

Project Acronym: STAR
Grant Agreement number: 956573 (H2020-ICT-2020-1 – Research and Innovation Action)
Project Full Title: Safe and Trusted Human Centric Artificial Intelligence in Future Manufacturing Lines
Project Coordinator: INTRASOFT International



Funded by the Horizon 2020
Framework Programme of the
European Union

DELIVERABLE

D5.3 – Digital Twins for Security and Safety – Initial version

Dissemination level	PU -Public
Type of Document	Report
Contractual date of delivery	31/03/2022
Deliverable Leader	SUPSI
Status - version, date	Final version – 31/03/2022
WP / Task responsible	WP5
Keywords:	Human digital models, human digital twin, Digital twin

Executive Summary

This document provides an overview of the activities and the results achieved within M5-M15 of Task 5.2 - Digital Twins for Worker Safety. The goal of T5.2 is to develop is to facilitate the human-machine collaboration, considering, understanding and anticipating the humans in the production system.

The main activities carried out during the T5.2 early stage are: 1) Literature review on exertion fatigue assessment (especially for workers); 2) selection of wearable devices review for collecting physiological data; 3) Definition of functional/non-functional specifications of the Fatigue Monitoring System (FaMS); 4) Architecture design; 5) Prototype implementation.

The main result of these first 11 months of the task is the prototype of the FaMS. After the introduction, the document provides an in-depth description of the software components, which compose the FaMS, responsible for the recognition of the physical fatigue exertion. The descriptions highlight implementation details and working logic. The FaMS takes in input the worker's physiological data from the devices enriched with static information, to compute physical fatigue exertion.

Deliverable Leaders:	SUPSI
Contributors:	RuG
Reviewers:	INTRA
Approved by:	Charalampos Ipektsidis, John Soldatos (INTRA)

Document History			
Version	Date	Contributor(s)	Description
V0.1	01/02/2022	SUPSI	Table of contents
V0.2	10/03/2022	SUPSI	First draft (section 1-3)
V0.3	17/03/2022	SUPSI	Second draft (section 1-5)
V0.6	29/03/2022	RuG	Christos Emmanouilidis (RuG) contributions
V0.7	31/03/2022	SUPSI	Updated version incorporating reviewer comments
V1.0	31/03/2022	INTRA	Final QA'ed version

Table of Contents

EXECUTIVE SUMMARY	2
TABLE OF CONTENTS.....	4
TABLE OF FIGURES.....	6
LIST OF TABLES.....	7
DEFINITIONS, ACRONYMS AND ABBREVIATIONS	8
1 INTRODUCTION.....	9
2 HUMAN FACTORS, FATIGUE AND STRESS IN THE INDUSTRY 4.0 ERA.....	10
2.1 PERCEIVED PHYSICAL EXERTION MEASUREMENT IN STAR	10
3 PERCEIVED EXERTION PREDICTION WITH FAMS	13
3.1 FAMS REQUIREMENTS	13
3.1.1 <i>Functional requirements.....</i>	<i>14</i>
3.1.2 <i>Non-functional requirements</i>	<i>14</i>
3.2 FAMS DESIGN.....	15
3.2.1 <i>Connected Workers Manager.....</i>	<i>15</i>
3.2.2 <i>Static Data Manager</i>	<i>16</i>
3.2.3 <i>Dynamic Data Manager.....</i>	<i>16</i>
3.2.4 <i>Classifier Executor.....</i>	<i>16</i>
3.2.5 <i>Multiclass Classifier.....</i>	<i>17</i>
3.2.6 <i>Result Converter</i>	<i>18</i>
3.2.7 <i>Stream Connector</i>	<i>18</i>
3.3 TECHNOLOGIES	18
3.3.1 <i>Machine Learning Framework.....</i>	<i>18</i>
3.3.2 <i>FaMS design language.....</i>	<i>25</i>
3.3.3 <i>Library for middleware connection.....</i>	<i>25</i>
3.3.4 <i>Deployment Technology.....</i>	<i>26</i>
3.4 WEARABLE DEVICES.....	26
4 MODELS SELECTION, TRAINING AND VALIDATION	29
4.1 ALGORITHM FOR FATIGUE PREDICTION	29
4.1.1 <i>Algorithms comparison.....</i>	<i>29</i>
4.1.2 <i>Selection of the machine learning algorithm</i>	<i>33</i>
4.2 FEATURE SELECTION AND MODEL TRAINING FOR PHYSICAL EFFORT PREDICTION	33
4.2.1 <i>Step 1: Preparing Static Data.....</i>	<i>34</i>
4.2.2 <i>Step 2: Preparing data regarding workers' perceived physical exertion</i>	<i>35</i>
4.2.3 <i>Step 3: Preparation and fusion of workers' dynamic data with perceived exertion data</i>	<i>36</i>
4.2.4 <i>Step 4: Data windowing, merging with static data and computation of additional</i>	<i>37</i>
<i>information.....</i>	<i>37</i>
4.2.5 <i>Step 5: Feature selection for training the machine learning model</i>	<i>39</i>
4.2.6 <i>Step 6: Training, validating and saving the machine learning model.....</i>	<i>42</i>
5 WORKER STREAMING DATA EVENT DETECTION	44
5.1 POSING THE EVENT DETECTION PROBLEM	44
5.2 CANDIDATE METHODS	44
6 CONCLUSIONS.....	46

REFERENCES 47

Table of Figures

FIGURE 1 FAMS - HDT CORE INFRASTRUCTURE INTEGRATION	13
FIGURE 2 FAMS GENERIC FLOWCHART	14
FIGURE 3 FAMS HIGH-LEVEL ARCHITECTURE COMPONENT DIAGRAM	15
FIGURE 4 AHP RESULTS	28
FIGURE 5 WEARABLES CLASSIFICATION (PARTIAL LIST)	28
FIGURE 6 CONFUSION MATRIX AFTER THE TRAINING AND VALIDATION PHASE	43

List of Tables

TABLE 1 BORG RPE SCALE	10
TABLE 2 BORG CR-10 SCALE.....	11
TABLE 3 SELECTED MACHINE LEARNING FRAMEWORKS.....	20
TABLE 4 ALGORITHMS COMPARISON	30
TABLE 5 STATIC DATA REGARDING WORKERS	34
TABLE 6 DATA RELATED TO THE PHYSICAL EFFORT PERCEIVED BY THE WORKERS DURING THE EXECUTION OF A TASK .	35
TABLE 7 WORKERS' PHYSIOLOGICAL DATA FROM SENSORS COMBINED WITH PHYSICAL EXERTION DATA.....	36
TABLE 8 STATIC, DYNAMIC DATA AND WORKERS' PERCEIVED EFFORT	37

Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
AI	Artificial Intelligence
CPS	Cyber-Physical Systems
FaMS	Fatigue Monitoring System
HDT	Human Digital Twin
WP	Work Package

1 Introduction

This document describes the work carried out in Task 5.2, which targets STAR's **Objective 4 - Human-centred simulations and digital twins for safe AI systems in manufacturing**. The goal of this task is to facilitate human-machine collaboration, considering, understanding and anticipating the humans in the production system. The task develops an Artificial Intelligence (AI) module capable to estimate the perceived fatigue exertion of the workers, by exploiting quasi-static data about the workers (e.g., age, height, weight), as well as dynamic data collected by wearable devices (e.g., heart rate).

The document sums up the activities carried out in the first 11 months of the task and it is structured in 6 sections. Section 1 provides an overview of the scope and the structure of this document. Section 2 includes a literature review on the concept of fatigue and stress in the Industry 4.0 era, together with an overview of the main applications realised so far. Section 3 introduces the module developed in T5.2, the Fatigue Monitoring System (FaMS), along with its requirements, design, and the technologies used for its implementation. The FaMS is one of the AI modules developed in WP5. Section 4 describes the adopted approaches for the training and selection of the AI models on which FaMS relies on. Section 5 provides an overview on the activities that will be carried out in the second part of the task dedicated to event detection. Finally, section 6 gives an overview of the next steps planned in the second part of Task 5.2, which will start at M16.

2 Human factors, fatigue and stress in the Industry 4.0 era

Recently, the adoption of Digital Twins (DTs) for workers’ well-being and health monitoring is getting attention [REF-01]. Different researchers provided a definition of DT applied for well-being monitoring:

- DTs enable the monitoring, understanding, and optimization of all functioning of humans, and provide constant health insight to improve quality of life and well-being [REF-02].
- Personal DTs are data-driven solution that depicts individuals’ health status based on regularly collected health parameters [REF-03].
- A DT is a simulation technology that delivers digital health insights while also allowing prediction and recommendation within a feedback loop [REF-04].

In literature, different types of fatigue have been defined:

- **Physical fatigue** involves the inability to maintain physical performance. It may be associated with reduced strength capacity, changes in motor control, and reduced proprioception. The physical fatigue could be responsible for decreased quality and productivity.
- **Cognitive fatigue** involves a reduction in information processing abilities due to mental workload. It leads to decreased competence, productivity, and error avoidance resulting from disengagement and low alertness.
- **Visual fatigue** results in decreased visual and perceptual performance and increased visual discomfort. It can lead to lower performance due to oculomotor strain and the effort involved in accommodation and convergence.

It is not easy to assign the perceived fatigue to only one of the three considered fatigue types, given the considerable overlap between them. It has been shown that the physical ability can be negatively affected by the mental demand. Moreover, workers may incorrectly interpret the visual fatigue as cognitive fatigue.

2.1 Perceived physical exertion measurement in STAR

A possible strategy for detecting workers' perceived physical exertion is to measure the physical exertion with the Borg RPE Scale, commonly used to assess the perceived exertion during physical activities and exercises. However, its use can be extended to assess the perceived fatigue during any type of physical activity. The values of this scale are enclosed in the range [6, 20], where 6 indicates no perceived exertion, and 20 represents the maximum exertion. Table 1 provides details about the Borg Scale, with notes to give a grounding reference to the reader.

Table 1 Borg RPE Scale

#	Level of Exertion	Instructions / Explanation
6	No exertion at all	
7		
7.5	Extremely Light	
8		

9	Very Light	Corresponds to a "very light" exercise. For a healthy person, it is like walking slowly at his or her own pace for some minutes
10		
11	Light	
12		
13	Somewhat hard	13 on the scale is "somewhat hard" exercise, but it still feels OK to continue.
14		
15		
16		
17	Hard (heavy)	17 "very hard" is very strenuous. A healthy person can still go on, but he or she really has to push him- or herself. It feels very heavy, and the person is very tired.
18		
19	Extremely Hard	19 on the scale is an extremely strenuous exercise level. For most people, this is the most strenuous exercise they have ever experienced.
20	Maximal Exertion	

In the STAR project, the modified version of the original scale has been adopted, i.e., the Borg CR-10 scale. In this version, the original range is projected in the interval [0, 10], where 0 indicates zero perceived exertion, and 10 represents the maximum exertion level. It is worth noting that in this new version, the spacing between stress levels is not linear. Details about the Borg CR-10 Scale are provided in Table 2.

Table 2 Borg CR-10 Scale

#	Level of Exertion	Instructions / Explanation
0	Nothing at all	
0.3		
0.5	Extremely Weak	Effort is just noticeable
1	Very Weak	
1.5		
2	Weak	Light exercise
2.5		
3	Moderate	
4		
5	Strong	Heavy Exercise
6		
7	Very Strong	
8		
9		
10	Extremely Strong	"Maximal" affordable exercise
.	Absolute Maximum	Highest Possible

From previous projects, namely COMPLEMENT and HUMAN (H2020 program), the Borg CR-10 Scale results are easier to understand for users, given its range. The Borg scale allows getting an approximate heart rate, a good indicator for predicting the perceived exertion of workers. Indeed, there appears to be a strong link between heart rate and aerobic capacity (VO2 max), an excellent indicator of perceived exertion, which however is difficult to measure in the working environment (it requires an invasive procedure to be measured). Since the heart rate is subject to large variations depending on other factors (e.g., age, physical conditions, environment), using it as the only feature to estimate the perceived exertion of workers may lead to unreliable results. To overcome this problem, within the STAR project additional physiological signals are collected from workers with wearable devices to derive more accurate estimates of perceived fatigue.

