Personalised Health Monitoring and Decision Support Based on Artificial Intelligence and Holistic Health Records

D4.7 – iHelp DSS suite with visual analytics II

WP4 Knowledge Management and Utilisation in the iHelp Platform

Dissemination Level:	Public
Document type:	Report
Version:	1.0
Date:	October 31, 2022

The project iHelp has received funding from the European Union's Horizon 2020 Programme for research, technological development, and demonstration under grant agreement no 101017441.

Document Details

Project Number	101017441
Project Title	iHelp - Personalised Health Monitoring and Decision Support Based on Artificial Intelligence and Holistic Health Records
Title of deliverable	iHelp DSS suite with visual analytics II
Work package	WP4 Knowledge Management and Utilisation in the iHelp Platform
Due Date	October 31, 2022
Submission Date	October 31, 2022
Start Date of Project	January 1, 2021
Duration of project	36 months
Main Responsible Partner	UPM
Deliverable nature	Report
Authors names	Ainhoa Azqueta (UPM), Marta Patiño (UPM), Oscar García (ICE), Eleftheria Kouremenou, George Manias (UPRC)
Reviewers names	Maritini Kalogerini, Giorgos Giotis (ATC), Aristodemos Pnevmatikakis (iSPRINT)

Document Revision History

Version History			
Version	**Date**	**Author(s)**	**Changes made**
0.1	2022-09-29	Ainhoa Azqueta (UPM)	Initial version and Table of Contents
0.2	2022-10-07	Ainhoa Azqueta (UPM)	Contents provided
0.3	2022-10-10	Ainhoa Azqueta Marta Patiño (UPM)	Contents for several sections
0.4	2022-10-13	Eleftheria Kouremenou, George Manias (UPRC)	Content provided in Section 5
0.5	2022-10-24	Marta Patiño (UPM)	Minor updates
0.6	2022-10-26	Aristodemos Pnevmatikakis (iSPRINT)	Document internal review
0.7	2022-10-26	Maritini Kalogerini (ATC), George Giotis (ATC)	Internal Review
0.8	2022-10-27	Ainhoa Azqueta (UPM), George Manias (UPRC)	Final review and updates
0.9	2022-10-28	Pavlos Kranas (LXS)	Quality review
1.0	2022-10-31	Dimosthenis Kyriazis (UPRC)	Final version

Table of Contents

Table of Figures

Executive summary

This deliverable reports the work done in the context of T4.3 – "Clinical DSS Suite with Visual Analytic Tools". The main objective of this task is to design and implement the decision support visualization component of the iHelp platform. On top of this, the goal of the DSS is to allow different stakeholders to run analytical functions and visualize analytic results in order to support decision and policy making.

This task started in the fourth month (M4) of the project thus this deliverable presents the results achieved during these first eighteen months of this task. In this context, a prototype of the DSS component has been designed and implemented. The DSS prototype allows the definition of dashboards to visualize results of analytics, the definition of analytics workflows and visualization of stored data while defining queries with low code.

The first prototype of the DSS was presented in D4.6 – "iHelp DSS suite with visual analytics I", in M12. The first version was based only on Node-RED, a browser-based editor for defining data flows by connecting predefined nodes. The first nodes implemented exclusively for iHelp were presented, which allowed the integration of the DSS with the Big Data Platform and the Analytic Workbench components. In addition, the DSS allowed the creation of Custom Dashboards that allowed to show the results of the queries performed on the data stored in the Big Data Platform and on the results obtained by the AI models provided by the Analytic Workbench. These Custom Dashboards could be visualised by the different types of users who have access to the DSS, the Model builder, the Health Care Professionals (HCPs), and the Policy makers.

The actual DSS implementation is based on Angular[1] and Node-RED, hence it follows the same principles. During the last ten months since the first version of this series of deliverables, a web interface based on Angular has been developed implementing part of the interfaces required by the iHelp project pilots. The integration with most of the iHelp components that interact with the DSS has also started. These are also detailed in the context of this deliverable. In the coming months, the implementation of the interfaces required by the pilots will be completed, the integration with all the components that interact with the DSS will be finalised and the interfaces will be presented to the Pilots so that they can start working with the DSS. The feedback from the pilots will be used to adjust the interfaces to their requirements. To this end, the third and final prototype of the DSS will be reported in deliverable D4.8 – "iHelp DSS suite with visual analytics III", due in month M32.

