Title:	**Document Version:**
D3.6 Data analytics services and cognitive applications M27	1.0

Project Number:	**Project Acronym:**	**Project Title:**
H2020-871493	DataPorts	A Data Platform for the Cognitive Ports of the Future

Contractual Delivery Date:	**Actual Delivery Date:**	**Deliverable Type*-Security*:**
M27 (March 2022)	M28 (April 2022)	O-PU

*Type: P: Prototype; R: Report; D: Demonstrator; O: Other; ORDP: Open Research Data Pilot; E: Ethics.

**Security Class: PU: Public; PP: Restricted to other programme participants (including the Commission); RE: Restricted to a group defined by the consortium (including the Commission); CO: Confidential, only for members of the consortium (including the Commission).

Responsible:	**Organisation:**	**Contributing WP:**
Miguel Bravo	ITI	WP3

Authors (organisation):	
Miguel Bravo (ITI)	Tristan Kley (UDE)
Francisco Valverde (ITI)	Xhulja Shahini (UDE)
Santiago Cáceres (ITI)	Enrique Miravet (ITI)

Abstract:

This deliverable presents the technical components to deliver AI based services in the context of the DataPorts Project. This overall goal has been addressed using two technical components: a Process-based analytics component developed by UDE, and an Automatic Model Training Engine implemented by ITI. Both technical components look for improving the current lack of ML capabilities in Port IT systems, introducing state-of-the-art technologies applied to the understanding of business process and prediction of KPIs from ports.

Keywords:

Analytics, Machine Learning, Big Data, Neural Networks

Revision History

Revision	Date	Description	Author (Organisation)
V0.1	22.02.2022	First version of the document	Miguel Bravo (ITI)
V0.2	28.02.2022	Content of Process-based Analytics	Miguel Bravo (ITI)
V0.3	01.03.2022	Integration and formatting	Miguel Bravo (ITI)
V0.4	09.03.2022	Modifications integrated	Miguel Bravo (ITI)
V0.5	11.03.2022	Version for UPV feedback	Miguel Bravo (ITI)
V0.6	14.03.2022	Version for revision	Miguel Bravo (ITI)
V0.7	15.03.2022	Version for internal reviewers	Miguel Bravo (ITI)
V1.0	26.04.2022	Final version	Miguel Bravo (ITI)

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement № 871493.

More information available at https://DataPorts-project.eu

INDEX

1 INTRODUCTION 7

1.1 DATAPORTS PROJECT OVERVIEW 7
1.2 DELIVERABLE PURPOSE AND SCOPE 7
1.3 DELIVERABLE CONTEXT 8
1.4 DOCUMENT STRUCTURE 8
1.5 DOCUMENT DEPENDENCIES 8

2 TECHNICAL OBJECTIVES 10

3 POSITION IN THE DATAPORTS ARCHITECTURE 12

4 AUTOMATIC MODEL TRAINING ENGINE 14

4.1 OVERVIEW 14
4.2 TECHNOLOGICAL DESCRIPTION 15
4.2.1 ARCHITECTURE OF THE COMPONENT 15
4.2.2 TRAINING WEB INTERFACE 17
4.2.3 COGNITIVE SERVICES 19
4.2.4 MACHINE LEARNING PIPELINES AND ALGORITHMS 20
4.2.5 MODELS TRACKING AND DEPLOYMENT 26
4.3 FUTURE WORK: ROADMAP 27
4.4 EXAMPLE OF USE: DEMONSTRATION 27
4.4.1 CREATION OF A COGNITIVE SERVICE 28
4.4.2 MANAGEMENT OF A COGNITIVE SERVICE 30
4.4.3 ANALYSIS OF THE RESULTS OF A COGNITIVE SERVICE 32
4.4.4 COMMON DEMONSTRATION 32
4.5 DEVELOPMENT STATUS (M27) 33

5 PROCESS-BASED ANALYTICS COMPONENT 36

5.1 OVERVIEW 36
5.2 TECHNOLOGICAL DESCRIPTION 36
5.2.1 PRESCRIPTIVE PROCESS MONITORING 37
5.2.2 EXPLAINABLE PREDICTIVE PROCESS MONITORING 39
5.3 FUTURE WORK: ROADMAP 40
5.4 EXAMPLE OF USE: DEMONSTRATION 41
5.4.1 PRESCRIPTIVE PROCESS MONITORING 41
5.4.2 EXPLAINABLE PREDICTIVE PROCESS MONITORING 44
5.4.3 COMMON DEMONSTRATION 45
5.5 DEVELOPMENT STATUS (M27) 47

6 RELEASE SUMMARY 48

7 CONCLUSIONS 50

8 REFERENCES AND ACRONYMS 51

8.1 REFERENCES 51
8.2 ACRONYMS 53

9 ANNEX A: EXTERNAL EXPERT REPORT 54

9.1 EXTERNAL EXPERT REPORT 54
9.1.1 DATA ACCESS AGENTS 54
9.1.2 SEMANTIC INTEROPERABILITY 55
9.1.3 DATA ABSTRACTION AND VIRTUALISATION 55
9.1.4 PROCESS BASED ANALYTICS 55
9.1.5 AUTOMATIC MODELS TRAINING ENGINE 56
9.1.6 FINAL CONCLUSIONS 56
9.2 RESPONSE FROM THE AUTHORS OF THE DELIVERABLE 56

LIST OF FIGURES

Figure 1 – DataPorts platform building blocks 12
Figure 2 - Analytics components in the architecture 13
Figure 3 – Interactions of the Automatic Model Training Engine 14
Figure 4 – Architecture of the Automatic Model Training Engine 15
Figure 5 – Row vs Columnar format 16
Figure 6 – Communication between the frontend and backend of AMTE 18
Figure 7 – Dockerized front-end and back end 19
Figure 8 – Cognitive Services implemented in AMTE 19
Figure 9 - Standard training pipeline 21
Figure 10 - Training Strategies 22
Figure 11 - Forecast of Average Vessel Berth Time for the next 5 months 25
Figure 12 – MLFlow integrated in the front-end of AMTE 27
Figure 13 – Dashboard main interface 28
Figure 14 - Services definition 28
Figure 15 - Dataset Selection Screen 29
Figure 16 - Configuration Screens 29
Figure 17 - Strategy selection screen 30
Figure 18 - Confirmation screen 30
Figure 19 - Selecting ‘More Information’ of a trained cognitive service 31
Figure 20 – Screen ‘More Information’ of a trained cognitive service 31
Figure 21 – Models of a trained cognitive service 32
Figure 22 – Results screen of a trained cognitive service 32
Figure 23 - Process-Based Analytics component 37
Figure 24 - Prescriptive Process Monitoring subcomponent 38
Figure 25 - Overview of Explainable Predictive Process Monitoring Component 40
Figure 26 - Asymmetric costs of proactive business process adaptation 41
Figure 27 - Learning behaviour; green: rate of adaptations; blue: earliness (0 = beginning, 1 = end of process); black: rate of correct adaptation decisions; red: overall reward/100 43
Figure 28 – Top five important attributes depending on prefix-length 44
Figure 10 - Exemplary API calls 46

LIST OF TABLES

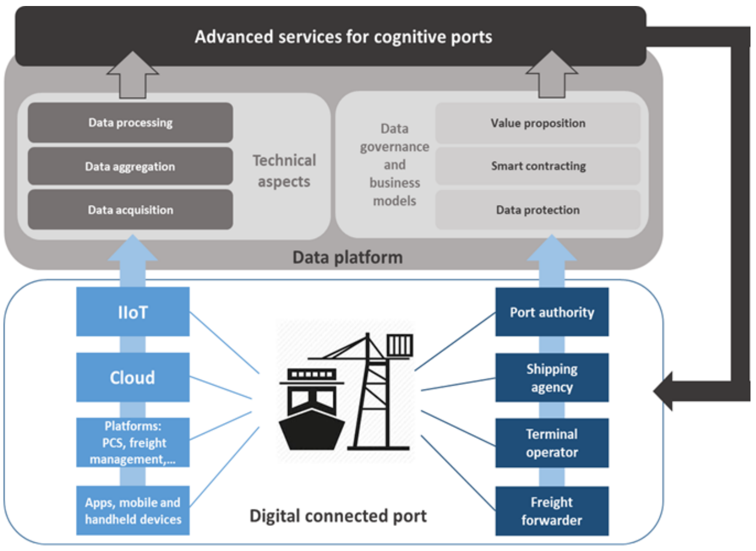
Table 1 – Implemented ML algorithms for Time Series Forecasting 23
Table 2 – Evaluation metrics for Time Series Forecasting 24
Table 3 - Models scoreboard for berth time predictor 24
Table 4 – Implemented ML algorithms for Missing values imputation 25

Table 5 – Evaluation metrics for Values Imputation 26
Table 6 – Reward function definition of the RL-agent 39
Table 7 - Data sets used in evaluation 42
Table 8 – Summary table of AMTE component release information 48
Table 9 – Summary table of PROCESS BASED ANALYTICS component release information 49
Table 10 – Acronyms 53

1 INTRODUCTION

1.1 DATAPORTS PROJECT OVERVIEW

DataPorts is a project funded by the European Commission as part of the H2020 Big Data Value PPP programme and coordinated by the ITI - Technological Institute of Informatics. DataPorts relies on the participation of 13 partners from five different nationalities. The project involves the design and implementation of a data platform, its deployment in two relevant European seaports connecting to their existing digital infrastructures and addressing specific local constraints. Furthermore, a global use case involving these two ports and other actors and targeting inter-port objectives, and all the actions to foster the adoption of the platform at European level.



Hundreds of different European seaports collaborate with each other, exchanging different digital data from several data sources. However, to achieve efficient collaboration and benefit from AI-based technology, a new integrating environment is needed. To this end, DataPorts project is designing and implementing an Industrial Data Platform.

The DataPorts Platform main aim is to connect to the different digital infrastructures currently existing in digital seaports, enabling the interconnection of a wide variety of systems into a tightly integrated ecosystem. In addition, it intends to set the policies for a trusted and reliable data sharing and trading based on data owners' rules and offering a clear value proposition. Finally, it also strives to leverage on the data collected to provide advanced Data Analytics services based on which the different actors in the port value chain could develop novel AI and cognitive applications.

DataPorts will allow to establish a future Data Space unique for all maritime ports of Europe and contribute to the EC global objective of creating a Common European Data Space.

1.2 DELIVERABLE PURPOSE AND SCOPE

Specifically, the DOA states the following regarding this Deliverable:

"*This deliverable will provide Big Data Analytics as a Service (BDAaaS) to the users of the data platform. Additionally, it will also include, built on top of these services, the needed algorithms to allow the development of cognitive applications*".

The purpose of this document is to report the developed software components, namely the Process-based Analytics and the Automatic Model Training Engine, to support such BDAaaS vision in the context of DataPorts. These software components fulfil several of the technical goals described in Task 3.4, specifically:

- The design and development of a set of data analytics services to support the development of ML models using the different sets of data available at the platform.

- These services are based on elastic or cloud-oriented technologies to simplify the customisation and deployment of the required technological blocks: databases, message brokers, ML frameworks and/or visualisation libraries.
- The analysis, evaluation, and selection of state-of-the-art machine learning (ML) algorithms to support the development of cognitive application/services for Ports.

1.3 DELIVERABLE CONTEXT

Its relationship to other documents is as follows:

Primary Preceding documents:

- D2.1 Industrial Data Platforms and Seaport Community Requirements and Challenges: provides the technical requirements of the platform and, specifically, the functionalities that must be supported by the analytics software layer. This deliverable defines the set of features to be accomplished.
- D3.3 Data analytics services and cognitive applications M18: presents the technical components to deliver AI based services in the context of the DataPorts Project. This goal has been addressed using two technical components: a Process-based analytics component developed by UDE, and an Automatic Model Training Engine implemented by ITI.

1.4 DOCUMENT STRUCTURE

This deliverable is broken down in the following sections:

- **Section 1 Introduction**: this section includes a brief description of the project, a description of the purpose and scope of the document, and its dependencies on other deliverables.
- **Section 2 Technological Objectives**: this section presents the alignment of the developed software with the specific goal established in the DataPorts project.
- **Section 3 Position in DataPorts Architecture**: this section briefly describes the interaction of the analytics components with the rest of the DataPorts architecture.
- **Section 4 Process-based Analytics Component**: this section updates the work carried out in the development of the Process-based Analytics Component, described in the deliverable D3.3 (M18) [2]. Additionally, it explains the technologies implemented and how it is expected to be used.
- **Section 5 Automatic Model Training Engine**: this section updates the work carried out in the development of the Automatic Model Training Engine component described in the deliverable D3.3 (M18). Additionally, it explains the technologies implemented and how it is expected to be used.
- **Section 6 Release Summary**: this section shows the release information summary table for each component of the deliverable at M27, including component name, version, source code, documentation, Docker image and dependencies.
- **Section 7 Conclusions**: this section briefly states the conclusions of the deliverable.

1.5 DOCUMENT DEPENDENCIES

This document is the second and last version of an iteration of a living deliverable detailing the components developed in T3.4 until M27. The previous version of the document, deliverable Data analytics services and cognitive applications M18, describes the first iteration of the Data Analytics services offered by the platform. The final version of the services is presented in this document.

The sections updated in D3.6 from D3.3 are:

- Section 1 updates the general information of the deliverable to the current period of delivery (M27).

- Section 3 includes the integration with other components of the platform, deeply described in the deliverable D3.5 Data processing services M27 [3], with new sequence diagrams describing in more detail the data flows between the DataPorts Platform components.
- Section 4 updates the information of D3.3 regarding the Process-Based Analytics component.
- Section 5 updates the information of D3.3 regarding the Automatic Model Training Engine component.
- Section 6 provides a new section with the technical release information of the components involved in this deliverable.
- Section 7 updates the deliverable conclusions to the current period of delivery (M27).

2 TECHNICAL OBJECTIVES

DataPorts' mission is to enable data sharing between port stakeholders. However, it is difficult to engage data owners in data sharing if there is not a business value in such task. Currently, data analytics is not a widely applied subject in the context of maritime ports, or "despite" the big potential of the gathered data. Big data technologies and ML frameworks are essential for the definition of cognitive services: services that provide business insights using the available data. Such services must hide the technological complexity using a cloud-oriented approach to simplify deployment and abstract the required technological components: databases, message brokers, frameworks, user interfaces, etc. Additional components of the DataPorts platform support such vision from a data sharing point of view, they combine data shared in the platform in a federated way (using standard connectors and a common data model) and avoid investments in physical infrastructures and provide data acquisition and processing capabilities. On top of such technical components, the platform must provide mechanisms to develop cognitive services using AI technologies.

This overall challenge has been addressed in the context of the project by two technical components: the Process-based Analytics component developed by UDE, and the Automatic Model Training Engine implemented by ITI. Both technical components aim to improve the current lack of ML capabilities in port domains, introducing state-of-the-art technologies applied to the understanding of business process and KPIs from ports. This goal is aligned with Objective 2 of the DataPorts Project "To design and validate the next-generation set of advanced interoperable data related and AI based services". To support this objective, this deliverable introduces advanced techniques from the AI domain but also adapt them as port-oriented services. As understanding analytics requirements of port business processes is key, these technical components are supporting the development of the pilots as stated in the Objective 1 of the project: "To address real-life data market use cases in two relevant European seaports, two global use cases including pilot deployment and evaluation of progress against benchmarking-existing deployments KPI's". The WP3 description summarises the main technical goal achieved by these components: "to design and develop a set of data analytics services to support the development of descriptive / predictive / prescriptive models using the different sets of data available at the platform." Additionally, task 3.4 involves the analysis, evaluation, and selection of state-of-the-art machine learning (ML) algorithms to support the development of cognitive services. Using different but complementary approaches, these two main components support the established goals from WP3. As a main outcome, it is expected to develop a set of cognitive services or applications in the context of the involved pilots. Next, the specific technical objectives of each component are introduced:

- **Process-based analytics**: the main goal of this component is to optimise a business process happening in the context of ports using machine learning techniques. A business process in the context of DataPorts can be the flow of vessels within the port's service area or transport (containers or goods) operation processes. By proactively predicting the future states of the ongoing process, the component provides forward-looking perspectives for the users to make decisions. To achieve this goal the component uses three techniques from the ML state-of-the-art: ensembles of deep learning models, online reinforcement learning and model induction from interpretable ML research.
- **Automatic Model Training Engine**: the main technical objective of this component is to reduce the delivery time of cognitive services. Using the data ecosystem already in place in the DataPorts platform, this component generates a ML model that generates predictions of relevant KPIs for port business. The training process is customised by a port stakeholder, with no expertise in analytics, according to their requirements and its domain knowledge to detect relevant data. The main advantage is that the component considers the ML techniques more suitable for the data available in current Port IT systems such as TOS (Terminal Operating System), PCS (Port Community System) or gate access control systems. Following a distributed approach using cloud-oriented technologies, the component generates several models to find out the most suitable for the task at

hand. Finally, this ML model is also packaged as a cognitive service, to extend the current functionality of the apps available in the port or to define new ones.

3 POSITION IN THE DATAPORTS ARCHITECTURE

From the services perspective of the DataPorts platform, the Process-based Analytics and Automatic Model Training Engine components are located inside the Analytics Services building block (See Figure 1).

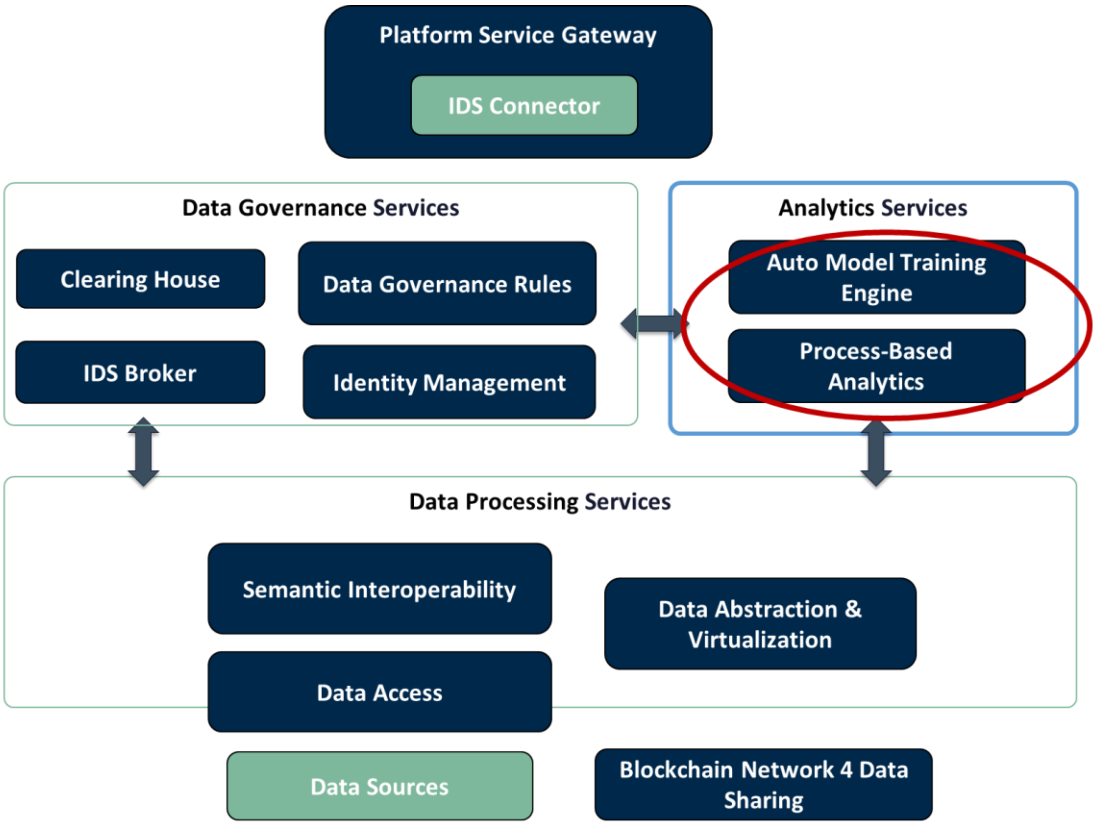


Figure 1 – DataPorts platform building blocks

Both components are data consumers and interact as the final step of the chain to deliver a cognitive service. Next, it is presented a summary of the main interactions with the rest of the components, as described in Deliverable D2.4:

- Semantic Interoperability API: This API provides information following the data model defined in the context of the project. Two subscription modes are available: historical data retrieved from already deployed data agents and the last received record. Using the historical data, it is possible to generate an input dataset for model training. Additionally, as it publishes the last record as soon as it is received, the communication with this API could enable the continuous generation of predictions in using the most recent data.
- Data Abstraction and Virtualisation Component: One of the features of this component is to apply several approaches to improve the data quality, for instance, marking or removing missing values. The quality of a ML model has a clear relationship with the quality of the input data. Therefore, this component will help to solve such issues without the need of specific implementations.
- Data Governance: This component manages the access to the datasets already available in the DataPorts ecosystem. Then, the main interaction expected is to request the metadata of the dataset: (e.g., column names, data types, sizes, etc.) and check if a dataset is available for performing analytics. This process is transparent to the user once it is logged into the system. Then, with the provided credentials, a list of the datasets available will be sent to the analytics component.

- Blockchain networks: In the context of the project several blockchain networks have been deployed to support the on-chain data sharing. This component is not required to interact with the analytics ones, but also could be considered as a data source to get data from or, if required, to publish the prediction outcomes of a model execution.
- External apps: External port-oriented apps, such as TOS or PCS, are potential consumers of the services developed using the analytics components. The main idea is that the resulting services or the trained model, could be integrated with such applications to improve their current functionality.

