Data Platform for the Connection of Cognitive Ports

Title:	Document Version:
D3.3 Data Analytics Services and Cognitive Applications M18	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*:
M18 (June 2021)	M18 (June 2021)	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:
Francisco Valverde	ITI	WP3

Authors (organisation):	
Miguel Bravo (ITI)	Francisco Valverde (ITI)
Santiago Cáceres (ITI)	Tsunghao Huang (UDE)
Paolo Calciati (ITI)	Tristan Kley (UDE)
Manuel Sánchez (ITI)	Andreas Metzger (UDE)

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	04.06.2021	First version of the document.	Miguel Bravo (ITI), Manuel Sánchez (ITI) and Francisco Valverde (ITI)
V0.2	07.06.2021	Contents for Process-based Analytics	Tsunghao Huang (UDE)
V0.3	21.06.2021	Updated with reviewers' comments	Paolo Calciatti (ITI), Francisco Valverde (ITI) and Tristan Kley (UDE)
V1.0	29.06.2021	Final version for submission	Santiago Cáceres (ITI) and Francisco Valverde (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

Copyright Statement

INDEX

LIST OF FIGURES

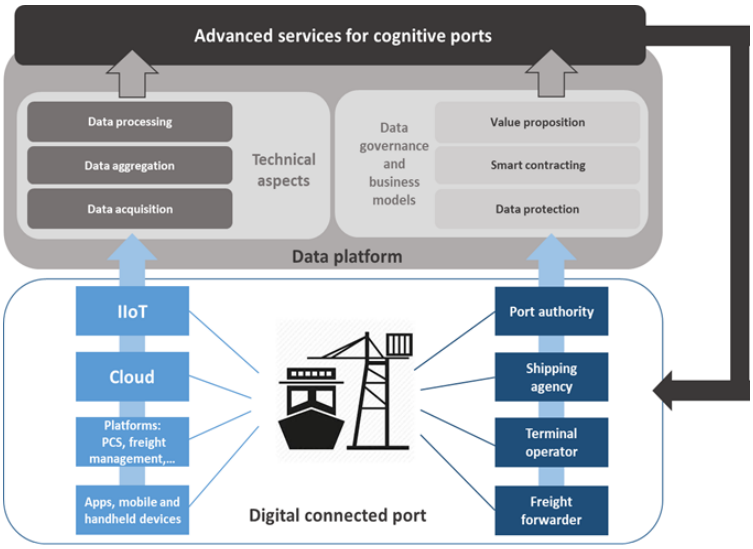
LIST OF TABLES

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 3.3, specifically:

- The design and development of a set of data analytics services for supporting 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 for supporting 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.

1.4 DOCUMENT STRUCTURE

This deliverable is broken down in the following sections:

- **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 details the work carried out in the development of the Process-based Analytics Component, specifically, the technologies, how it is expected to be used and the current development status.
- **Section 5** Automatic Model Training Engine: this section mirrors the previous one but detailing the Automatic Model Training Engine:
- **Section 6** Conclusions: this section briefly states the conclusions of the deliverable.

1.5 DOCUMENT DEPENDENCIES

This document is the first version of an iteration of a living deliverable detailing the components developed in T3.4 until M18. The next version of the deliverable is due on M30.

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 for supporting 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 for supporting 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).

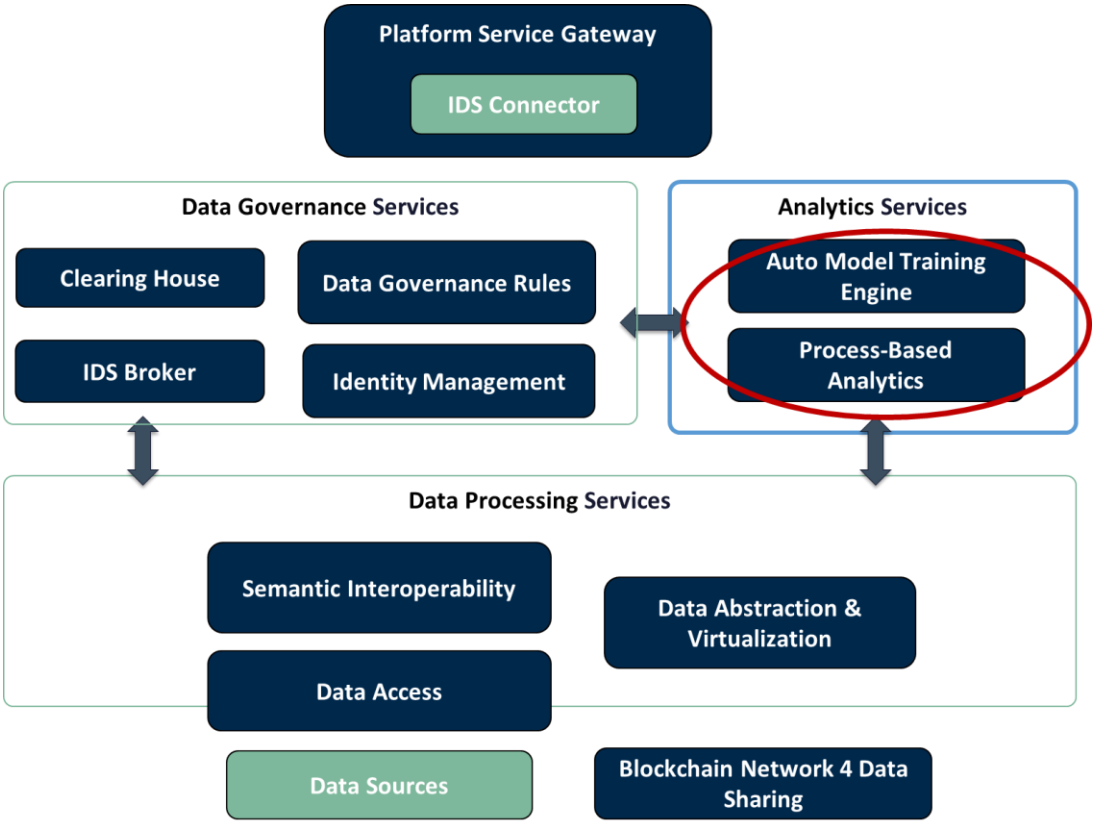


Figure 1 – DataPorts platform building blocks

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.