[1] https://angular.io/

1 Introduction

This deliverable, i.e., D4.7 – "iHelp DSS Suite with Visual Analytics II" describes the intermediate version of the iHelp Decision Support System (DSS). The iHelp DSS component will be the iHelp platform interface used by model builders, HCPs, and policy makers as a decision-making process system. The DSS architecture consists of five sub-components: the *Authentication and Authorization, the Workflow Editor, the Custom Dashboards, Explainable Dashboard Hub (EDH) and the Web Interface*. The DSS is integrated with the iHelp Big Data Platform, the Analytics Workbench, the Predictor and Risk identifiers, the Monitoring and Alerting System and the Personalized Predictor components. One of the main components of the DSS is the workflow interface component. This component allows the creation of analytical workflows, dashboards for the different types of users and the incremental design of SQL queries while the results of the query are visualized.

1.1 Objective of the Deliverable

The aim of this deliverable is to describe the intermediate version of the Decision Support System component developed under the scope of task T4.3 - *Clinical DSS Suite with Visual Analytic Tools*. More specifically, this deliverable describes the internal structure of the DSS, the use of the DSS, the provided interfaces and its deployment.

1.2 Structure of the Deliverable

This document is structured as follows: Section 1 introduces the deliverable and its main objectives, Section 2 presents the DSS architecture and objectives as well as its interaction with the rest of the iHelp components. Then, Sections 3, 4, 5 and 6 describe the Workflow interface, the Custom Dashboard interface, Models Explainable Dashboard Hub and the Web Interface DSS subcomponents respectively. Section 7 presents the deployment of the DSS both locally and remotely using Docker and Kubernetes. Finally, Section 8 describes future work and Section 9 presents the conclusions of the deliverable.

1.3 Updates since D4.6

The Decision Support System has been updated since the first version of this series of deliverables, i.e., D4.6 – "iHelp DSS suite with Visual Analytics I", was produced. The internal architecture of the DSS has been updated, including the addition of two new components, Explainable Dashboard Hub (EDH) and web interface components, sections 5 and 6 respectively. The EDH component has been added to help DSS users in understanding and feel more confident with the results obtained from the various AI models. The web interface has been added as a link between the other components of the DSS and other components of the iHelp platform. This component has been used for the creation of several interfaces to be used by all the pilots, such as the visualisation of patient data or the enrolment process of a new patient in the iHelp programme.

2 DSS Suite

This section describes the DSS component in the iHelp project, its objectives, internal architecture, and the interaction with other iHelp components.

2.1 Overview

The DSS is the user interface and access point to the iHelp platform. Several functionalities allow health care professionals/medical experts and policy makers to make decisions based on iHelp data analysis. The DSS provides query building, analytics, and visualization mechanisms to access the iHelp data and present them to iHelp users in a friendly way that eases the understanding of the iHelp platform's decision models. Technical users – the *model builders* – will find the required tools to design different dashboards for the visualization of data by health care professionals/medical experts and policy makers. The model builder uses a palette of SQL-like, analytical and visualization operators that will allow the creation of pipelines/workflows of transformations and visualizations over the results of the analytical algorithms. Moreover, a set of generic dashboards that help to both, the technical and non-technical users in the decision support process are provided.

2.2 Overall Objectives

The T4.3 – "Clinical DSS Suite with Visual Analytic Tools" has five main objectives. First, the DSS must display iHelp data and analytical algorithm results in customized dashboards to help health care professionals/medical experts and policy makers in the decision-making process. Second, the DSS must provide SQL-like, analytical and visualization functionalities in an easy and intuitive way. Third, provide a pipeline/workflow design interface to allow model builders the creation of the desired decision support dashboards. Fourth, provide a set of static dashboards common to all pilots that display patients' data stored in the Big Data Platform to be used by health care professionals during the decision-making process and to interact with patients. Fifth, include authentication and authorization for different types of users.

2.3 Internal Architecture

The DSS has five well differentiated sub-components: the *Authentication and Authorization, the Workflow Editor, the Custom Dashboards, Explainable Dashboard Hub (EDH) and the Web Interface*. Figure 1 depicts the DSS internal architecture and the relation among the sub-components.

The *Authentication and Authorization* component provides the DSS users the access point for different user types. This interface, that interacts with the Identity Manager, is in charge of user authentication. Currently, there are three types of users: model builders (data scientists), health care professionals/medical experts and policy makers. The Authentication and Authorization component gives access to the functionalities the DSS provides. The functionality available in the DSS will be different depending on the type of user that logs into the system.