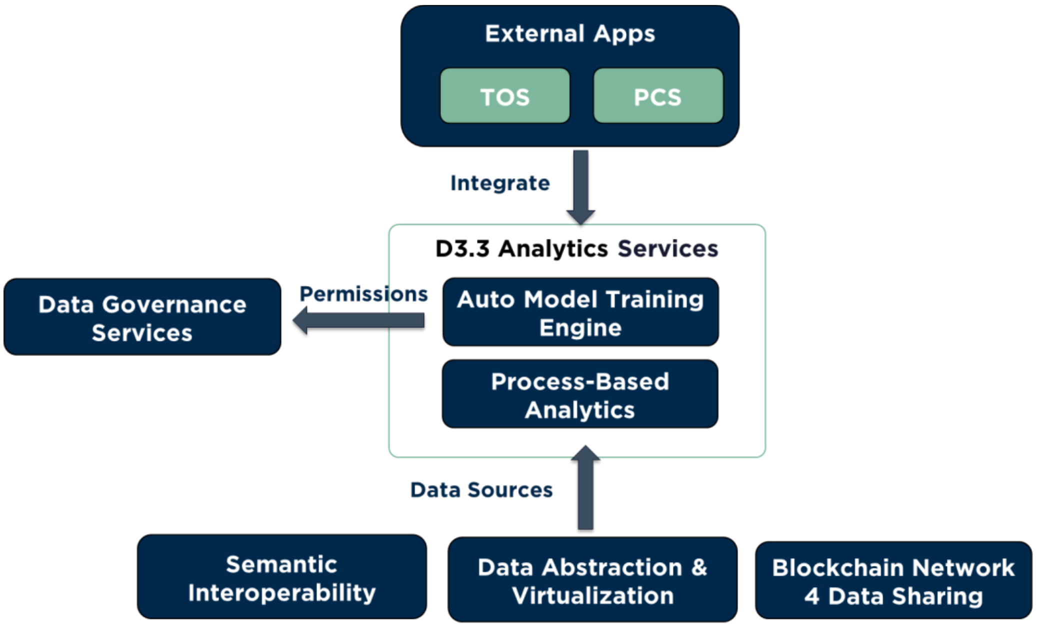


Figure 2 - Analytics components in the architecture

The interactions of both Automatic Model Training Engine and Process-based Analytics with the rest of the components of Dataports platform is fully explained in the deliverable *D3.5 Data processing services M27, section 3 Position in the Dataports architecture*”. For better understanding, sequence diagrams, amongst others, are included.

Regarding the Data Governance Services, broader information may be found in the deliverable *“D3.7 Permissioned Blockchain network M27, section 7 Data Processing Services, common example of use”* [4]

4 AUTOMATIC MODEL TRAINING ENGINE

4.1 OVERVIEW

The Automatic Model Training Engine is a technical solution to create cognitive services for Ports' business KPIs. From a set of already imported datasets, this service provides stakeholders a training dashboard to automatically create an underlying model to answer a specific port KPI, such as the ETD of a vessel or the Tons of goods expected for the following weeks. Data is not stored in the component but imported using the discovery and metadata mechanisms provided by the DataPorts platform. Such data is presented to the end-user to select a specific KPI or to discard irrelevant data. For instance, for the prediction of ETD of a vessel, the end-user could include the information of the arrival terminal. This process, usually named as feature engineering, is important in the further training process, as domain experts are the ones who truly understand which data is potentially useful in a business scenario.

Internally, the component implements a set of predefined training pipelines, using state-of-the-art ML algorithms for domains of regression, forecasting and classification, such as time series forecasting and values imputation. Using a distributed approach, several instances of such pipelines are executed simultaneously to find, without the need of manual intervention, the most accurate model for the end-user selected target. Then, the resulting model is wrapped as a REST Service which can be deployed as a cognitive service.

The Interactions of AMTE with the rest of the components of Dataports platform is fully explained with sequence diagrams, amongst others, in the deliverable *"D3.5 Data processing services M27, section 3 POSITION IN THE DATAPORTS ARCHITECTURE"*. For better understanding, a block diagram is showed in the following figure:

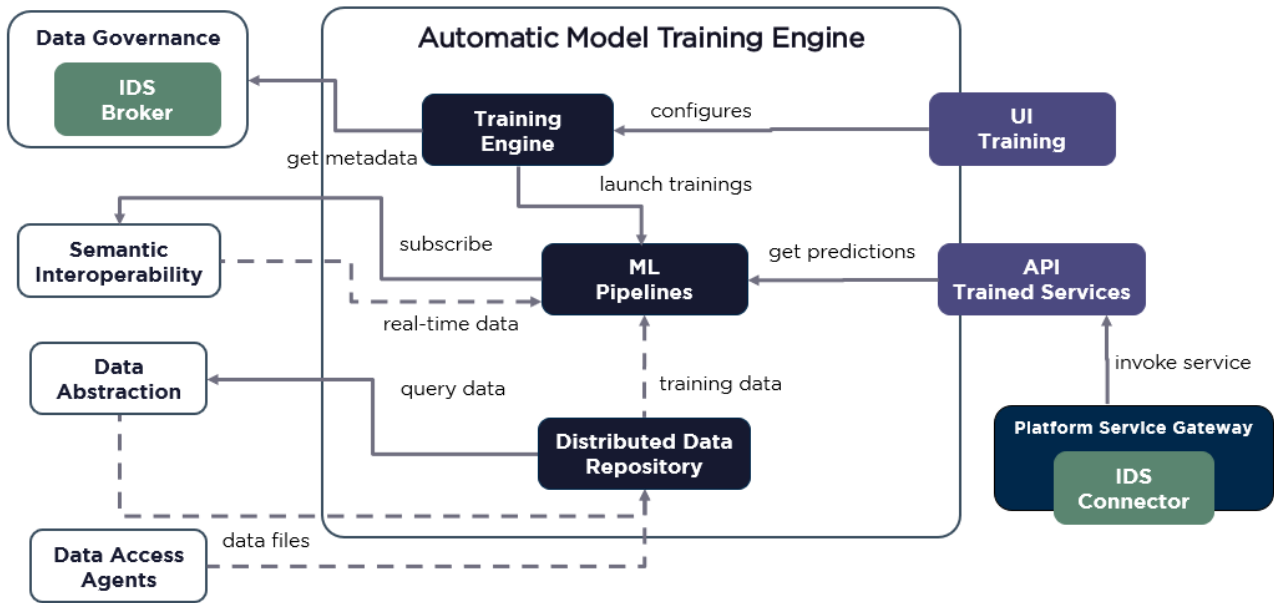


Figure 3 – Interactions of the Automatic Model Training Engine

Regarding the Data Governance Services, broader information may be found in the deliverable *"D3.7 Permissioned Blockchain network M27, section 7 DATA PROCESSING SERVICES COMMON EXAMPLE OF USE"*

4.2 TECHNOLOGICAL DESCRIPTION

4.2.1 Architecture of the component

This subsection shows and explains the architecture designed and the technologies selected to carry out the development of the Automatic Model Training Engine component.

One technical goal of this component is the distributed training of ML models, which it is not only more accurate, but also increases the training speed. The architecture and main technologies are summarised in Figure 4.

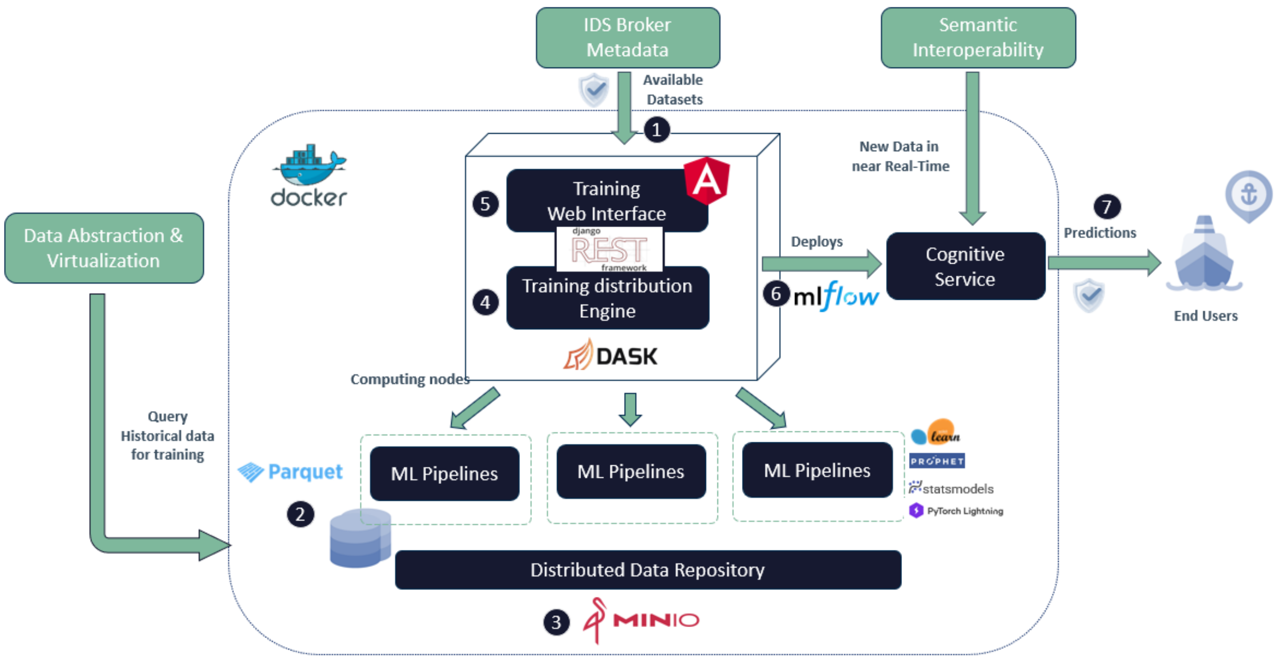
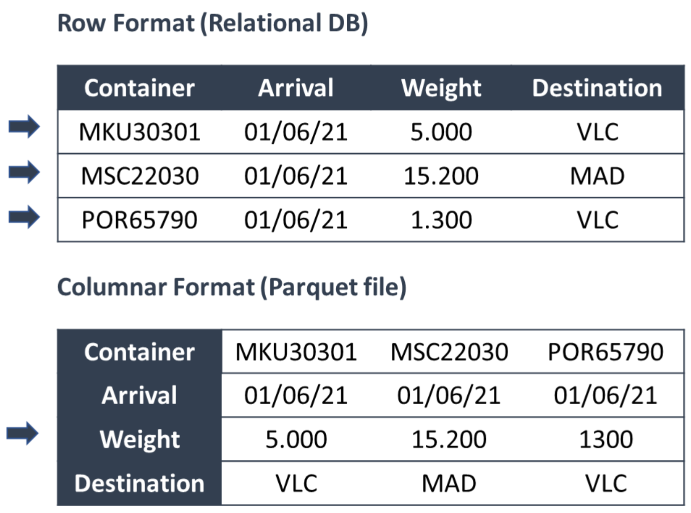


Figure 4 – Architecture of the Automatic Model Training Engine

As starting point, the main inputs of this component are datasets available in the context of the DataPorts ecosystem (Figure 4-1). Next, to achieve a distributed training approach, it is needed an approach to replicate the data in several hosts and avoid network latency due to data transfer. As file size is a relevant metric, Apache Parquet[1] has been selected to store datasets (Figure 4-2). Apache Parquet is a columnar file format that provides optimisations to speed up queries and it's a more efficient format than CSV or JSON. In Parquet files, data is stored in columns, i.e., which it is more efficient than the common row format for performing analysis over data. Specifically, this file format reduces the time to get the data from a single column, as Figure 5 shows. For instance, using a row format it is required to read every row to calculate the average of the weight, whereas using a columnar format with a single read all values are retrieved. Additionally, parquet offers a great data compression as the data type for each column is similar.

[1] Apache Parquet: https://parquet.apache.org/documentation/latest/



Row Format (Relational DB)

Container	Arrival	Weight	Destination
MKU30301	01/06/21	5.000	VLC
MSC22030	01/06/21	15.200	MAD
POR65790	01/06/21	1.300	VLC

Columnar Format (Parquet file)

Container	MKU30301	MSC22030	POR65790
Arrival	01/06/21	01/06/21	01/06/21
Weight	5.000	15.200	1300
Destination	VLC	MAD	VLC

Figure 5 – Row vs Columnar format

Imported datasets are stored and replicated in a MINIO[2] server (Figure 4-3). MINIO is a popular open source and distributed object storage technology utilized for its high performance. Therefore, the approach chosen replicates the parquet files into several nodes to store large data distribution workloads for future machine learning and analytics applications. Initially HDFS was selected as the distributed storage system, however due to technical circumstances, the technology was substituted by MINIO (see ANNEX A: EXTERNAL EXPERT REPORTANNEX A:). One of the key technical capabilities is to train simultaneously several ML models using different algorithms and parameters. (Figure 4-4). Then, the most accurate model is selected for deployment as described in section 5.2.2. This distributed training approach is implemented using Dask[3]. Dask is a framework that introduces parallelism mechanisms for Python scripts thus, enabling performance at scale for the most common ML frameworks. This technology fits well with the current analytics stacks, such the based on the scikit-learn framework, leveraging its execution in several distributed model. One great advantage of Dask is that with minimal Python code changes, the program can be run in parallel by taking advantage of the multi-host processing power. The main difference with Spark, one of the standards for this task, is that cluster configuration is simplified, and the programming paradigm is friendlier to data scientists. Additionally, Radiatus[4] was previously selected as distributing technology, however due to technical issues, the technology was finally avoided (see ANNEX A: EXTERNAL EXPERT REPORT).

Dask provides two mechanisms for this multi-host processing: Distribute the data in multiple hosts for training a single model or train several models using the same data in multiple hosts. For this component, we have followed the second approach. When a new training is defined, the Dask scheduler is responsible to send the different training configurations to the available nodes and get the resulting model with an accuracy metric. Additionally, we have also introduced parallelisation at the thread level, so several models are trained simultaneously in the same host CPU. With this improvement, the performance of the training process is greatly increased over standard ML frameworks.

To simplify the interaction of end-users with the component, a Training Web Interface was implemented (Figure 4-5) as described in the following section. The developed frontend communicates with Django API to start the distributed training according to the end-user requirements. When the backend receives a new training request, sends a notification to the Dask scheduler, which starts this process. The training task is

[2] MINIO: https://min.io/

[3] DASK: https://dask.org/

[4] RADIATUS: https://radiatus.iti.es/

performed asynchronously to not interrupt the interaction with the user interface. When the training is completed, the best model is deployed automatically (Figure 4-6) and can be used via its API.

4.2.2 Training Web Interface

This subsection describes the structure of the Training web interface developed and all the technologies used to carry out its implementation in the component.

The Training Web Interface implemented provides, to a potential user of the component, functionalities such as: importing datasets, selecting relevant variables, choosing the most suitable training strategy and finally, deploy or stop a trained cognitive service. This web interface is made up of two subcomponents: A front-end, developed in Angular Framework[5] following the Single-Page Application pattern, and an API back-end, developed in Python and supported by Django REST[6] framework.

4.2.2.1 FRONTEND: ANGULAR FRAMEWORK

The front-end was developed in Angular due to the following reasons: Angular is a Javascript framework written in the language called Typescript and is open source with clear advantages for developers providing a standard structure. Also, Angular code is modular, reusable, efficient and is faster than any other framework for frontends. Angular, uses (MVC Model–view–controller) architecture based on components, modules, and services. The components define views, and the services provide specific functionality. Modules define a group of components, directives, pipes, and services which are related to the application. Furthermore, Angular is one of the best security frameworks offering support to prevent two common HTTP vulnerabilities such as CSRF and XSRF and cross-site script inclusion (XSSI).

4.2.2.2 BACKEND: DJANGO FRAMEWORK. API REST DJANGO

The backend was developed using Django framework as the most suitable technology for the component purposes. Django is a high-level Python web framework to build websites or APIs. The architecture of this framework is clearly separated using DRY (Don’t Repeat Yourself) concept to eliminate needless repetition and reduce code. Moreover, Django REST offers generic classes for CRUD (Create, read, update, and delete) operations and offers a full complete authentication and authorization system. It also provides a complete ORM (Object–relational mapping) to manage all modern’s database technologies in an easy and common manner. One of the best features in Django REST is the security, offering protection for different web hacking attacks.

In addition, Django provides an authentication framework integrated with Django admin UI. It allows to handle user accounts, permissions, and groups that define whether a user can access to a view or has restricted content. Django authentications system restricts any content to a user through permissions which can be assigned to a specific group. Users can be included in a certain group restricted to view its content. This is very useful in different applications for different purposes such as deny content or full views only visible to users with elevated privileges.

4.2.2.3 BACKEND AND FRONTEND COMMUNICATION

Angular and Django API communicates over HTTP protocol. The client (Angular) sends messages called “requests” and server (Django) answer with messages called “responses”. HTTP protocol is stateless which means there is no link between two requests being successively carried out on the same connection.

[5] AngularJs: https://docs.angularjs.org/guide

[6] Django REST: ttps://www.django-rest-framework.org/

Angular provides a client HTTP API with the HttpClient service class in HttpClientModule with the ability to request typed response objects, request, and response interceptions. On the other hand, Django creates HTTP packets through django.http module. When message arrives, Django creates an HTTPRequest object that contains metadata about the request. Then Django forwards the request to the appropriate view. Each view is responsible for returning a HttpResponse object which contains all necessary data (calling any other necessary service like g et data from a DDBB) so the front may render the view properly.

The communication between both two frameworks is shown in the next figure:

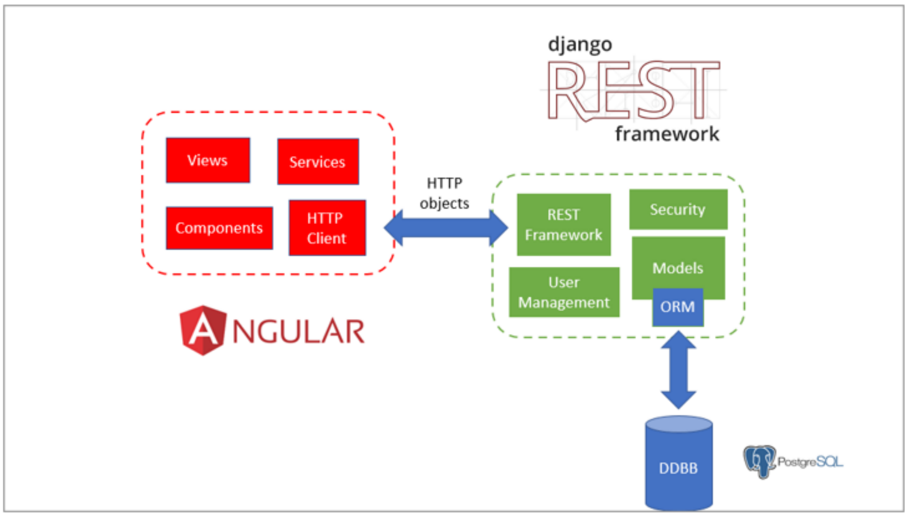


Figure 6 – Communication between the frontend and backend of AMTE

4.2.2.4 COMPONENTS PACKAGING

Finally, to be able to deploy the component in any infrastructure, all subcomponents have been packaged by using Docker[7]. Docker is an open-source containerization platform that allows users to package applications into containers. Therefore, the application may be easily deployed, abstracting of all complex configurations over any operating systems and environment.

To achieve this, Angular and Django frameworks are integrated inside Docker containers respectively. Both containers communicate with each other over HTTP protocol through a specific port. To allow the communication between both, it is necessary to configure some parameters and create two more containers which provide a database to persist data with PostgreSQL[8] and a friendly user interface to manage it, named Pgadmin[9].

On the one hand, Angular container creates a NodeJS[10] environment from scratch in an empty environment and installs all NodeJS necessary requirements (modules and services) to execute the Angular SPA application. On the other hand, Django has his own container with a system-base executing Python where all necessary dependencies are installed from scratch, without any external interference assuring a clean set up.

The full dockerized system is shown in next figure:

[7] Docker: https://www.docker.com/

[8] PostgreSQL: https://www.postgresql.org/

[9] https://www.pgadmin.org/

[10] NodeJS: https://nodejs.org/es/

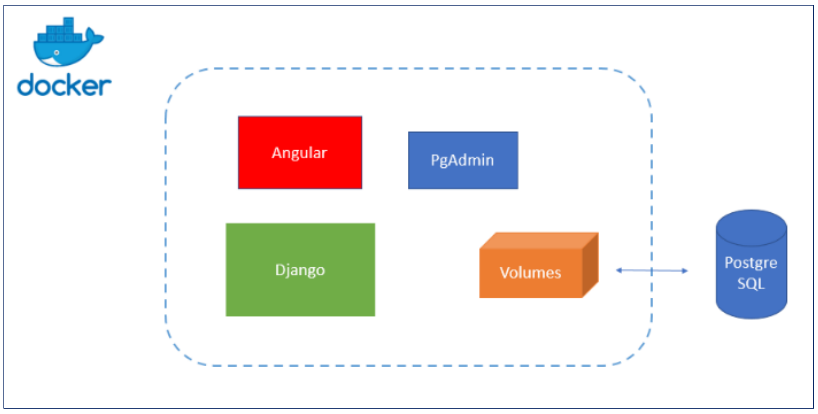


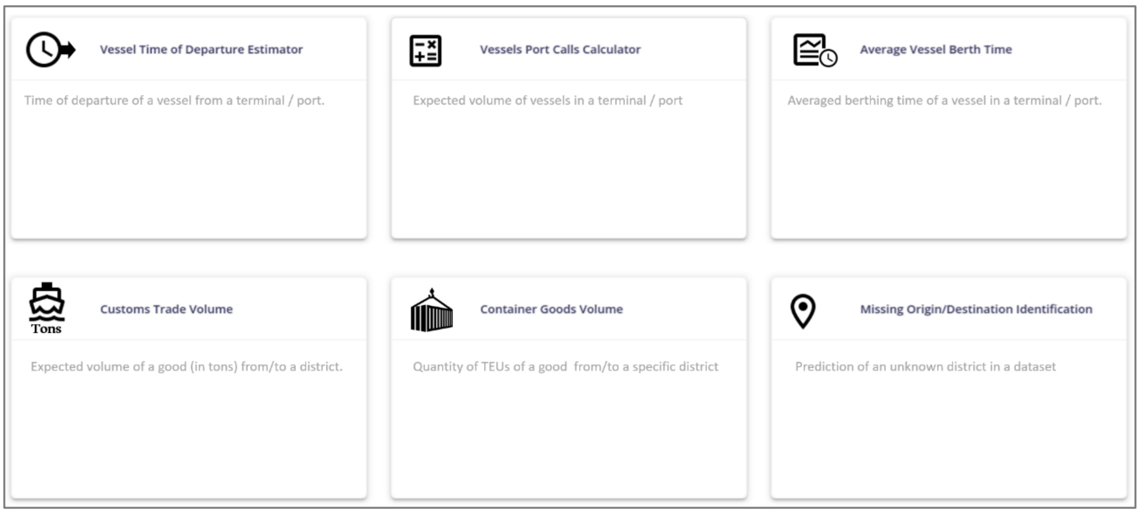
Figure 7 – Dockerized front-end and back end

To orchestrate that series of dockers at once, all the resources are deployed with docker-compose. Hence, all services and interconnections between all the elements are created throughout ports or dependencies, which allow the deployment of the whole application with only one command.

4.2.3 Cognitive Services

This section explains the work developed regarding the cognitive services implemented in the component.