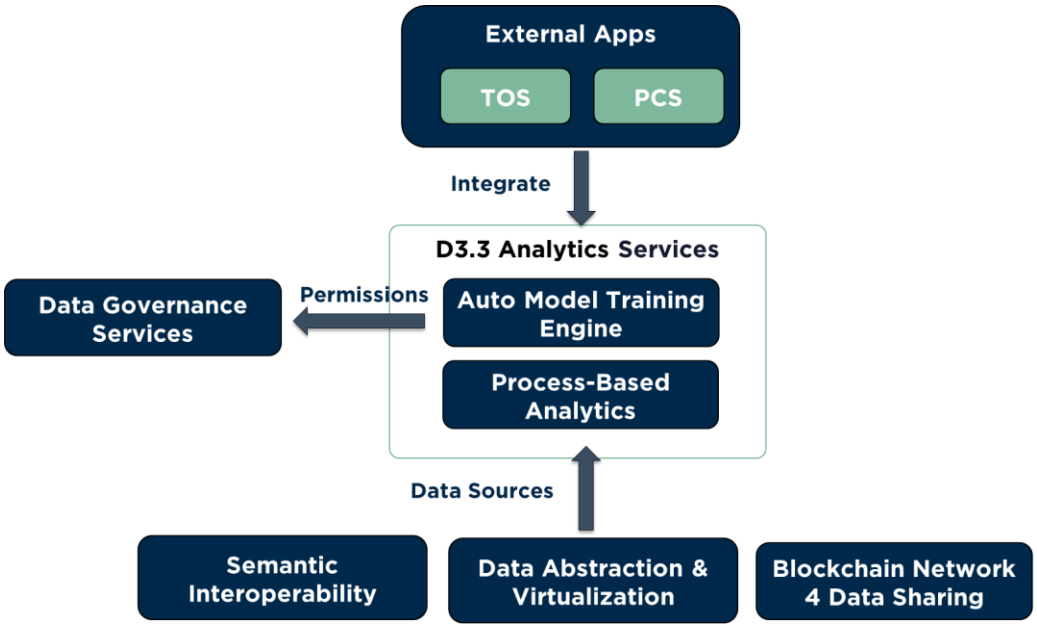


Figure 2 - Analytics components in the architecture

4 PROCESS-BASED ANALYTICS COMPONENT

4.1 OVERVIEW

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, predictive business process monitoring predicts how an ongoing case will unfold. To this end, predictive business process monitoring uses the sequence of events produced by the execution of a business case to make predictions about the future state of the case. If the predicted future state of the case indicates a problem, the ongoing case may be proactively adapted; e.g., by re-scheduling process activities or by changing the assignment of resources.

The Process-based Analytics component aims at optimising the business process 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 process. By proactively predicting the future states of the ongoing process, the component provides forward-looking perspectives for the users to make decisions.

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 to provide its predictive results to cognitive applications, which inform the end-users about the predictions. It consists of the following three main components. By exploiting advanced data analytics techniques and machine learning, these components offer decision support for terminal and process operators, thereby facilitating proactive management of port processes:

- Ensemble Predictive Process Monitoring: This component uses ensembles of deep learning models (recurrent neural networks) to provide accurate predictions for each point during process execution, i.e., in a streaming fashion.
- Prescriptive Process Monitoring: Building on process predictions, Online Reinforcement Learning allows automating the process on when to adapt a running process. We apply state-of-the-art Reinforcement Learning algorithms to the problem of identifying the signs of possible failure early and accurately.
- Explainable Predictive Process Monitoring: This component aims at providing interpretations on why a certain prediction is made by a black-box predictive model, in particular by the deep learning models used in the first component above. To generate highly accurate predictions and at the same time facilitate interpretability for predictive process monitoring tasks, we leverage the concept of model induction from interpretable machine learning (ML) research.

The technological stack used for our components consists of Python and Tensorflow.

4.2 TECHNOLOGICAL DESCRIPTION

4.2.1 Ensemble Predictive Process Monitoring

There are two quality-criteria for process predictions. First, predictions should be accurate. When adaptation decisions are based on inaccurate predictions, this may imply unnecessary adaptations (e.g., if a delay is falsely predicted) or missed adaptations (e.g., if an actual delay is not predicted). Second, predictions should be produced early during process execution. Earlier predictions leave more time for adaptations, which typically have non-negligible latencies.

As a solution to address this trade-off, this component computes a reliability estimate in addition to a prediction. To increase prediction accuracy and enable the computation of the reliability estimate, the component uses an ensemble of several individual machine learning prediction models. The ensemble

prediction for a checkpoint is computed as a majority vote, while the corresponding reliability estimate is computed as the fraction of individual model that predicted the majority class.

Reliability estimates help operators distinguish between more and less reliable predictions on a case-by-case basis. Together with the earliness indicators, reliability estimates can help operators decide whether to trust an individual prediction enough to adapt the running process. For example, a terminal operator may be informed that there is a high reliability prediction for a process to be delayed and that there are only a couple of hours remaining for any proactive action.

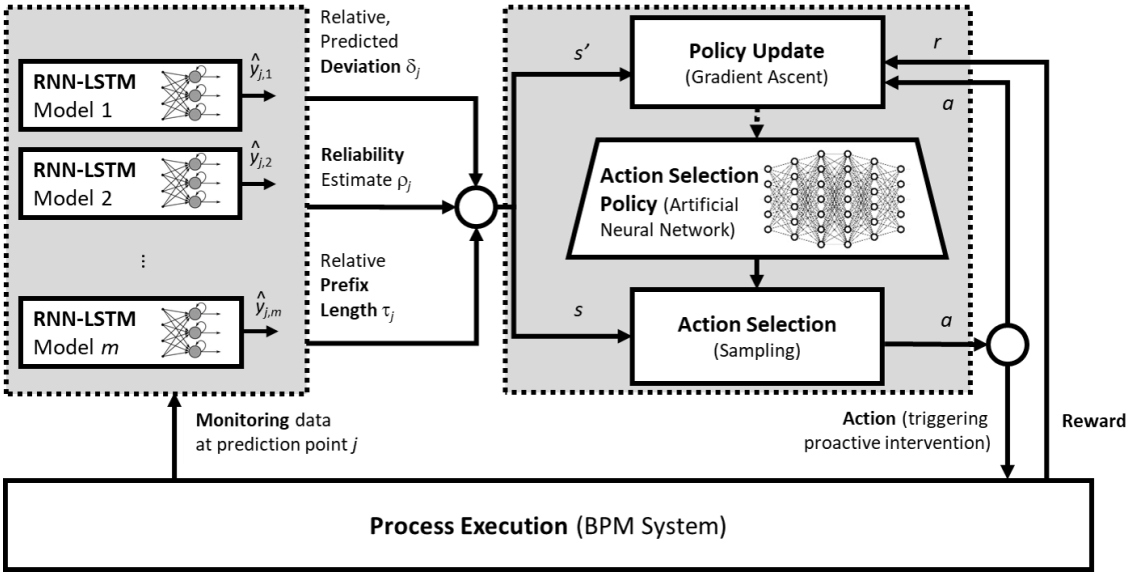


Figure 3 - Process-Based Analytics Components

Figure 3 shows the internal structure of the component. On the left-hand side is depicted the ensemble predictive process monitoring component It consists of an ensemble of Long Short-Term Memory [2] Recurrent Neural Network (LSTM). LSTM is a special recurrent neural network that not only has feedback connections but also special mechanisms in the LSTM cell to remember values over arbitrary time intervals. The main output of the component is the prediction about the future state of a monitored business process at each step of the process, as well as a reliability estimate of prediction while the main input of the component is monitoring data about the monitored business process, which allows the component to make its predictions.

4.2.2 Prescriptive Process Monitoring

By setting different thresholds for the reliability computed by the Ensemble Predictive Process Monitoring component, one can 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 component 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.

The RL-agent learns the effectiveness of an agent's actions through the agent's interactions with its environment. As shown on the right-hand side of Figure 1, the agent selects and executes an action a in response to environment state s. As a result, the environment transitions to s' and the agent receives a reward r for executing the action. The goal of RL is to maximise cumulative rewards.

