twin's RRTW framework which provides independence from historical data is significant. This capability dramatically lowers the barrier to deployment in legacy, resource-constrained, or remote water resource sites where historical records are sparse or nonexistent and instrumentation is minimal. In such sites with some minimal instrumentation such as sensors, pumps, PLCs and RTUs, in the absence of a treatment plant with SCADA, our Digital Twin can take the role of the MTU and guarantee the outflow of the groundwater to be of high quality. Also in the presence of an existing SCADA system, our DT can be integrated with its MTU as a plug-in software and add the capability of prediction to the whole system. The modular multi-agent design unifies sensing, reasoning, forecasting, visualization, and actuation within a single closed cognitive loop, and offers a new level of functional implementation, a fully operational technical blueprint, and a clear framework, applicable to the design of the Digital Twins of all kinds of physical entities in all substrate fields.

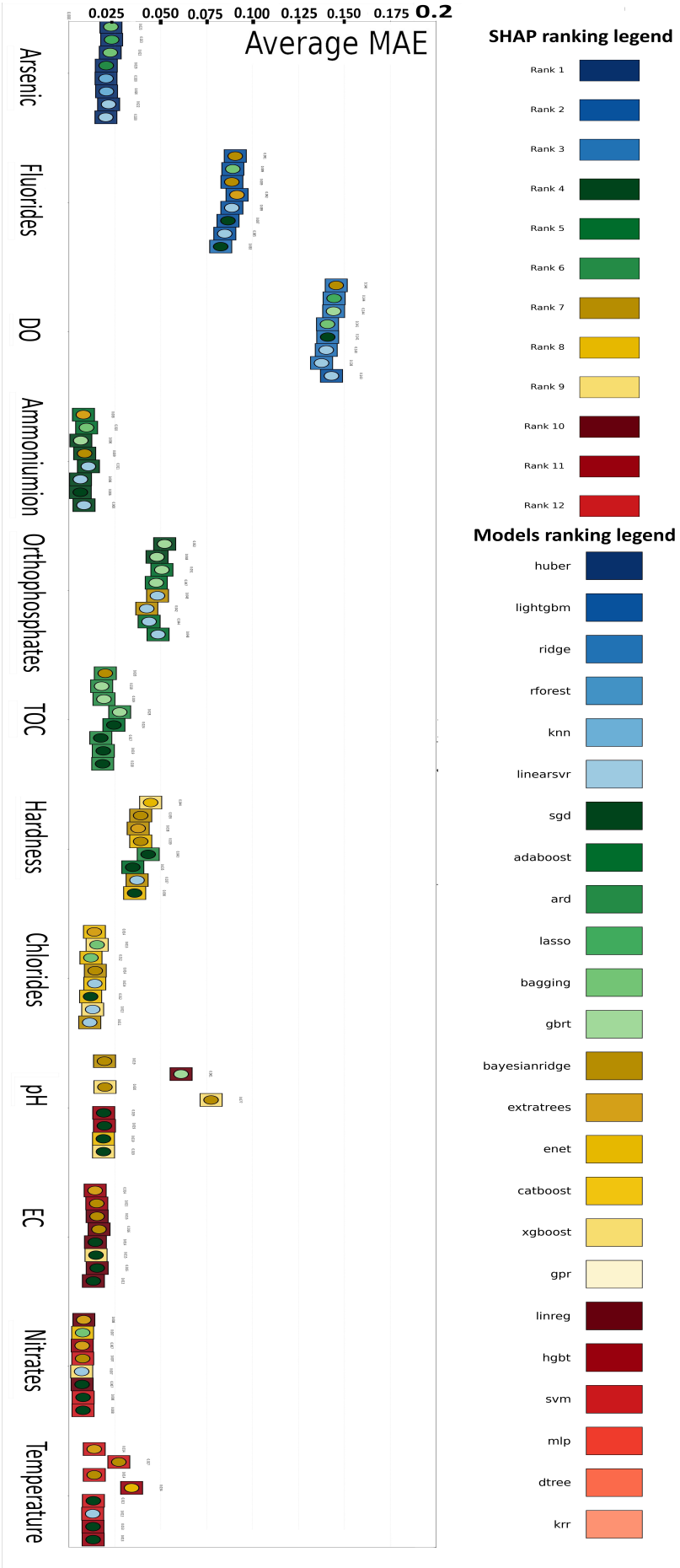

Figure 19: The colored schematic of the per-feature averages of SHAP-scores, assigned model rankings, and MAEs, for total 8 DT-RTUs of the first series and the second series with best performing models, the figure is best seen in horizontal page orientation