Cognitive Services may be understood as services that provide business insights using available data. Therefore, one of the main goals of the Automatic Model Training Engine is that a DataPorts' end user can use the component to create services that provide useful business information from the available data at the platform. With that respect, there were designed and developed a set of different cognitive services that address some real business needs of the ports. The developed cognitive services are showed the following figure:



Vessel Time of Departure Estimator

Time of departure of a vessel from a terminal / port.

Vessels Port Calls Calculator

Expected volume of vessels in a terminal / port

Average Vessel Berth Time

Averaged berthing time of a vessel in a terminal / port.

Customs Trade Volume

Expected volume of a good (in tons) from/to a district.

Container Goods Volume

Quantity of TEUs of a good from/to a specific district

Missing Origin/Destination Identification

Prediction of an unknown district in a dataset

Figure 8 – Cognitive Services implemented in AMTE

The detailed explanation of each cognitive service is described as follows:

(i) Vessel Time of Departure Estimator (VTDE)

Time Series Forecasting service. This service estimates the date and time of departure of an arriving vessel, based on the port/terminal/regular line associated to it. Once the arriving time is indicated, the ETD will be predicted and will be available to the user. This service currently utilizes historical data of port traffic from

ValenciaPort. To achieve the goal of the service, regression and statistics-based machine learning algorithms are used.

(ii) Vessels Port Calls Calculator (VPCC)

Time Series Forecasting service. This service calculates the amount of port calls that will be expected to occur in a specific port/terminal for a determined time horizon selected by the user. This service currently utilizes historical data of port traffic from ValenciaPort. To achieve the goal of the service, regression and statistics-based machine learning algorithms are used.

(iii) Average Vessel Berth Time (AVBT)

Time Series Forecasting service. This service calculates the berth time in average of a potential vessel docked in a port/terminal for a determined temporal horizon. It provides an idea of the saturation of the port (the longer the ships are docked, a greater volume of traffic is assumed). This service currently utilizes historical data of port traffic from ValenciaPort. To achieve the goal of the service, regression and statistics-based machine learning algorithms are used.

(iv) Customs Trade Volume (CTV)

Time Series Forecasting service. Estimation of the volume of a certain type of good (in tons) from/to a specific district in a certain time horizon in months. This service requires a historical dataset with the imported tons per day. This service currently utilizes historical data from the Spanish Customs Trade Agency. To achieve the goal of the service, regression and statistics-based machine learning algorithms are used.

(v) Missing Origin/Destination Identification (MODI)

Missing Values Imputation service. Prediction of an unknown district from the historical data. The identification of such district will be based on the information of port calls traceability of the given port. To achieve the goal of the service, classification and clustering machine learning algorithms are used. This service currently utilizes historical data of port traceability (hinterland) from ValenciaPort.

(vi) Container Goods Volume (CGV)

Time Series Forecasting service. Estimation of the quantity of TEUS of a certain type of good from/to a specific district in a certain time horizon specified by the user. This service requires a historical dataset with the imported TEUS per day at least. This service currently utilizes historical data of port traceability (hinterland) from ValenciaPort. To achieve the goal of the service, regression and statistics-based machine learning algorithms are used.

4.2.4 Machine Learning pipelines and algorithms

This section describes the Machine Learning (ML) pipelines and algorithms implemented in the training engine. Additionally, it describes the data management process from a DataPorts available dataset to a full trained machine learning model capable of making predictions over future unknown data. The overall approach is that several training pipelines are running in parallel using the technologies introduced in the previous section.

4.2.4.1 Machine Learning Pipelines

A training pipeline is composed of a series of processing steps in which each step applies a transformation to the data to prepare it for the machine learning algorithm. In some cases, independent steps may be run in parallel. The overview of the standard pipeline is showed in Figure 9. Green circles represent the main steps of the pipeline, whereas dark blue circles represent optional sub steps:

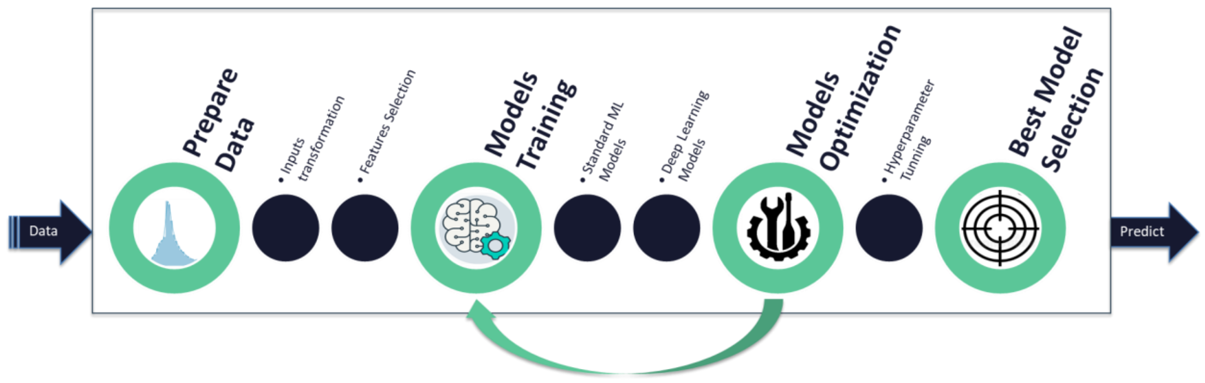


Figure 9 - Standard training pipeline

The main steps of this standard pipeline are:

- **Prepare Data:** The selected dataset is manipulated into a form that can accurately be ingested by the subsequent machine learning models. First, some technical pre-processing techniques are applied on the dataset to both enrich the dataset and deliver it cleaned. Next, by applying statistical and machine learning techniques, the features with most predictive power are selected, and the non-relevant variables are discarded.
- **Models Training:** A series of machine learning models are fed and trained by the incoming prepared dataset. Two different approaches are available: (1) Standard ML Models which use supervised and statistical models for the training, and (2) Deep learning models, specifically, neural networks with more than one hidden layer. The latter approach adds more complexity and technical resources; therefore, it more time is required more time for the last step. Additional details about the models are introduced in Table 1.
- **Models Optimisation:** This step performs a hyperparameter tuning: find the set of training parameters that optimizes the output of a trained model, for all the learning algorithms. If selected, it will drastically increase the training time, while substantially enhancing the predictive power of the trained models. For each optimisation or parameters set, a new model training step is required.
- **Best Model Selection:** This step calculates a set of quality metrics related to the resulting model. Once all models are trained, the model with the best quality metric obtained is selected and packaged to be deployed as a cognitive service.

In addition to this standard training pipeline, a set of four different training strategies are implemented. Each of the training strategies defines a custom configuration of the standard pipeline by adding or avoiding some of the steps. The selection of the most suitable training strategy is left up to the user and it will affect to the needed amount of time and computing resources to obtain the trained predictive model and the quality of the resulting model. The four available training strategies, with their specific steps are shown in Figure 10:

- **Fast Training Strategy:** A quick training process will be carried out to obtain the model outcome as fast as possible. Hence, some processing steps such as inputs transformations or features selection will be avoided. Additionally, the step of the search of the best hyperparameters combination, usually is the most time-consuming step due to its high computational cost. In the end a set of standard algorithms with default parameters will be trained and the model with the best quality metric will be selected to make future predictions.
- **Standard Training Strategy:** First, a set of mathematical transformations will be applied to the inputs to enrich the dataset and find intrinsic patterns within the data. After that, a thorough feature selection process will be performed so the most important independent features will be selected. That will enhance the data quality for the training process. The set of algorithms remains the same than in the

previous one, but better-quality metrics are expected with not so much overhead in the training time.

- **Optimum Training Strategy:** This strategy adds optimisation steps into the pipeline to find out the best possible training configuration. The main improvement resides in the hyperparameters optimisation step which performs a grid-search process where several combinations of parameters will be tried with each of the available algorithms. As each combination implies the training of a specific model, this strategy is recommended to be ran in a distributed environment with several nodes.
- **Deep Training Strategy:** This training strategy is like the standard strategy, but it differs in the utilisation of Deep Learning (DL) algorithms. In this case only neural networks with more than one hidden layer will be trained. DL algorithms are usually able to find very complex patterns within the data, but they require high computational cost and a lot of time to converge. Additionally, they do not highly benefit of the optimisation of the previous approach. This strategy is expected to be the most time consuming.

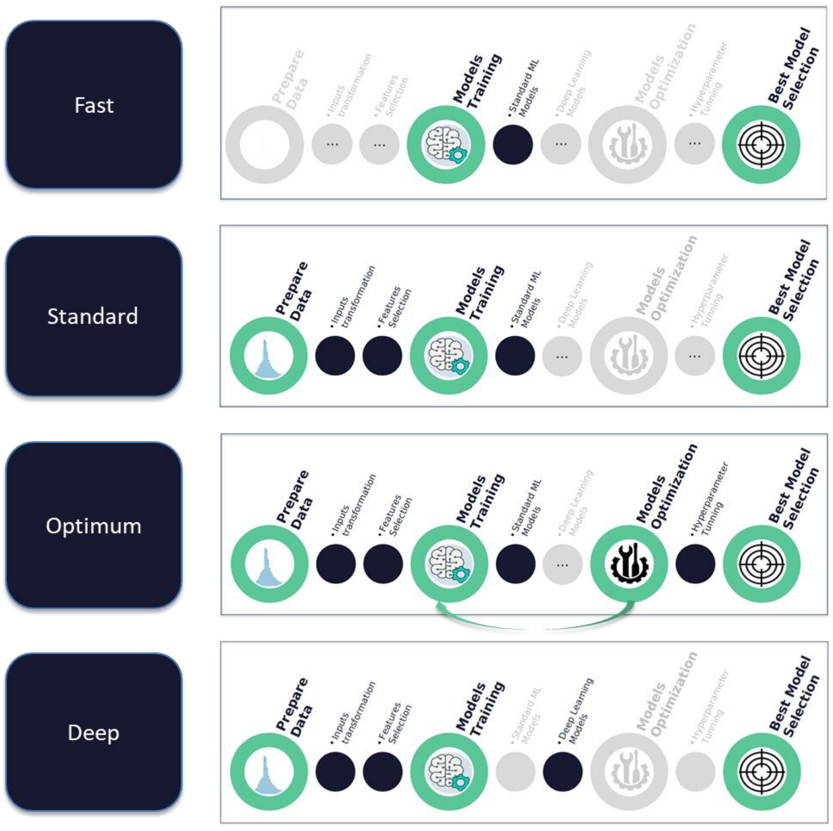


Figure 10 - Training Strategies

4.2.4.2 Machine Learning algorithms

As part of the strategy, the most critical decision in a ML pipeline is the ML algorithm selection in the Models training step of the pipeline. As our approach does not require manual intervention, a collection of available algorithms is selected beforehand and all of them are tested for training, according to the chosen strategy. The initial use cases proposed by the DataPorts pilots are specifically oriented to the prediction of KPIs using information from their specific IT systems, mainly PCS and TOS.

The tested algorithms have been selected due to their specific mathematical behaviour related to their predictive power regarding the initial datasets provided by the DataPorts data providers and to try to solve their business needs as accurately as possible.

In this context, the business predictive problems identified and thus, the groups of different algorithms, are of two natures: Time Series Forecasting (i) and Values Imputation (ii).

Time series forecasting domain

Giving a temporal series of data, try to guess how a specific value (for instance the number of imported containers) will behave in the near future. After performing an analysis of the state-of-the-art and some tests with benchmarking data, the prediction task is tackled by using regression and forecasting algorithms. Therefore, the selected algorithms are detailed in Table 1:

Time Series Forecasting algorithms				
Algorithm	Learning type	Model Type	Library	Prediction type
Multiple Linear Regression [5]	Supervised	Linear model	sklearn	Univariate TSF
KNN Regressor [6]	Supervised	Nearest Neighbours	sklearn	Univariate TSF
Multi-layer Perceptron NN [7]	Supervised	Neural Network	sklearn	Univariate TSF
Support Vector Regressor [6]	Supervised	Support Vector Machines	sklearn	Univariate TSF
Random Forest Regressor [6]	Supervised	Ensemble	sklearn	Univariate TSF
Gradient Boosting Regressor [8]	Supervised	Ensemble	sklearn	Univariate TSF
SARIMAX [9]	Statistical	Autoregressive	statsmodels	Univariate TSF
VARMAX [9]	Statistical	Autoregressive	statsmodels	Multivariate TSF
Prophet [10]	Statistical	Autoregressive	prophet	Univariate TSF
LSTM [11]	Deep Learning	Recurrent Neural Network	torch	Univariate TSF

Table 1 – Implemented ML algorithms for Time Series Forecasting

There exists plentiful number of metrics to evaluate ML models nowadays. However, only several are useful to evaluate ML models focused on Time Series (see Table 2). Those metrics give a roughly accurate idea on how good the models behave making predictions on future sequential and unknown data, to finally identify which is the best performing algorithm.

Specifically, in the context of the project, MAPE has been the selected as the preferred metric due to its easy understanding from a non-data-expert point of view. Although all metrics express somehow how accurate a model is, MAPE is the only metric that ranges from 0 to 100, thus making it easier for end users to understand. MAPE is a metric based on error calculation and it returns that error as a percentage, being the lower the percentage, the more accurate the model, hence a metric value of 10% would be better than 50%.

The analysis of the best metrics applied to Time Series Forecasting are represented in the following table:

Time Series Forecasting metrics				
Metric	Acronym	Type of metric	Boundaries	Explanation
Mean Absolute Percentage Error	MAPE	Error	[0, 100]	MAPE is the sum of the individual absolute errors divided by the demand (each period separately)
Mean Absolute Error	MAE	Error	[0, Inf)	How far on average are the predictions from the actual values
Root Mean Squared Error	RMSE	Error	[0, Inf)	Square root of the mean squared errors. RMSE gives relatively high weight to large errors
R-squared	R2	Quality	[0, 1]	How well the model explains the values of the dependent variable. Amount of variance of the output, explained by the model

Table 2 – Evaluation metrics for Time Series Forecasting

To prove the suitability of the pipelines and algorithms previously explained, a testing case was performed as follows:

First, the pipeline was tested with several datasets with time-series information to get a better know-how about the behaviour of the different algorithms and how the different parameter configuration influences the accuracy. Next, a small cognitive service was built, using historical vessel port calls from Valencia Port, with the goal of predicting the average berth time, i.e., the average time a vessel is berthed in the port executing an operation, in the following months. This KPIs provides an overall view of the expected traffic of the port helping to a better planning of the logistic operations. The name of this service is "Average vessel berth time" and, for testing purposes, the optimum strategy has been applied to find the best model to forecast 5 months ahead. The results obtained are as follows:

Model	Parameters	MAPE	Std
SARIMAX	'D': 0, 'P': 0, 'Q': 0, 'd': 0, 'p': 1, 'q': 0, 's': 0	25.840	0.00
SVR	'kernel': 'rbf'	26.540	0.00
Prophet	'seasonality_mode': 'additive'	27.430	0.00
NN	'hidden_layer_sizes': 100	28.808	1.85
LR	'fit_intercept': True	34.360	0.00
KNN	'n_neighbors': 5	43.590	0.00
RF	'n_estimators': 100	54.344	3.29
XGBoost	'loss': 'ls'	63.198	0.16

Table 3 - Models scoreboard for berth time predictor

As explained above, MAPE metric was chosen as it is easier to understand by end-users. As it can be observed in the previous table, the best model found by the pipeline is the SARIMAX with the default hyperparameters, reaching a MAPE of 25.8%. Therefore, that is the model selected for making forecasts over the future berth average time. Table 3 shows the forecasts with data of the five first months of 2021.

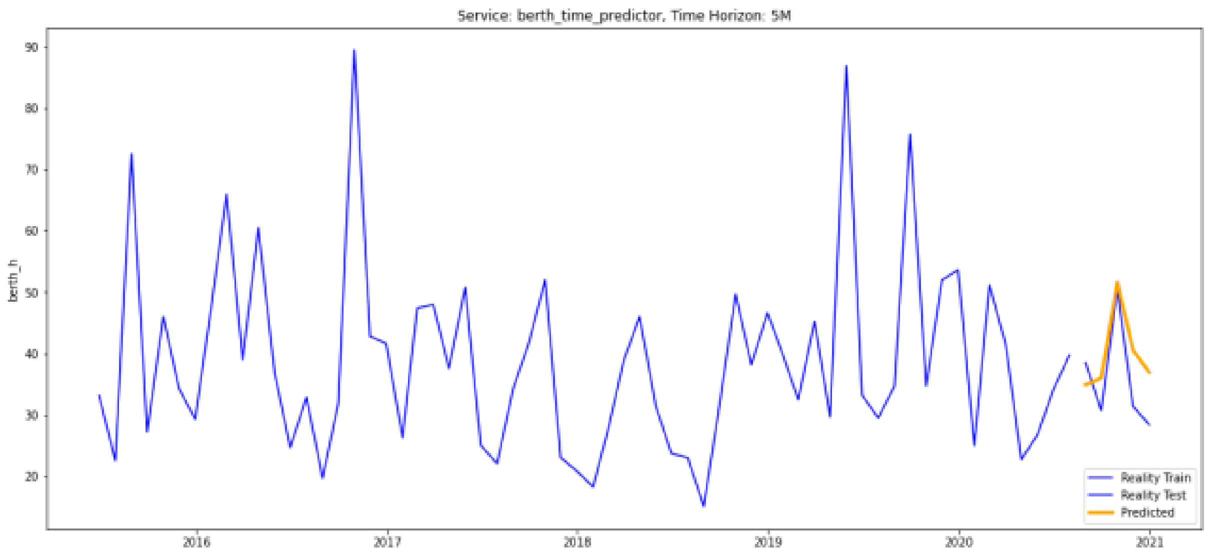


Figure 11 - Forecast of Average Vessel Berth Time for the next 5 months

As expected, the predicted series (in orange) is very close to the real one, as the MAPE metric of the best model is quite good. It is worth to mention that the prediction follows the trend expected by the KPI.

Values Imputation domain

Also known as missing values imputation, giving a certain dataset provided by a data provider, identify, and fill up the existing gaps within the data, of a potential target variable. For the prediction of missing values, the information of the predictors is used to project over new unknown values of the prediction. Hence, there have been identified a group of classification and imputation algorithms that may address the challenge with certain degree of success. The collection of selected algorithms is detailed in Table 4:

Imputation algorithms				
Algorithm	Learning type	Model Type	Library	Prediction type
Logistic Regression [12]	Supervised	Linear model	sklearn	Classification
KNN classifier [6]	Supervised	Nearest Neighbours	sklearn	Classification
Gaussian Naive Bayes [13]	Supervised	Bayes	sklearn	Classification
Decision Tree classifier [14]	Supervised	Tree	sklearn	Classification
Multi-layer Perceptron NN [7]	Supervised	Neural Network	sklearn	Classification
Random Forest Classifier [6]	Supervised	Ensemble	sklearn	Classification
AdaBoost classifier [15]	Supervised	Ensemble	sklearn	Classification
Gradient Boosting classifier [16]	Supervised	Ensemble	sklearn	Classification
XGBoost classifier [17]	Supervised	Ensemble	sklearn	Classification
KNN imputer [18]	Supervised	Regressor – RoundRobin	sklearn	Imputation
Iterative Imputer [18]	Supervised	Regressor – RoundRobin	sklearn	Imputation

Table 4 – Implemented ML algorithms for Missing values imputation

With respect to the predictions based on values imputation, different metrics are used to evaluate the quality of the results outcome from the models. In this case all metrics are quality based, meaning that they range from 0 to 1, being 0 the worst-case scenario, when the model has no predictive power at all, and being 1 the best case, when the model is able to perfectly predict the missing values.

The analysis of the best metrics applied to Values Imputation are represented in the following table:

Imputation metrics				
Metric	Acronym	Type of metric	Boundaries	Explanation
Accuracy	ACC	Quality	[0, 1]	Model hit ratio
Precision	Precision	Quality	[0, 1]	Indicates how reliable the model is. Level of false positives.
Recall	Recall	Quality	[0, 1]	It indicates how complete are the positive results that the model yields. Level of false negatives.
Area Under the Curve ROC	AUC	Quality	[0, 1]	Ability of the model to detect true positives and avoid false positives

Table 5 – Evaluation metrics for Values Imputation

4.2.5 Models tracking and deployment

This section describes the approach that was developed to let the end-user analyse the collection of machine learning models trained for a certain cognitive service.

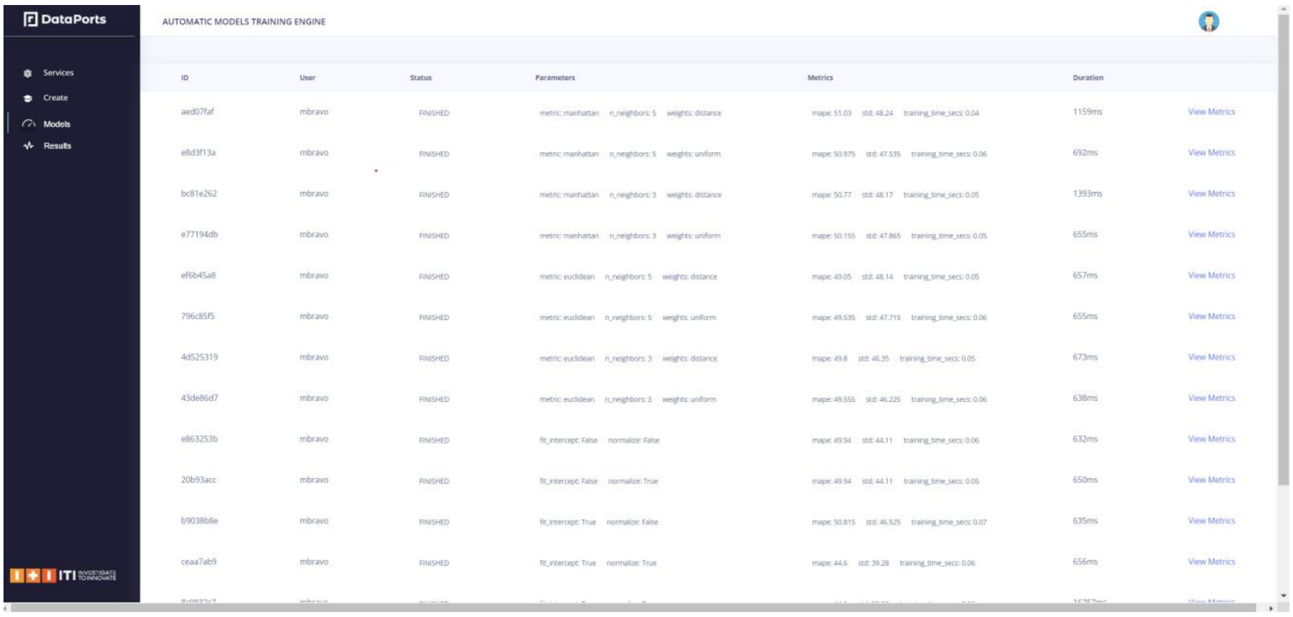
To enable the exploration of the trained models, **MLFlow**[11] was implemented. MLFlow is an open-source platform to manage the machine learning lifecycle, including amongst others, experimentation, models tracking and deployment of machine learning models. Additionally, MLFlow allows the integration with multiple platforms and libraries used to develop the component, such as Python, SciKitLearn, Statsmodels, Conda… and others.