3 Perceived exertion prediction with FaMS

Within the STAR project, a dedicated module, i.e., the Fatigue Monitoring System (FaMS) has been developed to predict the perceived physical exertion of workers, given their physiological parameters. The FaMS predicts the worker’s physical exertion on the Borg CR-10 Scale, and it helps in monitoring the overall safety level on the factory. More in details, the FaMS is connected with the STAR HDT Core Infrastructure with a two-way connector, as depicted in Figure 1: physiological data are read from the HDT middleware, while the estimated perceived physical exertion is sent back to the middleware.

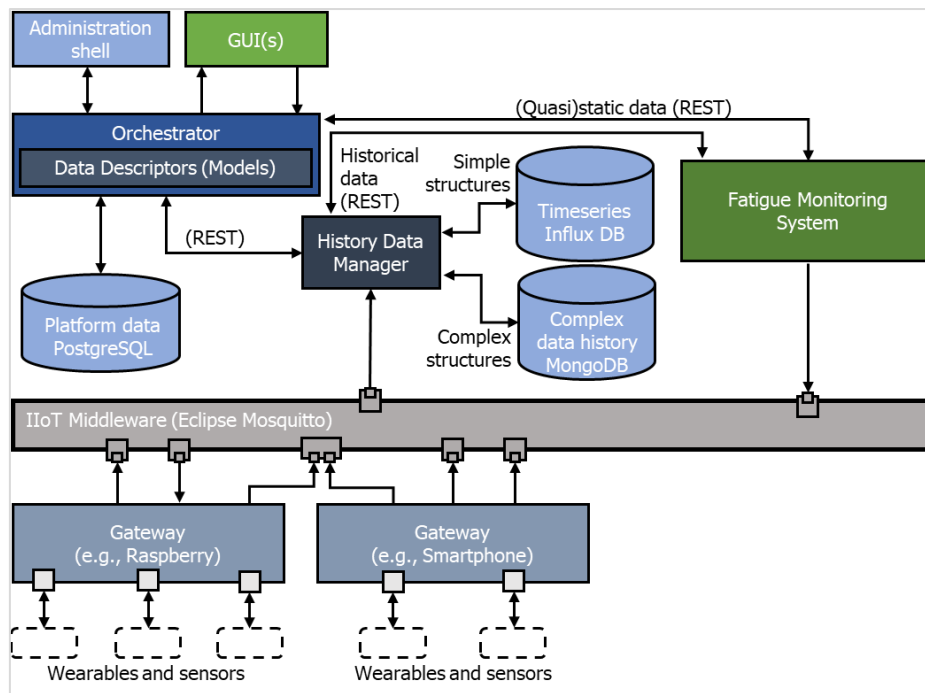


Figure 1 FaMS - HDT Core Infrastructure integration

In the following sections, an in-depth discussion is provided about FaMS, its requirements and functionalities.

3.1 FaMS requirements

This section outlines the functional and non-functional requirements of FaMS.

As depicted in Figure 2, the FaMS is a component that estimates the perceived exertion of workers based on *static data*¹ (e.g., age, weight, height, hiring date, maximum heart rate, % body fat, sex, working experience, grip strength, etc.), and *dynamic data* (e.g., current heart rate, skin temperature, galvanic skin response, etc.) of the worker. Static data are collected by questionnaires, while dynamic data are continuously measured with sensors attached to workers and/or the surrounding environment. In STAR, this activity is tackled by the HDT Core Infrastructure, thus the FaMS component has only to retrieve the two sets of static and dynamic data associated with the worker whose perceived exertion has to be estimated. The two datasets are then merged in a single dataset, where the real-time

¹ As per *static data* we refer to worker parameters that do not change over time (e.g., birthday), or change with a very low frequency (e.g., weight, skills).

measurements from sensors are enriched by contextual information. The datasets are then used to feed an AI prediction model that returns the degree of perceived worker’s fatigue according to the Borg CR-10 Scale.

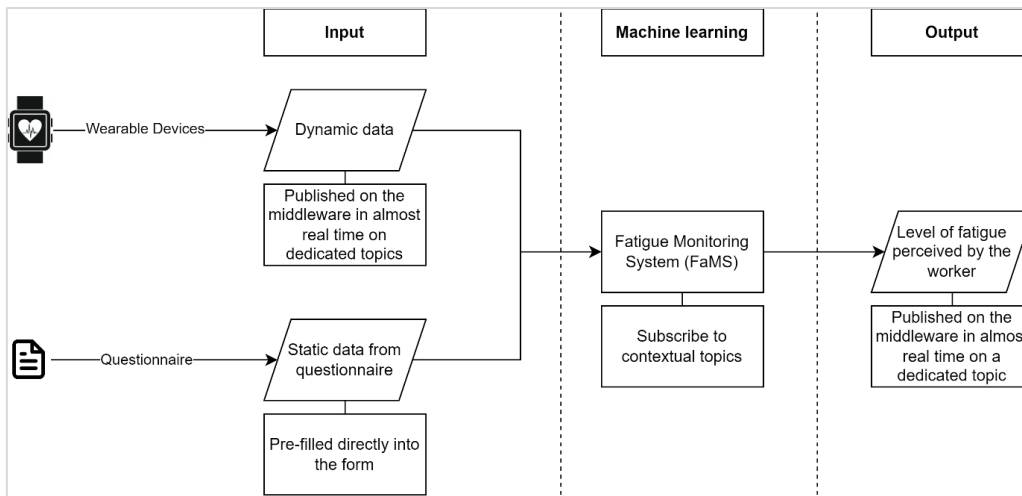


Figure 2 FaMS generic flowchart

The functionalities that the FaMS must implement are presented in the following two sections, divided between functional and non-functional requirements.

3.1.1 Functional requirements

The functional requirements of the FaMS are outlined below.

1. Communicate with the HDT through its REST API to retrieve static data about workers (i.e., CharacteristicValues);
2. Communicate with the HDT middleware (i.e., Mosquitto MQTT broker) to retrieve dynamic data (data streams from sensors);
3. Communicate with the HDT middleware (i.e., Mosquitto MQTT broker) to send the computed exertions (i.e., StateDescriptors);
4. Communicate with the HDT through its REST API to retrieve historical dynamic data from the HDM;
5. To consider both dynamic and static data to compute the perceived exertion;
6. Pre-process the incoming data to obtain relevant features (e.g., how to manage measurements with different timestamps);
7. Classify/identify the level of perceived fatigue exertion in the Borg CR-10 Scale using the input features;
8. To manage different types of classifiers to obtain a more reliable/accurate output;
9. To support labelling of new datasets (even autonomously by the worker) not only for training, but also for validation and continuous learning;
10. The FaMS has to provide To handle multiple workers simultaneously;
11. To interface with different types of brokers.

3.1.2 Non-functional requirements

The non-functional requirements of the FaMS are outlined below.

1. The FaMS shall rely on a single source of data, i.e., the HDT;
2. The FaMS shall send an output to the HDT broker with a frequency of at least x minutes (with x a configurable parameter of the module);

3. The FaMS parameters shall be configured via static files (e.g., JSON files).

3.2 FaMS design

To achieve high levels of scalability and flexibility, the FaMS high-level architecture has been designed as in Figure 3. Each component is decoupled from the other as much as possible, making it possible to change/customise a component with near-zero impact over the other. Moreover, the current architecture flexibility makes the FaMS easier to be adapted as the STAR project evolves. FaMS components are described in detail in the following sections.

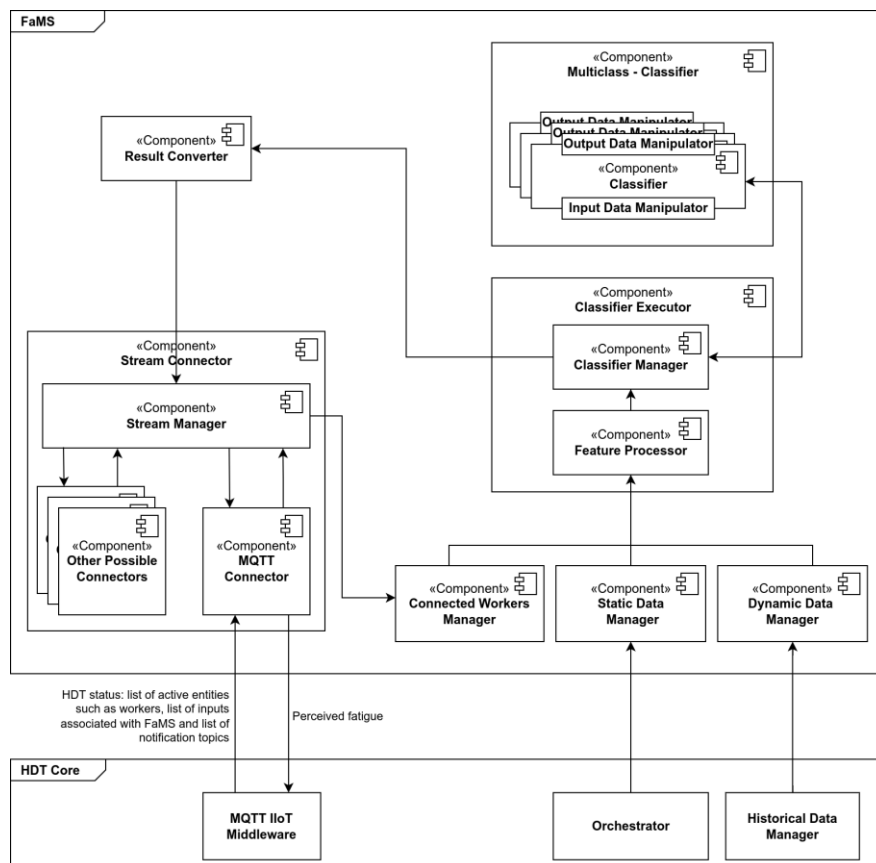


Figure 3 FaMS High-Level Architecture Component Diagram

3.2.1 Connected Workers Manager

This component is responsible for finding how many and which workers are currently connected to the HDT, i.e., which workers are working in the factory. In this way, the FaMS is able to retrieve only the information regarding active workers and then calculate their perceived fatigue exertion. The FaMS interacts with the HDT Orchestrator to get updates about the factory status. The interaction happens both via HTTP requests (using the Orchestrator REST API), or by subscribing specific notification topics on the IIoT middleware (where the Orchestrator publishes specific messages). For example, the Orchestrator publishes the status of active workers on the notification topic every time a worker starts a new working session. The Orchestrator publishes three possible states for workers:

- **"ACTIVE"**: the worker is connected to the system and should therefore be added to the list of active workers;
- **"INACTIVE"**: the worker has disconnected from the system and must be removed from the list of active workers;

- **"UPDATED"**: the static data about the worker has been updated, thus the FaMS must use the new data for future predictions.

Since the worker status is published on the IIoT middleware, the Connected Workers Manager relies on the Stream Manager (described in Section 3.2.2), to register a listener for notification events about workers. Whenever an event is published, the Connected Workers Manager adds it to an internal event queue to process events while maintaining the order of arrival.

The Connected Workers Manager also has the task of retrieving and updating the static information of workers. To complete this task, it uses the Static Data Manager component: when a new status is published for a worker, the Connected Workers Manager notifies the event to the Static Data Manager, which will retrieve/delete /update the static data based on the specific event type.