Conclusions

This dissertation contains the results of my research activities on the topic of Digital Twins. The sequence of the chapters represents the chronological order of such activities, which began by exploring the backgrounds and origins of the concept of the DT and studying and surveying the existing notable works performed in this field. As a result of such studies I found out a fact about the Digital Twins technology, which is the Digital Twins can be implemented in many different ways, but considering the strong theory and definition of this technology and observing the common tasks realized by the majority of the works under study, I was able to define a methodological approach towards creating my own contribution to the field, and that was the multi-agent architecture VACCSMEDUS which was explained in the third chapter. I also made use of the explainable AI which seemed to me to be a very strong tool for establishing the core analytical aspects of my work. The water quality features provided a very suitable input base for defining and designing a Digital Twin integrating an XAI and Machine Learning analytical core. This approach proved to be right after I could obtain good prediction results from the early stages of the project.

As stated previously, the main novelties include the mulit-agent architecture itself, the Recent-Real-Time concept and framework, and the SHAP-MAE-Engine which provides the system with competitive and self-enhancing mechanism that yields precise predictions. In fact it is this very same competitve mechanism which makes real-time data sifficient for precise predictions that span further in time, i.e. in our future window which its length is 20 percent of our Recent Real-Time Window (RRTW).

We were also able to identify from the results the most efficient 12 Machine Learning models out 24 efficient models, and validate their higher efficiency through comparative experimentation. In fact, Machine Learning models proved to be performing very well and remain stable in the case of using Recent-Real-Time Data (RRTD), considering that the Deep Learning models' deep architectures with millions of parameters demand large datasets. Therefore we can conclude that the Machine Learning models provided us with the performance that we need in the absence of historical data, and their lower computational costs make them suit better to our "Real-Time-Plus" online predictive DT. This is specifically important in the field of water quality monitoring and prediction because the deployment of a predictive system that does not require historical data can solve the problem of the absence of such data, which is a common issue in this field in many parts of the world. However the presence of historical data can be a bonus for our DT becuase we have the mechanism of reading also the historical data through CSV files and generate predictions. Our real-time online data is handled via MQTT connectivity which gives us operational flexibility in terms of physical range.

The full potential of digital twins for water quality management lies in integrating the complete ensemble of influencing factors such as hydraulic behavior, water chemistry, weather patterns, and seasonal variations, all integrated into unified predictive models. However, this remains an open challenge due to the scarcity of interconnected data spanning the entire source-to-treatment continuum. Water data is

typically disparate, fragmented across systems, and lacking the cohesive framework needed for such integration. Overcoming this data barrier represents the primary direction for future research, and provides the horizon toward which the present work could ultimately be expanded.

In the wider perspective, we can conclude that the Digital Twins are becoming increasingly popular, and we can see in the horizon of the progress of this technology a similar pattern such as the dotcom boom era.

Any work speaks better for itself, I hope I was able to explain the work in words in this dissertation, but the softwafre itself can be referred to in https://github.com/Alireza-Rahimi-Vaccsmedus/Vaccsmedus. It is best run as a Maven project in IntelliJ IDEA environment, there is a readme file containing the user's manual explaining the setup better.

Bibliography

[1] Grieves, M. Digital Twin: Manufacturing Excellence through Virtual Factory Replication; A White Paper; Michael Grieves, LLC: Melbourne, FL, USA, 2014.

[2] Grieves, Michael and Vickers, John, Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems, 2017, DOI: 10.1007/978-3-319-38756-7-4.

[3] T.H.-J. Uhlemann, C. Lehmann, R. Steinhilper, The digital twin: realizing the cyberphysical production system for industry 4.0, Procedia CIRP (2017) 335–340, https://doi.org/10.1016/j.procir.2016.11.152.

[4] Hazim El-Mounayri, Purdue School of Engineering and Technology, IUPUI Initiative for Product Lifecycle Innovation (IPLI), IUPUI, Integrated Model-based Systems Engineering (iMBSE) in Engineering Education, 2021 annual INCOSE international workshop, https://www.incose.org/iw2021/.