We formalise the learning problem of when to trigger a proactive process adaptation by defining actions a, states s and rewards r. 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 estimate 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 1 - 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 maximise 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.

4.2.3 Explainable Predictive Process Monitoring

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, we provide the users with interpretations in this component. We adopt interpretable model

induction techniques to generate separate interpretable models that approximate the behaviour of the black-box predictive models.

A graphical overview of the artifacts and activities of explainable predictive process monitoring component is shown in Figure 4. The Upper part of the figure depicts how individual prediction are is generated. As this is not the focus, it is represented by dashed lines. The lower part of the figure depicts how explanations are generated. This consists of three main stages: the approach starts with stage 0 by finding similar prefixes for initialisation. The purpose of this step is to ensure that we get an initial population consisting of realistic prefixes. Furthermore, the control flow attributes of the first population will be used by the mutation step later as well. Then, in step 1, the approach uses the genetic algorithm to generate neighbourhood instances. The genetic algorithm is used here to ensure we can have compact synthetic instances that are close to the 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 and with a pre-defined black box label. 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, after the stop criteria is met, the approach trains a decision tree for the derivation of explanations using the synthetic prefixes generated by the genetic algorithms. Based on the preference of the users, the explanation can be presented as factual or counterfactual explanations.

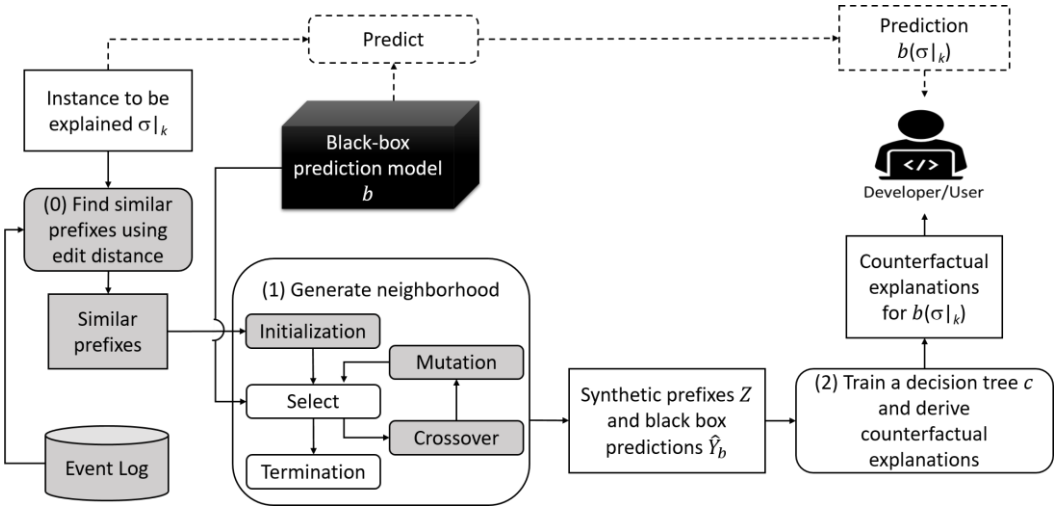


Figure 4 - Overview of Explainable Predictive Process Monitoring Component

4.3 EXAMPLE OF USE: DEMONSTRATION

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 3 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.

Research prototypes for our component have been developed. We evaluated these research prototypes using the aforementioned 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. .

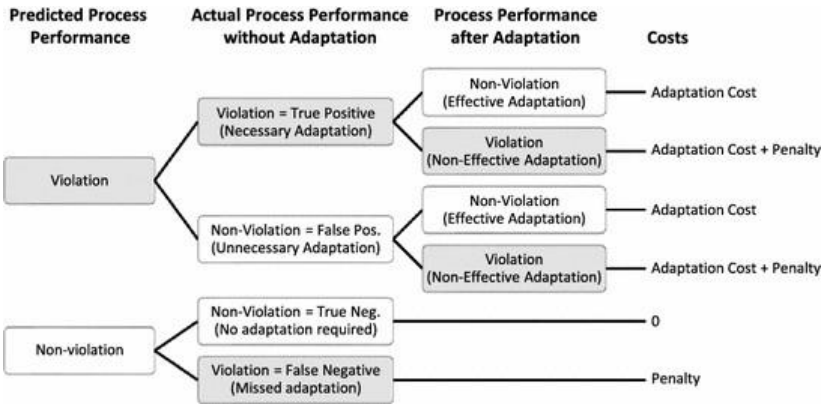


Figure 5 - Asymmetric costs of proactive business process adaptation

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
Cargo2000	Delayed air cargo delivery	27%	3,942	144
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 2 - Data sets used in evaluation

Using two of the data sets as an example, Figure 4 gives a first impression of the effect of the Ensemble Predictive Process Monitoring component without the Prescriptive Process Monitoring component. The figure shows the cumulated costs of the processes within a data set when using our approach to

dynamically (bold) adapt on the first prediction point for which the reliability of a prediction crosses a set threshold; as compared to **statically** (dashed) choosing a prediction point ahead of time and only making a prediction at that point, adapting if the reliability of the prediction exceeds the threshold. Every dashed line in the figures represents a different prediction point at which the adaptation takes place. The ordinate of the graphs matches the costs of the different static adaptation strategies and the dynamic strategy to the different reliability thresholds on the abscissa. The red line in each chart shows the costs without any adaptation. These costs also serve as baseline. We chose $\alpha_{max} = .9$ and $\alpha_{min} = .5$, reflecting the fact that early in the process there is a high chance that adaptation is effective, whilst at the very end, this chance is only 50%. Also, we show the results for two values of λ (relative adaptation costs). A $\lambda = .1$ reflects the situation where adaptation is rather cheap, whereas $\lambda = .4$ reflects a situation where it is more expensive.