The *Workflow Editor* interface provides to data scientists (model builders) users the visual tools for building models that process the data and ways to query the stored data. For that purpose, the Workflow Editor presents a palette of available operations and a workspace where SQL-like queries and workflows can be defined by dragging and dropping operators. The operators in the palette implement access to the iHelp data store, analytics, and prediction components available in the iHelp platform. The main users of the workflow interface are data scientists.

The Custom *Dashboards* interface sub-component allows the creation of customized dashboards to show the iHelp data and the results of applying the models and analytics. The dashboards present those results using different chart types and tables. The dashboard main users are health care professionals and policy makers.

Both the Workflow Editor and the Dashboard interface access external components, more concretely it uses the Big Data Platform, Analytics Workbench, Predictor and Risk Identifier, and Personalized Predictor in order to either define the data processing pipeline or model (Workflow) or show the results of the execution of that model. The Predictor and Risk Identifier and Personalized Predictor components are accessed through the Analytics Workbench. All these components are integrated within the DSS using their own REST APIs.

The Explainable Dashboard Hub (EDH) sub-component introduces several different dashboards that are used to monitor metrics and outcomes of the AI models providing easy-to-understand approaches by visualizing both the data and the insights derived from the utilization of different AI models. To this end, HCPs would be able to select multiple models for a wide range of scenarios and view numerous different visualizations and model comparisons. This will help the HCPs to enhance their knowledge and understanding on outcomes of the models. To achieve this goal this sub-component strongly integrates either directly with the external AI models, such as Risk Predictor and Risk Identifier via pickle files, or with the Analytic Workbench via appropriate APIs.

Last, the Web Interface sub-component provides a set of generic interfaces that have been defined in the context of deliverable D2.7 – "Functional and Non-Functional Specifications II". This sub-component is integrated with the Big Data Platform, Analytics Workbench, Predictor and Risk Identifier, Personalized Predictor and Monitoring and Alerting System components, and will be integrated with the Social Analyser and the Personalized Advisor, and the Mobile Application components during the second phase of the project. The web interfaces developed in the Monitoring and Alerting System, the Social Analyser and the Personalized Advisor components are embedded into the Web Interface component.