[5] Madni, Azad and Madni, Carla and Lucero, Scott, Leveraging Digital Twin Technology in Model-Based Systems Engineering, Systems, 7, 7, 2019, DOI: 10.3390/systems7010007.

[6] Kritzinger W, Karner M, Traar G, Henjes J, Sihn W, Digital Twin in manufacturing: A categorical literature review and classification, International Federation of Automatic Control, 2018, p.1016-1022.

[7] Negri E, Fumagalli L, Macchi M, A review of the roles of Digital Twin in CPS-based production systems, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM 2017, 11, 2017, p. 939-948.

[8] Z. Ren, J. Shi and M. Imran, Data Evolution Governance for Ontology-Based Digital Twin Product Lifecycle Management, IEEE Transactions on Industrial Informatics, vol. 19, no. 2, pp. 1791-1802, Feb. 2023, doi: 10.1109/TII.2022.3187715.

[9] M. Ertz et al., How transitioning to Industry 4.0 promotes circular product lifetimes, Ind. Marketing Manage., vol. 101, pp. 125–140, 2022.

[10] M. Xia, T. Li, T. Shu, J. Wan, C. W. de Silva, and Z. Wang, A two-stage approach for the remaining useful life prediction of bearings using deep neural networks, IEEE Trans. Ind. Inform., vol. 15, no. 6, pp. 3703–3711, Jun. 2019.

[11] Weyer S, Schmitt M, Ohmer M, Gorecky D, Towards Industry 4.0 - Standardization as the crucial challenge for highly modular, multivendor production systems, IFAC Symposium on Information Control in Manufacturing, 15, 2015, p. 579-584.

[12] J. Li et al., Big Data in product lifecycle management, Int. J. Adv. Manuf. Technol., vol. 81, no. 1–4, pp. 667–684, 2015.

[13] Tao F, Cheng J, Qi Q, Zhang M, Zhang H, Sui F, Digital twin-driven product design, manufacturing and service with big data, International Journal of Advanced Manufacturing Technology, 2017.

[14] Serries G, Digital Twin: towards a digital double for every physical object, ZDNet, 2016.

[15] Yvan Tchana, Guillaume Ducellier, Sébastien Remy, Designing a unique Digital Twin for linear infrastructures lifecycle management, Procedia CIRP, Volume 84, 2019.

[16] Stark R, Grosser H, Beckmann-Dobrev B, Kind S, Advanced Technologies in Life Cycle Engineering, In: 3rd International Conference on Through-life Engineering Services, Procedia CIRP, 22, 2014, p. 3-14.

[17] Gabor T, Belzner L, Kiermeier M, Beck MT, Neitz A, A Simulation-Based Architecture for Smart Cyber-Physical Systems, In: International Conference on Autonomic Computing, 2016, p. 374-379.

[18] Erikstad SO, Merging Physics, Big Data Analytics and Simulation for the Next-Generation Digital Twins, HIPER 2017 - High Performance Marine Vehicles, South-Africa: Zewenwacht, 2017, p. 139-149.

[19] van Dinter, Raymon and Tekinerdogan, Bedir and Catal, Cagatay, Predictive Maintenance using Digital Twins: A Systematic Literature Review, Information and Software Technology, 151, 2022, 107008, DOI: 10.1016/j.infsof.2022.107008.

[20] P. Papachatzakis, N. Papakostas, G. Chryssolouris, Condition based operational risk assessment an innovative approach to improve fleet and aircraft operability: maintenance planning, in: 1st European Air and Space Conference, Berlin, Germany, 2007, pp. 121–126.

[21] Z. Kang, C. Catal, B. Tekinerdogan, Remaining useful life (Rul) prediction of equipment in production lines using artificial neural networks, Sensors 21 (2021) 932.

[22] Jay, L., W. Fangji, Z. Wenyu, G. Masoud, L. Linxia, and S. David, Prognostics and Health Management Design for Rotary Machinery systems—Reviews Methodology and Applications, Mechanical Systems and Signal Processing 42: 314–334, 2014, doi:10.1016/j.ymssp.2013.06.004.

[23] Aivaliotis, P., K. Georgoulias, Z. Arkouli, and S. Makris, Methodology for Enabling Digital Twin Using Advanced Physics-based Modelling in Predictive Maintenance, Procedia CIRP 81: 417–422, 2019, doi:10.1016/j.procir.2019.03.072.