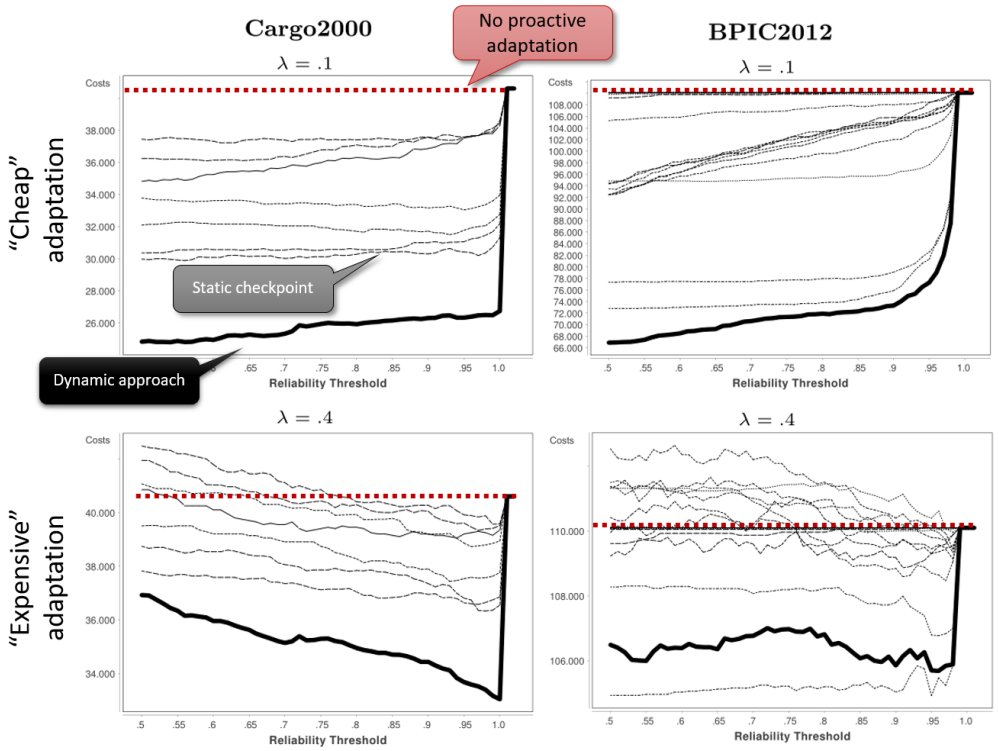


Figure 6 - Comparison of costs in evaluation

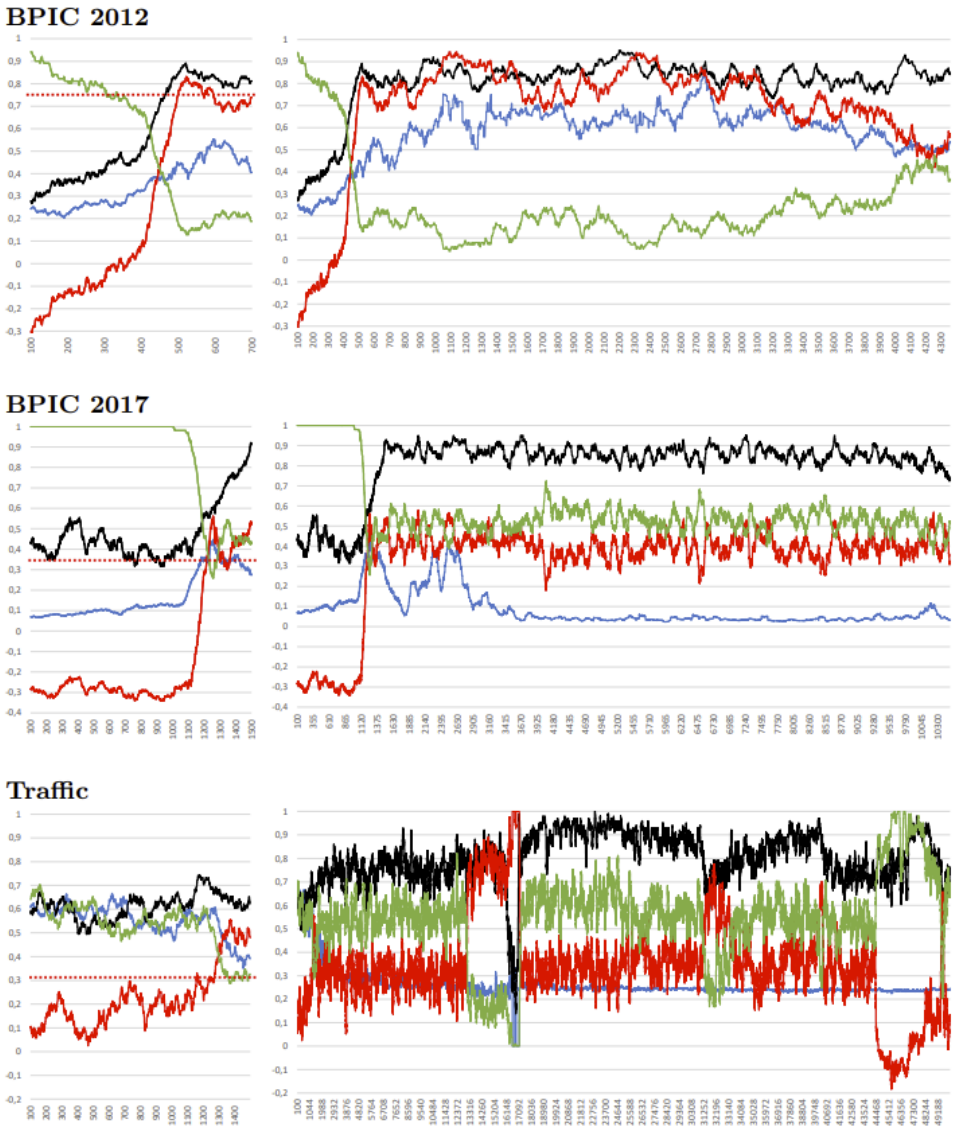
Overall (i.e., considering all possible situations), the average savings of using dynamic adaptations compared to static adaptations are 9.2% for Cargo2000, 27.2% for Traffic, 15.1% for BPIC2012, and 35.8% for BPIC2017. Across all four data sets, average savings are 27%. We conclude that the dynamic approach can deliver cost savings compared to the static approach, with a high chance that it is better than not performing any proactive adaptation at all.

Using the same cost model similar 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 5 show the respective results for the three data sets. As Cargo2000 is a comparatively small data set, it could not be used in this evaluation.

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).

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.



(a) Until convergence (b) Complete

Figure 7 - 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.

4.4 DEVELOPMENT STATUS (M18)

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 and pending steps toward M18 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	Research prototype and experimental results are available
Pending	Adaptation for DataPorts APIs and interconnection with 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	A description regarding the data requirements and the training of ML models will be available.

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	There is currently no such dataset available to test this feature.
Pending	Not Started

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	We are coordinating with partners to understand the data sources. Afterwards, possibility to fulfil this feature will be accessed.
Pending	Not started

5 AUTOMATIC MODEL TRAINING ENGINE

5.1 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, specifically from the time series forecasting domain. 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 defined goal. Then, the resulting model is wrapped as a REST Service which can be deployed as a cognitive service.

5.2 TECHNOLOGICAL DESCRIPTION

5.2.1 Data processing and visualisation

One technical goal of this component is the distributed training of ML models, which not only is more accurate, but also increases the training speed. The architecture and main technologies are summarised in Figure 8.