3.2.2 Static Data Manager

This component is responsible for retrieving static data about workers from the HDT. When the FaMS needs to know the static information about a worker, it delegates the task of retrieving these data to the Static Data Manager. The component retrieves data by directly interacting with the Orchestrator component via REST API. Static information is then retrieved and persisted in a local data structure to avoid duplicated requests to the Orchestrator. The local data structure is updated whenever asked by the Connected Workers Manager.

3.2.3 Dynamic Data Manager

This component is responsible for retrieving historical dynamic data about workers. When the FaMS needs historical dynamic data such as the physiological data from sensors, it relies on the Dynamic Data Manager. Thanks to this component, it is possible to retrieve the dynamic data associated with a worker in a specific period of time (from timestamp S1 to timestamp S2). This component interacts with the Historical Data Manager (HDM) component, available in the HDT Core infrastructure, via REST API.

3.2.4 Classifier Executor

This component has the task of executing the classifier by feeding them with data from the Static Data Manager and Dynamic Data Manager. Before triggering the exertion estimation, the data undergo a preprocessing phase. The results coming from the classifiers are then forwarded to the Result Converter component (see Section 3.2.6 for details).

3.2.4.1 Features Processor

This component is responsible for preparing the data to be used by the Classifier Manager, in charge of the computation and estimation task. The operations performed by this component can be divided into two main phases:

- **Data merging where static data are coupled with dynamic data and processed to derive additional and enriching information:** Static and dynamic data are prepared and structured in order to merge them into a single dataset: some features are renamed, dynamic data from various sensors are temporally aligned (they are collected with different frequencies) and then a windowing process is performed on them. Further additional and enriching features are obtained (e.g., the Body Mass Index is computed from the weight and height of the workers; the time

elapsed since the start of the shift is computed by subtracting the starting shift time from the current observation timestamp). This step is of paramount importance as it provides context to the dynamic data. For example, a high value of heart rate could be considered alarming if no contextual information is given. Without having other context information. The same value, if accompanied with extra information, may be evaluated not critical (e.g., because the worker just finished to complete a heavy task). At the end of this process, the dataset contains a list of *observations*.

- **Data preparation and adaptation for the multiple classifiers used for the calculation of the exertion:** the merged dataset may require additional transformations depending on the type of classifier that will be used to compute the fatigue. For example, some classifiers cannot work with string data and require to encode the values into an appropriate format.

3.2.4.2 Classifier Manager

This component is responsible for managing the classifiers that can be executed by the FaMS. The current release of FaMS comes with 3 pre-trained models, which are Random Forest classifiers trained on a single dataset, but with different hyper-parameters (e.g., number of trees and features) and preprocessing strategies (e.g., different windowing and binning strategies). The list of classifiers to consider is a parameter of the FaMS and can be set during the configuration phase. Once the classifiers are instantiated, the component runs the prediction of perceived fatigue exertion for all the observations given as input. The final value is obtained by aggregating the results for all the observations (the aggregation function can be configured, and it defaults to LAST, i.e., only the last observation is considered).

The Classifier Manager runs the fatigue computation by assigning a thread for each instantiated classifier. The actual value of the calculated fatigue is obtained by performing the statistical mode of the results provided by the classifiers: in this way a majority vote is performed, which allows the selection of the most frequent value among the results. This strategy helps in mitigating possible errors produced by classifiers).

3.2.5 Multiclass Classifier

This component deals with predicting the fatigue perceived by a worker, relying on machine learning models.

The main task of the FaMS is to estimate the degree of fatigue exertion perceived by a worker. The perceived fatigue is identified by a value on the Borg CR-10 Scale, where 0 represents no perceived fatigue and 10 represents the maximum perceived fatigue. The FaMS addresses this problem as a classification problem, thus assigning a value to the eleven possible values (which are seen as classes) with multi-class classifiers.

The current version of FaMS comes with classifiers trained in previous projects, using datasets from real production environments. However, the next releases of FaMS will include a functionality to (re-)train models by using historical data from HDM as new training data. New models can then be saved and imported into the FaMS during the configuration phase: in this way it is possible to specify which trained models to use for fatigue exertion estimation.

3.2.6 Result Converter

This component is responsible for converting the results provided by the Classifier Manager into the format required by the HDT. Depending on the classifier, it may be necessary to add/remove information (e.g., some classifiers provide the "degree of confidence" of their predictions). The specific transformations applied by this component depend on the selected classifiers, as well as the specific machine learning framework used. Finally, this component is also in charge of sending error messages to the stream connector in case of missing results.

3.2.7 Stream Connector

This component is responsible for managing data streams from/to the HDT. It is worth noting that the Stream Connector decouples the FaMS from the actual technology used to implement the IIoT middleware, thus providing a great flexibility. The Stream Connector is responsible for both retrieving and publishing data from/to the IIoT middleware.

The current FaMS release includes an MQTT connector to support the Eclipse Mosquitto MQTT broker, the IIoT middleware used in the STAR HDT Core infrastructure. However, new connectors will be released in the future to support other brokers (e.g., Apache Kafka or FIWARE Orion Context Broker).

3.3 Technologies

This section discusses the technologies used to develop and implement the FaMS. Each of them will be presented and the reasons for their choice is given.

3.3.1 Machine Learning Framework

In this section, the analysis that identified Scikit-learn as the most appropriate machine learning framework for the current project is addressed.

3.3.1.1 Selection Criteria

At this point, the machine learning frameworks to estimate the fatigue are presented and evaluated. The main evaluation criteria included in the assessment are:

- **License:** It is possible to distinguish between Open Source and Commercial software.

A license is Open Source if it allows the software to be used, modified and shared freely. There are two main types of Open-Source licenses: the Copyleft license and the Permissive license. The Copyleft license allows code to be modified and new work based on it to be distributed, as long as any new work or adaptation is distributed under the same software license. The Permissive License contains minimum requirements for how the software may be modified or redistributed. It can simply be defined as a non-Copyleft Open-Source license. An Open-Source software license has the advantage of being adaptable to the specific needs of the project. In fact, if some features are missing (for example, certain algorithms) it is possible to implement them and add them to the software. However, there is a risk of poor support and consequently, the project may suffer.

The use of a commercial license must be evaluated in terms of costs and benefits, both in the development and distribution phases. The end user must agree to pay the cost of the commercial software.

To avoid exploitation problems, it is therefore preferable to use a Permissive Open-Source license.

- **Algorithms:** since at present no single algorithm has yet been identified that prevails over others in the task of predicting worker perceived fatigue from physiological data, the number of different machine learning algorithms that a framework has to offer is an important feature for its selection. This leaves the ability to quickly change algorithm types without having to change machine learning frameworks. Also, it should be considered that some machine learning algorithms need a preliminary data transformation before being used. Therefore, not only the variety of algorithms offered by a framework, but also the tools and features offered (e.g., data processing tools) should be taken into account.
- **Architectures:** the architectures supported by the framework can have direct consequences on efficiency and speed in systems using machine learning. In particular, the training phase of machine learning models is the most critical phase that takes a long time to complete. Some techniques, to speed up the training phase, exploit parallel computing. To achieve good levels of parallelisation, several methods can be used: using dedicated clusters, using resources available on the cloud, or using the GPU of a single server. Since the training phase is repeated multiple times to obtain a machine learning model capable of making quality predictions, it is necessary to take into account any factors that may make this process faster. Consequently, the architectures that the machine learning framework can support are an important factor to consider.

3.3.1.2 Frameworks

Table 3 shows the frameworks considered in the comparison.

Table 3 Selected machine learning frameworks

Framework	License	Architecture	Streaming	Languages	Algorithms
Matlab	Commercial	Standalone	No	Matlab	<p>Classification: Logistic Regression, Support Vector Machines, Neural Networks, Naïve Bayes Classifier, Decision Trees, Discriminant Analysis, Nearest Neighbors, Ensemble Classification, Generalized Additive Model</p> <p>Regression: Linear Regression, Nonlinear Regression, Generalized Linear Models, Decision Trees, Neural Networks, Gaussian Process Regression, Support Vector Machine Regression, Ensemble Regression</p> <p>No information available on feature transformation algorithms</p> <p>https://www.mathworks.com/discovery/supervised-learning.html</p>
Amazon ML	Commercial	Cloud	Yes	C++, Go, Java, Ruby	<p>The types of algorithms available are: Binary Classification, Multi-class Classification, Regression</p> <p>There are few details on the available algorithms on the web (e.g. Neural Networks, Bayesian Classifiers, Decision Trees). Offers feature preprocessing algorithms but concrete algorithms are not listed</p> <p>https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html</p>
Azure ML	Commercial	Cloud	Yes	R, Python	<p>Classification: Two-class Logistic Regression, Two-class Decision Forest, Two-class Boosted Decision tree, Two-class Neural Networks, Two-class Averaged Perceptron, Two-class Support Vector Machine, Multiclass Logistic Regression, Multiclass Decision Forest, Multiclass Boosted Decision Tree, Multiclass Neural Networks, One-vs-all Multiclass</p> <p>Regression: Linear Regression, Decision Forest Regression, Boosted Decision Tree Regression, Neural Networks Regression</p> <p>Clustering: K-means Clustering</p> <p>https://docs.microsoft.com/en-us/azure/machine-learning/how-to-select-algorithms</p>
OpenNN	LGPLv3	OpenMP, CUDA	No information available	C++	<p>Neural Networks, Deep Neural Networks</p> <p>https://www.opennn.net/</p>

WEKA	GPLv2	Standalone	To be developed	Java	Classification: Naive Bayes, Decision Tree, K-Nearest Neighbours, Support Vector Machines, Neural Networks, Random Forest Regression: Logistic Regression e Linear Regression https://weka.sourceforge.io/packageMetaData/
Keras over Tensorflow	MIT	Standalone, TPU	Not officially	C++, Python	State of the Art for Neural Networks and Deep Neural Networks https://keras.io/ https://www.tensorflow.org/
Spark ML	Apache	Standalone, cluster	Yes	Scala, Java, Python, R	Classification: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Gradient-boosted Tree classifier, Multilayer Perceptron Classifier, Linear Support Vector Machine, One-vs-Rest Classifier, Naive Bayes, Factorization Machines Classifier Regression: Linear Regression, Generalized Linear Regression, Decision Tree Regression, Random Forest regression, Gradient-boosted Tree Regression, Survival Regression, Isotonic Regression, Factorization Machines Regressor https://spark.apache.org/docs/latest/ml-classification-regression.html
Scikit-learn	Open Source, commercially usable - BSD license	Standalone	No	Python	Scikit-learn offers a large variety of machine learning and feature processing algorithms, the main ones are listed below: Support Vector Machines, Nearest Neighbors, Generalised Linear Models, Naive Bayes, Decision Trees, Stochastic Gradient Descent, Neural Networks Models (supervised), Multiclass and Multilabel algorithms, Clustering, Biclustering, Manifold learning, Gaussian Mixture models, Novelty and Outlier Detection, Neural Networks Models (unsupervised) https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html https://scikit-learn.org/stable/supervised_learning.html https://scikit-learn.org/stable/unsupervised_learning.html

After initially reviewing all the 8 frameworks listed above, only WEKA, Keras over Tensorflow, Spark ML, and Scikit-learn have been analysed in detail. OpenNN has not been taken into consideration because it has an offer of machine learning algorithms mainly limited to Neural Networks and Deep Neural Networks. Although these are excellent tools, the range offered is too limited.

The main reason behind the choice of these frameworks is the avoidance of commercial solutions. Open-Source licensed software is preferable to commercially licensed software because it offers more support from developers' communities working on Open-Source projects. This allows leveraging quick responses to recurring problems to develop robust solutions during project implementation. After these considerations, Matlab, Amazon ML, and Azure ML have been excluded from the analysis. Nevertheless, the possibility to switch to a Commercial license is always valid.