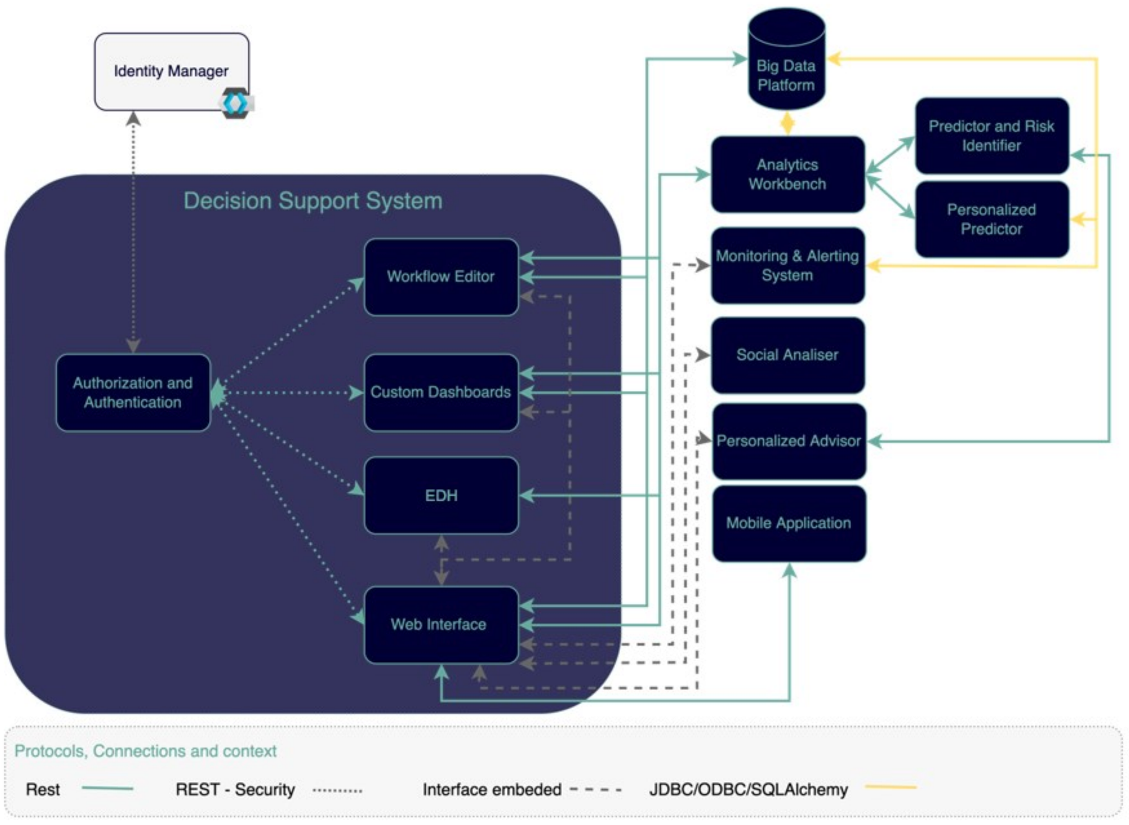


Figure 1: DSS internal architecture

2.3.1 User Interaction with the DSS

The interaction of a model builder, a HCP, and a policy maker user with the DSS is presented in Figure 2. First, the user must access the DSS through the Login interface. Based on the privileges of the user, the DSS will give access to some DSS components. If the user has a model builder user role, the DSS will give access to the Workflow Editor, Custom Dashboards or EDH components. Otherwise, for health care professional role users the DSS will give access to the Custom Dashboards or EDH components. The access to the platform is done through the user login interface. Finally for the policy maker user, the DSS will give access to the web interface where the dashboards provided by the iHelp Social Analyzer component will be presented. When a user registers in the iHelp platform, he/she is assigned one of the above roles. The model builder user will design the workflows and the custom dashboards that will be used by health care professionals or policy maker users. The Query-builder and Workflow interface interact with the iHelp Big Data Platform reading the data and injecting them into the Analytical models provided by the Analytics Workbench or into the Predictor and Risk Identifier or the Personalized Predictor components. The model builder can create dashboards through the Dashboard interface which will show the results of the models and queries designed in the Workflow interface.

On the other hand, the web interface contains a series of generic interfaces that allow the health care professional to access patient data, access the monitoring and alert system, the social analyser and interact with the patient by sending messages through the mobile application (the Healthentia[2] mobile app), as well

[2] https://healthentia.com/

as having access to the custom dashboard implemented by the model builder and the EDH component interface to analyse the results obtained by the AI models.

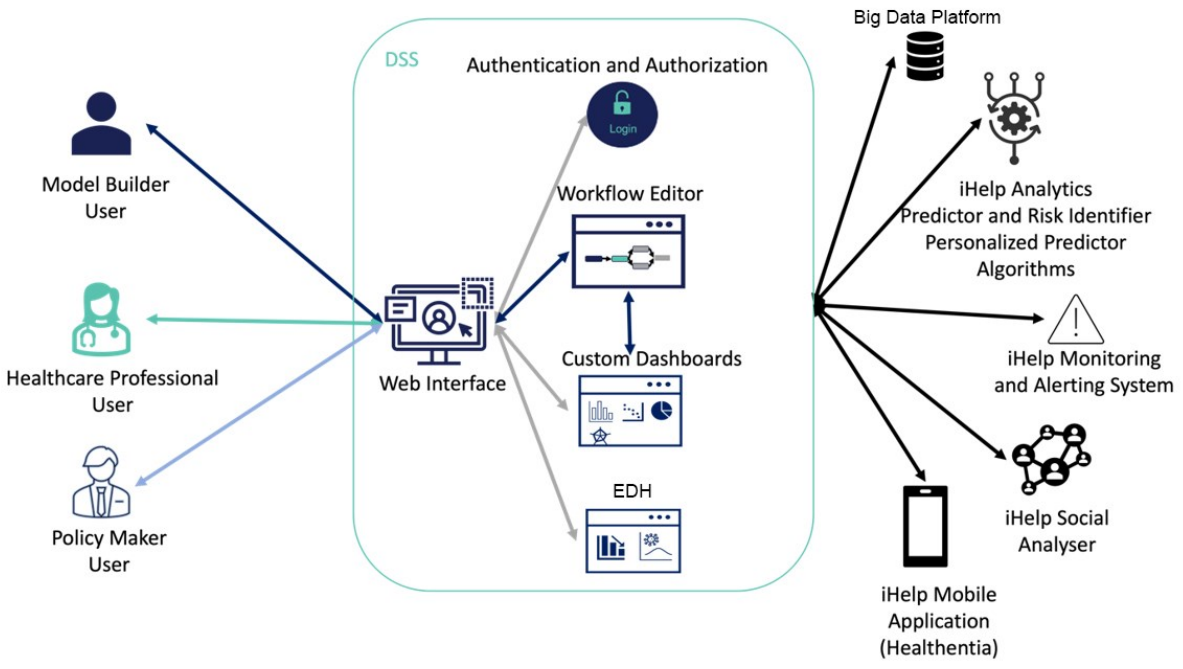

DSS
Authentication and Authorization
Login
Workflow Editor
Custom Dashboards
EDH
Web Interface
Model Builder User
Healthcare Professional User
Policy Maker User
Big Data Platform
iHelp Analytics Predictor and Risk Identifier Personalized Predictor Algorithms
iHelp Monitoring and Alerting System
iHelp Social Analyser
iHelp Mobile Application (Healthentia)