The available functionalities and benefits are the following:

- Analyse the collection of algorithms trained
- Visualize the metrics obtained by each model
- Perform a deep comparison of models
- Investigate the hyperparameters used

MLFlow was integrated with the front-end of the component, so a potential end-user can make use of its potential. See the following figure:

[11] MLFLow: https://mlflow.org/



ID	User	Status	Parameters	Metrics	Duration	
aed07faf	mbravo	FINISHED	metric: manhattan n_neighbors: 5 weights: distance	mape: 51.03 std: 48.24 training_time_secs: 0.04	1159ms	View Metrics
e8d3f13a	mbravo	FINISHED	metric: manhattan n_neighbors: 5 weights: uniform	mape: 50.975 std: 47.535 training_time_secs: 0.06	692ms	View Metrics
bc81e262	mbravo	FINISHED	metric: manhattan n_neighbors: 3 weights: distance	mape: 50.77 std: 48.17 training_time_secs: 0.05	1393ms	View Metrics
e77194db	mbravo	FINISHED	metric: manhattan n_neighbors: 3 weights: uniform	mape: 50.155 std: 47.865 training_time_secs: 0.05	655ms	View Metrics
ef6b45a8	mbravo	FINISHED	metric: euclidean n_neighbors: 5 weights: distance	mape: 49.05 std: 48.14 training_time_secs: 0.05	657ms	View Metrics
796c85f5	mbravo	FINISHED	metric: euclidean n_neighbors: 5 weights: uniform	mape: 49.535 std: 47.715 training_time_secs: 0.06	655ms	View Metrics
4d525319	mbravo	FINISHED	metric: euclidean n_neighbors: 3 weights: distance	mape: 49.8 std: 46.35 training_time_secs: 0.05	673ms	View Metrics
43de86d7	mbravo	FINISHED	metric: euclidean n_neighbors: 3 weights: uniform	mape: 49.555 std: 46.225 training_time_secs: 0.06	638ms	View Metrics
e863253b	mbravo	FINISHED	fit_intercept: False normalize: False	mape: 49.94 std: 44.11 training_time_secs: 0.06	632ms	View Metrics
20b93acc	mbravo	FINISHED	fit_intercept: False normalize: True	mape: 49.94 std: 44.11 training_time_secs: 0.05	650ms	View Metrics
b9038b8e	mbravo	FINISHED	fit_intercept: True normalize: False	mape: 50.815 std: 46.525 training_time_secs: 0.07	635ms	View Metrics
ceaa7ab9	mbravo	FINISHED	fit_intercept: True normalize: True	mape: 44.6 std: 39.28 training_time_secs: 0.06	656ms	View Metrics

Figure 12 – MLFlow integrated in the front-end of AMTE

4.3 FUTURE WORK: ROADMAP

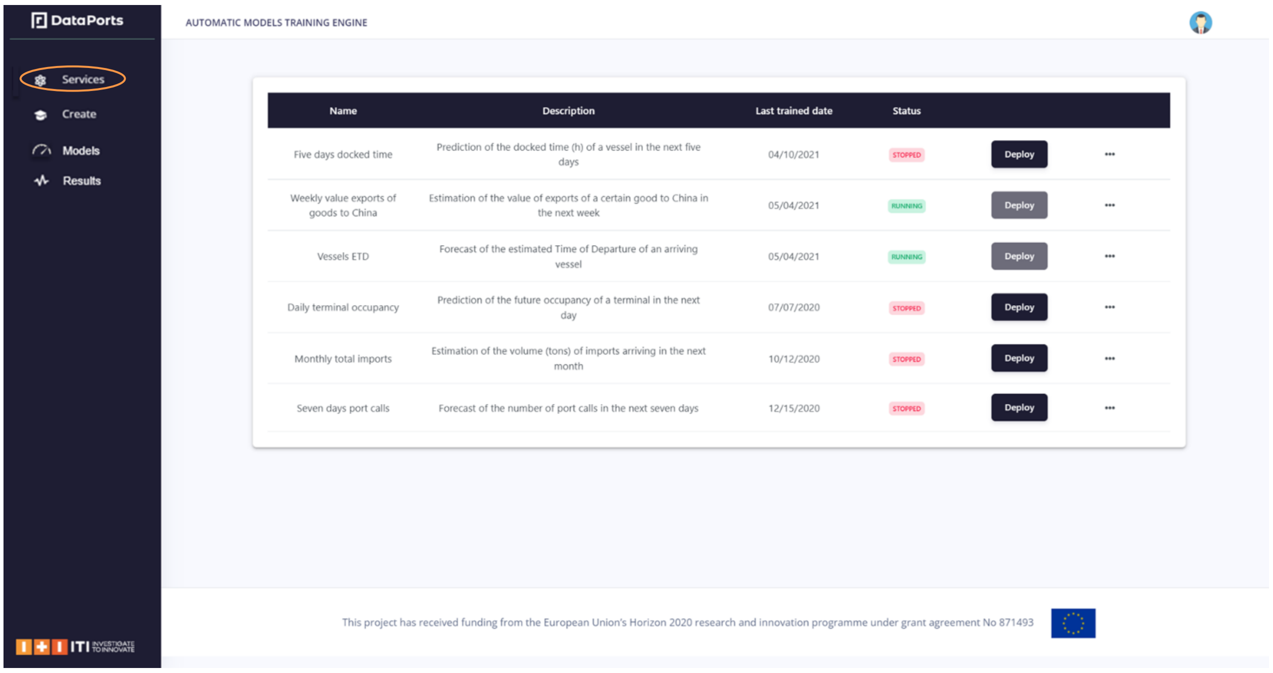
Regarding the plan for future versions of the Automatic Model Training Engine up to M30 (final version of the platform), the roadmap consists of the following planned tasks:

- Contributing to the generation of insight and knowledge from the datasets offered in the platform. If new datasets are ingested by the platform, new and broader value can be extracted.
- Inclusion of new predictive tasks apart from the existing Time Series Forecasting and Values Imputation to offer a wider selection of possibilities
- Enrichment of the machine learning models catalogue to update the collection with novelty algorithms that may perform differently
- Creation of additional Cognitive Services based on business needs to increase the offer currently available

4.4 EXAMPLE OF USE: DEMONSTRATION

A frontend is available to the user in the URL selected in the deployment configuration. The main interface presents a list of the already existing services and their status, i.e., if they are currently running or not. Additionally, a menu is provided on the left-hand side (see Figure 13) with the following entries:

- Services: Returns to the main interface to list the existing services
- Create: Opens a wizard to define a new service and start the training of the underlying ML models
- Models: Shows detailed information of the collection of ML models for each trained cognitive service
- Results: Presents the predictions made by a trained cognitive service as well as an interactive dashboard to analyse the results



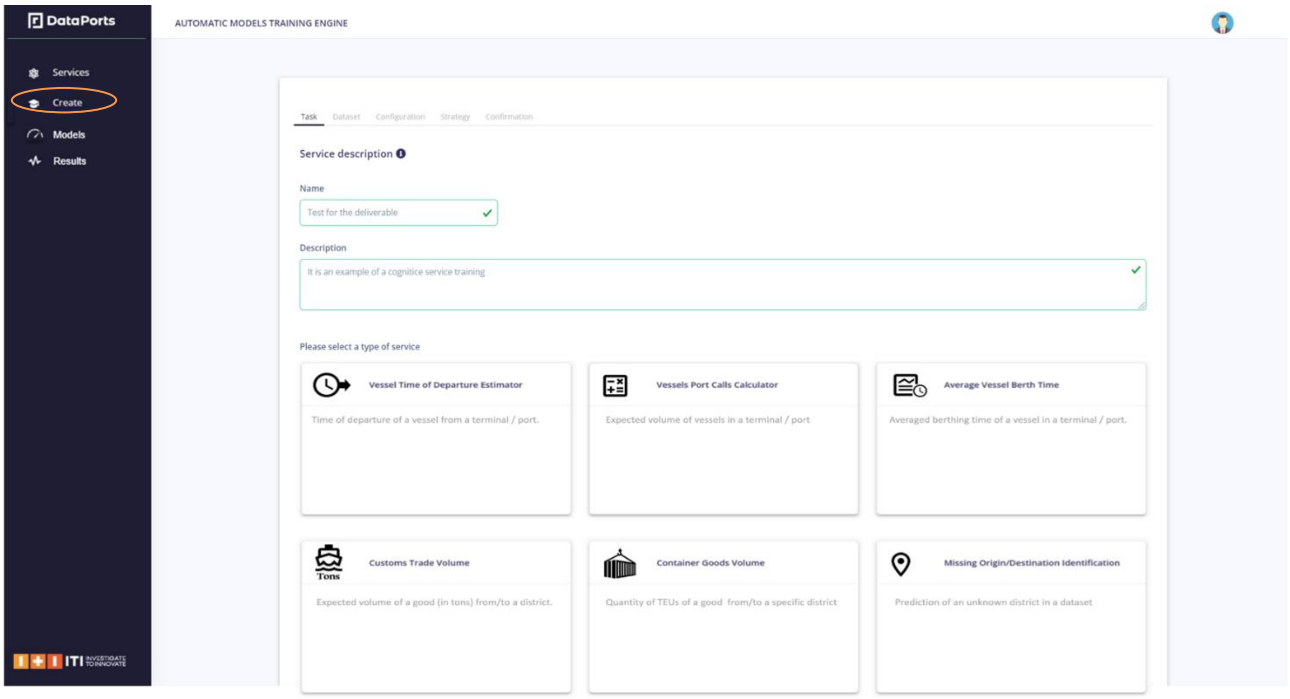
Name	Description	Last trained date	Status		
Five days docked time	Prediction of the docked time (h) of a vessel in the next five days	04/10/2021	STOPPED	Deploy	•••
Weekly value exports of goods to China	Estimation of the value of exports of a certain good to China in the next week	05/04/2021	RUNNING	Deploy	•••
Vessels ETD	Forecast of the estimated Time of Departure of an arriving vessel	05/04/2021	RUNNING	Deploy	•••
Daily terminal occupancy	Prediction of the future occupancy of a terminal in the next day	07/07/2020	STOPPED	Deploy	•••
Monthly total imports	Estimation of the volume (tons) of imports arriving in the next month	10/12/2020	STOPPED	Deploy	•••
Seven days port calls	Forecast of the number of port calls in the next seven days	12/15/2020	STOPPED	Deploy	•••

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871493

Figure 13 – Dashboard main interface

4.4.1 Creation of a cognitive service

To create a new service, the wizard as show in Figure 14 is provided. In the first step, "Task", the end-user defines a service name and its optional description about what it is expected to accomplish. Then, the wizard provides several cognitive services to fulfil a specific type of prediction, such as forecast the ETD of a vessel or calculate the received tons of a specific good for the next months.



Service description

Name

Test for the deliverable

Description

It is an example of a cognitice service training

Please select a type of service

Vessel Time of Departure Estimator — Time of departure of a vessel from a terminal / port.

Vessels Port Calls Calculator — Expected volume of vessels in a terminal / port

Average Vessel Berth Time — Averaged berthing time of a vessel in a terminal / port.

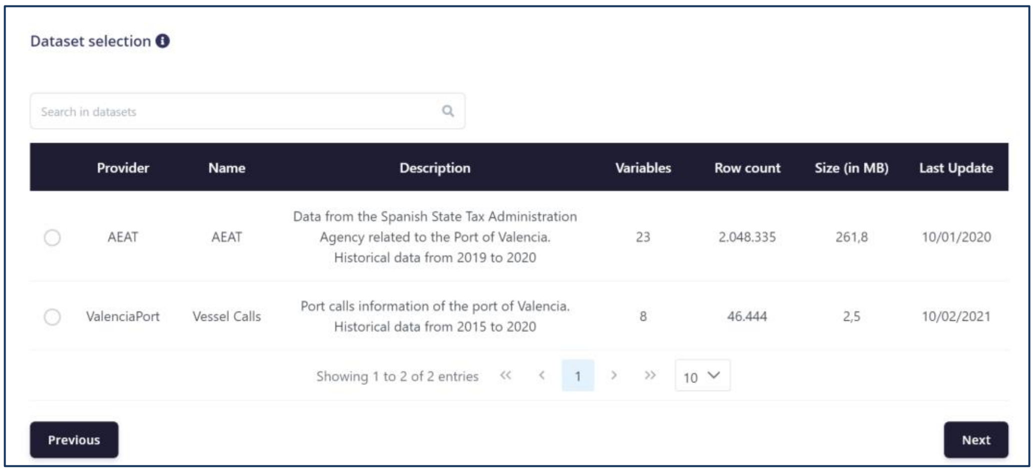
Customs Trade Volume — Expected volume of a good (in tons) from/to a district.

Container Goods Volume — Quantity of TEUs of a good from/to a specific district

Missing Origin/Destination Identification — Prediction of an unknown district in a dataset

Figure 14 - Services definition

The second step of the wizard is named Dataset (see Figure 15). In this step, the user is presented a searchable list of available datasets is presented on the DataPorts platform, which provides the data required to create the service. Only datasets with the required data are available for selection. For instance, if the user requires to predict the vessel ETD, only datasets with historical records of the arrival and departure of vessels are shown. This dataset filtering is achieved using the functionality of the Semantic Interoperability API and the DataPorts data model. It is important to highlight those datasets are only available if permissions are already granted to an organisation using the data governance functionality of the platform. To be compliant with the security and governance guidelines, they are deleted from the component infrastructure when the training process is finished.


Dataset selection

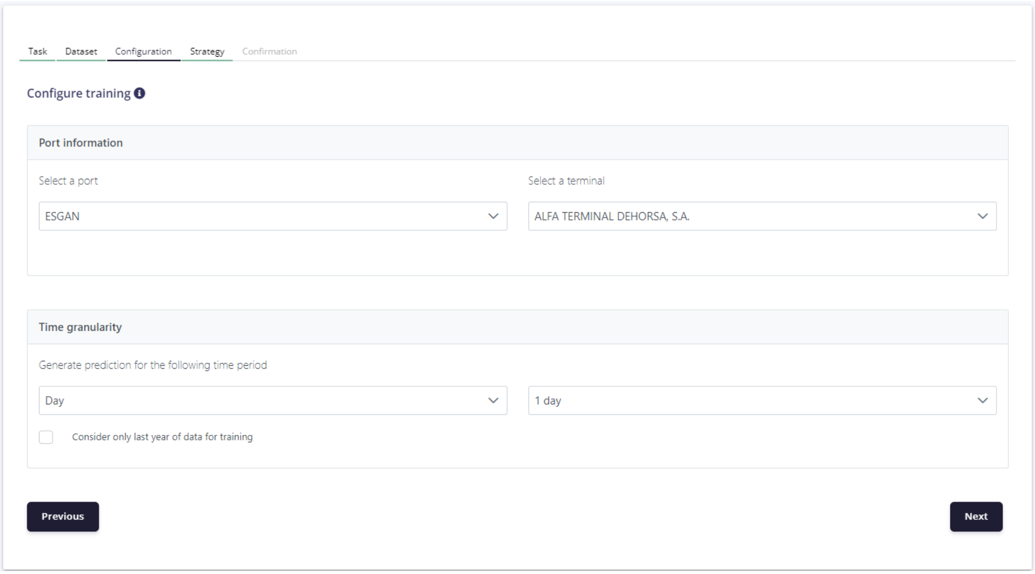
Search in datasets

	Provider	Name	Description	Variables	Row count	Size (in MB)	Last Update
○	AEAT	AEAT	Data from the Spanish State Tax Administration Agency related to the Port of Valencia. Historical data from 2019 to 2020	23	2.048.335	261,8	10/01/2020
○	ValenciaPort	Vessel Calls	Port calls information of the port of Valencia. Historical data from 2015 to 2020	8	46.444	2,5	10/02/2021

Showing 1 to 2 of 2 entries « ‹ 1 › » 10

Figure 15 - Dataset Selection Screen

The third step, called Configuration (see Figure 16), is specific for each service. In this screen, the user can choose additional information or options to be considered for training the underlying models. For instance, in the configuration of the service Vessel Average Berth Time, the end-user could select consider only vessels arriving to a specific terminal or port, as terminals in the same port have different processes and traffic, mixing information from different ones could lead to low accuracy in the prediction. Additionally, the service could be configured to predict using a chosen time granularity: predict the average berth time for the next days, weeks, or months. A proper configuration of such options could greatly impact the model's accuracy. In the case of the configuration of Cognitive Services associated with a time variable, the option "Consider only last year of data for training" will be available. Hence, more accurate models may be trained as the training data is closer to the time of the prediction. It is worth to mention, that only options relevant to the end-user expertise are available, avoiding options related with the underlying ML training process.



Configure training

Port information

Select a port: ESGAN

Select a terminal: ALFA TERMINAL DEHORSA, S.A.

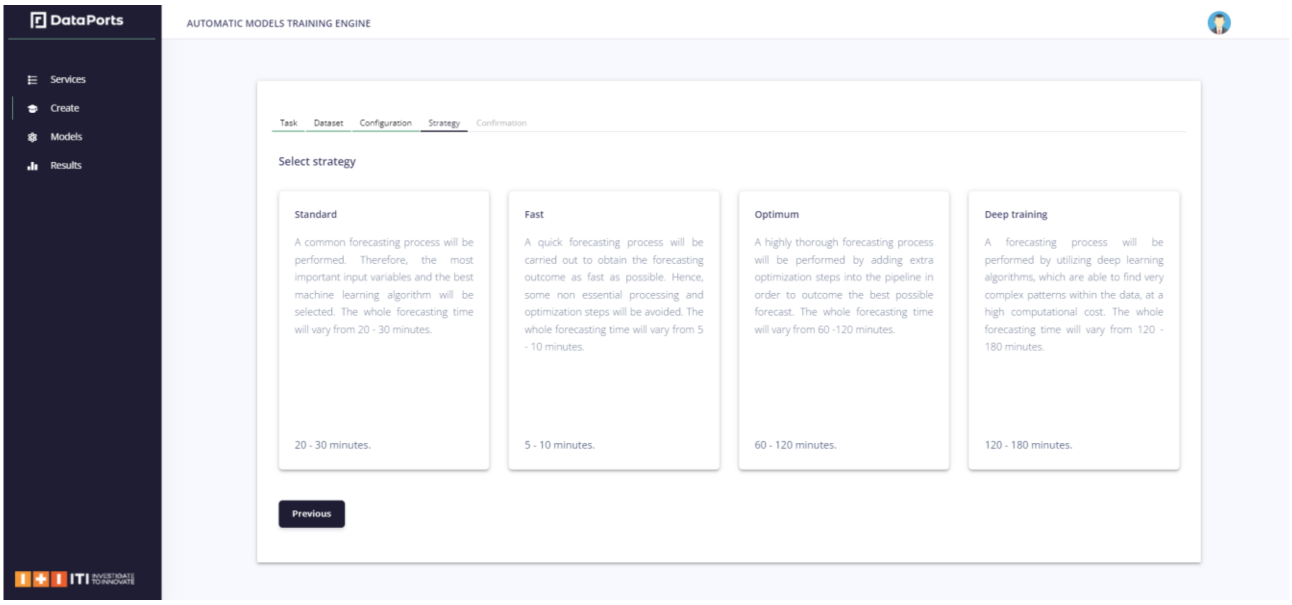
Time granularity

Generate prediction for the following time period: Day / 1 day

☐ Consider only last year of data for training

Figure 16 - Configuration Screens

The fourth step is named *Strategy* (see figure below). This screen shows a list of the different training strategies, already introduced in the previous section. The end-user must select any of them depending on its time availability and performance requirements. Depending on the strategy selected as specific set of algorithms and associated parameters will be automatically choose by the training engine.



Select strategy

Standard
A common forecasting process will be performed. Therefore, the most important input variables and the best machine learning algorithm will be selected. The whole forecasting time will vary from 20 - 30 minutes.
20 - 30 minutes.

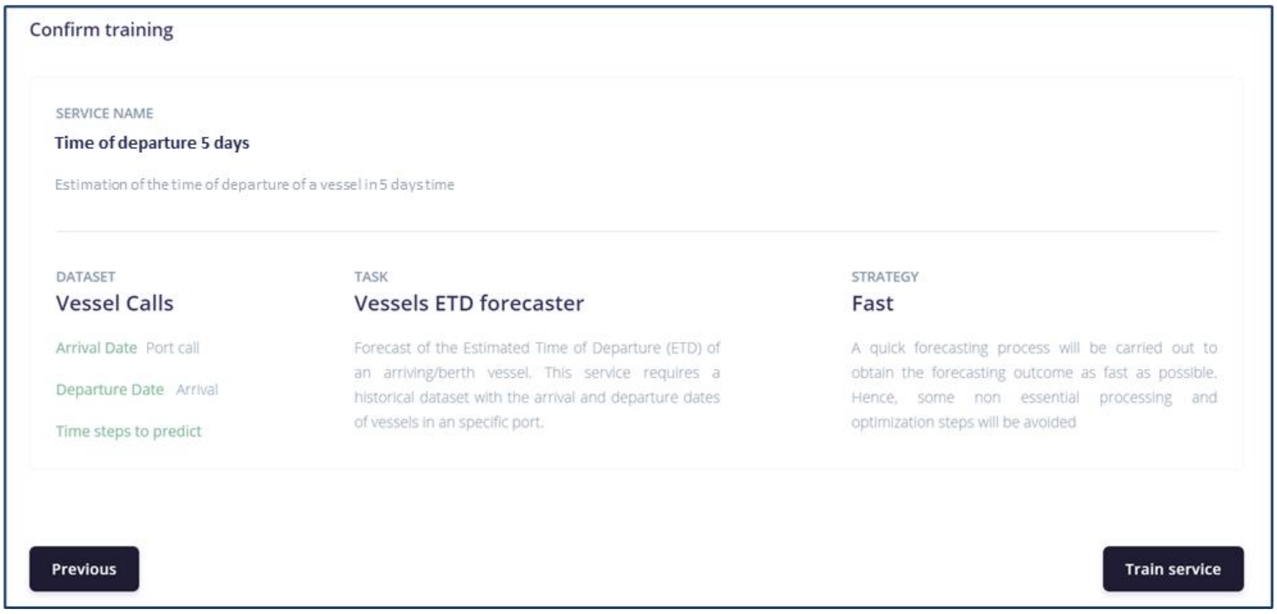
Fast
A quick forecasting process will be carried out to obtain the forecasting outcome as fast as possible. Hence, some non essential processing and optimization steps will be avoided. The whole forecasting time will vary from 5 - 10 minutes.
5 - 10 minutes.