[24] P. Aivaliotis, K. Georgoulias and G. Chryssolouris, The use of Digital Twin for predictive maintenance in manufacturing, International Journal of Computer Integrated Manufacturing, 32:11, 1067-1080, 2019, DOI: 10.1080/0951192X.2019.1686173.

[25] Sresakoolchai, Jessada and Kaewunruen, Sakdirat, Railway infrastructure maintenance efficiency improvement using deep reinforcement learning integrated with digital twin based on track geometry and component defects, Scientific Reports, 13, 2023, DOI: 10.1038/s41598-023-29526-8.

[26] R. Rossini, D. Conzon, G. Prato, C. Pastrone, J. Reis, G. Gonçalves, REPLICA: a solution for next generation iot and digital twin based fault diagnosis and predictive maintenance, in: CEUR Workshop Proceedings, 2020, pp. 55–62.

[27] T. Wang, Z. Liu, M. Liao, N. Mrad, Life prediction for aircraft structure based on Bayesian inference: towards a digital twin ecosystem, in: Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM, 2020.

[28] P.K. Rajesh, N. Manikandan, C.S. Ramshankar, T. Vishwanathan, C. Sathishkumar, Digital twin of an automotive brake pad for predictive maintenance, Procedia Comput. Sci. 165 (2019) 18–24.

[29] I. Linkov et al., Changing the resilience paradigm, Nature Climate Change, vol. 4, no. 6, pp. 407–409, May 28, 2014, doi: 10.1038/nclimate2227.

[30] A. S. Jin et al., Resilience of Cyber-Physical Systems: Role of AI, Digital Twins, and Edge Computing, IEEE Engineering Management Review, vol. 50, no. 2, pp. 195-203, 1 Secondquarter, June 2022, doi: 10.1109/EMR.2022.3172649.

[31] Saracco, R., Digital twins: Bridging physical space and cyberspace, Computer 2019, 52, 58–64.

[32] Al Faruque, M.A.; Muthirayan, D.; Yu, S.Y.; Khargonekar, P.P., Cognitive Digital Twin for Manufacturing Systems, In Proceedings of the 2021 Design, Automation and Test in Europe Conference and Exhibition, Grenoble, France, 1–5 February 2021, pp. 440–445.

[33] Eirinakis, P.; Lounis, S.; Plitsos, S.; Arampatzis, G.; Kalaboukas, K.; Kenda, K.; Lu, J.; Rožanec, J.M.; Stojanovic, N., Cognitive Digital Twins for Resilience in Production: A Conceptual Framework, Information 2022, 13, 33, https://doi.org/10.3390/info13010033.

[34] Dmitry Ivanov, Intelligent digital twin (iDT) for supply chain stress-testing, resilience, and viability, International Journal of Production Economics, Volume 263, 2023, 108938, ISSN 0925-5273.

[35] G. Cabri, A. Rahimi, Exploitation of Digital Twins in Smart Manufacturing, 2024 IEEE 21st Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2024, pp. 759-764, https://doi.org/10.1109/CCNC51664.2024.10454782.

[36] Y. Lu, C. Liu, K.I. Wang, H. Huang, X. Xu, Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues, Robotics and Computer-Integrated Manufacturing, Volume 61, 2020, 101837, https://doi.org/10.1016/j.rcim.2019.101837.

[37] K. Zhang, T. Qu, D. Zhou, H. Jiang, Y. Lin, Digital twin-based opti-state control method for a synchronized production operation system, Robotics and Computer-Integrated Manufacturing, Volume 63, 2020, 101881, https://doi.org/10.1016/j.rcim.2019.101881.

[38] J. Leng, H. Zhang, D. Yan, Q. Liu, X. Chen, D. Zhang, Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart workshop, Journal of Ambient Intelligence and Humanized Computing, Volume 10, Issue 3, 2019, pp. 1155–1166, https://doi.org/10.1007/s12652-018-0881-5.

[39] L. Fan, K. Ding, G. Liu, Digital twin technology within intelligent manufacturing, Manufacturing Technology & Machine Tool, Issue 7, 2019, pp. 61–66, https://doi.org/10.19287/j.cnki.1005-2402.2019.07.010.

[40] L. Li, B. Lei, C. Mao, Digital Twin in Smart Manufacturing, Journal of Industrial Information Integration, Volume 26, 2022, 100289, https://doi.org/10.1016/j.jii.2021.100289.