3.3.1.2.1 WEKA

WEKA is a collection of machine learning algorithms used for data mining tasks. These algorithms can be directly on the WEKA's platform, or they can be called from a Java implementation.

Advantages:

- To offer a wide variety of machine learning algorithms such as Naive Bayes, Decision Tree, K-Nearest Neighbours, Support Vector Machines, Neural Networks, Random Forests.
- To offer data pre-processing, association rules and visualisation tools.
- To be a framework widely used in the scientific community, ensuring the accuracy of algorithms.

Disadvantages:

- To have a GPLv2 Copyleft license. Consequently, for a code developed with this framework, it is required to be released with at least the GPLv2 license.
- To not have a native streaming component.

3.3.1.2.2 Keras over Tensorflow

Keras is a Python open-source library for machine learning. Keras offers an abstraction layer to simplify the creation of neural networks without dealing with the underlying low-level library. Indeed Keras exploited different (and customisable) backend libraries for machine learning, including TensorFlow, Microsoft Cognitive Toolkit (CNTK), and Theano. TensorFlow is a free and open source software library for machine learning. It is mainly used for training neural networks on CPUs or GPUs (thanks to its integration with CUDA or OpenCL libraries). The software is licenced under the Apache 2.0 Licence, and its flexible architecture enables an easy deployment on a variety of platforms (clusters, mobile devices, edge devices).

Advantages:

- To have a deployment on a very large set of devices (e.g. desktop computers, clusters, servers) through the use of a single API. This makes it easy to deploy the test environment in a distribution environment.
- To represent the state of the art in the field of Neural Networks and Deep Neural Networks.

- To have an easy integration in any kind of commercial product thanks to the MIT license of Keras and the Apache 2.0 license of Tensorflow.

Disadvantages:

- To offer machine learning algorithms primarily limited to Neural Networks and Deep Neural Networks. However, this limitation can be overcome thanks to the possibility of implementing customized algorithms.
- To not have an integrated component that manages data streaming.

3.3.1.2.3 Spark ML

Apache Spark is a unified analytics engine for large-scale data processing. It provides high-level APIs in Java, Scala, Python and R, as well as an optimised engine that supports general purpose execution graphs. It also features a considerable set of tools such as: Spark SQL for SQL and structured data processing, MLlib for machine learning operations, GraphX for graph processing, and Structured streaming for incremental computation and stream processing.

Spark ML was developed on top of Apache Spark. It is an open source cluster computing system that allows speeding up the data analysis phase. To do this Spark allows loading data into memory and querying it repeatedly faster than in disk-based systems by leveraging the primitives for in-memory cluster computing. Its high-level APIs allow distributed data sets to be quickly manipulated as local collections. In addition, through Scala or Python shells, it can be used to interactively query big data. MLlib is Spark's machine learning library. Its goal is to make practical machine learning easy and scalable. At a high level, it provides the following tools:

- **ML algorithms:** common machine learning algorithms such as classification, regression, clustering and collaborative filtering.
- **Feature processing:** feature extraction, transformation, dimensionality reduction and selection.
- **Pipelines:** tools to build, evaluate and tune machine learning pipelines.
- **Persistence:** save and load algorithms, models and pipelines.
- **Utilities:** linear algebra, statistics, data management, etc.

Advantages:

- To have native integration with Apache Kafka, that in future projects could be chosen as middleware.
- To have the possibility to manage data streaming.
- To query data directly from Scala or Python shells.
- To persist the machine learning models developed during the training phase on disk and then reuse them in the production code by exporting them easily.
- To provide excellent performance in processing large amounts of data in a distributed environment.

Disadvantages:

- To have a scarce supply of machine learning algorithms for Deep Neural Networks, Convolutional Neural Networks and Neural Networks.

3.3.1.2.4 Scikit-learn

Scikit-learn is an Open-Source library for use in the field of machine learning in Python. It has an exhaustive collection of the most efficient tools for statistical modelling and machine learning. It has various algorithms for: classification, regression, clustering, Support Vector Machines, Random Forests, Gradient Boosting, K-means and DBSCAN. In addition, it is designed to interact with Python's NumPy and SciPy numerical and scientific libraries. The library uses a unified and consistent Python interface to implement various pre-processing, machine learning, visualisation, and cross-validation algorithms. Scikit-learn has many features for machine learning, including:

- **Supervised learning algorithms:** the collection of algorithms includes most of the well-known supervised machine learning algorithms such as linear regression, Decision Trees, Support Vector Machines and Bayesian methods.
- **Unsupervised learning algorithms:** the collection of algorithms includes factoring, cluster analysis, principal component analysis and Unsupervised Neural Networks.
- **Feature extraction:** features can be extracted from text and images
- **Cross-validation:** the accuracy and validity of supervised models on never-before-seen data can be checked with the help of Scikit-learn
- **Dimensionality reduction:** With this feature, the number of attributes in the data can be reduced for subsequent visualisation, synthesis and selection of features.
- **Clustering:** this feature allows the clustering of unlabeled data
- **Ensemble methods:** predictions from different supervised models can be combined using this feature

Advantages:

- To have a library distributed under a BSD license and therefore legal and licensing restrictions are minimal.
- To be able to persist the machine learning models developed in the training phase on disk and then reuse them in production code by exporting them easily.
- To have excellent performance if data can be stored in RAM.
- To possess a fast learning curve for use.
- To be constantly updated and supported by several authors, contributors and a large international community.
- To have a website that provides elaborate API documentation for users who want to integrate the algorithms with their platforms.

Disadvantages:

- To have a poor offering of machine learning algorithms with respect to Deep Learning.
- To not have an integrated component that handles streaming data.

3.3.1.2.5 Framework selection

All four frameworks have advantages and disadvantages that affect the selection of the ideal candidate for the STAR project. Considering the four main criteria to evaluate the frameworks, the importance of working with tools that have an Open-Source license and a wide range of appropriate machine learning algorithms stands out. Last but not least, the data processing performance also has its influence. In addition to these decisive factors,

experience with a given framework also played a role: it helps to save a lot of time in the project's initial phases and obtain higher quality results in the short time available.

Considering all the factors just exposed, the choice has fallen on Scikit-learn, which is characterised by:

- A BSD Open-Source license;
- A good offer of machine learning algorithms constantly updated and good tools for the pre-processing of features;
- A large community of international collaborators;
- A detailed documentation;
- Excellent performance if data are contained in RAM.

The two negative aspects of this framework are:

- The scarce current offer of Deep Learning algorithms;
- The lacking of an integrated component that manages the data streaming.

3.3.2 FaMS design language

Since the machine learning part was developed using Scikit-learn library in Python, it was found interesting to develop the whole FaMS with the same language. In fact, after the training phase of the machine learning model in charge of estimating the perceived fatigue exertion of the workers, Scikit-learn allows to export the trained model; since the FaMS is developed in Python, the model can be easily imported into the FaMS without the need to go through a wrapper. Moreover, since the HDT Core Infrastructure integrates MQTT Eclipse Mosquitto (See D5.1), Python is a worthy choice since the Paho Python library allows to easily integrate MQTT.

A critical point regarding this language concerns its execution speed. The program is based only on a core and multithreading is present only at the abstract level. However, for this application, it is not critical.

3.3.3 Library for middleware connection

To integrate the FaMS with the HDT Core Infrastructure, the Paho Python library² released under EDL/EPL license has been used.

This library provides a client class with support for MQTT v5.0, MQTT v3.1.1, and v3.1 on Python 2.7 or 3.x. It provides some support features to make the operations of publishing single messages to an MQTT server extremely easy.

The main features supported by Paho Python are listed below:

- MQTT 3.1
- MQTT 3.1.1
- MQTT 5.0
- LWT
- SSL / TLS
- Automatic Reconnect
- Offline Buffering

² <https://www.eclipse.org/paho/index.php?page=clients/python/index.php>

- WebSocket Support
- Standard TCP Support
- Non-Blocking API
- Blocking API
- QoS 0
- QoS 1
- QoS 2
- Authentication

The ability to persist messages is a missing feature, but this is not a problem as it is possible to implement this functionality directly in the FaMS.

3.3.4 Deployment Technology

The technology chosen for the FaMS deployment is Docker. Docker is released under the Apache License 2.0. Docker is a set of Platform as a Service products that take advantage of virtualisation at the operating system level to deliver software in packages called containers. Containers, each of which contains its own software, libraries and configuration files, are isolated from each other. Communication between them is allowed through well-defined channels. Containers use fewer resources than virtual machines since each of them uses the services of a single operating system.

More in detail, Docker is able to package an application (including dependencies) in a virtual container that can run on any Linux, Windows or macOS machine. In this way, applications can run on-premises, in public and private clouds.

Given the lightweight nature of containers, several of them can run simultaneously on a single server or virtual machine. Moreover, it is possible to take advantage of kernel features to monitor the execution of containers.

3.4 Wearable devices

To select wearable devices to use within the STAR project, the following requirements have been identified:

- To collect physiological data of the worker to allow an AI-based module to estimate the perceived fatigue exertion;
- To transmit collected data from the device to an external platform in near-real-time;
- To comply with an industrial environment characterised by dust, vibrations and shocks;
- To be non-invasive and comfortable for workers, with no obstructions.

To select the devices, a technological database containing a list of 110 wearable devices currently available along with their features in the market has been defined. The scope of our database are smartwatches and smartbands belonging to commercial and medical segments only. To build the initial device list, we searched the Web for the terms "smartband" and "smartwatch", and we inspected results available in scientific articles, comparisons and rankings in blogs, and producers' websites.

The first selection out of this first list has been made by removing obsolete systems and those that do not fit the application requirements. The approach that has been followed implies the inclusion of the latest models for the selected devices, while obsolete models and the ones no longer available in the market have been excluded. The rationale behind this

choice is that usually new models come with more features, in some cases also fixing some bugs. Moreover, given that the application requires the development of specific connectors to allow the communication between the devices layer and the software layer, relying on out-of-commerce and obsolete devices has been considered not viable.

Given the above constraints, the initial list has been short listed to a second list of 89 devices. The resulting dataset includes a classification of all the device features, including:

- General characteristics: device type, wearing position, screen presence, internal memory size (if any), battery duration (estimated), battery duration (estimated) with active sensors (e.g., GPS), charging time, battery capacity, waterproof rating (IP or ATM), weight and price.
- Communication protocols: we checked the support for ANT+, Bluetooth Low Energy (BLE), WiFi. SDK availability: availability of an SDK to develop real-time application to transmit data directly from the device, or to communicate via a smartphone application, or other generic APIs enabling communication (e.g., via Web platforms). Physiological metrics: capabilities to measure features relevant to the fatigue estimation, namely electromyography (EMG), electrocardiogram (ECG / EKG), bioimpedance, heart rate (HR), heart rate variability (HRV), interval RR/intra-beat interval (IBI), peak to peak interval (PPI), skin temperature (ST), galvanic skin response (GSR) / electrodermal activity (EDA), blood pressure, oxygen saturation, volume of oxygen consumed per minute (VO2 max), Pulse Plethysmogram (PPG)/Pulse Blood Volume (BVP).
- Installed sensors: presence of specific sensors, including heart rate meter, SpO2 sensor, pedometer, accelerometer, gyroscope, magnetometer, barometer, GPS, altimeter, thermometer, microphone, compass, ambient light sensor, optical sensor, NFC.