Optimum
A highly thorough forecasting process will be performed by adding extra optimization steps into the pipeline in order to outcome the best possible forecast. The whole forecasting time will vary from 60 -120 minutes.
60 - 120 minutes.

Deep training
A forecasting process will be performed by utilizing deep learning algorithms, which are able to find very complex patterns within the data, at a high computational cost. The whole forecasting time will vary from 120 - 180 minutes.
120 - 180 minutes.

Figure 17 - Strategy selection screen

The final step is named *Confirmation* (see figure below). In this step, the detailed information of all the choices and selections made during the wizard are presented for reviewing. Once confirmed, the button *Train Service* starts the distributed training process of a ML model that implements the required service.



SERVICE NAME
Time of departure 5 days
Estimation of the time of departure of a vessel in 5 days time

DATASET
Vessel Calls
Arrival Date Port call
Departure Date Arrival
Time steps to predict

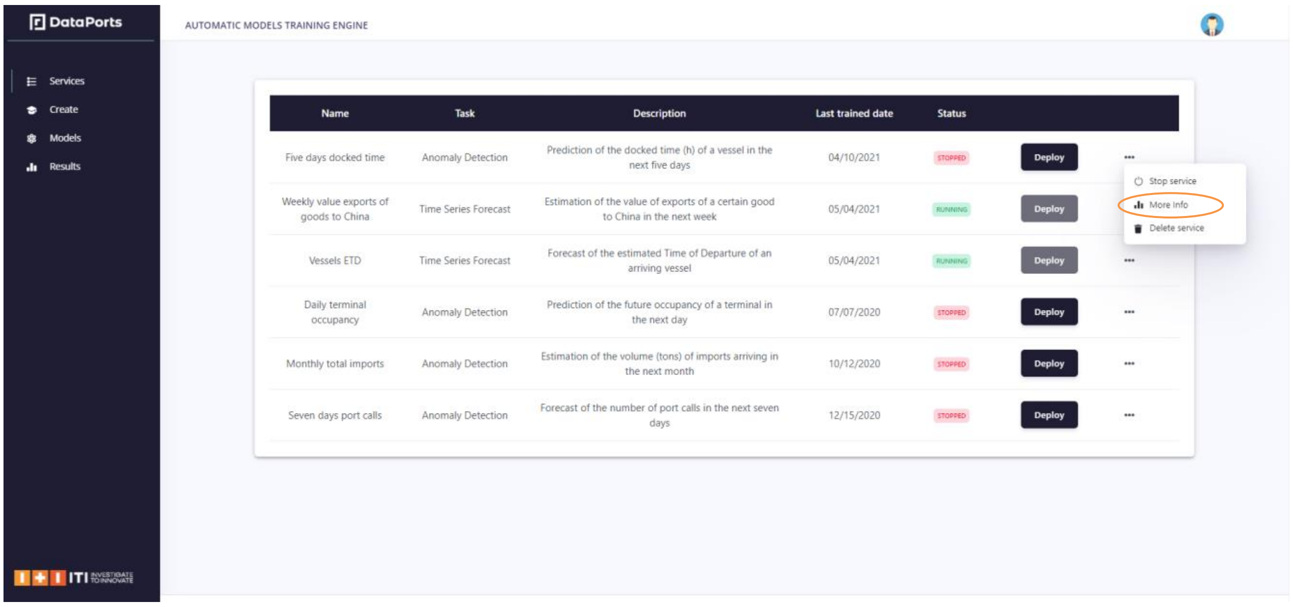
TASK
Vessels ETD forecaster
Forecast of the Estimated Time of Departure (ETD) of an arriving/berth vessel. This service requires a historical dataset with the arrival and departure dates of vessels in an specific port.

STRATEGY
Fast
A quick forecasting process will be carried out to obtain the forecasting outcome as fast as possible. Hence, some non essential processing and optimization steps will be avoided

Figure 18 - Confirmation screen

4.4.2 Management of a cognitive service

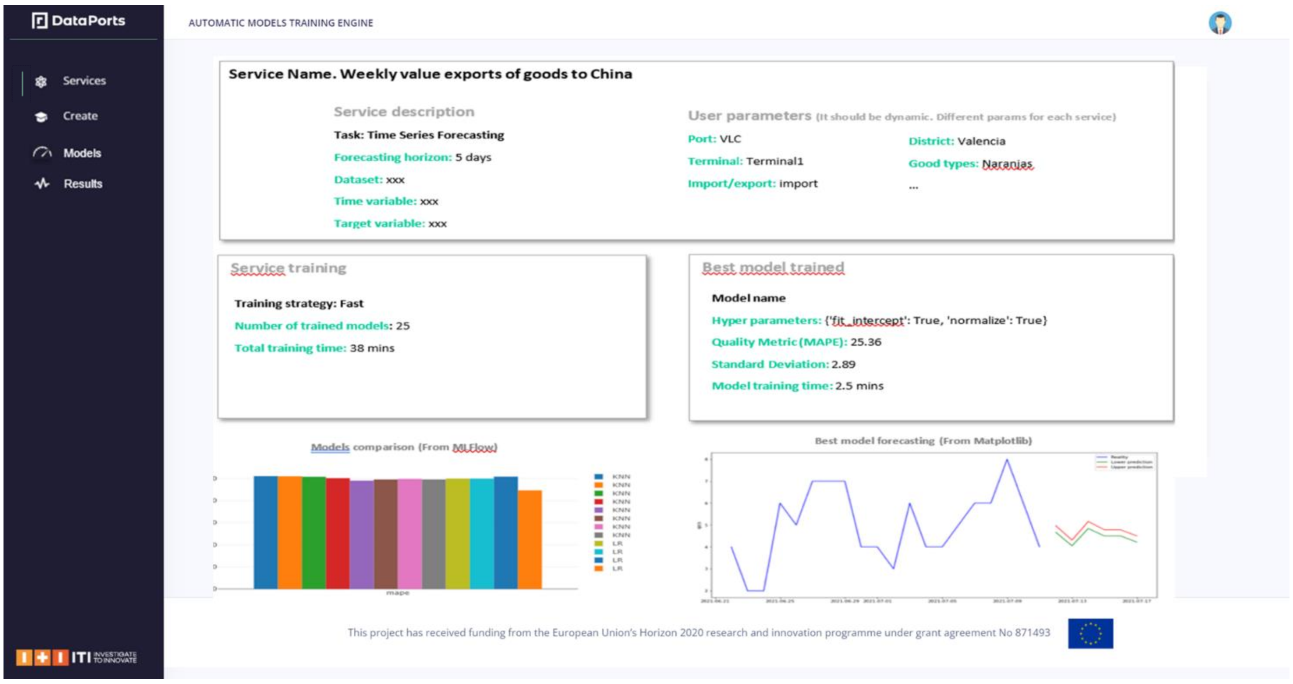
Once the model is trained the service is available on the main interface. Services can be deployed, stopped, or deleted. The interface also provides a link button with "more information" about the service such as the algorithm selected, the training time and the current accuracy metrics



Name	Task	Description	Last trained date	Status		
Five days docked time	Anomaly Detection	Prediction of the docked time (h) of a vessel in the next five days	04/10/2021	STOPPED	Deploy	•••
Weekly value exports of goods to China	Time Series Forecast	Estimation of the value of exports of a certain good to China in the next week	05/04/2021	RUNNING	Deploy	
Vessels ETD	Time Series Forecast	Forecast of the estimated Time of Departure of an arriving vessel	05/04/2021	RUNNING	Deploy	•••
Daily terminal occupancy	Anomaly Detection	Prediction of the future occupancy of a terminal in the next day	07/07/2020	STOPPED	Deploy	•••
Monthly total imports	Anomaly Detection	Estimation of the volume (tons) of imports arriving in the next month	10/12/2020	STOPPED	Deploy	•••
Seven days port calls	Anomaly Detection	Forecast of the number of port calls in the next seven days	12/15/2020	STOPPED	Deploy	•••

Figure 19 - Selecting 'More Information' of a trained cognitive service

Once the button "More Information" is clicked, the following screen appears:


DataPorts
AUTOMATIC MODELS TRAINING ENGINE
Services
Create
Models
Results
Models comparison (From MLFlow)
mape
KNN
LR
Best model forecasting (From Matplotlib)
ITI INVESTIGATE TO INNOVATE

Service Name. Weekly value exports of goods to China

Service description

Task: Time Series Forecasting
Forecasting horizon: 5 days
Dataset: xxx
Time variable: xxx
Target variable: xxx

User parameters (It should be dynamic. Different params for each service)

Port: VLC
Terminal: Terminal1
Import/export: import
District: Valencia
Good types: Naranjas
...

Service training

Training strategy: Fast
Number of trained models: 25
Total training time: 38 mins

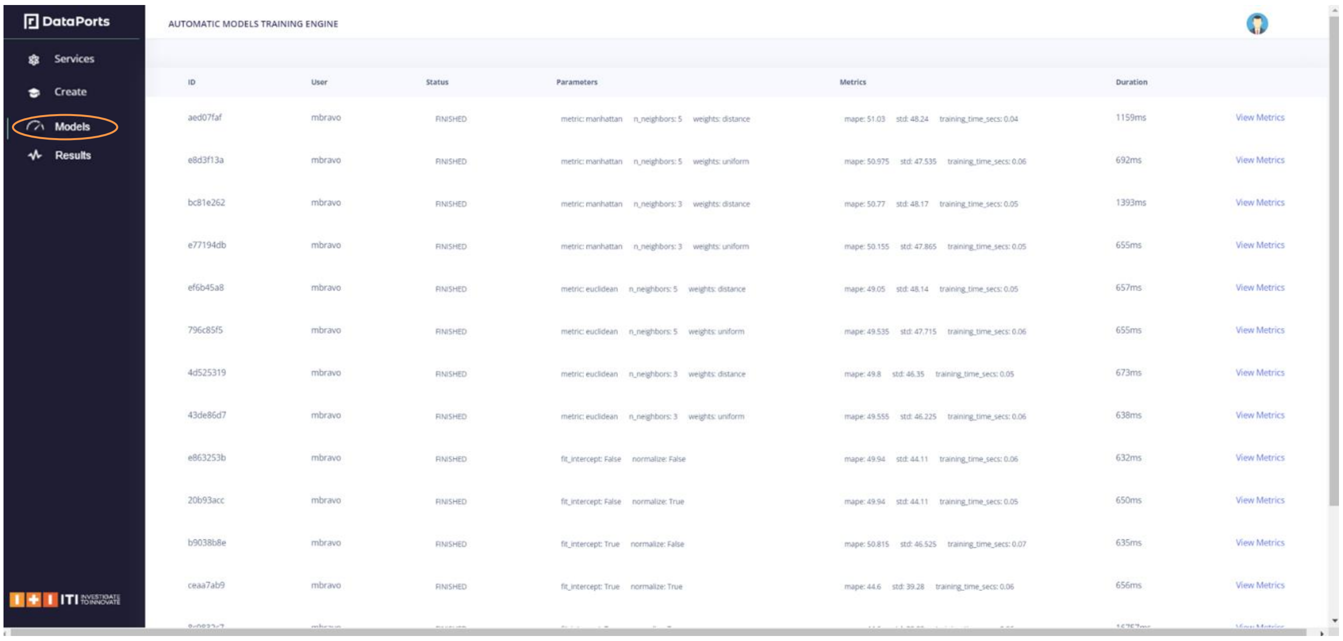
Best model trained

Model name
Hyper parameters: {'fit_intercept': True, 'normalize': True}
Quality Metric (MAPE): 25.36
Standard Deviation: 2.89
Model training time: 2.5 mins

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871493

Figure 20 – Screen 'More Information' of a trained cognitive service

When a service is trained, it is possible to analyse the detailed information of the collection of Machine Learning Algorithms trained, by selecting the tab 'Models' on the left panel.



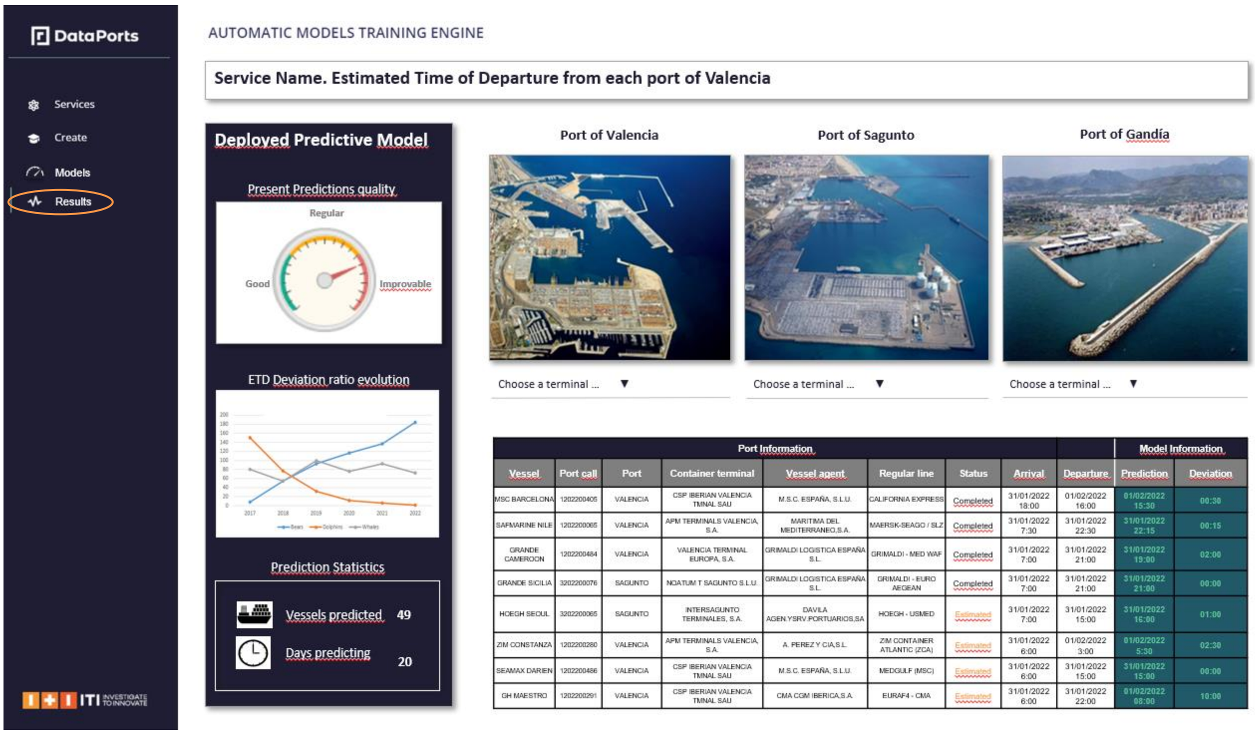
AUTOMATIC MODELS TRAINING ENGINE

ID	User	Status	Parameters	Metrics	Duration	
aed07faf	mbravo	FINISHED	metric: manhattan n_neighbors: 5 weights: distance	mape: 51.03 std: 48.24 training_time_secs: 0.04	1159ms	View Metrics
e8d3f13a	mbravo	FINISHED	metric: manhattan n_neighbors: 5 weights: uniform	mape: 50.975 std: 47.535 training_time_secs: 0.06	692ms	View Metrics
bc81e262	mbravo	FINISHED	metric: manhattan n_neighbors: 3 weights: distance	mape: 50.77 std: 48.17 training_time_secs: 0.05	1393ms	View Metrics
e77194db	mbravo	FINISHED	metric: manhattan n_neighbors: 3 weights: uniform	mape: 50.155 std: 47.865 training_time_secs: 0.05	655ms	View Metrics
ef6b45a8	mbravo	FINISHED	metric: euclidean n_neighbors: 5 weights: distance	mape: 49.05 std: 48.14 training_time_secs: 0.05	657ms	View Metrics
796c85f5	mbravo	FINISHED	metric: euclidean n_neighbors: 5 weights: uniform	mape: 49.535 std: 47.715 training_time_secs: 0.06	655ms	View Metrics
4d525319	mbravo	FINISHED	metric: euclidean n_neighbors: 3 weights: distance	mape: 49.8 std: 46.35 training_time_secs: 0.05	673ms	View Metrics
43de86d7	mbravo	FINISHED	metric: euclidean n_neighbors: 3 weights: uniform	mape: 49.555 std: 46.225 training_time_secs: 0.06	638ms	View Metrics
e863253b	mbravo	FINISHED	fit_intercept: False normalize: False	mape: 49.94 std: 44.11 training_time_secs: 0.06	632ms	View Metrics
20b93acc	mbravo	FINISHED	fit_intercept: False normalize: True	mape: 49.94 std: 44.11 training_time_secs: 0.05	650ms	View Metrics
b9038b8e	mbravo	FINISHED	fit_intercept: True normalize: False	mape: 50.815 std: 46.525 training_time_secs: 0.07	635ms	View Metrics
ceaa7ab9	mbravo	FINISHED	fit_intercept: True normalize: True	mape: 44.6 std: 39.28 training_time_secs: 0.06	656ms	View Metrics

Figure 21 – Models of a trained cognitive service

4.4.3 Analysis of the results of a cognitive service

Finally, the results of the cognitive services may be visualized by using the option 'Results' on the left panel.



AUTOMATIC MODELS TRAINING ENGINE

Service Name. Estimated Time of Departure from each port of Valencia

Port Information									Model Information	
Vessel	Port call	Port	Container terminal	Vessel agent	Regular line	Status	Arrival	Departure	Prediction	Deviation
MSC BARCELONA	1202200405	VALENCIA	CSP IBERIAN VALENCIA TMNAL SAU	M.S.C. ESPAÑA, S.L.U.	CALIFORNIA EXPRESS	Completed	31/01/2022 18:00	01/02/2022 16:00	01/02/2022 15:30	00:30
SAFMARINE NILE	1202200065	VALENCIA	APM TERMINALS VALENCIA, S.A.	MARITIMA DEL MEDITERRANEO,S.A.	MAERSK-SEAGO / SLZ	Completed	31/01/2022 7:30	31/01/2022 22:30	31/01/2022 22:15	00:15
GRANDE CAMEROON	1202200484	VALENCIA	VALENCIA TERMINAL EUROPA, S.A.	GRIMALDI LOGISTICA ESPAÑA S.L.	GRIMALDI - MED WAF	Completed	31/01/2022 7:00	31/01/2022 21:00	31/01/2022 19:00	02:00
GRANDE SICILIA	3202200076	SAGUNTO	NOATUM T SAGUNTO S.L.U.	GRIMALDI LOGISTICA ESPAÑA S.L.	GRIMALDI - EURO AEGEAN	Completed	31/01/2022 7:00	31/01/2022 21:00	31/01/2022 21:00	00:00
HOEGH SEOUL	3202200065	SAGUNTO	INTERSAGUNTO TERMINALES, S.A.	DAVILA AGEN.Y SRV.PORTUARIOS,SA	HOEGH - USMED	Estimated	31/01/2022 7:00	31/01/2022 15:00	31/01/2022 16:00	01:00
ZIM CONSTANZA	1202200280	VALENCIA	APM TERMINALS VALENCIA, S.A.	A. PEREZ Y CIA,S.L.	ZIM CONTAINER ATLANTIC (ZCA)	Estimated	31/01/2022 6:00	01/02/2022 3:00	01/02/2022 5:30	02:30
SEAMAX DARIEN	1202200486	VALENCIA	CSP IBERIAN VALENCIA TMNAL SAU	M.S.C. ESPAÑA, S.L.U.	MEDGULF (MSC)	Estimated	31/01/2022 6:00	31/01/2022 15:00	31/01/2022 15:00	00:00
GH MAESTRO	1202200291	VALENCIA	CSP IBERIAN VALENCIA TMNAL SAU	CMA CGM IBERICA,S.A.	EURAF4 - CMA	Estimated	31/01/2022 6:00	31/01/2022 22:00	01/02/2022 08:00	10:00

Figure 22 – Results screen of a trained cognitive service

4.4.4 Common demonstration

To test all the components and their proper interactions, a common demonstration has been designed. The demonstration is aimed to show the interactions of the core functional components of the DataPorts platform, by testing the technical items to acquire, process and analyse the seaports-related data coming from the existing seaport ecosystem.

A full explanation of the common demonstration may be found in the deliverable "D3.5 Data processing services M27".

4.5 DEVELOPMENT STATUS (M27)

Current source code and Docker deployments of the components are available in following URLs of the DataPorts repository:

- Dashboard: https://gitlab.iti.upv.es/dataports/dev/dataports_project
- Training Pipelines & ML algorithms: https://egitlab.iti.es/dataports/analytics/amte-pipeline

As a summary this section details, for each requirement defined in D2.1 and supported to some extend by this component, the status and pending steps.

ID 3.23	
Description	The components of the DataPorts Platform could be virtualized, in order to ease its deployment and portability
Status	All developed sub-components were packaged as Docker containers and using Docker-compose scripts to simplify the deployment. These containers were tested in an infrastructure based on Openstack
Pending	Apply the same approach to the pending subcomponents and to the required integration software with the pilots

ID 3.26	
Description	As an end-user, I want the platform to provide cognitive services specific to ports requirements, so that I could improve my decision-making processes and/or KPIs
Status	As presented in Section 4.2.4, there was developed a data pipeline that trains ML models with datasets available in the DataPorts platform to predict unknown items with business value. In addition, a web user interface was developed to support the end-user in this process, as it is depicted in Section 4.2.2
Pending	Add usability improvements. Besides, feedback from pilots will be considered

ID 3.27	
Description	As a developer, I want an abstraction mechanism regarding the implementation and set-up details of the data sources connection, so that the deployment will be faster and easier
Status	Data sources metadata will be available from the Data Governance component
Pending	The integration process between AMTE and the Data Governance component has already started and is under discussion

ID 3.28	
Description	As an end-user, I want software components based on State-of-the-Art Machine Learning (ML) algorithms, specifically customized to the port's domain, so that I could easily and automatically create models
Status	The implemented data pipeline utilizes a wide collection of state-of-the-art ML

	algorithms focused on the identified business needs so far: Time series forecasting and Values Imputation. These algorithms were benchmarked with real port's data.
Pending	Test them using a distributed training approach in a cloud infrastructure, currently only local tests were performed.