[41] J. Rios, A. Bernard, A. Bouras, S. Foufou, Product Lifecycle Management and the Industry of the Future, 14th IFIP WG 5.1 International Conference, PLM 2017, Seville, Spain, July 10-12, 2017, Revised Selected Papers, Lecture Notes in Computer Science, Vol. 517, Springer, 2018.

[42] R. Müller, M. Vette, A. Geenen, Skill-based Dynamic Task Allocation in Human-Robot- Cooperation with the Example of Welding Application, Procedia Manufacturing, Volume 11, 2017, pp. 13–21, https://doi.org/10.1016/j.promfg.2017.07.117.

[43] A.A. Malik, A. Bilberg, Digital twins of human robot collaboration in a production setting, Procedia Manufacturing, Volume 17, 2018, pp. 278–285, https://doi.org/10.1016/j.promfg.2018.10.047.

[44] Z.Y. Zhou, J.H. Liu, C.T. Tang, Z.J. Li, Implemented assembly process technology for complex products development, Computer Integrated Manufacturing Systems, Volume 20, Issue 8, 2014, pp. 1859-1869.

[45] Y. Wu, L.Y. Yao, H.X. Xiong, C.B. Zhuang, H.R. Zhao, J.H. Liu, Quality control method of complex product assembly process based on digital twin technology, Computer Integrated Manufacturing Systems, Volume 25, Issue 6, 2019, pp. 1568-1575.

[46] H. Ma, H. Zhou, H. He, G. Jiao, S. Wei, A Digital Twin-Based Approach for Quality Control and Optimization of Complex Product Assembly, 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), Dublin, Ireland, 2019, pp. 762-767, https://doi.org/10.1109/AIAM48774.2019.00157.

[47] W. Sarni, C. White, R. Webb, K. Cross, R. Glotzbach, Digital Water: Industry Leaders Chart the Transformation Journey, Technical Report, IWA Publishing: London, UK, 2019.

[48] G. Karmous-Edwards, P. Conejos, K. Mahinthakumar, S. Braman, P. Vicat-Blanc, J. Barba, Foundations for building a digital twin for water utilities, Smart Water Report (SWAN/Water online), 2019.

[49] Implementation of the Digital Twin in Water 4.0. Cabri, G., Rahimi, A. In 32nd International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2024, Reggio Emilia, Italy, June 26-28, 2024, pages 38–43, 2024. IEEE.

[50] J.M. Curl, T. Nading, K. Hegger, A. Barhoumi, M. Smoczynski, Digital Twins: The Next Generation of Water Treatment Technology, Journal AWWA, 2019, https://doi.org/10.1002/awwa.1413.

[51] M. Garrido-Baserba, L. Corominas, U. Cortés, D. Rosso, M. Poch, The Fourth-Revolution in the Water Sector Encounters the Digital Revolution, Environmental Science & Technology, Volume 54, Issue 8, 2020, pp. 4698-4705, https://doi.org/10.1021/acs.est.9b04251.

[52] H.M. Ramos, A. Kuriqi, M. Besharat, E. Creaco, E. Tasca, O.E. Coronado-Hernández, R. Pienika, P. Iglesias-Rey, Smart Water Grids and Digital Twin for the Management of System Efficiency in Water Distribution Networks, Water, Volume 15, Issue 6, 2023, 1129.

[53] R. Ranjbar, E. Duviella, L. Etienne, J.M. Maestre, Framework for a digital twin of the Canal of Calais, Procedia Computer Science, Volume 178, 2020, pp. 27-37.

[54] Future City Flow Project, Sweden Water Research, https://www.swedenwaterresearch.se/en/projekt/future-city-flow-3/.

[55] International Water Association (IWA), Operational digital twins in the urban water sector: case studies, 2023, https://iwa-network.org/publications/operational-digital-twins-in-the-urban-water-sector-case-studies/.

[56] Jacobs, Creating the First Digital Twin for PUB's Changi Water Reclamation Plant, 2023, https://www.jacobs.com/newsroom/press-release/jacobs-creating-first-digital-twin-pubs-changi-water-reclamation-plant.

[57] C. Bonilla, B. Brentan, I. Montalvo, D. Ayala-Cabrera, J. Izquierdo, Digitalization of Water Distribution Systems in Small Cities, a Tool for Verification and Hydraulic Analysis: A Case Study of Pamplona, Colombia, Water, Volume 15, Issue 21, 2023, 3824, https://doi.org/10.3390/w15213824.