Each device considered in the dataset has been analysed individually by integrating pieces of information from several sources: manufacturers' websites, technical data sheets, manuals and online articles. Each of these characteristics individually contributes to determining the adequacy of the device for the identified application. In this classification, some features are not filled because of the limited information available, especially for general devices. For example, some manufacturers do not release the complete specifications, or the same were not easy to find/access by the authors. However, the classification is sufficiently wide to cover the scope of this research. To make a step ahead in selecting the devices, the primary features have been identified, starting from the application requirements. These include screen presence, internal memory, weight, communication protocols (ANT+, BLE, Wi-Fi), data stream via device, and data stream via web-platform. Primary features have been classified, adopting the AHP method. 10 experts have been involved to define the weight of each primary feature. Results are summed up in the following figures.

Criteria		more important ?	Scale
A	B	A or B	(1-9)
Screen presence	Internal memory	B	3
	Weight	B	1
	ANT+	B	5
	BLE	B	7
	Wifi	B	3
	Real time	B	9
	Platform	B	7
Internal memory	Weight	A	3
	ANT+	B	5
	BLE	B	7
	Wifi	B	3
	Real time	B	7
	Platform	B	5
	Weight	B	5
Weight	ANT+	B	5
	BLE	B	7
	Wifi	B	3
	Real time	B	9

Table	Criterion	Comment	Weights	+/-
1	Screen presence		3.0%	0.9%
2	Internal memory		5.3%	2.6%
3	Weight		3.0%	1.1%
4	ANT+		8.2%	1.1%
5	BLE		26.6%	5.5%
6	Wifi		5.9%	1.3%
7	Data streaming		36.0%	14.6%
8	Platform		11.9%	4.9%
9		for 9&10 unprotect the input sheets and expand the question section ("+" in row 66)	0.0%	0.0%
#			0.0%	0.0%

Result	Eigenvalue	Lambda:	8.385	MRE:	33.7%
Consistency Ratio	0.37	GC:	0.14	Psi:	8.3%
		CR:	3.9%		

Figure 4 AHP results

Binary features like screen presence, communication protocols (ANT+, BLE, Wi-Fi), data stream via device, data stream via web-platform are simple to assess (1 if present, 0 otherwise). For weight and memory, a normalisation from 0 to 1 has been adopted, where 0 corresponds to the worst device (e.g., device with no internal memory) and 1 to the best device (e.g., device with the largest memory).

By sorting the various devices in descending order by the value of the score, a ranking of the best devices has been obtained on the basis of the primary features. From this classification, a third list of 9 devices (10%) has been identified. These 9 devices have been then evaluated according to the following secondary features: Positioning on the body, measurement of at least 4 metrics, internal memory, battery life. All the 9 devices have been carefully evaluated according to these features. This assessment allowed to select 6 devices: Garmin Venu 2, Garmin Instinct Solar Tactical Edition, Polar H10, Garmin Vivosmart 4, Empatica E4, Samsung Galaxy Watch 3. At this point it was necessary to carry out an even more in-depth analysis of each of the devices chosen, considering, in particular, the details of the SDK and the format of the metrics tracked, plus any operations that are performed on the data and information aggregate. This was done in the implementation phase.

Brand	Model	Type	Position	Screen	Internal memory	Internal Storage [G]	Battery life [h]	Battery life GPS [h]	Charging Time [h]	Battery capacity [mAh]	IP level/Water rating	Weight [g]	Normalized weight	Price [Euro]	ANT+	Bluetooth Low Energy	WiFi	Device (data stream)	App	Web platform	EHR (Electrocardiogram)	ECG/ECG (Electrocardiogram)	Blood pressure	HRV (Heart Rate Variability)	HRV (Heart Rate)	RR Interval/HRV (Inter Beat Interval)	PPV (Peak to Peak Interval)	Skin Temperature (ST)	GSR (Galvanic Skin Response)	Blood pressure	Pulse OX Blood Oxygen Saturation	VO2 max	PPG (Pulse Plethysmogram)/HRV	Heart rate monitor (HRM)	SpO2 sensor	Pedometer	Accelerometer	Gyroscope	Magnetometer	Biometer	GPS	Altimeter	Thermometer	Microphone	Compass	Ambient light sensor	Optical sensors	NFC	Score
Xiaomi	Mi Watch	Smartwatch	Wrist	X	X	8GB	384	50	420	5 ATM	32	0.82	130	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.3735		
Amazfit	Band 5	Tracker	Wrist	X	X	16MB	360		2	125	5 ATM	24	0.68	35	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.3755	
Amazfit	GTR 2	Smartwatch	Wrist	X	X	3GB	168	15	2.5	471	5 ATM	40	0.75	170	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.4305	
Amazfit	T-Rex Pro	Smartwatch	Wrist	X	X	3GB	432	40	1.5	390	5 ATM	60	0.58	170	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.3665	
Amazfit	X	Smartwatch	Wrist	X	X	1024MB	168		2	205	5 ATM	47	0.63	330	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.3658			
Honor	Band 6	Tracker	Wrist	X	X		336				5 ATM	23	0.63	49	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.3220		
Realme	Watch S Pro	Smartwatch	Wrist	X	X		336			420	5 ATM	63	0.56	130	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.3658			
Realme	Band	Smartband	Wrist	X	X		144			90	5 ATM	20	0.32	25	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.3235		
Huawei	Watch Fit	Smartwatch	Wrist	X	X	4GB	240	12			5 ATM	27	0.68	79	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.5528		
Huawei	TalkBand B6	Smartband	Wrist	X	X	16MB	36			120	IP67	29	0.64	160	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.5523		
Huawei	Band 5	Smartband	Wrist	X	X		336				5 ATM	25	0.68	59	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.5523		
Huawei	Watch 3 Pro	Smartwatch	Wrist	X	X	16GB	120	22			5 ATM	70	0.50	409	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.542		
Scosche	Rhythm2.0	Tracker	Wrist	X	X		24				5 ATM	0.00	0.00	66	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.627		
Scosche	Rhythm2.0	Tracker	Wrist	X	X		24				5 ATM	0.00	0.00	62	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.68			
Coros	Ventix	Smartwatch	Wrist	X	X		1080	60	2		15 ATM	54	0.63	495	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.81		
Coros	Pace 2	Smartwatch	Wrist	X	X		480	30	2		5 ATM	29	0.64	165	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.6953		
Coros	AlpenPro	Smartwatch	Wrist	X	X		720	40	2		10 ATM	59	0.58	413	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.6088		
Biotan	Faou 360	Tracker	Chest	X	X		4320	192	15		IP67	18	0.93	3096	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.707		
Audib	S	Tracker	Chest	X	X		108	336			IPX7	15	0.96		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.4668			
Audib	Audib	Tracker	Chest	X	X		48	3	400			46	0.70	230	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.459			
Aidmed	Aidmed	Tracker	Chest	X	X							0.00	495	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.172			
Equival	eq02+ LifeMonitor	Tracker	Chest	X	X		8GB	48			IP67	0.00	0.00		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.438			
Equival	EX eq02+ LifeMonitor	Tracker	Chest	X	X		8GB	48			IP67	0.00	0.00		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.438			
BioSouris	Bionton	Tracker	Arm	X	X		120				720	IP67	40	0.75	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0.4605			

Figure 5 Wearables classification (partial list)

4 Models selection, training and validation

The following sections explain the processes that led to the choice of the machine learning algorithm. The selection of the appropriate features and the necessary training steps are also exposed. Finally, the modalities with which it is possible to evaluate the reliability and the accuracy of the carried-out work are reported.

4.1 Algorithm for fatigue prediction

This section presents the rationale that leads to the identification of the Random Forest as the most suitable machine learning algorithm for the first release of the FaMS.

Because of its flexibility, FaMS can use different types of machine learning algorithms. At the current stage, it was decided to search in the literature for studies comparing different techniques and machine learning algorithms, thus being able to analyse the main advantages and disadvantages of each of them. The idea is to select the most suitable algorithm for the current state of the project. However, in the second period of the T5.2, different types of algorithms will be quantitatively evaluated, and a test phase will be carried out to compare their performance. Since this situation has already been addressed in past projects very similar to the current one, the analyses carried out in these works were also considered.

4.1.1 Algorithms comparison

There are several machine learning algorithms suitable for classification problems in the literature. Each of them has advantages and disadvantages depending on the application case.

The FaMS, at its current state, has two features to be considered in selecting the algorithm:

- The physical effort perceived by a worker is expressed by an integer value in the range $[0, 10]$, which is considered as a class. Therefore, the chosen algorithm should be able to solve a multiclass classification problem.
- The dataset currently available to perform the training and validation of the model contains a limited amount of observations (221 observations, with 45 features), and a pain point is the difficulty in collecting new datasets from real workers in real industrial environments. It follows that the selected algorithm should provide accurate predictions even with a small dataset, so that to mitigate the risk of lacking data also in the future.

The list of considered algorithms is presented in Table 4, by highlighting the main advantages and disadvantages.

Table 4 Algorithms comparison

Description	Advantages	Disadvantages
<p>Logistic Regression is a classification machine learning algorithm used to find the probability of success and failure of an event. It is normally used when the dependent variable is of binary nature. It allows categorising data into discrete classes by studying the relationship from a given set of labelled data. In this algorithm, from a set of data, the probabilities describing the possible outcomes are modelled using a logistic function.</p>	<p>algorithm designed for classification; easy to implement, interpret and efficient to train; allows to understand the influence of several independent variables on a single outcome variable; fast in classifying unknown data; good accuracy for many simple data sets and good performance when the data set is linearly separable; not very prone to overfitting, but can overfit in high-dimensional datasets.</p>	<p>algorithm originally designed for binary classification, but thanks to some techniques and strategies (e.g., One-vs-Rest or One-vs-One) it is possible to extend its functionality for multiclass classification problems; if the number of observations is less than the number of features, the Logistic Regression can lead to overfitting; Logistic Regression assumes that there is a linear relationship between the dependent variable and the independent variables.</p>
<p>Naive Bayes is a probabilistic classifier and is based on Bayes' theorem, with the assumption that the features are independent of each other. Each feature has the same weight.</p>	<p>working with small datasets; fast and efficient training and prediction time; based on a probabilistic approach; working for multiclass classification problems</p>	<p>all the features have the same relevance; features must be independent;</p>
<p>K-Nearest Neighbours algorithm assumes that similar things exist in close proximity. In other words, similar things are close to each other. This non-linear proximity-based classification algorithm is Lazy Learning in that it does not attempt to build a general internal model, but simply stores instances of the data. The classification is computed by a simple majority vote of the k nearest neighbours of each point. These k nearest neighbours are selected based on Euclidean distance. Among these k neighbours, the new point is assigned to the category in which there is the largest number of neighbours.</p>	<p>easy to interpret easy to implement; working for multiclass classification problems; very accurate if fed with a sufficient amount of data.</p>	<p>it suffers from noise and not relevant features it struggles with high dimensionality data; it tends to overfit; fixing the number of clusters (K) may not be straightforward and the process is very expensive</p>

<p>Decision Tree is a flowchart with a tree structure, where each inner node denotes a test on an attribute (a condition), each branch represents a test result (True or False), and each leaf node (end node) contains a class label. Based on this tree, divisions are made to differentiate the classes in the original provided dataset. The classifier predicts which class a new datum belongs to according to the decision tree.</p>	<p>simple to understand; requires little data preparation; it can manage both numerical and categorical data; can be applied to both linear and non-linear problems.</p>	<p>poor results on very small data sets as overfitting can easily occur; very complex trees can be created that do not generalize well to the problem; can be unstable as small variations in the data can lead to a completely different tree.</p>
<p>Random Forest is a type of supervised machine learning algorithm based on Ensemble Learning. Ensemble Learning is a type of learning where different types of algorithms or the same algorithm are combined multiple times to form a more powerful prediction model. The Random Forest combines multiple algorithms of the same type, i.e., multiple Decision Trees, resulting in a forest of trees. This algorithm can be used for both regression and classification tasks.</p>	<p>the algorithm is not biased, as there are several trees each of which is trained on a subset of data. Basically, the Random Forest is based on the power of the "crowd", so the overall bias of the algorithm is reduced; reduction of overfitting; in most cases higher performance than the Decision Tree; very stable: even introducing new data into the dataset, the overall algorithm is not affected much, since new data may impact one tree, but it is very difficult for it to impact all trees; excellent performance; can also be applied to non-linear problems; it can handle both numerical and categorical data; the algorithm works well even when the data has missing values or has not been scaled well; works well even with small to medium sized datasets, as opposed to Neural Networks which require large amounts of data to train efficiently;</p>	<p>due to the complexity of the algorithm more time is required for the training phase than in other algorithms; manual identification of the best number of trees; because of the great number of trees decision joined to form a forest, many computational resources are required.</p>
<p>Support Vector Machine is used as a linear or non-linear classifier depending on the kernel</p>	<p>works relatively well when there is an understandable margin of dissociation between</p>	<p>poorly suited for nonlinear problems;</p>