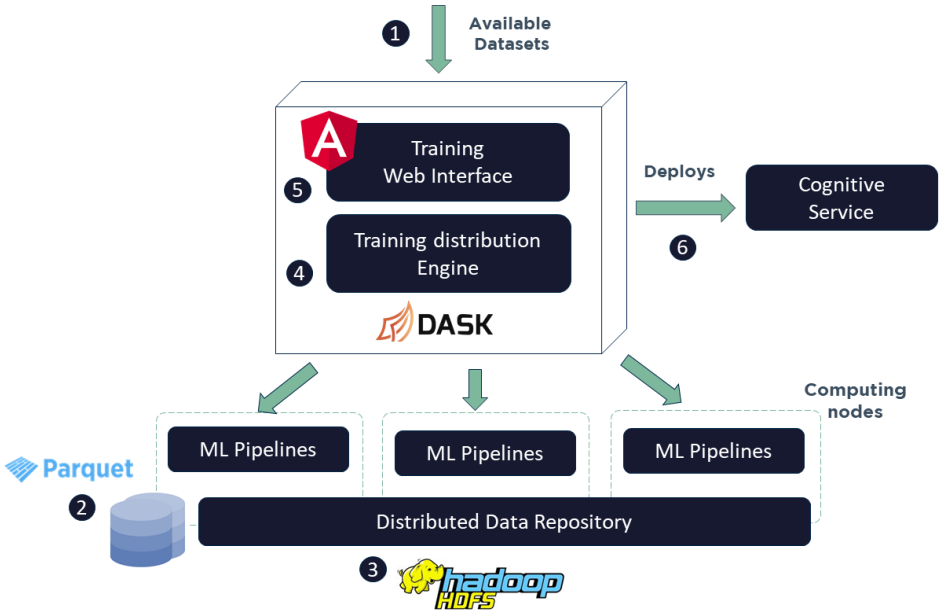


Figure 8 - Architecture of the Automatic Model Training Engine

As starting point, the main input of this component are datasets available in the context of the DataPorts ecosystem (Figure 8-1). Next, to achieve a distributed training approach, first 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 8-2). Apache Parquet is a columnar file format that provides optimisations to speed up queries and is 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 9 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.

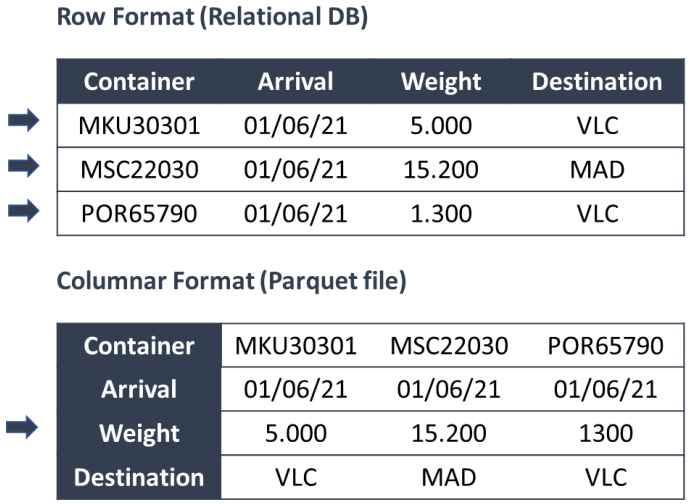


Figure 9 - Row vs Columnar format

Imported datasets are transformed to parquet files and then stored and replicated in one HDFS[2] server (Figure 8-3). This server is composed of several physical nodes with a replica of the parquet file in each of them. Using a standard filesystem as HDFS, this replication process is transparent. Additionally, this approach guarantees that each model reads the data from the node in which it is deployed and, thereby, the network traffic is reduced.

One of the key technical capabilities is to train simultaneously several ML models using different algorithms and parameters (Step 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.

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. The second approach has been followed for implementing this component. 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, this scheduler introduces parallelisation at the thread level, so several

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

[2] HDFS: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html

[3] Dask: https://docs.dask.org/en/latest/

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.

Finally, for simplifying the interaction of end-users with the component, a web dashboard is implemented (Figure 8-5) to support functionalities, such as importing datasets, selecting the relevant variables, choosing the most suitable training strategy (see next section) and finally, deploy or stop a trained service. This dashboard is made up of two subcomponents: a frontend developed in AngularJs[4], following the Single-Page Application pattern, and an API backend developed in Python and supported by FastApi.

FastApi [5] is an asynchronous Python framework to provide high performance, easy to learn and oriented to the generation of friendly documentation. In FastApi, for each declared endpoint, it is created a new entry in an Open API specification, which helps to document all the developed methods in a standard way. Other frameworks taken into consideration were Flask, Falcon and Django. But finally, FastApi was selected for its nice support to the asynchronous paradigm and a straightforward process to generate the Open API specification.

The developed frontend communicates with the Fast 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 performed asynchronously to not interrupt the interaction with the user interface. When the training is completed, the best model is deployed automatically (Figure 8-6) and can be used via its API.

5.2.2 Machine Learning algorithms and pipelines

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.

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 10. Green circles represent the main steps of the pipeline whereas black circles represent optional sub steps:

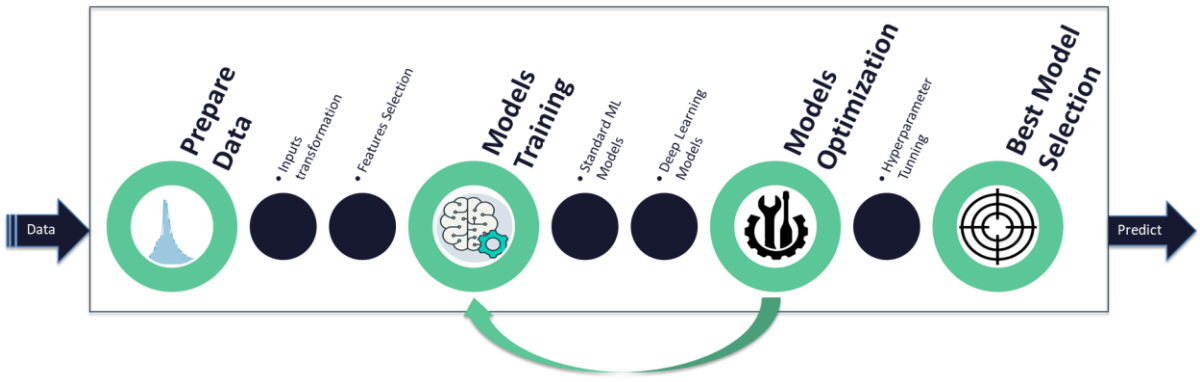


Figure 10 - Standard training pipeline

The main steps of this standard pipeline are:

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

[5] FastAPI: https://fastapi.tiangolo.com/

- **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 3.
- **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 11:

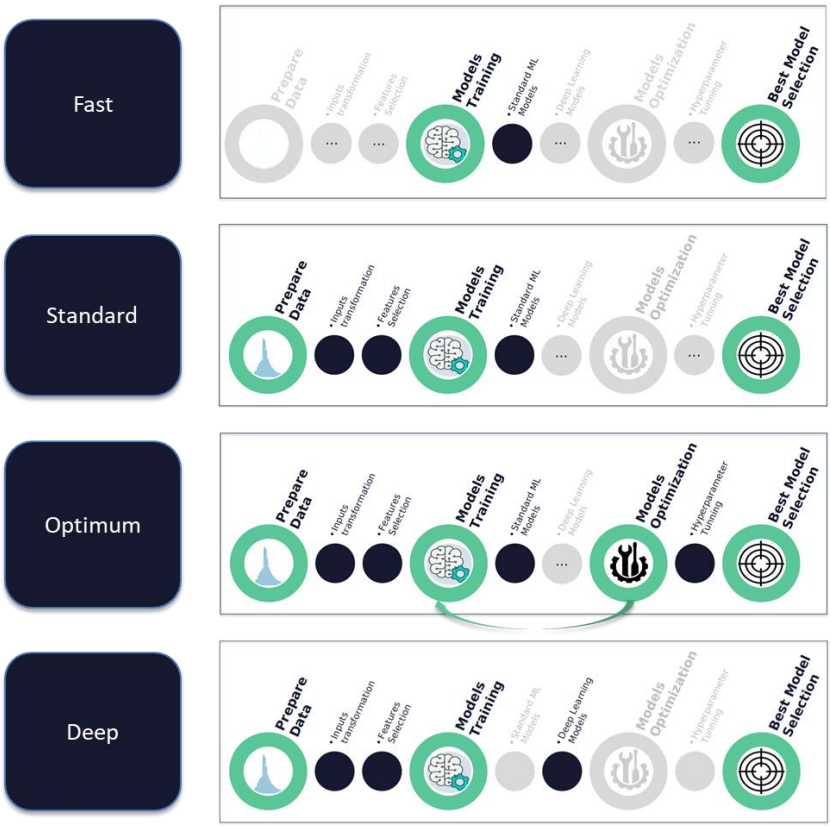


Figure 11 - Training Strategies

- **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.

As part of the strategy, the most critical decision in a ML pipeline is the ML algorithm chosen in the Models training step of the pipeline. As our approach does not require manual intervention, a set of available algorithms are selected beforehand and all of them are tested for training, according to the selected 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. In this context, the predictive problems to be solved are mainly related to the 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 selected algorithms are detailed in Table 3:

Algorithm	Learning type	Model Type	Library	Prediction type
Multiple Linear Regression [3]	Supervised	Linear model	sklearn	Univariate TSF
KNN Regressor [4]	Supervised	Nearest Neighbors	sklearn	Univariate TSF
Multi-layer Perceptron NN [5]	Supervised	Neural Network	sklearn	Univariate TSF
Support Vector Regressor [4]	Supervised	Support Vector Machines	sklearn	Univariate TSF
Random Forest Regressor [4]	Supervised	Ensemble	sklearn	Univariate TSF
Gradient Boosting Regressor [6]	Supervised	Ensemble	sklearn	Univariate TSF
SARIMAX [7]	Statistical	Autoregressive	statsmodels	Univariate TSF
VARMAX [7]	Statistical	Autoregressive	statsmodels	Multivariate TSF

Prophet [8]	Statistical	Autoregressive	prophet	Univariate TSF
LSTM [9]	Deep Learning	Recurrent Neural Network	torch	Univariate TSF

Table 3 - ML algorithms selected.

The tested algorithms have been selected due to their specific mathematical behaviour related to the prediction of time series, and specifically, with the initial datasets provided from the DataPorts project. All of them can learn a historical time series dataset and forecast future predictions. This list of algorithms will expand as the pilots' requirements evolve.

There exists plentiful number of metrics to evaluate ML models nowadays. However, only several are useful to evaluate ML models focused on Time Series. Those metrics give a roughly accurate idea on how good the models are to make predictions on future sequential and unknown data. Specifically, in the context of the pipeline the following metrics are selected to classify which is the best performing algorithm:

Metric	Acronym	Type of metric	Boundaries	Explanation
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
Mean Absolute Percentage Error	MAPE	Error	[0, 100]	MAPE is the sum of the individual absolute errors divided by the demand (each period separately)
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 4 - Evaluation metrics for Time Series Forecasting

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 5 - Models scoreboard for berth time predictor

In this evaluation, MAPE metric is chosen as it is easier to understand by end-users. As 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. Figure 12 shows the forecasts with data of the five first months of 2021.

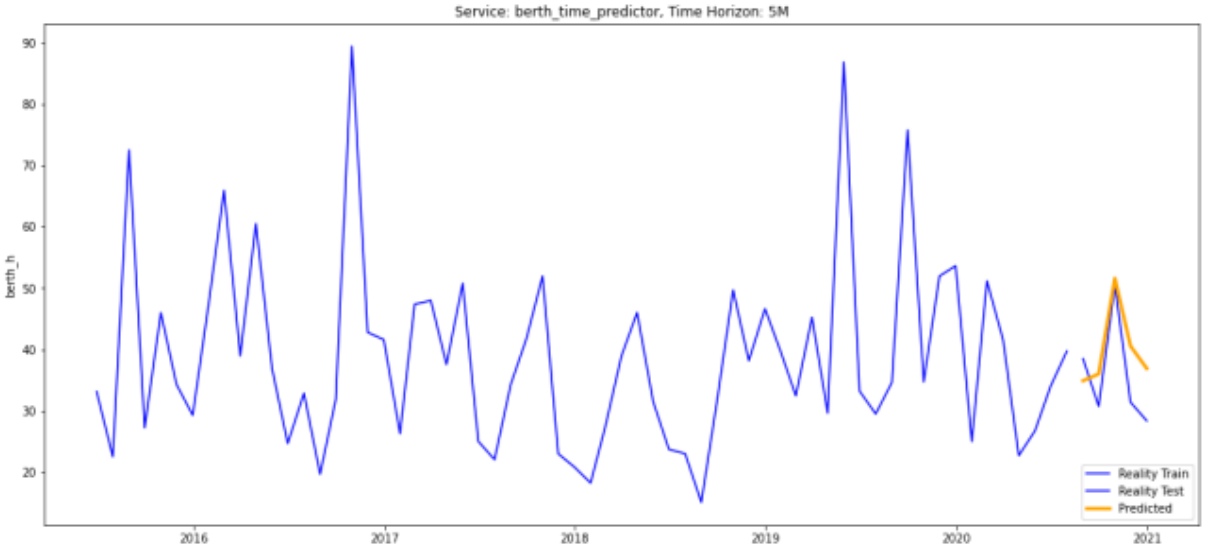


Figure 12 - Forecast of Average Vessel Berth Time for the next 5 months

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.

5.3 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 model.
- Training: Shows the Dask training interface (just for illustrative purposes).

Figure 13 – Dashboard main interface

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.