Figure 2: User interaction with the DSS

2.4 Related Work

With the increasing interest in the field of data science, several tools have emerged to help in the definition of analytics processes and visualize the associated results. Some of the available tools are more oriented to the creation of workflows for accessing input datasets for the execution of analytical algorithms by just drag and dropping different types of nodes and wiring them together. Other tools are more oriented to the visualization of the data and the results obtained from the execution of the analytical algorithms. Well-known and widely used workflow-oriented tools are: Rete[3], Apache Airflow[4], Node-RED[5] and Apache Superset[6]. Rete is a modular framework developed at MIT that allows the creation of flows in a web browser. This framework, which is integrated with Angular[7], requires to code the logic of the different nodes and the interactions between the wired nodes. Apache Airflow is a platform that allows the creation of workflows as direct acyclic graphs of tasks. These workflows support scheduling and deploying tasks among nodes of a cluster, providing integration with several cloud infrastructures like AWS[8] and Google

[3] https://rete.js.org/#/
[4] https://airflow.apache.org/
[5] https://nodered.org/
[6] https://superset.apache.org/
[7] https://angular.io/
[8] https://aws.amazon.com/es/?nc2=h_lg&aws-products-analytics.sort-by=item.additionalFields.productNameLowercase&aws-products-analytics.sort-order=asc&aws-products-business-apps.sort-by=item.additionalFields.productNameLowercase&aws-products-business-apps.so

Cloud[9]. The main drawback of the Apache Airflow tool is the large learning curve and the complexity of its deployment. Node-RED is a workflow web browser-based editor. It is backed by a large community that creates and shares custom created nodes to be used in the design of workflows. Node-RED provides a dashboard creation interface that allows the design of dashboards using nodes and the visualization of results produced by the workflows using different visualization widgets embedded in the dynamic dashboards. Apache Superset is the successor of the Apache Airflow enhanced with a data visualization interface that allows the creation of customizable dashboards. The learning curve and deployment complexity are the major drawbacks of this tool.

Some of the most popular data visualization tools are: Grafana[10], Tableau Public[11], Stashboard[12] and Freeboard[13]. Grafana allows the visualization of data collected from different databases like Prometheus[14], Graphite[15], InfluxDB[16] and PostgreSQL[17]. Grafana allows the creation of dynamic dashboards and the installation of several plugins for the incorporation of different visualization widgets to the available ones. Tableau Public is a tool for data visualization and business analytics. It allows the creation of public dashboards and requires a license to create private ones. Due to that, the usability of this tool is limited in its free version. Finally, Stashboard and Freeboard were created to monitor IoT devices. Stashboard has a very limited visualization set of widgets, while Freeboard has a wide set of visualization widgets, but it requires a license to make created dashboards private.

For the implementation of iHelp DSS the Node-RED tool was selected to be utilized for several reasons. First, it is fully open source and has a great community that contributes with the development of visualization widgets that fit with the requirements of the iHelp platform. Second, it is very well documented. This is a fundamental aspect for the DSS development. Third, it has a specific module for the creation and design of dashboards with visualization widgets that can be integrated as part of the analytical workflows. Fourth, the learning curve is small. The customized nodes are created providing JavaScript files. Last but not least, the installation and deployment are much easier in comparison with other tools.

2.5 Positioning and Relation with Other Components

The DSS component is the user entry point to the iHelp platform. The iHelp platform architecture (Figure 3) shows the relations of the DSS component, highlighted in a blue circle, with the rest of the iHelp components, namely the Big Data Platform, developed under the scopes of T4.4 – “Big Data Platform and Knowledge Management System”, the Analytic Workbench, developed under in T4.2 – “Model Library: Implementation and Recalibration of Adaptive Models”, the Predictor and Risk identifier and the Personalized Predictor, developed in T5.1 – “Techniques for Early Risk Identification, Predictions and Assessment”, the Personalized Advisor developed under the scope of T5.2 – “Design of Personalised