<p>used. To separate two classes, a line is drawn that possesses a maximum margin. This line is found to be equidistant from both sets. Two other lines are also drawn on either side known as support vectors. This type of algorithm learns from support vectors, unlike others that learn from correct and incorrect data.</p>	<p>classes; productive in high-dimensional spaces; efficient in instances where the number of dimensions is greater than the number of samples; efficient in terms of memory; not affected by outliers; not sensitive to overfitting.</p>	<p>not the best choice in the presence of a large number of features; not acceptable for large data sets.</p>
---	---	---

4.1.2 Selection of the machine learning algorithm

To qualitatively select the best algorithm, the following characteristics of the application have been considered:

- From the functional requirements, the need is to have a multiclass classifier.
- At the current stage of the project there is still very little data available. In fact, the dataset available consists of only 221 observations and 45 variables.
- Since Scikit-learn was chosen as the machine learning framework, the selected machine learning algorithm must be well supported by it.

Logistic Regression and Support Vector Machine have been devised for binary classification problems. They have also the possibility of being adapted for multiclass classification problems through the use of some strategies like the One-vs-Rest and the One-vs-One. The first trains a binary classification model for each class with respect to all other classes, and predictions are then made using the most confident model (the class that gets a higher probability score will represent the "final prediction"). The second, trains a binary classification model for each pair of classes and in this case, the prediction is made using a majority count (the class that has been predicted more times than the others will represent the "final prediction"). At the current stage, it is preferred to use algorithms directly applicable to multiclass problems (Naive Bayes, K-Nearest Neighbours, Decision Tree, Random Forest).

According to the selection criteria outlined in section 3.3.1.1, the Random Forest proved to be the most appropriate multiclass classification algorithm. In fact, Random Forest is supported by Scikit-learn and can be used for multiclass classification problems. In addition, its characteristics allow it to perform well even with small datasets. Moreover, it manages unbalanced datasets (by weighting classes so that to obtain a weighted accuracy). It is robust to outliers, less sensitive to noise and being an ensemble method combats overfitting thus improving prediction performance. During the training phase, the Random Forest is able to evaluate the importance of the features helping to understand which of them are more relevant. Finally, Random Forest is often reported in the literature as one of the most efficient multiclass classification algorithms and, in past projects with similar applications, it was been always the best performing algorithm for multiclass classification problems.

The choice of this algorithm should not be considered final, as the project is still designed to be flexible to possible changes. Consequently, should it be necessary, a different algorithm can be used with little effort. As the task progressed, it was envisioned that models for fatigue estimation trained with different machine learning algorithms could be used simultaneously within FaMS. In this way, it may be possible to obtain more accurate and reliable estimations. In the future, when more data is available for the training phase, it will then be possible to implement a quality testing phase to select additional algorithms, comparing their actual performance.

4.2 Feature selection and model training for physical effort prediction

This section presents the process that allows selecting the features included in the trained artificial intelligence model. The process consists of six steps:

1. Preparation of workers' static data;

2. Preparation of data relating to the physical effort felt by workers;
3. Preparation and merging of workers' dynamic data with perceived exertion data (step 2);
4. Windowing process of the data obtained in step 3
Fusing the data obtained from the windowing process with the static data from step 1
Enrichment of the data with new information derived from the available data
5. Selection of features for the training phase of the machine learning model using the data from step 4
6. Training and validation of the machine learning model using the features identified in step 5
Saving of the obtained model for its use within the FaMS

In the following sections, these phases are analysed in detail.

4.2.1 Step 1: Preparing Static Data

This step aims to prepare static data from workers to be used in model training for predicting workers' perceived fatigue. These data have been collected through questionnaires completed by workers. In this way, each worker is profiled. The questionnaire data are persisted in a file in CSV format: the first line of the file represents the header, while the following lines represent the questionnaire data belonging to the workers (each line refers to a specific worker). After retrieving the data from the CSV file, the questionnaire data are prepared for use in the next steps. For example, the age of the workers is calculated from their birth date, as well as their maximum heart rate; the header and format of some columns are changed to aid understanding. The redundant information is removed, such as that used in the Örebro questionnaire to calculate and estimate a psychophysical assessment of the worker. In fact, the Örebro score already contains all the information needed to provide a psychophysical assessment and consequently, if retained in the profiling, the latter would appear as a duplication. Furthermore, since the original data from the questionnaire were in Spanish, it was decided to translate them into English. Once the preparations are completed, these data are persisted in a new CSV file with the same format as that of the questionnaire. In this way, it is possible to use this information in the following phases.

Table 5 shows what the data used to profile workers look like at the end of Step 1.

Table 5 Static data regarding workers

Data	Unit	Data type	Acquisition method
Worker ID	-	Integer	Questionnaire
Sex	Female/Male/Other	String	Questionnaire
Date of birth	Date format: "yyyy-mm-dd"	Date	Questionnaire
Age	-	Integer	Computed
Hiring date	Date format: "yyyy-mm-dd"	Date	Questionnaire
Working age	-	Integer	Computed
Weight	Kg	Integer	Measured
Height	cm	Integer	Measured
Maximum heart rate	bpm	Integer	Computed
Waist circumference	cm	Integer	Measured
Body fat	%	Float	Measured
Grip strength (right and left) (mean and	Kg	Float	Measured

standard deviation)			
Smoker	Yes/No	String	Questionnaire
Number of workouts per week	-	Integer	Questionnaire
Training intensity	None/ Low/ Medium/ High	String	Questionnaire
Work task experience	None/ Low/ Medium/ High	String	Questionnaire
Psychophysical assessment (OMPQ)	Örebro Score	Integer	Örebro questionnaire

4.2.2 Step 2: Preparing data regarding workers' perceived physical exertion

This stage is used to prepare data regarding the physical exertion felt by workers during the course of their work shift for use in training the model to predict fatigue.

The data processed at this stage is critical to the creation of the fatigue prediction model. In fact, it is information that provides the real effort perceived by workers while performing a specific task expressed by means of Borg's CR-10 Scale. To obtain this information, the worker was asked to indicate the degree of physical exertion perceived while performing a specific task. These data are obtained from a file in CSV format: the first line of the file represents the header, while the following lines represent the collected data regarding the fatigue perceived by workers and the task they performed at that moment (each line refers to a specific worker associated with date and time). After retrieving the data from the CSV file, the data is prepared for use in the next steps. For example, the header and format of some columns are changed to make them easier to understand and the "Timestamp" column is added. Rows belonging to unidentifiable workers (the worker ID is not present) are also removed. At the end of these operations, the data obtained are saved in a new CSV file with the same structure of the original one. In this way, it is possible to use this information in the following phases.

Table 6 shows how the data, used to identify the physical exertion felt by workers during their work shift, look at the end of Step 2.

Table 6 Data related to the physical effort perceived by the workers during the execution of a task

Data	Unit	Data type	Additional information
Worker ID	-	String	Worker Identifier
Work task ID	-	Integer	Identifier of the task performed by the worker
Detection time	Date format: "yyyy-mm-dd HH:MM:SS"	Date	Date and time of data collection expressed in the format "yyyy-mm-dd HH:MM:SS"
Timestamp	-	Integer	Date and time of data collection expressed in seconds elapsed since 1970-01-01
Worker perceived fatigue	Value in the range [0, 10]	Integer	Asked the worker while performing the task. It is a value that lies in the range [0, 10], where 0 is the perceived zero fatigue and 10 is the perceived maximum fatigue
Notes	-	String	Additional notes describing the task performed by the worker

4.2.3 Step 3: Preparation and fusion of workers' dynamic data with perceived exertion data

In this phase two main operations are carried out: first the data coming from the sensors applied to the workers during their work session are retrieved, and then these data are merged with the information obtained in Step 2 (for more information see section 4.2.2).

This data fusion is essential to give context to the information coming from the sensors: in this way, physiological parameters are accompanied by indications regarding the effort perceived by workers during a specific activity in the work shift.

Each sensor stores its data in separate CSV format files. The first line of the file represents the header, while the following lines represent data collected by the sensor referring to a worker at a given time (each line refers to a specific worker associated with date and time). During the implementation of this project, data were available from three different types of sensors that detected: heart rate, temperature, and galvanic skin response of workers.

After retrieving the sensor data, they are prepared to be merged with the data from Step 2. For example, redundant information is removed, and the header and format of some columns are changed to aid understanding.

At this point, data from sensors and from Step 2 are merged into a single dataset. Given that different datasets have different timestamps, a simple join operation is not enough: data are then concatenated so that to have sorted timestamps with data from sensors in different columns (null values are present when a metric is missing for a timestamp; null values will be managed during the downstream windowing operation described in Section 4.2.4).

Finally, the obtained data are saved in a new CSV file with the same structure as the starting one. In this way, it is possible to use this data in the following steps.

In Table 7 it is possible to see how the data of the sensors joined with those of the effort perceived by the workers at the end of Step 3 are presented.

Table 7 Workers' physiological data from sensors combined with physical exertion data

Data	Unit	Data type	Additional information
Worker ID	-	String	Worker Identifier
Work task ID	-	Integer	Identifier of the task performed by the worker
Detection time	Date format: "yyyy-mm-dd HH:MM:SS"	Date	Date and time of data collection expressed in the format "yyyy-mm-dd HH:MM:SS"
Timestamp	-	Integer	Date and time of data collection expressed in seconds elapsed since 1970-01-01 ("yyyy-mm-dd")
Heart rate	bpm	Float	Heart rate of a worker detected by sensor while performing a task
Galvanic skin response	microsiemens	Float	Galvanic skin response of a worker detected by the sensor while performing a task
Skin temperature	°C	Float	Skin temperature of a worker detected by the sensor while performing a task
Worker perceived fatigue	Value in the range [0, 10]	Integer	Asked the worker while performing the task. It is a value that lies in the range [0, 10], where 0 is the perceived zero fatigue and 10 is the perceived maximum fatigue

Notes	-	String	Additional notes describing the task performed by the worker
--------------	---	--------	--

4.2.4 Step 4: Data windowing, merging with static data and computation of additional information

This phase deals with three main operations: first, the data obtained in Step 3 are windowed, then they are merged with the static data obtained in Step 1, and finally, additional information is obtained from the existing data. Both the data obtained in Step 3 and the data obtained in Step 1 are persisted in files in CSV format: the first line of the file represents the header, while the following lines represent the data associated with the workers. First, the data obtained in Step 3 (representing those of the sensors and perceived exertion) are retrieved and a windowing operation is performed on them with a configurable time window (default: 1 minute). The goal is to investigate a subset of the data in more detail over a period of time in order to learn about some statistical properties about the data in the time series. Specifically, for the data of each sensor, it is desired to obtain minimum value, maximum value and average value in the time window of 1 minute. After this first operation, all rows in which there is no value for the effort perceived by the worker at a specific time and all rows in which there are no values for the sensor data are removed from the table. This results in a table with no incomplete data. To improve the information of the static data, additional information is computed from the existing information. The new information that is calculated is listed below:

- Body mass index
- Weight to height ratio
- Ponderal Index
- Is the grip strength (left and right) higher than expected?
- Hour (at the time of data collection)
- Minute (at the time of data collection)
- Work shift
- Time elapsed since the start of the work shift
- Theoretical Borg value (min, max and mean), i.e., perceived fatigue

The data obtained from the windowing process are then merged with the static data enriched with additional information in a single table. Since the column representing the percentage of body fat of the workers is composed only of missing data, it is removed from the table.