ID 3.29	
Description	As an end-user, I want a distributed AI platform, so that huge data volumes and time-consuming tasks could be achieved
Status	A distributed approach for training several models simultaneously based on Dask, was implemented, as it is described in Section 4.2.1. This approach also is reliable enough to deal with huge data volumes.
Pending	It is expected to migrate the component to RADIATUS (ITI's Big Data Analytics platform), so the data pipeline may be distributed automatically in a cloud environment.

ID 3.30	
Description	As an end-user, I want to use continuous data streams, so that the platform provides predictions in near real-time
Status	The integration with the Orion Context Broker from the Semantic Interoperability API will support this requirement, as a subscription to this broker is considered a data stream.
Pending	The integration process between AMTE and the Semantic Interoperability component has already started and is under discussion.

ID 3.33	
Description	As a data consumer, I want to get the list of the available data sources and all the methods provided by the platform to subscribe or request data on demand
Status	Data sources information will be available from the Data Governance developed by CERTH. A mock-up API has been defined to test the integration with the developed dashboard.
Pending	The integration process between AMTE and the Data Governance component has already started and is under discussion

ID 3.34	
Description	As a data consumer, I want to subscribe to an available subscription provided by the DataPorts Platform
Status	The integration with the Orion Context Broker from the Semantic Interoperability API will support this requirement, as a subscription to this broker is considered a data stream
Pending	The integration process between AMTE and the Semantic Interoperability component has already started and is under discussion

ID 3.35	
Description	As a data consumer, I want to be able to cancel a current subscription, so that I stop

	receiving data modifications
Status	Same reasoning that the previous requirement.
Pending	The integration process between AMTE and the Semantic Interoperability component has already started and is under discussion.

ID 3.45	
Description	The DataPorts Platform must provide and API for developers in order to build specific applications or services
Status	A first iteration with Django REST Framework was designed with a set of general methods for: retrieving a dataset, start the training of a specific service and start/stop the deployment of an already trained ML models. In addition to this, the API has methods to get predictions taking as input the current data from external components or apps.
Pending	Add additional services to be developed in the context of the pilot into the API

5 PROCESS-BASED ANALYTICS COMPONENT

5.1 OVERVIEW

The aim of the Process-based Analytics component's is to optimise seaport's business processes by leveraging state-of-the-art machine learning techniques. Examples for a business processes in the context of DataPorts are the flow of vessels within the port's service area, or the transport operations of containers or goods. Based on continuously predicting the future states of ongoing processes (facilitated by prediction models which may be trained via the Automatic Model Training Engine), the component provides forward-looking decision support for process managers to take proactive actions.

Failing business processes, like delayed delivery of cargo containers, can be costly to the entity conducting the business. To prevent business processes from failing and the costs associated with said failure, the Process-based Analytics component uses predictions of how an ongoing business process will unfold and enhances these predictions with alarms that prescribe when a process manager should proactively intervene, as well as with explanations on why a certain prediction was made.

The Process-based Analytics component is part of the Advanced Big Data Analytics layer of the DataPorts platform. It analyses business processes by using both historic and real-time data available inside the DataPorts platform. It consists of the following two main subcomponents that were developed in the DataPorts project:

- **Prescriptive Process Monitoring:** Building on process predictions provided by prediction models, this sub-component employs Online Reinforcement Learning to automate the process on when to adapt a running process. This subcomponent applies state-of-the-art Reinforcement Learning algorithms to the problem of identifying the signs of possible failure early and accurately. Details about the underlying concepts and technologies can be found in [25, 26].
- **Explainable Predictive Process Monitoring:** This subcomponent provides interpretations on why a certain prediction is made by a black-box prediction model. This subcomponent uses a model-agnostic technique, which generates a simple model that mimics the behaviour of the black-box model and uses the simple model to explain the predictions. The explanations are given in the form of counterfactuals. Details about the underlying concepts and technologies can be found in [27].

5.2 TECHNOLOGICAL DESCRIPTION

This subsection describes the architecture and technologies behind the Process-Based Analytics component. Each of the two main subcomponents of the Process-based analytics component, is described in detail by their own respective sections further below. In addition, this section also describes the API of the Process-based analytics component and the Event Logs Repository. An overview of the architecture can be seen in Figure 23.

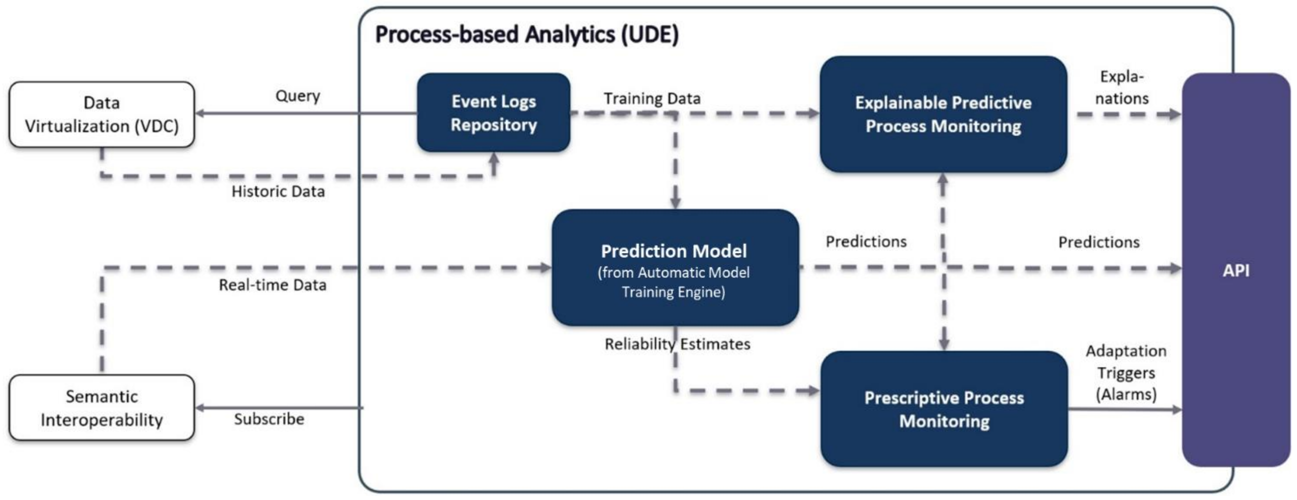

Process-based Analytics (UDE)
Data Virtualization (VDC)
Query
Historic Data
Event Logs Repository
Training Data
Explainable Predictive Process Monitoring
Expla-nations
Prediction Model (from Automatic Model Training Engine)
Predictions
Predictions
Real-time Data
Reliability Estimates
Semantic Interoperability
Subscribe
Prescriptive Process Monitoring
Adaptation Triggers (Alarms)
API

Figure 23 - Process-Based Analytics component

The Event Logs Repository queries the Data Abstraction and Virtualization component for historical data. The Event Logs Repository saves the received data in a NoSql database. The reason for choosing a NoSql database is because they are specifically optimized for big data applications. Moreover, they provide low latency, and flexible data models.

The event log repository pre-processes the raw data and saves them, i.e., both raw and pre-processed data. An interface is created to provide a convenient base class for using the prediction model with different datasets. An important functionality of this interface is to extract the metadata of the dataset being used and pre-process it. This interface also ensures that all traces are always in the correct format and dimensionality needed to train the Ensemble model, despite the absence of a static type of system (as Python is used).

All subcomponents of the Process-based Analytics component have been build using the Tensorflow library. Tensorflow is an open-source machine learning and artificial intelligence library with a focus on deep neural networks. It completely integrates with the popular data library Numpy and offers a high-level interface in the form of Keras to simplify the use of common neural network building blocks and layers. As such it is a natural fit to be used for the implementation of the component's functionalities described in the later sections.

The Process-based Analytics component's API, built to communicate with the cognitive services platform is implemented using Django and the Django REST framework. Django is a Python-based open-source web framework for developing web server applications. The Django REST framework allows building Web APIs for Django based web servers. Both Django and Django REST offer various features that allow the rapid development of clean code by emphasizing reusability through low coupling and architectural choices. Due to this, and Django's ability to scale, it is a perfect fit to be used in the development of the component.

5.2.1 Prescriptive Process Monitoring

This section describes the Prescriptive Process Monitoring subcomponent, whose goal is to provide accurate prediction to business process operators. Besides accuracy, this subcomponent aims at providing failure predictions as early as possible. To do so it uses the output given by the Ensemble Predictive Process Monitoring subcomponent, namely the prediction and its corresponding reliability estimate. Earlier predictions are generally less reliably, thus on average being given a lower reliability estimate. Still, with more time to react, earlier predictions are also inherently favourable to later ones. A process manager could define a reliability threshold, only reacting to failure predictions with reliabilities greater than the threshold.

By setting different thresholds for the reliability computed by the Ensemble Predictive Process Monitoring subcomponent, a process manager could trade the earliness of adaptation actions against their accuracy. Yet, how to set a concrete threshold that is optimal in the given situation remains open. Another approach, which eliminates the need to manually set a threshold, is to empirically determine a threshold. This so-called empirical thresholding is performed via a dedicated training process involving a separate training dataset and knowledge about the concrete cost structure of process execution. This ensures that the threshold is optimal for the training data used and the given cost structure. However, the threshold may not remain optimal over time due to non-stationarity of process environments, data, and cost structures.

The Prescriptive Process Monitoring subcomponent uses an alternative approach that does not require manually determining a threshold nor a-priori information to empirically determine such threshold. In fact, it does not aim to determine an optimal threshold at all. Instead, it uses online reinforcement learning (RL) to learn at run time when to trigger an adaptation based on the predictions and their reliability estimates.

Figure 24 shows a schematic of the internal structure of the subcomponent. For each point of process monitoring data s, an action a is selected by the subcomponent's internal policy, which constitutes the adaptation triggers. Using a combination of the previous process monitoring data s, the next process monitoring data s' and the selected action a, a reward function computes a reward r. The policy is then updated to maximise the received rewards. As such, the RL-agent learns the effectiveness of an agent's actions through the agent's interactions with its environment.

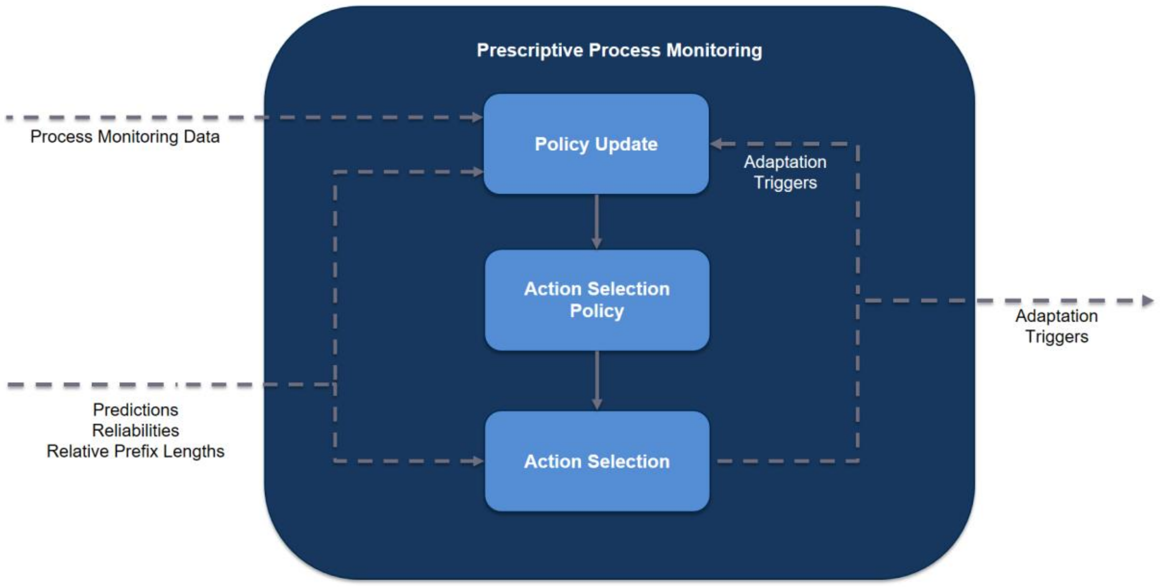


Figure 24 - Prescriptive Process Monitoring subcomponent

To formalise the learning problem described above, the actions a, states s and rewards r are in the following defined. We define an action a as either triggering (a = true) or not not triggering (a = false) a proactive process adaptation. We build the state s from the output of the predictive monitoring system, which includes the predicted deviation δ_j, the reliability estimates ρ_j as well as information about the current prediction point j given as the relative prefix length τ_j, i.e., each state s is represented by $s = (\delta_j, \rho_j, \tau_j)$. In addition to the relative deviation δ_j and the prediction reliability ρ_j, we also use the relative prediction point τ_j of the current case as input. Using τ_j provides an important signal to the RL algorithm about the earliness of the prediction. This relative prediction point τ_j can be computed by dividing the prediction point j by the case length.

Finally, the most important part of formalising the learning problem is to define suitable rewards r. By giving a reward function, one expresses the learning goal in a declarative fashion. As mentioned above, the aim of the learning process is to maximise cumulative rewards. Therefore, finding a suitable reward function is key to successful learning. We thus formulate strong rewards for each of the prediction contingencies as shown in Table 1.

	Predicted Violation	Predicted Non-violation
Actual Violation	$+1 * (1 - \tau_j)$ *(necessary adaptation)*	-1 *(missed adaptation)*
Actual Non-violation	$-.5 - .5 * (1 - \tau_j)$ *(unnecessary adaptation)*	$+1$ *(no adaptation)*

Table 6 – Reward function definition of the RL-agent

We break down the RL problem into suitable episodes, each episode matching the execution of a single case. For each prediction point (process activity), our approach decides whether to adapt or not. Whenever the approach decides to adapt or when the end of the case is reached, we provide a reward r as described above; otherwise, we provide a reward of zero. In order not to discount the reward received at the end of the case, we consequently set the discount factor of the RL algorithm to $\gamma = 1$. The discount factor is a standard hyper-parameter in RL and defines the relevance of future rewards.

As a concrete RL algorithm, we use proximal policy optimisation (PPO). PPO is a policy-based RL algorithm that uses a neural network to directly represent a policy function. By updating the weights of the neural network, the algorithm tunes this policy function to map actions to environment states in such a way that it maximises the expected long-term rewards. These updates are governed by a clipped loss function that can be optimized using gradient decent methods, yet still prevents overly large and thus destructive policy changes during a single optimisation step. PPO is also rather robust for what concerns hyper-parameter settings. Thereby, we avoid the need for extensive hyper-parameter tuning compared to other policy-based RL algorithms. As neural network architecture we use a multi-layer perceptron (MLP) architecture with two hidden layers of 64 neurons each. The input layer consists of three neurons representing the three state variables; the output layer consists of one neuron representing the action variable.

5.2.2 Explainable Predictive Process Monitoring

RNN ensembles are complex models that cannot be interpreted by humans, i.e., the end-user cannot understand why the model made a specific prediction. In the context of explainable AI, such complex models are called black-box models. Using black-box models without being able to interpret their decisions has potential risks. Such risks could hinder the acceptance and trust of users in adopting the predictive assistant system. Therefore, in addition to predictions, the Process-Based Analytics component provides the users with explanations in the explainable predictive process monitoring component. The explanations can be used by operators to understand why the model has made a specific prediction. This can help the operator decide what action should be taken and help in determining the likelihood of the model's prediction being correct.

The method used to explain model predictions, is model-agnostic. This means this method can be used with any black-box model. The method generates an interpretable model for each prediction of the black-box model, with training data sampled close to the local decision boundary of the black-box. This allows the interpretable model to "mimic" the black-box model in the local area around the prediction. Then, it extracts counterfactual explanations from the interpretable model.

To show the process of explanation generation, a graphical overview of the artefacts and activities of explainable predictive process monitoring component is shown in Figure 25. The Upper part of the figure depicts how individual predictions are generated using the black-box model. The lower part of the figure depicts how explanations are generated. This process has three main stages: the approach starts with stage 0 by finding similar prefixes for initialisation in the training dataset. By only using already existing prefixes, an initial population consisting of realistic prefixes is ensured. Furthermore, the control flow attributes of the first population will be used by the mutation step later as well. In stage 1, the approach uses a genetic algorithm to generate neighbourhood instances. The genetic algorithm is used to ensure we can have compact synthetic instances that are close to the local decision boundary of the black box. This is done by iteratively optimising a fitness function, which tries to find the instances that are as close as possible to the instance to be explained but not exactly the instance to be explained. The method generates a fixed number of prefixes for each possible black-box prediction class. The set of all selected prefixes is called the neighbourhood, as it represents the neighbourhood of the instance to be explained near the black boxes' decision boundary. During the mutation step, the control flow attributes can only be mutated by sampling from the control flow attributes from the first population. This mechanism ensures that all the control flow attributes in the synthetic instances are realistic by construct. Finally, in step 2, a decision tree for the extraction of explanations is trained using the synthetic prefixes as a training dataset. The explanations are presented to the user in the form of counterfactual explanations.

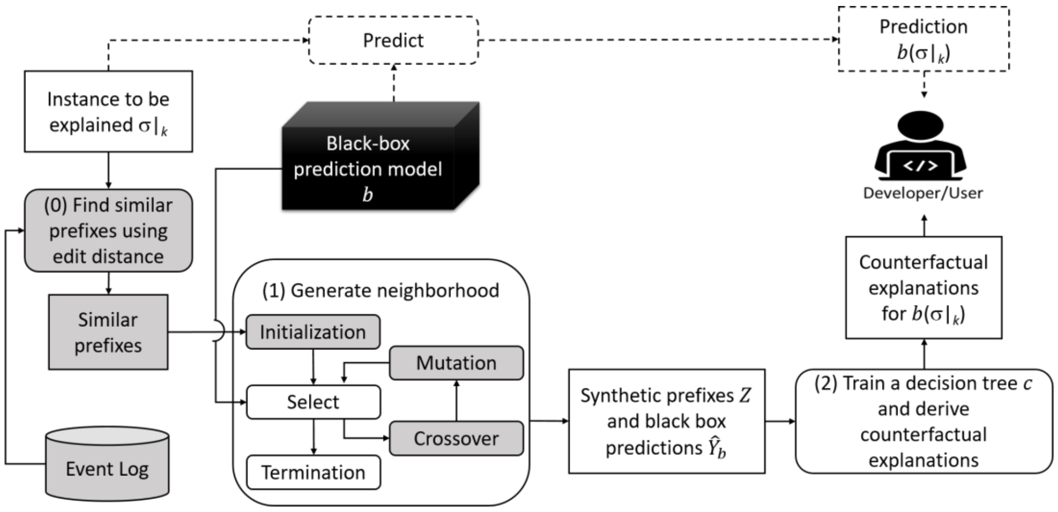

Predict
Prediction $b(\sigma|_k)$
Instance to be explained $\sigma|_k$
Black-box prediction model b
Developer/User
(0) Find similar prefixes using edit distance
Similar prefixes
(1) Generate neighborhood
Initialization
Mutation
Select
Crossover
Termination
Event Log
Synthetic prefixes Z and black box predictions $\hat{Y}_b$
(2) Train a decision tree c and derive counterfactual explanations
Counterfactual explanations for $b(\sigma|_k)$

Figure 25 - Overview of Explainable Predictive Process Monitoring Component

5.3 FUTURE WORK: ROADMAP

The following section describes our ambitions for the Process-based Analytics component after month 30 of the project, when the component will be fully integrated with and deployed alongside the rest of the DataPorts platform.

- **Experimental Usage:** We plan on using the implementation of the Process-based Analytics component as well as the use-case data in scientific experiments to further our research in Predictive Business Process Monitoring and Explainability (XAI). Furthermore, we plan to use our component with other use-cases that will be incorporated in the DataPorts platform in the future and extend the functionalities of our component to meet the expectations and needs of the end-users.
- **Scientific Publications:** We plan on disseminating in additional scientific publications the insights gained by using the Process-based Analytics component with DataPorts use-cases and further experiments
- **Research Prototype:** As part of the scientific publications, we plan on releasing the implementation of the Process-based Analytics component's subcomponents in the form of research prototypes to open-source repositories

5.4 EXAMPLE OF USE: DEMONSTRATION

5.4.1 Prescriptive Process Monitoring

Proactive process adaptation entails asymmetric real-world costs. On the one hand, one may face penalties in case of violations, e.g., due to contractual arrangements (e.g., SLAs) or due to loss of customers. On the other hand, adapting the running business processes may incur other costs, e.g., due to executing roll-back actions or due to scheduling alternative process activities.

To understand the way our approach to proactive process adaptation impacts theses real world costs, we use a cost model that assigns execution costs to every business case. Figure 26 shows this cost model, which incorporates the two cost drivers. In this model, costs depend on (1) the actual process performance if no adaptation was taken, (2) whether the prediction was accurate, and (3) whether a business process adaptation was effective, i.e., whether the adaptation indeed resulted in a non-violation (also see [28]).

Research prototypes for our component have been developed. We evaluated these research prototypes using the cost model. Besides the cost model, this evaluation has also been governed by several variables, the *Reliability threshold* θ, the *Relative adaptation costs* λ and the *Adaptation effectiveness* α. The next paragraphs concern themselves with describing these variables.

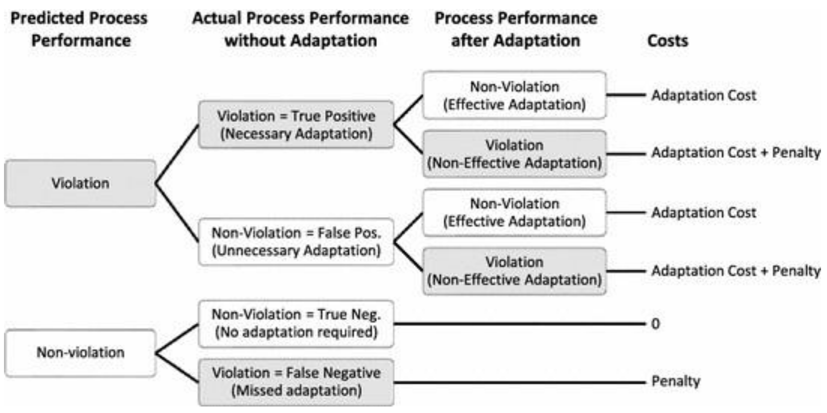

Predicted Process Performance
Actual Process Performance without Adaptation
Process Performance after Adaptation
Costs
Violation
Non-violation
Violation = True Positive (Necessary Adaptation)
Non-Violation = False Pos. (Unnecessary Adaptation)
Non-Violation = True Neg. (No adaptation required)
Violation = False Negative (Missed adaptation)
Non-Violation (Effective Adaptation)
Violation (Non-Effective Adaptation)
Non-Violation (Effective Adaptation)
Violation (Non-Effective Adaptation)
Adaptation Cost
Adaptation Cost + Penalty
Adaptation Cost
Adaptation Cost + Penalty
0
Penalty

Figure 26 - Asymmetric costs of proactive business process adaptation

Reliability threshold $\theta \in [.5, 1]$ As introduced earlier, we compare the performance of our Prescriptive Process Monitoring component to the process of empirical thresholding. When using empirical thresholding, a proactive adaptation is only triggered if the reliability of a predicted violation is equal to or greater than the pre-defined reliability threshold. Alternatively, our Prescriptive Process Monitoring component does away with this threshold and uses a RL-agent to decide based on the reliability when to trigger an adaptation.