[9] https://cloud.google.com/gcp/?hl=es&utm_source=google&utm_medium=cpc&utm_campaign=emea-es-all-es-bkws-all-all-trial-e-gcp-1010042&utm_content=text-ad-none-any-DEV_c-CRE_495030365279-ADGP_Hybrid%20%7C%20BKWS%20-%20EXA%20%7C%20Txt%20~%20GCP%20~%20General%23
[10] https://grafana.com/
[11] https://public.tableau.com/en-us/s/
[12] https://www.stashboard.org/
[13] https://freeboard.io/
[14] https://prometheus.io/
[15] https://graphiteapp.org/
[16] https://www.influxdata.com/
[17] https://www.postgresql.org/

Prevention and Intervention Measures”, the Mobile Application developed in T5.3 – “Delivery Mechanisms for Personalised Health care and Real-time Feedback”, the Social Analyser developed in T5.4 – “Social Analytics for the Study of Societal Factors and Policy Making”, and the Monitoring and Alerting developed under the scope of T5.5 – “Monitoring, Alerting, Feedback and Evaluation Mechanisms”.

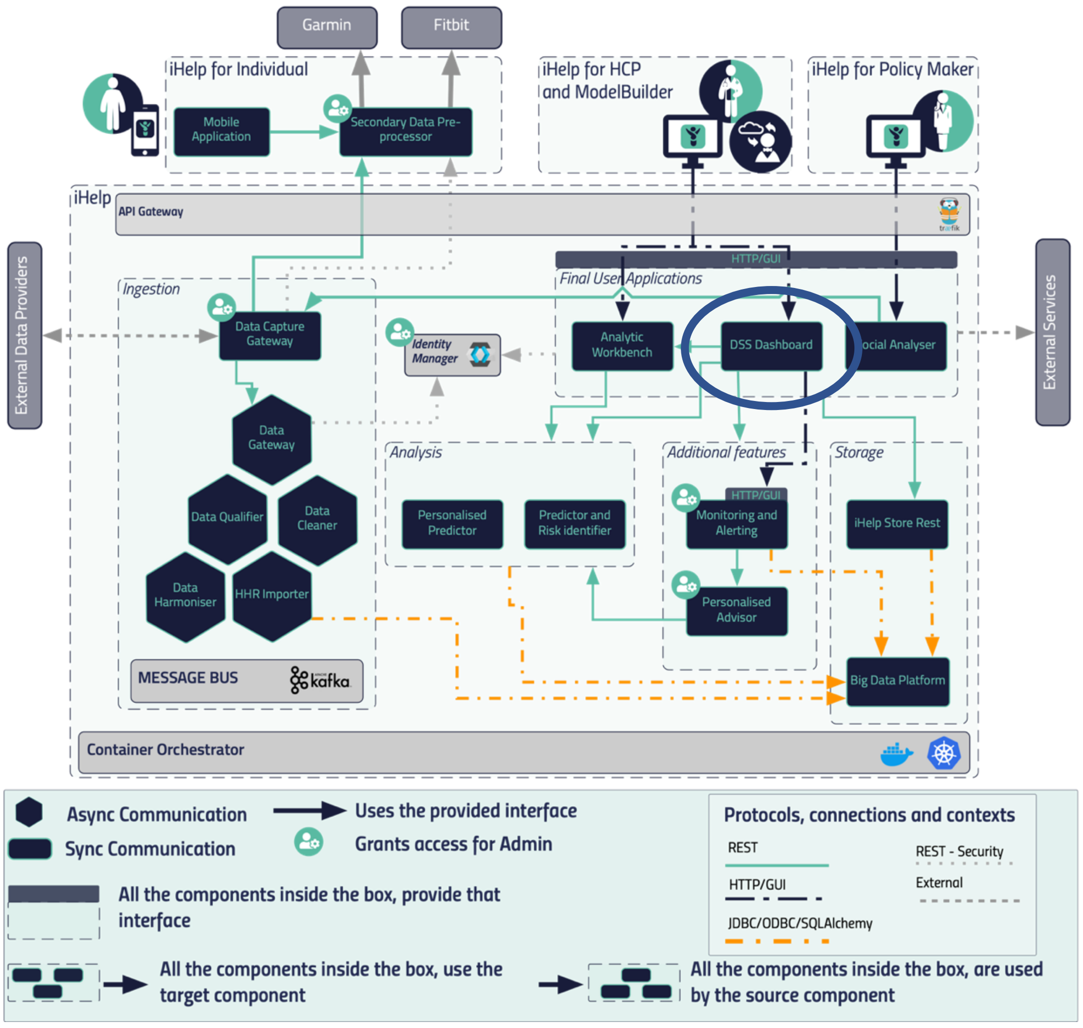


Figure 3: iHelp platform architecture

3 Workflow Interface