[58] Rossman, L.A.; Woo, H.; Tryby, M.; Shang, F.; Janke, R. Manual Del Usuario de EPANET 2.2; EPA, US Environmental Protection Agency: Washington, DC, USA, 2002.

[59] M. Bernard, Journal AWWA, Volume 116, Issue 1, January/February 2024.

[60] M.G. Kim, M. Bartos, A digital twin model for contaminant fate and transport in urban and natural drainage networks with online state estimation, Environmental Modelling & Software, Volume 171, 2024, 105868, https://doi.org/10.1016/j.envsoft.2023.105868.

[61] J.E. Quansah, B. Engel, G.L. Rochon, Early warning systems: a review, Journal of Terrestrial Observation, Volume 2, Issue 2, 2010, pp. 24-44.

[62] S. Pasika, S.T. Gandla, Smart water quality monitoring system with cost-effective using IoT, Heliyon, Volume 6, Issue 7, 2020, e04096, https://doi.org/10.1016/j.heliyon.2020.e04096.

[63] V. Lakshmikantha, A. Hiriyannagowda, A. Manjunath, A. Patted, J. Basavaiah, A.A. Anthony, IoT based smart water quality monitoring system, Global Transitions Proceedings, Volume 2, Issue 2, 2021, pp. 181-186, https://doi.org/10.1016/j.gltp.2021.08.062.

[64] C. Puttamadappa, B.D. Parameshachari, Demand side management of small scale loads in a smart grid using glow-worm swarm optimization technique, Microprocessors and Microsystems, Volume 71, 2019, 102886, https://doi.org/10.1016/j.micpro.2019.102886.

[65] M.A. Mutri, A.R.A. Saputra, I. Alinursafa et al., Smart system for water quality monitoring utilizing long-range-based Internet of Things, Applied Water Science, Volume 14, 2024, 69, https://doi.org/10.1007/s13201-024-02128-z.

[66] Y. Qiu, H. Liu, J. Liu, D. Li, C. Liu, W. Liu, J. Wang, Y. Jiao, A Digital Twin Lake Framework for Monitoring and Management of Harmful Algal Blooms, Toxins, Volume 15, Issue 11, 2023, 665, https://doi.org/10.3390/toxins15110665.

[67] S. Wang, L. Qing, X. He, Y. Zhang, Granular cumulative probability curve drawing system based on Canvas, Microcomputer & Its Applications, Volume 36, Issue 18, 2017, pp. 97-100.

[68] P. Rengarajan, A Comparison of 3D and 2D U-Net Convolutional Networks for Segmentation in FIB-SEM Imagery, Microscopy and Microanalysis, Volume 28, Issue S1, 2022, pp. 3064-3066, https://doi.org/10.1017/S1431927622011651.

[69] R. Bogdan, C. Paliuc, M. Crisan-Vida, S. Nimara, D. Barmayoun, Low-Cost Internet-of-Things Water-Quality Monitoring System for Rural Areas, Sensors, Volume 23, Issue 8, 2023, 3919, https://doi.org/10.3390/s23083919.

[70] R.K. Makumbura, L. Mampitiya, N. Rathnayake, D.P.P. Meddage, S. Henna, T.L. Dang, Y. Hoshino, U. Rathnayake, Advancing water quality assessment and prediction using machine learning models, coupled with explainable artificial intelligence (XAI) techniques like shapley additive explanations (SHAP) for interpreting the black-box nature, Results in Engineering, Volume 23, 2024, 102831, https://doi.org/10.1016/j.rineng.2024.102831.

[71] R. Makubura, D.P.P. Meddage, H.M. Azamathulla, M. Pandey, U. Rathnayake, A simplified mathematical formulation for water quality index (WQI): a case study in the Kelani River Basin, Sri Lanka, Fluids, Volume 7, Issue 5, 2022, 147, https://doi.org/10.3390/fluids7050147.

[72] K.D. Siriwardhana, D.I. Jayaneththi, R.D. Herath, R.K. Makumbura, H. Jayasinghe, M.B. Gunathilake, U. Rathnayake, A simplified equation for calculating the water quality index (WQI), Kalu River, Sri Lanka, Sustainability, Volume 15, Issue 15, 2023, 12012.