Relative adaptation costs $\lambda \in [0, 1]$ To be able to concisely analyse and present our evaluation results, we assume constant costs and penalties, as they are described in the beginning of this section. Thus, the costs of a process adaptation, ca, are expressed as a fraction of the penalty for process violation, cp, i.e., $ca = \lambda * cp$. We thereby can reflect different situations that may be faced in practice concerning how costly a process adaptation in relation to a penalty may be. Choosing $\lambda > 1$ would not make sense, as this leads to higher costs than if no adaptation is performed.

Adaptation effectiveness $\alpha \in (0, 1]$ If an adaptation results in a non-violation, we consider such an adaptation effective. We use α to represent the fact that not all adaptations might be effective. More concretely, α represents the probability that an adaptation is effective. We do not consider $\alpha = 0$ as this means that no adaptation is effective. To reflect the fact that earlier prediction points may be favoured as they provide more options and time for proactive adaptations, we vary α in our evaluation in such a way that α linearly decreases over the course of process execution. This means that the probability for effective proactive adaptations diminishes towards the end of the process. To model this, we define α_{max} as the α for the first prediction point in the process instance, and α_{min} as the α for the last prediction point.

We use four data sets from different sources for the evaluation described above. Table 2 provides key characteristics of these data sets.

Name	Positive class	Positive class ratio	Process instances	Process variants
Traffic	Unpaid traffic fine	46%	129,615	185
BPIC 2012	Unsuccessful credit application	52%	13,087	3,587
BPIC 2017	Unsuccessful credit application	59%	31,413	2,087

Table 7 - Data sets used in evaluation

Using this cost model evaluations have been made for the Prescriptive Process Monitoring component. After training the Ensemble Predictive Process Monitoring component on roughly 2/3 of each dataset, it is used on the remainder of the datasets to generate the input for the RL-agent. Figure 27 shows the respective results for the three data sets.

The charts show how the rate of adaptations, earliness, the rate of correct adaptation decisions, and overall rewards evolve (costs are discussed further below). We measure earliness in terms of relative prefix-length when an adaptation was made, i.e., 0 means an adaptation was made at the beginning of the process, while 1 means it was made at the end. Charts start at case # 100, because the points in the charts are averaged over the last 100 cases for stability reasons (also see [28].

Part (a) of the charts shows the learning process until convergence can be observed, while part (b) shows the learning process for the whole test data set. We consider convergence to happen as soon as cumulative rewards averaged over the last 100 cases reaches the cumulative rewards averaged across the whole data set, which is indicated as the dashed red line.

Across all four data sets, the convergence of the learning process is evident when observing the development of the reward curve. Convergence happens after around 500 cases for BPIC 2012, 1200 for BPIC 2017 and Traffic. It can also be seen that the approach indeed is able to learn when to adapt in order to maximise rewards. For all three data sets, the approach starts with a very high rate of adaptations that are triggered very early in the process. However, this has negative impact on rewards, as the approach has not yet learned that (1) adaptations should only be triggered for positive predictions, (2) not all predictions may be accurate, (3) there is a trade-off between accuracy and earliness. This is also evident in the rate of correct adaptation decisions (which is very low before convergence). After the point of convergence is reached, it can be observed that the approach has learned to be more conservative with triggering adaptations (the rate of adaptations goes down), and that later predictions may be more accurate (earliness goes up). This results in a higher rate of correct adaptation decisions and thus a higher reward.

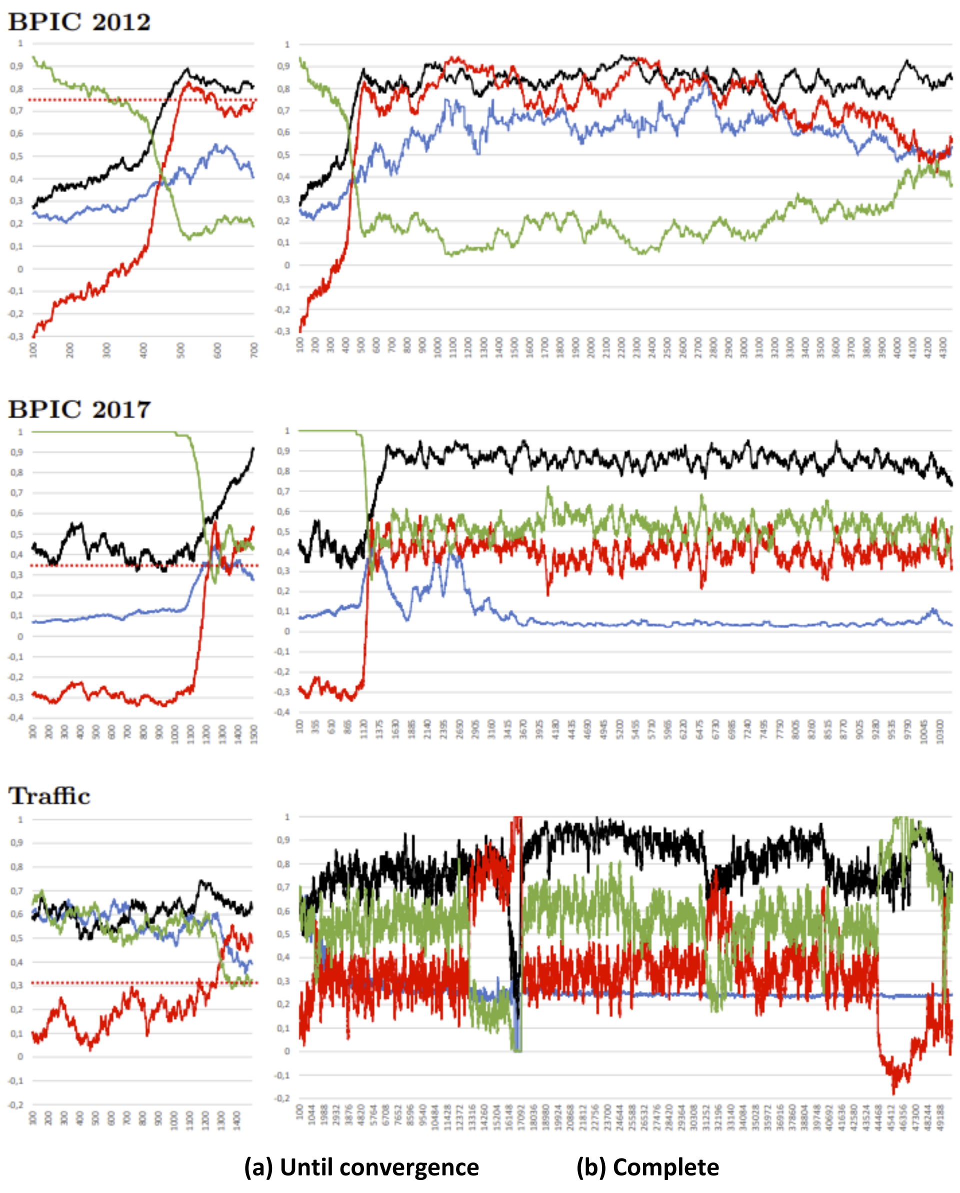


Figure 27 - Learning behaviour; green: rate of adaptations; blue: earliness (0 = beginning, 1 = end of process); black: rate of correct adaptation decisions; red: overall reward/100

Having observed convergence of learning, the comparison of process execution costs of the RL-agent with the process execution costs when using empirical thresholding shows promising results for all data sets. Empirical thresholding being an even stronger baseline than the dynamic approach mentioned earlier, as the threshold is not arbitrarily chosen, but an optimal threshold is computed for a subset of the data set and then applied to the rest.

Results indicate that the proactive process adaptations triggered by our approach result on average, when varying different settings of λ, in 6.1% ($\alpha_{min} = .5$) resp. 6.4% ($\alpha_{min} = 0$) less process execution costs when compared to empirical thresholding. Only the first excerpt of the Traffic data set was chosen for this evaluation, as using proactive process adaptations triggers on the full Traffic results in considerably above average savings.

As mentioned above, one of the main advantages of the RL approach is to capture non-stationarity in the data. The Traffic data set shows such no stationarity between around case # 14,000 and # 16,000, and again after around case # 45,000. Deeper analysis shows that this is because the average prediction accuracy for cases # 14,000 to # 16,000 is 65% higher than the average accuracy for the whole data set, while for all cases after case # 45,000 the average accuracy is 51% lower than the average accuracy for the whole data set.

In the presence of non-stationary, the RL approach shows high improvements over empirical thresholding, leading to 20.3% lower costs on average for the whole Traffic data set for both settings of α_{min}. Overall, this leads to average savings of 8% resp. 12.2% when we compare the RL approach for all three complete data sets against empirical thresholding.

5.4.2 Explainable Predictive Process Monitoring

To give an impression for the nature and quality of explanations we first examine whether the most important attributes indicated by the explanations correspond to the domain knowledge (also see [30]). We use one of the data sets from above: BPIC2017, which was used in the BPI contest 2017.

It is expected that the logic learned by the black box prediction model would reflect a certain degree of domain knowledge as accuracy increases along the prefix length. Since involving the process owner of the BPIC2017 event log was not possible, we compare our results with the findings of the winners of the BPIC2017 challenge, for which this dataset was published. Figure 28 shows the top five most frequent attributes for each input length examined.

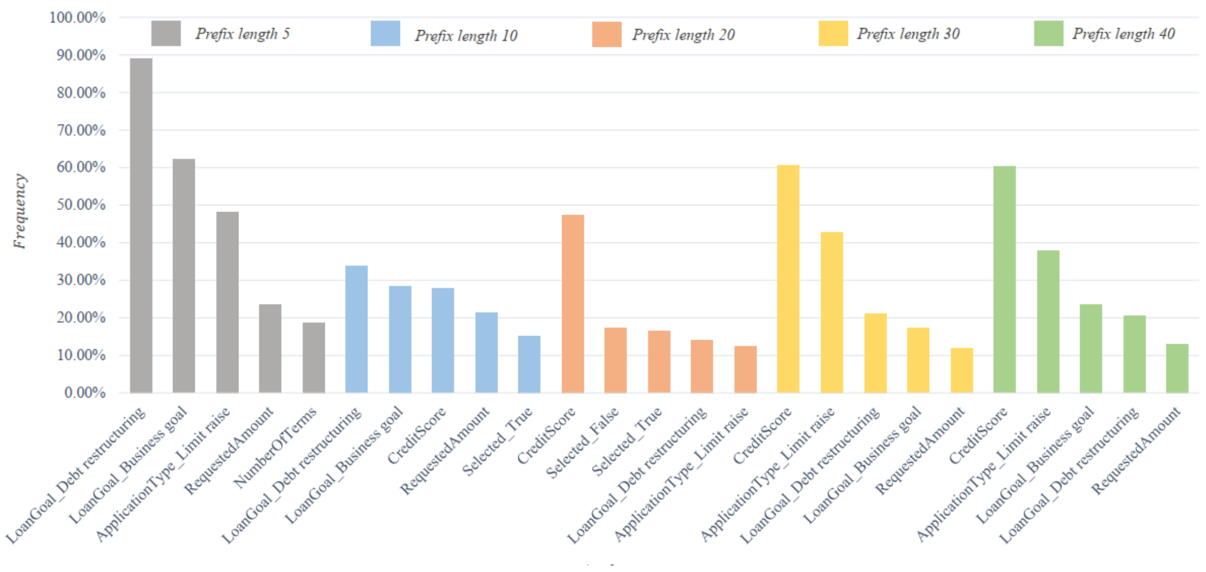


Figure 28 – Top five important attributes depending on prefix-length

The result shown in Figure 28 thus gives a broader view regarding what the black box prediction model has learned for different input lengths. As the figure shows, the most important attribute for prefix lengths 20, 30, and 40 is CreditScore. This corresponds to the findings of the BPI contest professional group winners, where they found CreditScore is among the most important attributes because of the predictive analysis. The other frequent attribute is ApplicationType Limit raise and it is the second most frequent important attribute for prefix length 30 and 40. This corresponds to the findings of the BPI contest academic group winners, who found that the institute approved a significantly higher fraction of applications of the type "limit raise" (73.37%) than "new credit" (52.61%). It is therefore another reasonable predictor used by the black box.

Figure 28 also shows that the LoanGoal attributes are used by the black-box model quite often to predict the outcome (especially for prefix lengths 5 and 10) although LoanGoal has little influence on the outcome of the case according to the BPI contest analysis. However, these attributes could be the most frequent decisive attributes available for the black box considering prefixes 5 and 10 are still quite early in the process.

Complementing this high-level assessment of explanation quality, we give a concrete example for explanations generated. To this end, we use a particular input data of prefix length 30.

The instance to be explained has the following event attributes for O_Create_Offer: FirstWithdrawalAmount = 0, NumberOfTerms = 126, MonthlyCost = 250, CreditScore = 0, OfferedAmount = 25000, Selected = true, ac- cepted = false, and it has the following case attributes case RequestedAmount = 25000, case LoanGoal = Existing_loan_take_over, case ApplicationType = new_credit. The corresponding black-box prediction is A_Denied. By observing the control flow of the prefix, one can find that the prefix is at the later stage of the application process. The application process has gone through application creation and assessment. An offer is created and the customer has been contacted a few times to send incomplete files.

The explanation for the instance to be explained is:

{CreditScore ≤ 323, Case_LoanGoal = Existing loan takeover, Case_ApplicationType = New credit } → A_Denied

The following are examples of two counterfactual rules (with changes from the factual rule highlighted in grey):

{CreditScore > 323} → A Pending

{CreditScore ≤ 323, Case_LoanGoal = Existing loan takeover, Case_ApplicationType != New credit } → A_Pending

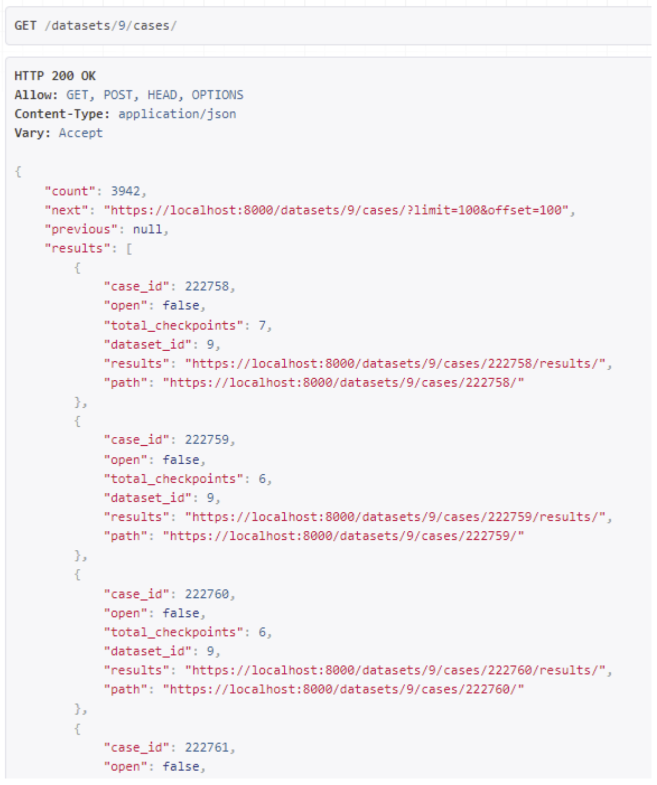
These two counterfactual rules provide the least changes one has to make to obtain the desired outcome. Again, the derived explanations match the findings of the BPI contest winners: Applications of type “limit raise” or applicants with higher credit scores indeed have higher acceptance rates (recall that state A Pending means accepted).

5.4.3 Common Demonstration

In the common demonstration the Process-based Analytics component (PBAC) will be using the PCS Traffic dataset. The PCS Traffic dataset contains information about port and logistic data of the vessels arriving and leaving the Valencia Port. Historic data from the Data abstraction & virtualization (DAV) component is used for training the prediction model. The Semantic Interoperability (SIC) component supplies the Process-based Analytics component with real-time data used to make real-time predictions about currently running processes in the port.

Using the real-time PCS Traffic data, the prediction model will make predictions on the outcome of the described port-calls, that is, the PBAC component will inform the port operators about the future of a current process. Specifically, the model will predict if the process will end in the estimated time, or not. Depending on the expected outcome and the prediction, the Explainable Predictive Process Monitoring subcomponent will create an explanation for the current prediction and what could be a possible adjustment for it. The Prescriptive Process Monitoring subcomponent will create an alarm if a predicted deviation from the expected outcome is sufficiently reliable to warrant acting on it.

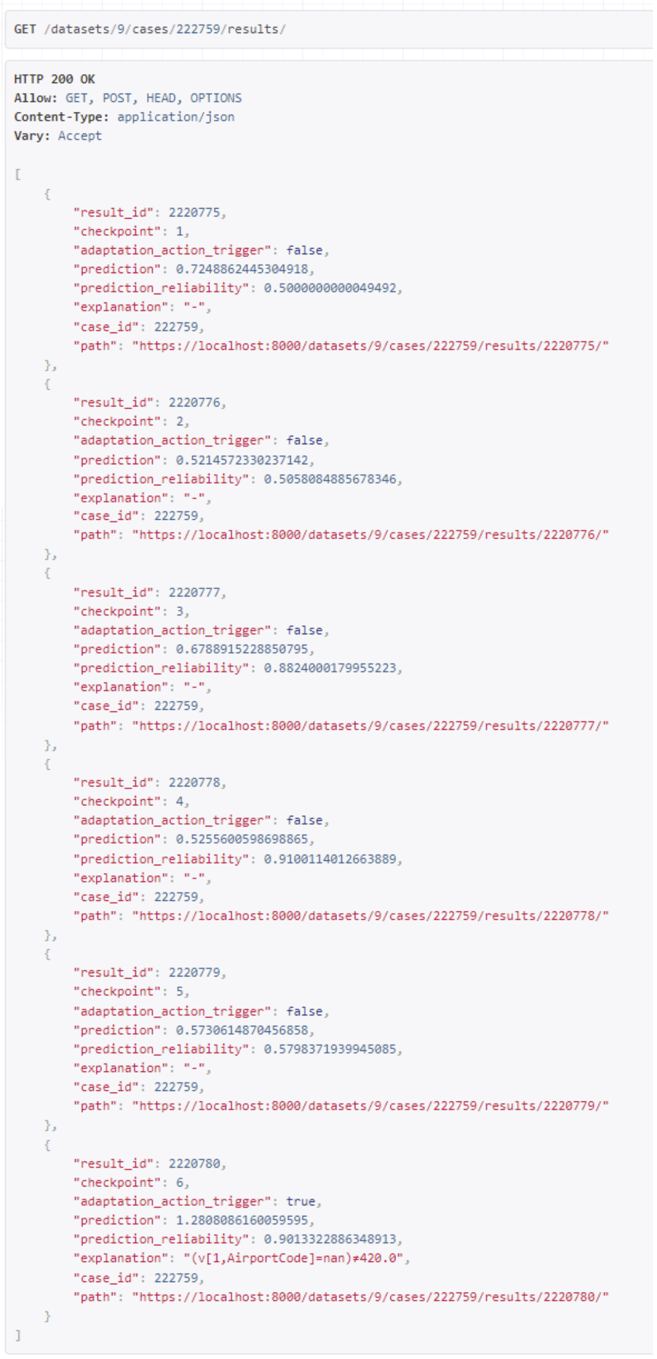
The results of the Process-based Analytics component will be made available by the API subcomponent. The API subcomponent implements a REST API, which allows querying the results of the component by dataset, process-instance (case) and step of the process (checkpoint). The API subcomponent additionally allows for consumers to subscribe to datasets and cases specifically to be notified whenever new results for them become available. Figure 29 shows the results of two exemplary API call, one querying for cases within a dataset and querying for the results of one case specifically.



```
GET /datasets/9/cases/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 3942,
    "next": "https://localhost:8000/datasets/9/cases/?limit=100&offset=100",
    "previous": null,
    "results": [
        {
            "case_id": 222758,
            "open": false,
            "total_checkpoints": 7,
            "dataset_id": 9,
            "results": "https://localhost:8000/datasets/9/cases/222758/results/",
            "path": "https://localhost:8000/datasets/9/cases/222758/"
        },
        {
            "case_id": 222759,
            "open": false,
            "total_checkpoints": 6,
            "dataset_id": 9,
            "results": "https://localhost:8000/datasets/9/cases/222759/results/",
            "path": "https://localhost:8000/datasets/9/cases/222759/"
        },
        {
            "case_id": 222760,
            "open": false,
            "total_checkpoints": 6,
            "dataset_id": 9,
            "results": "https://localhost:8000/datasets/9/cases/222760/results/",
            "path": "https://localhost:8000/datasets/9/cases/222760/"
        },
        {
            "case_id": 222761,
            "open": false,
```



```
GET /datasets/9/cases/222759/results/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "result_id": 2220775,
        "checkpoint": 1,
        "adaptation_action_trigger": false,
        "prediction": 0.7248862445304918,
        "prediction_reliability": 0.5000000000049492,
        "explanation": "-",
        "case_id": 222759,
        "path": "https://localhost:8000/datasets/9/cases/222759/results/2220775/"
    },
    {
        "result_id": 2220776,
        "checkpoint": 2,
        "adaptation_action_trigger": false,
        "prediction": 0.5214572330237142,
        "prediction_reliability": 0.5058084885678346,
        "explanation": "-",
        "case_id": 222759,
        "path": "https://localhost:8000/datasets/9/cases/222759/results/2220776/"
    },
    {
        "result_id": 2220777,
        "checkpoint": 3,
        "adaptation_action_trigger": false,
        "prediction": 0.6788915228850795,
        "prediction_reliability": 0.8824000179955223,
        "explanation": "-",
        "case_id": 222759,
        "path": "https://localhost:8000/datasets/9/cases/222759/results/2220777/"
    },
    {
        "result_id": 2220778,
        "checkpoint": 4,
        "adaptation_action_trigger": false,
        "prediction": 0.5255600598698865,
        "prediction_reliability": 0.9100114012663889,
        "explanation": "-",
        "case_id": 222759,
        "path": "https://localhost:8000/datasets/9/cases/222759/results/2220778/"
    },
    {
        "result_id": 2220779,
        "checkpoint": 5,
        "adaptation_action_trigger": false,
        "prediction": 0.5730614870456858,
        "prediction_reliability": 0.5798371939945085,
        "explanation": "-",
        "case_id": 222759,
        "path": "https://localhost:8000/datasets/9/cases/222759/results/2220779/"
    },
    {
        "result_id": 2220780,
        "checkpoint": 6,
        "adaptation_action_trigger": true,
        "prediction": 1.2808086160059595,
        "prediction_reliability": 0.9013322886348913,
        "explanation": "(v[1,AirportCode]=nan)≠420.0",
        "case_id": 222759,
        "path": "https://localhost:8000/datasets/9/cases/222759/results/2220780/"
    }
]
```

Figure 29 - Exemplary API calls

5.5 DEVELOPMENT STATUS (M27)

The current version of the source code is available in the DataPorts repository: https://egitlab.iti.es/dataports/analytics/process_based_analytics. Next the status of component with respect to D2.1 [20] and pending steps are presented.