The Workflow interface sub-component allows the model builder user to design analytical pipelines/workflows, to incrementally design queries for accessing the iHelp data store, as well as to design the visualization dashboards for the rest of non-technical users of the platform. This component is based on the Node-RED low code programming tool. Node-RED was created to help users to program by wiring nodes available in different palettes. A node in Node-RED is the basic building block of a workflow that processes the data received from the previous nodes. For instance, Figure 4 shows a simple workflow example using three different types of nodes: one *inject* node (grey node), one *function* node (light orange) and one *debug* node (green). The *inject* node (named timestamp in the figure) is used for the manual triggering of the workflow execution. That is, when the node is clicked, a message is produced and sent to the next node. The *function* node executes JavaScript code, for instance it parses the timestamp of the message received from the previous inject node to HH:MM:SS format and sends the processed timestamp message to the debug node. The debug node displays messages in the *Debug sidebar* within the Node-RED editor.

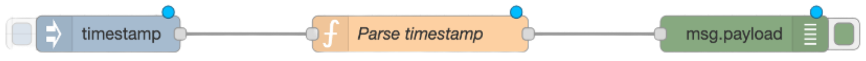
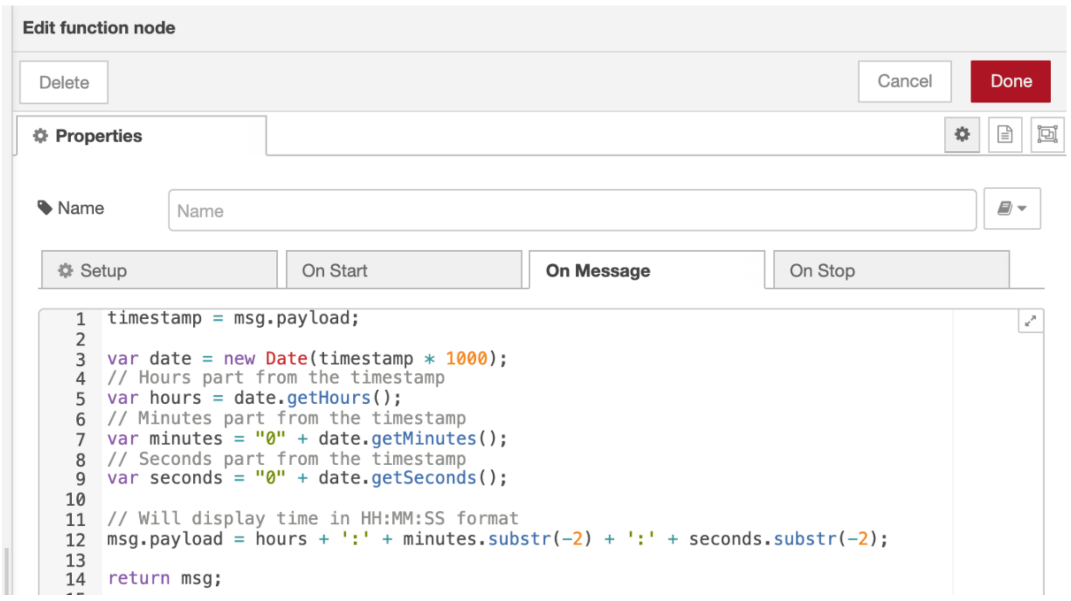