In Table 8, it is possible to see how the static, dynamic and perceived exertion data of the workers look at the end of Step 4.

Table 8 Static, dynamic data and workers' perceived effort

Data	Unit	Data type	Additional information
Worker ID	-	String	Worker Identifier
Work task ID	-	Integer	Identifier of the task performed by the worker
Detection time	Date format: "yyyy-mm-dd HH:MM:SS"	Date	Date and time of data collection expressed in the format "yyyy-mm-dd HH:MM:SS"

Timestamp	-	Integer	Date and time of data collection expressed in seconds elapsed since 1970-01-01 ("yyyy-mm-dd")
Heart rate (min, max, media)	bpm	Float	Heart rate of a worker detected by sensor while performing a task
Galvanic skin response (min, max, media)	microsiemens	Float	Galvanic skin response of a worker detected by the sensor while performing a task
Skin temperature (min, max, media)	°C	Float	Skin temperature of a worker detected by the sensor while performing a task
Worker perceived fatigue	Value in the range [0, 10]	Integer	Asked the worker while performing the task. It is a value that lies in the range [0, 10], where 0 is the perceived zero fatigue and 10 is the perceived maximum fatigue
Notes	-	String	Additional notes describing the task performed by the worker
Sex	Female/Male/Other	String	Questionnaire
Date of birth	Date format: "yyyy-mm-dd"	Date	Questionnaire
Age	-	Integer	Computed
Hiring date	Date format: "yyyy-mm-dd"	Date	Questionnaire
Working age	-	Integer	Computed
Weight	Kg	Integer	Measured
Height	cm	Integer	Measured
Maximum heart rate	bpm	Integer	Computed
Waist circumference	cm	Integer	Measured
Body fat	%	Float	Measured
Grip strength (right and left) (mean and standard deviation)	Kg	Float	Measured
Smoker	Yes/No	String	Questionnaire
Number of workouts per week	-	Integer	Questionnaire
Training intensity	None/ Low/ Medium/ High	String	Questionnaire
Work task experience	None/ Low/ Medium/ High	String	Questionnaire
Psychophysical assessment	Örebro Score	Integer	Örebro questionnaire

(OMPQ)			
Body mass index	Kg/m ²	Float	Computed
Weight to height ratio	Kg/cm	Float	Computed
Ponderal Index	Kg/m ³	Float	Computed
Is the grip strength (left and right) higher than expected?	0 or 1 (where 0 means no, and 1 means yes)	Integer	Computed
Hour	-	Integer	Computed. Represents the hour (at the time of data collection)
Minute	-	Integer	Computed. Represents the minute (at the time of data collection)
Work shift	morning_shift/ afternoon_shift	String	Computed. Up to 14:00 is considered morning_shift, after 14:00 is considered afternoon_shift
Time elapsed since the start of the work shift	minutes	Integer	Computed
Theoretical Borg value (min, max and mean)	-	Integer	Computed

The data obtained are ready to face Step 5, which is responsible for identifying the most suitable features for the training phase of the machine learning model. Unfortunately, at the current stage of the project, the available data are very few: the complete table is composed of only 221 rows and 45 columns. With such a small amount of data it is difficult to properly train a machine learning model (i.e., to achieve a satisfiable level of generalization). As the project progresses, it will be possible to acquire new data to improve the training phase.

The data is then randomly divided into two datasets: the dataset for the training and validation phases (consisting of 80% of the total data) and the dataset to perform the test phase (consisting of 20% of the total data) of the machine learning model. The data is saved in three CSV files: one file containing all the data, one file containing the data for training the model, and one file containing the data for testing the model. In this way, it is possible to use this data in the following steps.

4.2.5 Step 5: Feature selection for training the machine learning model

This phase aims to identify the most suitable features for the training of the machine learning model.

At the current state of the project, the data in possession to be used for training a machine learning model for predicting worker fatigue are very few: the complete table consists of just 221 rows and 45 columns. In addition to this, there is the problem of unbalanced data

on the perceived effort of workers, i.e., the class to be predicted. Moreover, some classes have just a few observations, making it difficult to both training and testing the model in such classes.

Given the situation, three techniques were identified to try to overcome these issues:

- class weights
- data binning
- stratified sampling

The technique of **class weights** consists of indicating to the algorithm to consider the distortion of the classes by giving different weights to classes. The classifier is instructed to assign class weights inversely proportional to their respective frequencies. In this way, wrong predictions have a different impact over the total accuracy of the model. The purpose is to penalise the misclassification error made by the minority class by setting a higher class weight and at the same time reducing the weight for the majority class.

Data binning (or bucketing) is a data pre-processing technique used to reduce the effects of small observational errors. The original data values, which fall in a specific small interval, are replaced by a value representative of that interval, often the central value. This is a form of quantisation.

Stratified sampling is a sampling technique in which samples are selected in the same proportion (by dividing the data set into groups called "strata" based on a characteristic) in which they appear in the data set. Some classification problems do not have a balanced number of samples for each class label. It is therefore desirable to divide the dataset into training set and validation set, in a way that the proportions of samples in each class, as observed in the original dataset, are preserved.

Starting with the three techniques listed above, five variants used for feature selection were developed:

- none of the three techniques applied
- application of class weights
- application of data binning
- application of class weights and data binning
- application of class weights, data binning and stratified sampling

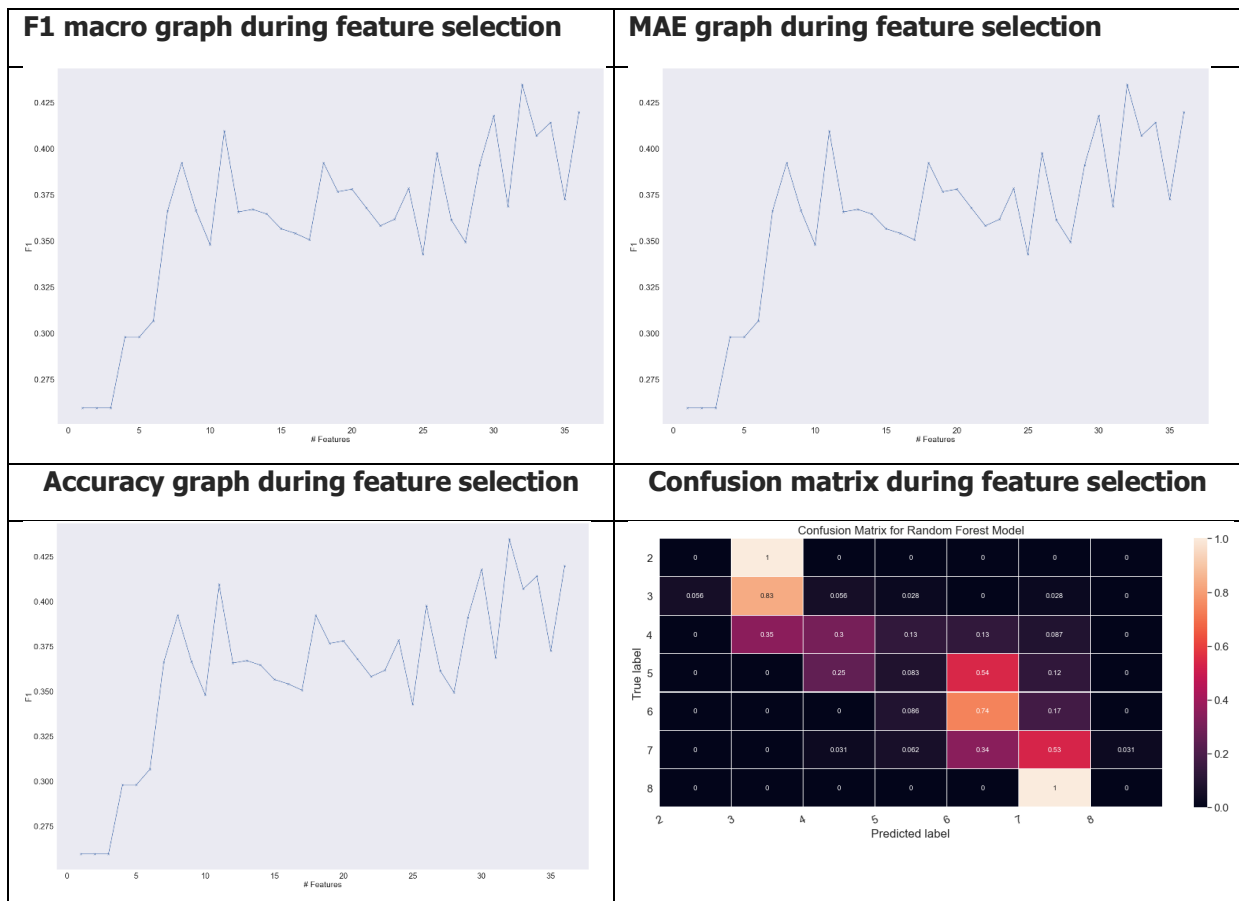
At the current stage of the project, the classification algorithm chosen is the Random Forest. The operations carried out in Step 5 are therefore aimed at identifying the parameters to obtain the best performance for this classification algorithm. In the case of the Random Forest, it is a matter of selecting the most appropriate features and number of trees used in the algorithm. All five variants follow the same steps:

1. the dataset for the training phase prepared in Step 4, persisted in a CSV format file, is uploaded
2. since several machine learning models require all input data to be numeric, the encoding of the data is performed (data in String format are transformed into numeric format). For ordinal data, the Ordinal Encoding approach was chosen, while for categorical data the One-Hot Encoding approach was chosen
3. the value that the Random Forest classifier must predict is the fatigue perceived by workers. To try to identify the most appropriate features for prediction, initially,