[73] S. Sen, B. Deora, V. Vaishnav, Explainable Deep Learning for Time Series Analysis: Integrating SHAP and LIME in LSTM-Based Models, Journal of Information Systems Engineering and Management, 2025.

[74] K. Kanagarathinam, S. Krishnan, R. Manikandan, Water quality prediction: A data-driven approach exploiting advanced machine learning algorithms with data augmentation, Journal of Water and Climate Change, 2024, https://doi.org/10.2166/wcc.2023.403.

[75] M.H. Nishat, M.H.R.B. Khan, T. Ahmed et al., Comparative analysis of machine learning models for predicting water quality index in Dhaka's rivers of Bangladesh, Environmental Sciences Europe, Volume 37, 2025, 31, https://doi.org/10.1186/s12302-025-01078-w.

[76] S. Kundu, P. Datta, P. Pal, K. Ghosh, A. Das, B.K. Das, Unveiling the hidden connections: Using explainable artificial intelligence to assess water quality criteria in nine giant rivers, Journal of Cleaner Production, Volume 492, 2025, 144861, https://doi.org/10.1016/j.jclepro.2024.144861.

[77] B. Ngwenya, T. Paepae, P.N. Bokoro, Monitoring ambient water quality using machine learning and IoT: A review and recommendations for advancing SDG indicator 6.3.2, Journal of Water Process Engineering, Volume 73, 2025, 107664, https://doi.org/10.1016/j.jwpe.2024.107664.

[78] M. Abdelhedi, H. Gabtni, Performance comparison of various machine learning models for predicting water quality parameters in the Chebika Zone of Central Tunisia, Earth Science Informatics, Volume 17, 2024, pp. 4245-4259, https://doi.org/10.1007/s12145-024-01370-y.

[79] H.A. Afan, W.H.M. Wan Mohtar, F. Khaleel, A.H. Kamel, S.S. Mansoor, R. Alsultani, A.N. Ahmed, M. Sherif, A. El-Shafie, Data-driven water quality prediction for wastewater treatment plants, Heliyon, Volume 10, Issue 18, 2024.

[80] UK Water Industry Research (UKWIR). (2022). A Roadmap for Digital Twins in the Water Sector (Report Ref. No. 22/WM/03/2). London: UKWIR.

[81] Zeyu Ma, Yunyi Zhu, Chunsheng Chen, Ting Li, Yanan Li, Xiaoding Li, Yuan Wang, T. David Waite, Jing Guan,Towards the digitalization of water treatment facilities: A case study on machine learning-enabled digital twins, Journal of Water Process Engineering,Volume 77, 2025,108316,ISSN 2214-7144,

[82] abdelfatah, S., Alferes, J., and Colpaert, P.: Opportunities and challenges of interoperable Data sharing in the field of Circular water, EGU General Assembly 2025, Vienna, Austria, 27 Apr–2 May 2025, EGU25-16919, https://doi.org/10.5194/egusphere-egu25-16919, 2025.

[83] Haimi H, Awaitey A, Kiran A, Larsson T, Blomberg K, Elvander F, Petäjä E, Mulas M, Sahlstedt K, Mikola A. Integrating data-driven models and process expertise in soft-sensor design for a wastewater treatment digital twin application. Water Sci Technol. 2025 Nov;92(9):1308-1327. doi: 10.2166/wst.2025.154. Epub 2025 Oct 22. PMID: 41236065.

[84] Robert M. Brown, Norman I. McClelland, Robert A. Deininger, and Richard G. Tozer, "A Water Quality Index—Do We Dare?" Water and Sewage Works, vol. 117, no. 10, pp. 339–343, 1970.

[85] Lundberg, Scott & Lee, Su-In. (2017). A Unified Approach to Interpreting Model Predictions. 10.48550/arXiv.1705.07874.

[86] S. Boyer, SCADA: Supervisory Control and Data Acquisition, 4th ed.,International Society of Automation (ISA), 2009.

[87] M. Wooldridge, An Introduction to multi-agent Systems, 2nd ed., Wiley, 2009.

[88] H. S. Peavy, D. R. Rowe, and G. Tchobanoglous, Environmental Engineering. New York, NY, USA: McGraw–Hill, 1985.

[89] World Health Organization, *Guidelines for Drinking-water Quality*, 4th ed., 2022.

[90] U.S. Environmental Protection Agency, *National Primary Drinking Water Regulations*, 2023.