ID 3.26	
Description	As an end-user, I want the platform to provide cognitive services specific to ports requirements, so that I could improve my decision-making processes and/or KPIs
Status	First implementation of the Process Based Analytics component is available
Pending	Finalising the API offered to DataPorts' cognitive services based on the integration of the Process Based Analytics component with the other components

ID 3.28	
Description	As an end-user, I want software components based on State-of-the-Art Machine Learning (ML) algorithms, specifically customized to the port's domain, so that I could easily and automatically create models
Status	Research prototype and experimental results are available
Pending	The adaptation of the component's implementation to data from DataPorts' use-cases is in progress. Once the Process Based Analytics component has been integrated with the other components it will provide its analytics results based on the use-case data

ID 3.30	
Description	As an end-user, I want to use continuous data streams, so that the platform provides predictions in near real-time
Status	The Semantic Interoperability component will make real-time data available for the Process Based Analytics component
Pending	The integration process between the Process Based Analytics component and the Semantic Interoperability component is in progress

ID 3.31	
Description	As a data scientist, I want the platform to deal with the original data sources heterogeneity, real-time (streaming) or persistent data, relational or non-relational databases
Status	The Data Virtualisation component will make historic data available to the Process Based Analytics component while the Semantic Interoperability component will make real-time data available. Through the interface to both components the Process Based Analytics component will be able work with any kind of data source
Pending	The integration process between the Process Based Analytics component and the Semantic Interoperability component as well as the Data Virtualisation component has already started and is in progress.

6 RELEASE SUMMARY

In this section the release information summary table for each component of the deliverable at M27 is showed, including component name, version, source code, documentation, Docker image and dependencies.

AUTOMATIC MODEL TRAINING ENGINE

Component	Subcomponent	Release Information	Details and Links
Automatic Model Training Engine	Training Distribution Engine	Version	1.0
		Source Code	https://egitlab.iti.es/dataports/analytics/amte-pipeline
		Binaries (optional)	-
		Documentation	https://platform.dataports-project.eu/docs/autotraining/
		Docker Image (optional)	-
		Dependencies (optional)	Dask, Python, RADIATUS, MLFlow, Docker, Training Web Interface
	Training Web Interface	Version	1.0
		Source Code	https://egitlab.iti.es/dataports/analytics/amte_dashboard
		Binaries	-
		Documentation	https://platform.dataports-project.eu/docs/autotraining/
		Docker Image	-
		Dependencies	Angular, Django REST Framework, Docker, Training Distribution Engine
	APIs	Version	1.0
		Source Code	https://egitlab.iti.es/dataports/analytics/amte_dashboard/-/tree/main/backend
		Binaries	-
		Documentation	https://platform.dataports-project.eu/docs/autotraining/api/
		Docker Image	-
		Dependencies	Django REST Framework, Docker, Training Web Interface, Training Distribution Engine

Table 8 – Summary table of AMTE component release information

PROCESS BASED ANALYTICS

Component	Subcomponent	Release Information	Details and Links
Process-based Analytics component	Prescriptive Process Monitoring	Dependencies	Tensorflow (service deployed with the docker-compose)
		Version	1.0
		Source Code	https://egitlab.iti.es/dataports/analytics/process_based_analytics
		Binaries	-
		Documentation	https://platform.dataports-project.eu/docs/processanalytics/
		Docker Image	-
	Explainable Predictive Process Monitoring	Dependencies	Tensorflow (service deployed with the docker-compose)
		Version	2.0
		Source Code	https://egitlab.iti.es/dataports/analytics/process_based_analytics
		Binaries	-
		Documentation	https://platform.dataports-project.eu/docs/processanalytics/
		Docker Image	-
	API and Event Logs Repository	Dependencies	- (service deployed with the docker-compose)
		Version	1.0
		Source Code	https://egitlab.iti.es/dataports/analytics/process_based_analytics
		Binaries	-
		Documentation	https://platform.dataports-project.eu/docs/processanalytics/
		Docker Image	-

Table 9 – Summary table of PROCESS BASED ANALYTICS component release information

7 CONCLUSIONS

This deliverable has presented the two main components that fulfil the task T3.4 of the project "T3.4 Data analytic and AI services for cognitive applications".

Regarding the Automatic Model Training Engine, the following results are highlighted:

- A dashboard for configuring the training process has been developed to enhance the usability and involvement of end-user for developing cognitive services. The dashboard allows the end-user to create new cognitive services, analyse the collection of machine learning models trained, and visualize and interact with the results obtained from the predictions of a cognitive service.
- A wide collection of ML algorithms has been defined and implemented after performing an extensive analysis of the State of the Art. The algorithms are mainly focused on two different domains, detected after identifying the port's main business needs regarding data analysis: Time Series Forecasting and Values Imputation. Therefore, the component provides regression, classification, imputation and forecasting algorithms.
- A cloud distributed training approach using RADIATUS guarantees that we could scale up the training several models with different approaches.
- All the integrations with other components have been initiated, as described in D3.5

The most relevant results of the Process Based Analytics component are summarised as follows:

- Two prototypical subcomponents are developed, namely the prescriptive and explainable process monitoring subcomponents.
- All two subcomponents were evaluated using real business processes datasets. The underlying concepts and prototypical implementation, together with experimental results were published at three prestigious, international research conferences.
- While the validation is phase is still ongoing, initial results indicate that the developed component has potentials to benefit port stakeholders.

Final evaluation results using the scenarios from the pilots use cases will be reported by the end of the project.

8 REFERENCES AND ACRONYMS

8.1 REFERENCES

[1] "AGA – Annotated Model Grant Agreement," p. 750.

[2] D. Consortium, " D3.3 Data Analytics services and Cognitive Applications M18," 2021.

[3] D. Consortium, "D3.5 Data processing services M27," 2021.

[4] D. Consortium, "D3.7 Permissioned Blockchain network M27," section 7 Data Processing Services common example of use.

[5] D. C. Montgomery, E. A. Peck and G. G. Vining, Introduction to Linear Regression Analysis, John Wiley & Sons, 2021.

[6] A. C. Müller and S. Guido, Introduction to machine learning with Python: a guide for data scientists, " O'Reilly Media, Inc.", 2016.

[7] H. Kvamme, Ø. Borgan and I. Scheel, "Time-to-event prediction with neural networks and Cox regression," *arXiv preprint arXiv:1907.00825,* 2019.

[8] J. Brownlee, XGBoost With Python: Gradient Boosted Trees with XGBoost and scikit-learn, Machine Learning Mastery, 2016.

[9] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[10] S. J. Taylor and B. Letham, "Forecasting at Scale," *The American Statistician,* vol. 72, pp. 37-45, 2018.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, p. 8024–8035.

[12] F. A. N. &. P. D. Thabtah, "A machine learning autism classification based on logistic regression analysis," *Health Inf Sci Syst 7,* 2019.

[13] S. D. L. X. C. B. K. R. C. a. D. C. J. Ren, "Naive Bayes Classification of Uncertain Data," *2009 Ninth IEEE International Conference on Data Mining,* 2009.

[14] P. Tu and J. Chung, "A new decision-tree classification algorithm for machine learning," *Proceedings Fourth International Conference on Tools with Artificial Intelligence, Arlington, VA, USA,* 1992.

[15] V. V. Alexander Vezhnevets, "Modest AdaBoost' – Teaching AdaBoost to Generalize Better," Moscow State University , 2005.

[16] P. S. Pritika BahadEmail, Study of AdaBoost and Gradient Boosting Algorithms for Predictive Analytics, Part of the Algorithms for Intelligent Systems book series (AIS), 2019.

[17] I. L. Cherif and A. Kortebi, "On using eXtreme Gradient Boosting (XGBoost) Machine Learning algorithm for Home Network Traffic Classification," *2019 Wireless Days (WD),* 2019.

[18] F. a. V. G. a. G. A. a. M. V. Pedregosa, "Scikit-learn: Machine Learning in Python," *Journal of*

Machine Learning Research, vol. 12, pp. 2825--2830, 2011.

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation,* vol. 9, p. 1735–1780, 1997.

[20] D. Consortium, “Industrial Data PLatforms and seaport community requirements and challenges M15,” 2021.

[21] ITI, “RADIATUS,” [Online]. Available: https://radiatus.iti.es/.

[22] Dask, “DASK,” [Online]. Available: https://dask.org/.

[23] Apache, “MINIO,” [Online]. Available: https://min.io/.

[24] EC, “H2020 Annotated Model Grant Agreement,” [Online]. Available: http://ec.europa.eu/research/participants/data/ref/h2020/grants_manual/amga/h2020-amga_en.pdf.

[25] A. Metzger, T. Kley and A. Palm, “Triggering proactive business process adaptations via online reinforcement learning,” *18th Int’l Conference on Business Process Management (BPM 2020), ser. Lecture Notes in Computer Science, Sevilla, Spain, Springer,* vol. 12168, September 13-18, 2020.

[26] A. Palm, A. Metzger and K. Pohl, “Online reinforcement learning for self-adaptive information systems,” in *32nd Int’l Conference on Advanced Information Systems Engineering (CAiSE 2020)*, Grenoble, France, Springer, June 8-12, 2020.

[27] A. Metzger, T. Kley and A. Palm, “Triggering Proactive Business Process Adaptations via Online Reinforcement Learning,” in *International Conference on Business Process Management*, 2020.

[28] A. Palm, A. Metzger and K. Pohl, "Online Reinforcement Learning for Self-adaptive Information Systems," in *Advanced Information Systems Engineering*, Cham, 2020.

[29] W. F. et al., “PyTorch Lightning,” *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning,* vol. 3, 2019.

8.2 ACRONYMS

Acronym List	
AMTE	Automatic Model Training Engine
API	Application Programming Interface
CSV	Comma Separated Values
DL	Deep Learning
ETD	Estimated Time of Departure
KPI	Key Performance Indicator
LSTM	Long-short Term Memory
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
PC	Project Coordinator
PCS	Port Community System
RL	Reinforcement Learning
SLA	Service Level Agreement
TOS	Terminal operating system

Table 10 – Acronyms

9 ANNEX A: EXTERNAL EXPERT REPORT

9.1 EXTERNAL EXPERT REPORT

Following recommendations after the first review meeting, the consortium has looked for external opinion about the choice of technologies used to implement the functionalities foreseen in the DataPorts platform, specifically the comment was:

"Another recommendation concerns the choice of the technology used to implement some modules. In this sense, the experts suggest the consortium to look for some professional advice (from technical architects for example) in order to validate how appropriate those technologies are in the context of a (big) data platform. For instance, one of the modules includes a software to filter and process incoming data. The technology chosen is a web application based on the Flask framework, which may not be suitable for handling big volumes of data."

The technical architect asked to carry out the evaluation has been Dr. Jordi Arjona Aroca. Dr. Arjona has a Telecommunications engineering degree by the Universitat Politècnica de València (UPV) in 2008. He has two MsC, one in industrial computer science, automation and robotics by the UPV and a second one in telematics engineering by the Universidad Carlos III de Madrid, where he also got his PhD in Telematic Engineering, cum laude, in 2015 in a joint program with IMDEA Networks Institute. During his PhD he had the opportunity to spend 6 months in Beijing, in the Institute for Computing Technology (ICT) of the Chinese Academy of Sciences and of 3 months in IBM India Research Labs.

Besides his PhD position at IMDEA Networks Institute, he worked for 2.5 years as a Postdoctoral Researcher, and later, as Member of Technical Staff, at Nokia Bell Labs in Dublin. Moving back to Spain, he worked at the Fundacion Valenciaport for 19 months, before joining ITI in April 2019. During these years he has participated in numerous European projects, even as a member of the External Advisory Board (RECAP). Similarly, he has multiple publications in international conferences and journals as well as some patents.

The following subsections collects his thoughts and findings after carefully reading the documentation available: D2.4 Platform architecture and specifications (M24), D3.5 Data Processing Services (M27) and D3.6 Data Analytics and Services and Cognitive Applications (M27). Each subsection collects feedback component by component, plus an overall platform analysis in the final conclusion's sections.

Finally, it is worth mentioning that the analysis has been carried out from a purely big data analysis perspective, not considering other project requirements. As an example, an IDS architecture implementation may imply to use a technology not suitable for big data analysis, it may happen that a technology which satisfies a concrete functionality may be a perfect fit for big data analysis, but incompatible with IDS architecture. In those cases, an analysis of pros and cons have been carried out together with a benchmarking exercise to find the most suitable solution for DataPorts. These cases are documented in WP3 deliverables.

9.1.1 Data Access Agents

Data ingestion in DataPorts is performed through the Data Access Agents. The selected tool is Cygnus, from the FIWARE ecosystem. Cygnus is based on Apache Blume and, hence, using a Source-Channel-Sink approach. Flume, or in this case Cygnus, is a solid solution for several reasons: Because, apparently, the complexity of the required data flows is low, only data ingestion is performed, and the work is on a Hadoop ecosystem. However, it could have been interesting to consider tools like Apache NiFi or, at least knowing why it has not been considered or why it has been discarded (for instance being an overkill). Considering the whole picture, e.g., the use of FIWARE components, Cygnus seems a solid and sufficient tool for this task.

9.1.2 Semantic Interoperability

The semantic interoperability component is necessary to ensure that data coming from the different ports involved in the project is homogeneous from an ontological point of view. Constructing this ontology is challenging because, even when data in different ports is or should be similar, the degree of digitization is very heterogeneous and so is the data available. Hence, the subsets of data available in each port may have had a small intersection. This however should be partially solved, or the impact smoothed, thanks to the participation of the ValenciaPort foundation, related to the port of Valencia, one of the most important ports in Europe, highly digitized and that has participated in projects devising partial ontologies on some of the most common procedures in ports. Assuming, then, that the partners have the knowledge and means to construct a solid data model, we must focus on the technologies used.

To this regard, the components used in DataPorts make sense from the point of view of their attachment to the FIWARE ecosystem. DataPorts uses the Orion Context Broker and NGSI specifications. These are widely known and accepted technologies that are, for instance, key components of the Connecting Europe Facility digital catalogue. They can be considered as solid components. Similarly, their use jointly with MongoDB seems solid enough. The main risk could have been that Orion cannot provide the functionality for requesting historical data, but this has been covered with a component devised and implemented in purpose. These solutions seem totally valid to, later, provide services or integrate with initiatives like IDSA brokers (to publish datasets metadata) or CKAN to publish the datasets themselves.

9.1.3 Data Abstraction and Virtualisation

The main technologies involved in the Data Abstracion and Virtualisation component in DataPorts are Apache Spark, Apache NiFi and MongoDB. The component is well structured and each layer well motivated and with clear goals defined. The selection of Spark for the Preprocessing and Filtering software (PaFS), focused on the preprocessing, cleaning, and filtering of data over other tools, such as Flink or Storm, is well justified from my humble point of view. Apache Spark is a well-known and widely used tool in the BigData community and I think its use is totally justified here, especially due to its scalability.

The functionality of the Virtual Data Repository (VDR) resembles a data lake. This data lake relies completely in MongoDB, which is, from my point of view, a perfectly valid and solid solution as it reflects its current positioning in the market. However, I am surprised that the solution only considers document-based storage. Although this is also a consequence of the previous layers, totally oriented to this type of data, it is relatively surprising that no other type of storage has been considered, like for storing objects. In any case, considering this limitation, the selection of MongoDB as storage technology is completely justified.

Finally, the selection of Apache NiFi and Apache Spark for the Virtual Data Container (VDR) seems to me, again, to be solid and completely justified, especially considering the ease for interacting with Spark from NiFi, or the usability of NiFi.

9.1.4 Process Based Analytics

The process-based analytics component is composed by three main subcomponents performing different types of monitoring: an ensemble predictive process, a prescriptive process, and an explainable predictive process. These seems to me three valid approaches to put in value the operation of the underlying components (data access, semantic interoperability, or DAV) moreover considering the kind of data available. However, I think the analysis we must perform is on the technologies used to do it. To this regard, the use of TensorFlow and Keras seems a perfectly valid approach, widely used in the community and in more complex and ambitious projects, that perfectly covers the need of this component. Same applies to the use of Python or Django.

9.1.5 Automatic Models Training Engine

There are several aspects to look at in this component. Starting with the training web interface, the use of frameworks like Angular or Django, even when I am not an expert in these matters, seems correct. In addition, we are talking about frameworks which are, again, widely used in similar circumstances and known to provide good results. The use of auxiliary tools like PostgreSQL is also justified and I do not think impacts in a substantial way and, finally, using NodeJS is quite a standard practice as well. Regarding the training distribution engine, the main technologies to use are Radiatus and Dask. Dask enables the parallelization of Python code and, given the availability of hardware resources, seems a valid option to easily leverage them without increasing the complexity on the design of the pipelines, which we will comment later. Radiatus, although not an extended tool given its early stage (TRL~7), will perfectly meet the purpose for what it is intended, also allowing a nice management of hardware resources. Coming back to the pipelines, using statsmodel, scikit learn and similar Python libraries is sufficient and solid, from my point of view, for the services being proposed.

Finally, we look at data formats and storing technologies. On the one hand, the use of Apache Parquet is justified, from my point of view, and brings in some advantages over standard row formats or json. Regarding the storage, I do not think using HDFS is totally justified over other alternatives like Minio. It is true, however that HDFS is sufficient and works smoothly with other tools used in this component, however, Minio may have been a better approach.

9.1.6 Final Conclusions

After reviewing the proposed architecture and the different elements in detail, I do not see or think that there is a technical risk. There can always be opinions on whether this or that piece of software could have been a better match for a particular functionality but, from my point of view, the selected technologies suffice to meet the goals of the project.

Moreover, I not only think that the technologies are sufficient considering the different blocks individually, in isolation, also I do not foresee that there may be problems regarding their integration to have a fully functional framework. Therefore, my feedback is positive, and I am not concerned about the future success of the project.

9.2 RESPONSE FROM THE AUTHORS OF THE DELIVERABLE

The responsible of the deliverable D3.6 have thoroughly examined the current external expert report, to evaluate the fit of the technological approaches agreed in the WP3 of Dataports. The main purpose of the technologies selected was to tackle all requirements indicated in the description of the WP3 such as: ports oriented cognitive services with State-of-the-art ML algorithms, distributed AI platform, etc.

The specific information regarding the selection and description of the selected technologies is detailed in the following sections of the deliverable D3.6:

- Section 4.2 Technological description: describes the architecture and technologies behind the Process-Based Analytics component. LSTM, Tensorflow, Django REST Framework
- Section 4.2.1 Ensemble Predictive Process Monitoring: LSTM, Tensorflow
- Section 4.2.2 Prescriptive Process Monitoring: Reinforcement Learning
- Section 4.2.3 Explainable Predictive Process Monitoring: Recurrent Neural Networks

- Section 5.2.1 Architecture of the component: Describes the architecture designed and the technologies selected to develop of the Automatic Model Training Engine component: DASK, MLFlow, Python, Angular, Django REST, Parquet, HDFS, Docker.
- Section 5.2.2.1 Frontend: Angular Framework. Explains the selection of Angular as framework to develop the frontend
- Section 5.2.2.2 Backend: Django Framework. API REST DJANGO. Describes Django as the choice the carry out the backend of the component
- Section 5.2.2.4 Components packaging. Depicts the use of Docker to package all developed subcomponents into one entity
- Section 5.2.4 Machine Learning pipelines and algorithms: Illustrates the different Python libraries utilized to develop the Training Engine: Sklearn, Statsmodels, prophet, torch
- Section 5.2.5 Models tracking and deployment: Explains the technology MLFlow as the choice for model’s tracking and deployment

To address the improvement comments made by the expert reviewer, the responsible of the deliverable D3.6 developed the following responses:

- Comment 1

Regarding the training distribution engine, the main technologies to use are Radiatus and Dask. Dask enables the parallelization of Python code and, given the availability of hardware resources, seems a valid option to easily leverage them without increasing the complexity on the design of the pipelines, which we will comment later. Radiatus, although not an extended tool given its early stage (TRL~7), will perfectly meet the purpose for what it is intended, also allowing a nice management of hardware resources.

Radiatus [21] was the chosen technology to automatically handle the distribution and parallelization of the data processing and model trainings required in AMTE. In a nutshell, Radiatus is a platform for building elastic big data analytics services in the cloud developed by ITI. However, Radiatus will finally not be implemented in the AMTE component due to a set of technical circumstances. First, RADIATUS is still in the process of upgrading to a newer version, which besides to the implementation of additional state-of-the-art Big Data technologies, it also aims to include automatic distributed processing and storage. Unfortunately, this new version is not ready yet. Second, although the previous version already implemented a stable set of Big Data Technologies, distributed processing was not yet achieved, so it could not meet the requirements of the AMTE component. As a result, the inability to use Radiatus led the AMTE responsible to address the component requirements by implementing DASK [22], which is perfectly capable of handling the indicated distribution of training and processing tasks developed in Python, as the reviewer concluded.

- Comment 2

Regarding the storage, I do not think using HDFS is totally justified over other alternatives like MINIO. It is true, however that HDFS is sufficient and works smoothly with other tools used in this component, however, MINIO may have been a better approach.

At first, HDFS was selected as the preferred storage technology because it was already implemented in Radiatus, thus making it easy for the developers to use it as storage system as the configuration complexity was automatically avoided. Nonetheless, as explained in the comment 1, Radiatus will not be finally utilized in AMTE. Hence, an alternative was analysed by the responsible of D3.6 and agreeing with the technical reviewer point of view, MINIO [23] has been selected as storage technology. As an object store, MINIO is an open-source technology that can store a great lot of unstructured data and may perfectly serve for the needs of AMTE for storing imported Dataports datasets for training purposes.

Finally, although this document is the final version of the D3.6, if any further change is needed soon, it will be properly indicated and explained in the corresponding communication sources and channels.