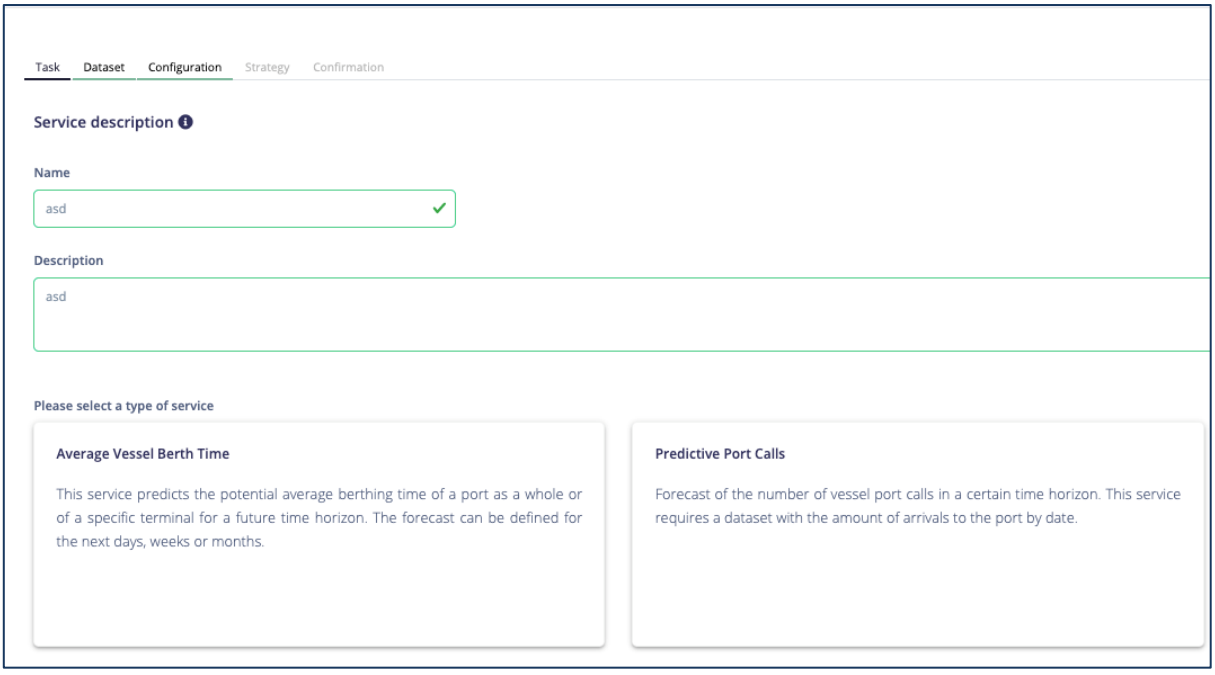


Figure 14 - Services definition

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.

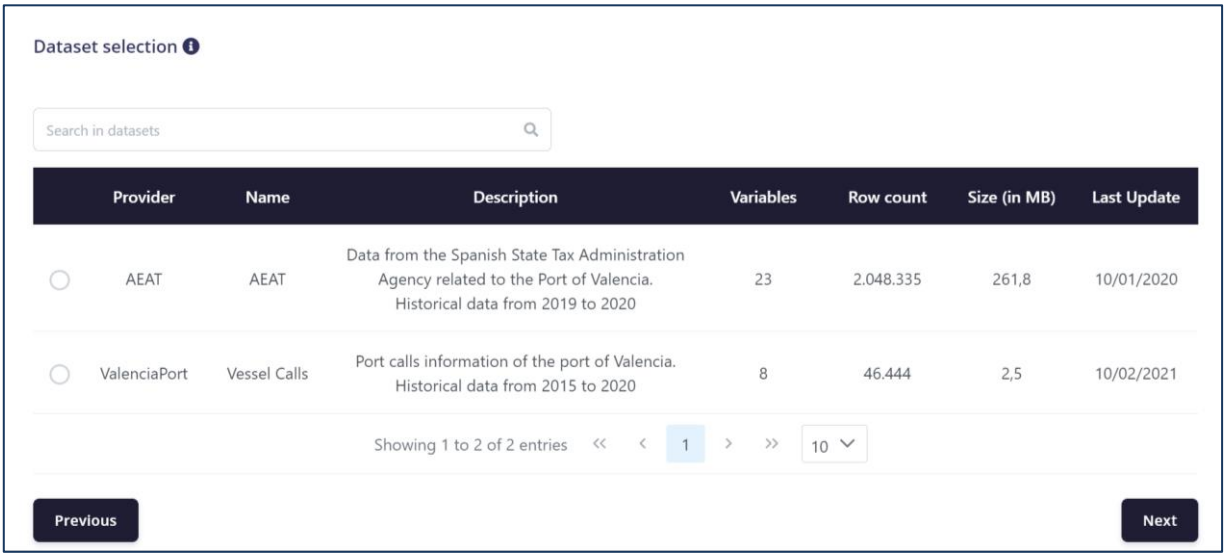


Figure 15 - Dataset Selection Screen

The third step, called *Configuration* (see Figure 16), its specific for each service. In this screen the user can choose additional information or options to be considered for training the underlying model. For instance, in the configuration of the Vessel Average Berth Time service (Figure 16 - up), the end-user could select if its relevant to consider the information regarding to the arrival terminal for the prediction, even to consider only vessels arriving to a specific terminal. 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 the Vessel ETD service se (Figure 16- down), the only available option is to include information related with vessel regular line. 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.

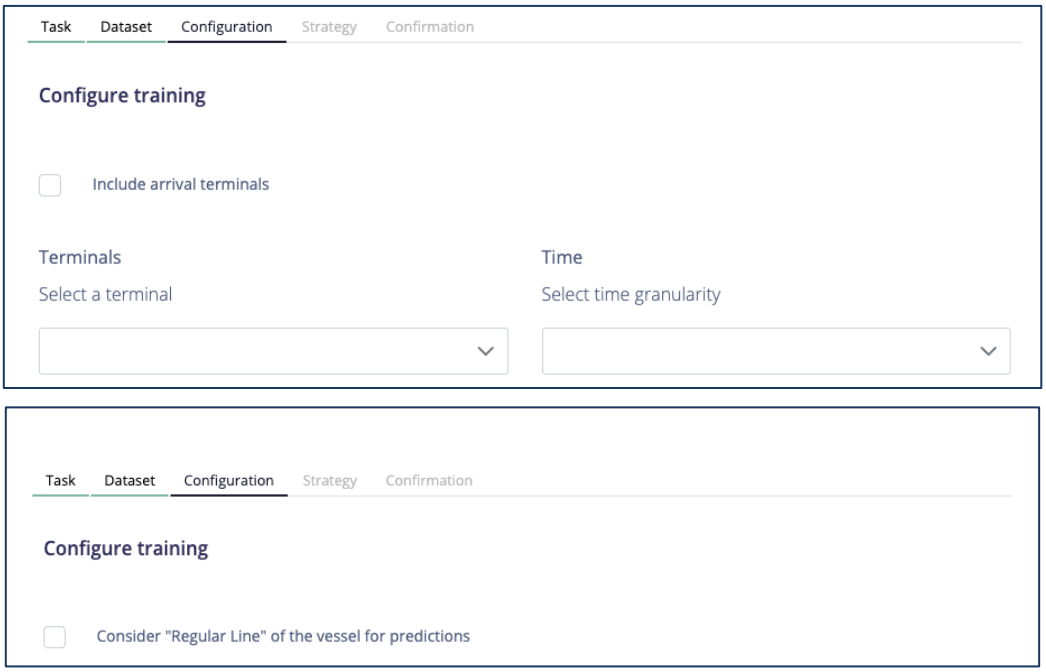


Figure 16 - Configuration Screens

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.

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.

Figure 18 - Confirmation screen

When a new service is being trained, it is possible to view the running parallelisation process with Dask (see Figure 19). The Dask interface shows the specific training tasks that are being carried out, the number of workers assigned to the whole process and the current resources used. Additionally, the task stream shows the flow parallel tasks to provide a concise view of the system performance as a whole and to find bottlenecks. This view is not expected to be shown to an end-user, but it is useful from the development point of view.