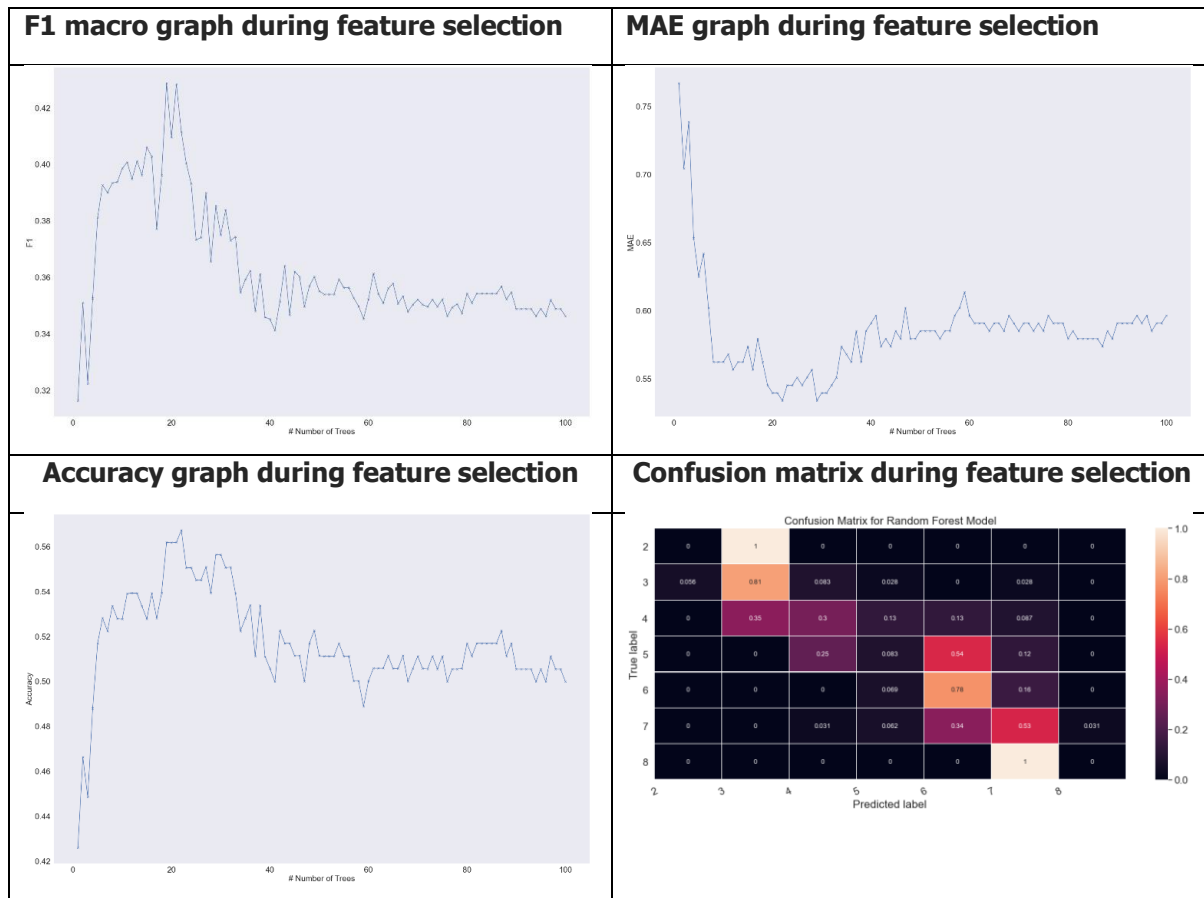
correlation coefficients are calculated for each of them in relation to perceived fatigue. A list containing the features is then created in descending order: from the one with the highest absolute correlation coefficient to the one with the lowest absolute correlation coefficient

4. the list obtained at point 3 is then exploited to select the most suitable features to identify worker fatigue; in a first phase the most relevant features are sought and in a second phase the most appropriate number of trees used in the algorithm is sought. Four types of metrics are used to choose these parameters: F1 macro, accuracy, mean absolute error (MAE) and confusion matrix
5. to select the features a model is trained using the cross-validation method (with 5 folds): at each iteration, the feature with the lowest correlation coefficient in relation to the effort is discarded. This procedure continues until the feature with a higher correlation coefficient with the physical effort is isolated. To identify the most appropriate set of features the metrics indicated in point 4 are used
6. once the most useful features for fatigue prediction have been identified, the search for the ideal number of trees for the Random Forest algorithm is performed. Also, in this case, a model is trained using the method of cross validation (with 5 folds). At each cycle, using the features selected in point 5, the number of trees is increased. Their quantity varies from 1 to 100. To identify the most appropriate number of trees, the metrics indicated in point 4 are used
7. at the end of these processes, the best parameters for training the Random Forest are obtained.

The graphs and the confusion matrix show an example of the feature selection process. In the specific case, the optimal number of features identified is 11.



The graphs and confusion matrix show an example of the process of selecting the most appropriate number of trees. In this specific case, the optimal number of trees identified is 22.



4.2.6 Step 6: Training, validating and saving the machine learning model

This phase aims to train and validate the machine learning model responsible for predicting workers' perceived fatigue. The data to be used are those obtained at the end of Stage 4. As this data is present in limited quantity, obtaining a quality classifier with the process of training and validation of the same is not so simple. In addition to this, there is the fact that the data on the perceived effort of the workers, i.e., the class to be predicted, is unbalanced. As in Step 5, the three techniques used to address these difficulties were as follows:

- class weights
- data binning
- stratified sampling

Thanks to them, as before, five variants used for training and validation of the machine learning model have been developed:

- none of the three techniques applied
- application of class weights
- application of data binning
- application of class weights and data binning
- application of class weights, data binning and stratified sampling

The algorithm chosen for this phase of the work, as already indicated, is the Random Forest classifier. Starting from the features and the number of trees identified in Step 5, the training, validation and saving process of the machine learning model is performed. All five variants follow a common pattern for achieving the processes just mentioned. A more detailed explanation of this scheme is presented below:

1. the set of data for the training phase and that for the validation phase prepared in Step 4, persisted in files in CSV format, are loaded
2. since for several machine learning models it is essential that all input data are numeric, the data encoding is performed (data in String format are transformed into numeric format). For ordinal type data, Ordinal Encoding is used, while for categorical type data, One-Hot Encoding is used. These operations are performed both on the data set for the training and on the one for the validation of the model
3. using the training dataset, the machine learning model (the Random Forest classifier) is trained, exploiting the parameters identified in Step 5 (most appropriate features and number of trees). This yields a classifier trained to predict workers' perceived fatigue
4. using the F1 macro, accuracy, mean absolute error (MAE) and confusion matrix metrics, the performance of the machine learning model obtained in point 3 is estimated. To do this, use the data set for validation
5. finally, the trained model and features are saved: the model is persisted in a JOBLIB file, while the features and their data type are saved in a CSV file. In this way, it is possible to import and use the trained model comfortably in FaMS

In Figure 6, the confusion matrix shows the results obtained after training and validating the fatigue classification model.

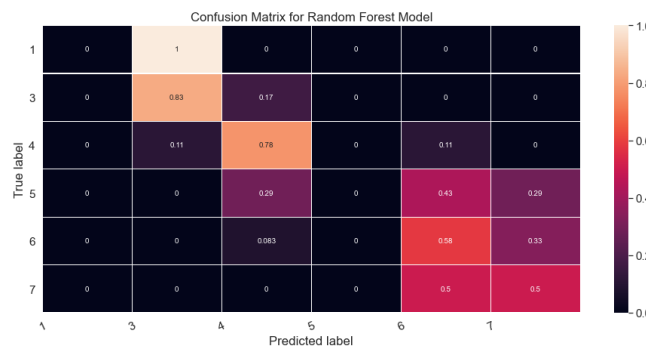


Figure 6 Confusion matrix after the training and validation phase

5 Worker streaming data event detection

While the whole data process chain described in the earlier version is extremely valuable in associating worker information and measurements to different fatigue states, in practice it is also valuable to track the monitored parameters for signs of changes. This might provide early warnings of growing fatigue or indicate time points of change in the conditions. Some of the changes could be related to the worker fatigue state but others can be broadly contextual or operational – for example, a change in the work activities of the worker. When applied in practice, monitoring the worker’s state might not have access to all contextual factors which may signify a change in the worker’s condition. In such cases the ability to detect change events in the monitored signal can trigger additional process chains, for example for further data processing, signal segmentation, classification, but crucially also such changes may signify the needs for further changes in the operational practice during work activities.

Reliable identification of significant changes in human monitoring data streams is the start of any event-based proactive approach for more efficient work activities and eventually combined human state and process state management. This can be pursued via time series-based change point detection (CPD). The main difference with such methods is that they need to perform on-the-fly, i.e., although, of interest, it is not sufficient to process the whole time series measurement a-posteriori. Naturally, for streaming data event detection, it is very hard to establish the ground truth of change events. However, an approach that is calibrated continuously to individual worker condition and state is one that can uncover patterns in the work practice that may contribute to reducing the efficiency of the worker, the production itself, and ultimately also have a detrimental effect on the worker well-being.

It is worth noting that the event detection in the scientific literature enters in multiple but not identical forms. Relevant terms include outlier, novelty, anomaly, event/change, concept drift, and detection, as well as time series segmentation.

5.1 Posing the event detection problem

In a multivariate time series sequence $\mathbf{X}_1, \dots, \mathbf{X}_t, \dots, \mathbf{X}_N$, CPD seeks to identify non-overlapping sub-sequences, also known as segments, partition blocks or product partitions, denoted as:

$$[\mathbf{X}_{t_0}, \dots, \mathbf{X}_{t_1}], [\mathbf{X}_{t_1+1}, \dots, \mathbf{X}_{t_2}], \dots, [\mathbf{X}_{(t_{k-1})+1}, \dots, \mathbf{X}_{t_k}]$$

such that each observation within the same segment follows the same probability density function but observations belonging to two consecutive sequences follow different probability density functions:

$$P[\mathbf{X}_t] = P[\mathbf{X}_{t+1}] \quad \forall t_i \leq t < t_{i+1}, \quad P[\mathbf{X}_t] \neq P[\mathbf{X}_{t+1}] \quad \forall t \in t_i$$

for $i = 0, 1, 2, \dots, k - 1$.

The time indexes t_1, t_2, \dots, t_{k-1} represent the separation between the sub-sequences and are called change points. This is a problem that has been addressed in different ways over the year [REF-6][REF-7] but the above notation is consistent with several such definitions.

5.2 Candidate methods

Among the methods available in the literature, relevant methods for streaming data event detection include

- Autoregressive integrated moving average fault detection (ARIMAFD) [REF-08]
- Various Bayesian approaches and specifically Bayesian online CPD (BOCPD) [REF-06]
- Isolation forest [REF-09].

We are currently performing an exploratory analysis of the available data in order to prioritise the most suitable method to employ for the event detection problem (or CPD).

6 Conclusions

In the next period, the task will focus on the test, fix and improve the current prototype. Specific efforts will be dedicated to the system integration with the components developed within tasks T5.1.

Moreover, the FaMS will be validated first in laboratory scenarios, to obtain different and the to support one of the Philips use-cases, to obtain also labelled datasets for training of different models. Finally, new type of features will be included in the models, such as the contextual and task's features and a user manual will be realised.

References

- [REF-01] Ferdousi, R., Laamarti, F., Hossain, M. A., Yang, C., & El Saddik, A. (2022). Digital twins for well-being: an overview. *Digital Twin*, 1(7), 7.
- [REF-02] El Saddik A, Badawi H, Velazquez RAM, et al.: Dtwins: a digital twins ecosystem for health and well-being. *IEEE COMSOC MMTCC Commun. Front.* 2019; 14: 39–43.
- [REF-03] Bagaria N, Laamarti F, Badawi HF, et al.: Health 4.0: Digital twins for health and well-being. *Connected Health in Smart Cities*. Springer. 2020; 143–152.
- [REF-04] Patrone C, Galli G, Revetria R: A state of the art of digital twin and simulation supported by data mining in the healthcare sector. *Advancing Technology Industrialization Through Intelligent Software Methodologies, Tools and Techniques*. IOS Press. 2019; 318: 605–615.
- [REF-05] Adams, R.P., and MacKay, D.J.C., (2007), "Bayesian Online Changepoint Detection," Oct. 2007.
- [REF-06] Hinkley, D.V., Hinkley, E.A., (1970) "Inference about the change-point in a sequence of binomial variables," *Biometrika*, vol. 57, no. 3, pp. 477–488, 1970, doi: 10.1093/biomet/57.3.477.
- [REF-07] Kozitsin, V., Katser, I., Lakontsev, D., (2021) "Online Forecasting and Anomaly Detection Based on the ARIMA Model," *Applied Sciences*, vol. 11, no. 7, p. 3194, doi: 10.3390/app11073194.
- [REF-08] Zhang, C., Wu, R., Li, G., Cui, W., Jiang, Y., (2020), "Change detection method based on vector data and isolation forest algorithm," *Journal of Applied Remote Sensing*, vol. 14, no. 02, p. 1, doi: 10.1117/1.JRS.14.024516.