[91] European Union, Directive (EU) 2020/2184 on the quality of water intended for human consumption.

[92] UNESCO–IHP, *Groundwater Resources Assessment*, International Hydrological Programme, 2012.

[93] FAO AQUASTAT, *Global Water Quality and Monitoring Overview*, Food and Agriculture Organization, 2016.

[94] Murphy, K.P. (2012) Machine Learning: A Probabilistic Perspective. MIT Press.

[95] P. J. Huber and E. M. Ronchetti, *Robust Statistics*, 2nd ed., Wiley, 2009, pp. 176–182.

[96] URL: `https://scikit-learn.org/stable/auto_examples/ensemble/plot_hgbt_regression.html`.

[97] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 785-794).

[98] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Advances in Neural Information Processing Systems.

[99] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, 6639–6649.

Acknowledgments

I continue the acknowledgments from the preface. I would like to express my sincerest thanks towards very dear Dr. Marilena Ravetto from "Direzione Sistemi Informativi (Ufficio RTD) e Assicurazione Qualità" for her continuous enormous support since I was a master's student in this institution, she and her dear family including dear Dr. Stefano Pedrazzi from "Ufficio dottorati", my dear friend David and Mr. Andras Gemes and Mrs. Marinella Marchetti, have been like a family to me for over 6 years. My sincerest thanks to dear Dr. Giuseppe Gatti from the administrative office who has always supported me all these years, and dear Professor Alice Ruini from the department of Physics for her very kindly support for years. Dears Professor Marko Bertogna my second supervisor, and Professor Andrea Marongiu the director of the PhD program of Computer and Data Science have been very supportive and kind to me, I thank them sincerely very much. I also thank sincerely Dr. Letizia Musto from UNIMORE, she has been very kind and supportive to me. I also to thank sicerely very much all UNIMORE personel including the vigilanza staff and Mr. Nicola who have always been very kind to me. Much sincere thanks to Officer Nicola from the Polizia ferrovia di Modena. My further sincere thanks goes to my internships supervisors Dr. Cristina Nizzoli from OT-Consulting in Reggio Emilia, Dr. Miquel Sarrias Monton and Dr. Eloisa Vargiu from Cetaqua in Cornella de Llobregat, in Catalonia, Spain. Also my dear friends in Reggio Emilia Mr. Antonio Proscita, Mrs. Aneta Przeworska, Mrs. Turia, my very dear friends Ing. Alessandro Prandi, Ing. Paolo Cerva and his kind wife Francesca, Veronica Bardot, Dottoressa Marzia and Mr. Maurizio from ENPA animals support institution, all have been very nice to me and I sincerely thank them all very much. I thank very much my colleagues in the PhD program of CDS-TSI especially Federico Motta, it has been an honor learning and growing alongside them. I thank Dr. Georgia Franchini and the whole sport group I am a member of, exercising with them at CUS was a big part of keeping my spirits high during the program. My sincere thank also goes to my dears Papa Don Giorgio "Doriano" Carraro in Castellina Scalo, and Mrs. Miranda Rabitti in Correggio. My sincere thank goes to Mr. Reza Moghimi and his wife, my neighbors and compatriots, their presence has always been heartwarming, I wish the best for them and their newborn darling Ryra. I should also express my sincere thanks to my oldest friends Milad Askian, Masood Nikoofar and Hamidreza Amiri for their long-time friendship that has kept me in good emotional state. My sincerest thanks goes to Dr. Hossein Cameron Khanlar, a longtime friend of my father's and of myself, whose support and kind guidance has always been very important and helpful. I also thank sincerely very much my parents Mohammad-Mehdi and Mehri Rahimi, their support all these years throughout my educational path was an essential factor for achieving my goals. My grandparents souls in heaven may become happy, I remember their kindness towards me, they are always in my heart, Kobra, Sudabeh, Mohammad-Vali and Hossein.

Last but not least, my pets, "Mamu" the cat here with me in Italy, and my very beloved twin dogs "Mike and Tommy" in Iran from whom I have been away for more than 6 years, I wish and hope I can bring them into the most beautiful and wonderful country on earth Italia very soon, I have not passed a second not remembering and feeling them in my mind and heart.

Much thanks also the readers of this work.

I thank my God above all, may my prayers be granted, God bless everyone.

Alireza Rahimi, 11/03/2026