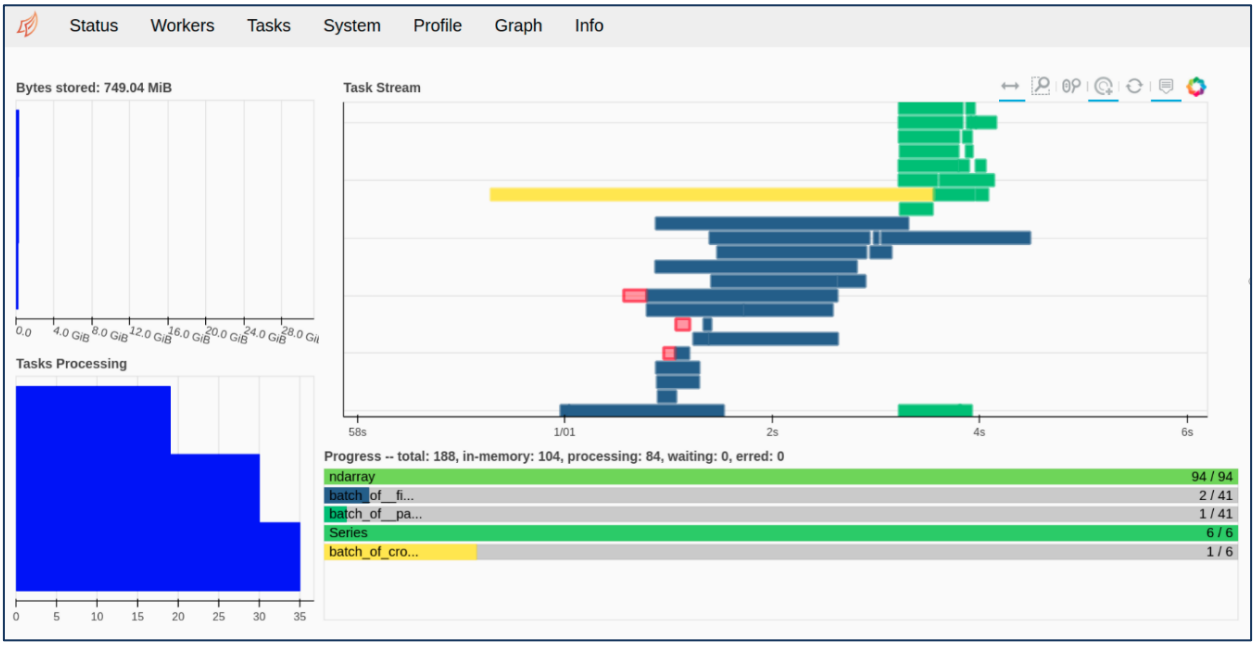


Figure 19 - Dask dashboard

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.

Figure 20 - Services Deployment Screen

Finally, the training process and the management of the services can also be done programmatically using an API.

5.4 DEVELOPMENT STATUS

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

- Dashboard: https://egitlab.iti.es/dataports/analytics/amte-dashboard
- 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 are packaged as Docker containers and using Docker-compose scripts to simplify the deployment. These containers have been tested in an infrastructure based on Openstack. Additional procedures and guidelines are provided to support the local deployment without Docker or to modify the current deployment configuration.
Pending	Apply the same approach to the subcomponents to be finished 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 5.2.2, it has been developed a standard pipeline that trains ML models with datasets available in the DataPorts platform to predict future values over a specified time horizon. The resulting services are available using an Open API specification.

	A web user interface has been already developed to support the end-user in this process, as it is depicted in Section 5.3.
Pending	Mainly, add improvements related with usability in the user interface such as the visualization of the predictions in charts or provide information about quality metrics of the trained model. Specific configuration screen for additional the cognitive services will be implemented. Feedback from pilots will also be taken into consideration.

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 in the Semantic Interoperability API and IDS broker. The release of the first version of this components is due to M18, so integration mechanisms will start in the following months.
Pending	Not started

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 uses wide set of different ML algorithms, specifically, from the state-of-the-art about Time series forecasting and regression. These algorithms have been benchmarked with data similar to the one found in ports information systems.
Pending	Include additional algorithms to enrich the current set if it is required by the pilots use cases. Test them using a distributed training approach in a cloud infrastructure, currently only local tests have been 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 have been already implemented using Dask, as it is described in Section 5.2.1. This approach also is reliable enough to deal with huge data volumes.
Pending	At this moment (M18) the overall distribution of the algorithms and data normalization is implemented. In the following months, it is expected to implement the hyperparameters optimization process and data distribution amongst nodes not implemented yet.

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	To support this requirement, first, a data stream is required but there is no such dataset from pilots able to generate such stream. 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	Not Started

ID 3.33	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 in the Semantic Interoperability API and IDS Broker, both developed by UPV and CERTH respectively. The release of the first version of these components is due to M18. A mockup API has been defined to test the integration with the developed dashboard.
Pending	Not Started

ID 3.34	
Description	As a data consumer, I want to subscribe to an available subscription provided by the DataPorts Platform
Status	This requirement is equivalent to 3.30 and has not started because subscriptions are managed with the Semantic Interoperability API due to M18.
Pending	Not Started

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	Not Started

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 of an Open API is already available with a set of general methods for: retrieving and selecting variables from a dataset, start the training of a specific service and start/stop the deployment of an already trained ML models, obtaining the stored datasets as well as selecting any of their variables. 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.

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

- Three prototypical subcomponents are developed, namely ensemble, prescriptive, and explainable process monitoring components.
- All three components have been evaluated using real-life datasets. Prototypes and experimental results are available as published research papers for ensemble and prescriptive subcomponents. Same materials can be provided for explainable process monitoring upon request.
- The evaluation results showed that the developed component has potentials to benefit the project. As the data extraction from pilot (supported by project partners) is ongoing, we expect to validate these promising results in the following iteration.

Regarding the Automatic Model Training engine, we highlight the following results:

- A dashboard for configuring the training process has been developed to enhance the usability and involvement of end-user for developing cognitive services. Initial feedback from end-users is positive in both regards.
- An extensive benchmarking of Time Series Forecasting approaches, resulting in a set of candidate algorithms to support the predictive goals of the project. They also have been included in our training strategies.
- A distributed training approach, using Dask and a cloud infrastructure, guarantees that we could scale up the training several models with different approaches.

As next steps, it is expected to validate the components with the initial scenarios from the pilots use cases. Datasets provided by the pilots will be used to support this task and to improve the current developed features in both components.

7 REFERENCES AND ACRONYMS

References

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

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

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

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

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

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

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

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

[9] 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.

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

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

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

7.1 ACRONYMS

Acronym List	
API	Application Programming Interface
CP	Consortium Plenary
CSV	Comma Separated Values
DL	Deep Learning
ETD	Estimated Time of Departure
HDFS	Hadoop Distributed File System
IDS	International Data Spaces
KPI	Key Performance Indicator
LSTM	Long-short Term Memory
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
PC	Project Coordinator
PCS	Port Community System
PMB	Project Management Board
PPR	Project Periodic Report
QM	Quality Management
RM	Risk Management
RL	Reinforcement Learning
SLA	Service Level Agreement
TM	Technical Manager
TOS	Terminal operating system
UDE	University of Duisburg-Essen
WPL	Work Packages Leaders

Table 6 – Acronyms