Figure 4: Workflow example

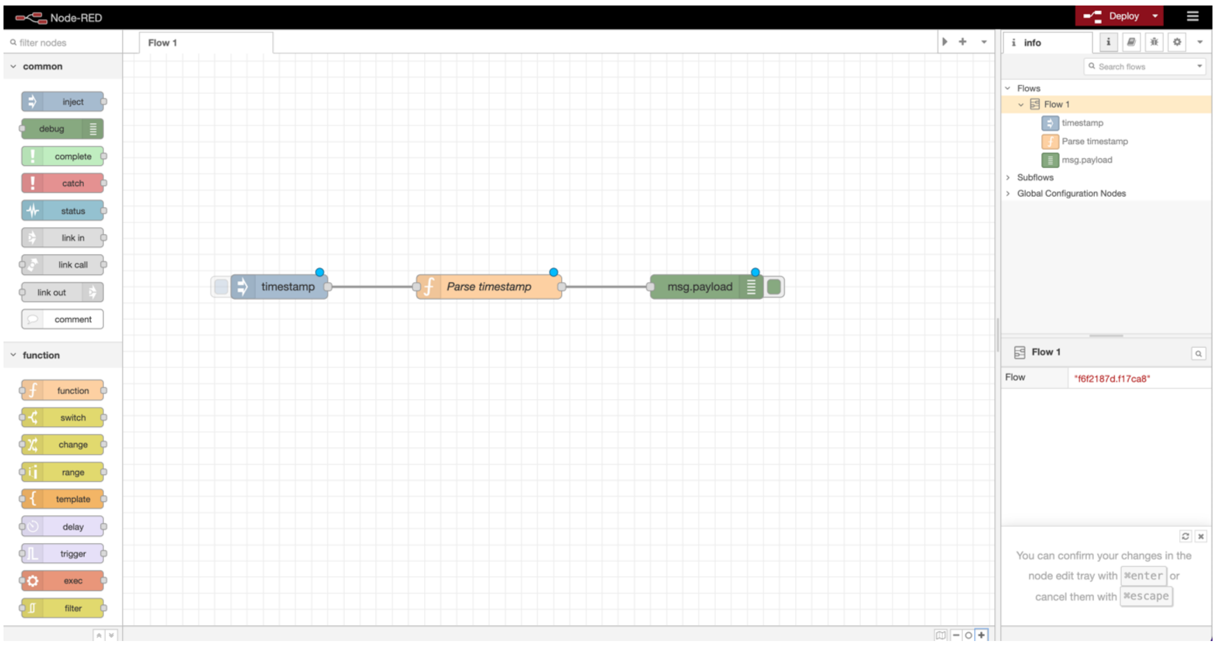
All nodes can be configured through a configuration node edit dialog that appears by clicking on the node. Figure 5 shows the configuration dialog for a function node. Different parameters can be defined in the configuration dialog. In this case the name of the node, *test function*, is included and the JavaScript code is included here.



```
timestamp = msg.payload;

var date = new Date(timestamp * 1000);
// Hours part from the timestamp
var hours = date.getHours();
// Minutes part from the timestamp
var minutes = "0" + date.getMinutes();
// Seconds part from the timestamp
var seconds = "0" + date.getSeconds();

// Will display time in HH:MM:SS format
msg.payload = hours + ':' + minutes.substr(-2) + ':' + seconds.substr(-2);

return msg;
```

Figure 5: Configuration dialog for a function node

The core nodes palette[18] includes a default set of basic nodes for creating workflows. These nodes are *inject, debug, function, change, switch* and *template*. The change node can be used to modify the message properties, the switch node allows sending messages to different branches of the workflow and the template node generates text using messages to fill a template. The core nodes palette can be extended by importing available palettes, or creating custom palettes.



Flow "f6f2187d.f17ca8"

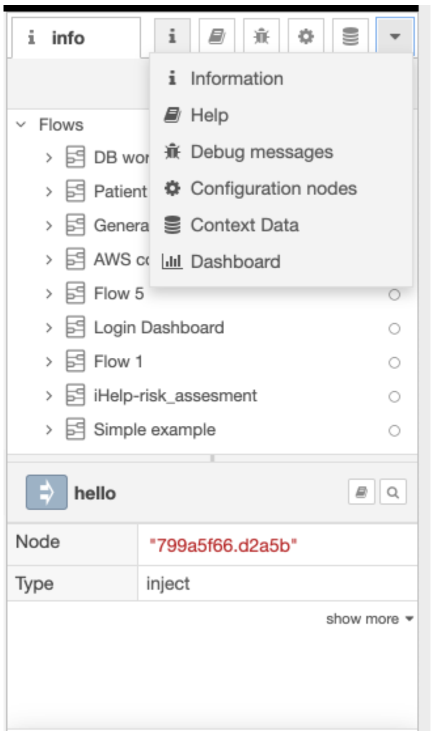
You can confirm your changes in the node edit tray with ⌘enter or cancel them with ⌘escape

Figure 6: Node-RED editor interface

Figure 6 shows the Node-RED editor interface. The different palettes and their nodes are shown on the left side. The sidebar on the right of the editor provides different tools within the editor to for providing about the different workflows, nodes used and to access the debug console among others. The workspace placed in the central part is the Workflow Editor which used for the design of workflows by dragging nodes from the palette and wiring them together.

The *sidebar* (Figure 7) includes different panels like the information panel, the nodes´ help panel, the *debug messages* for showing the values of the different messages generated during a workflow execution and the *configuration nodes* panel. The *context data* panel shows the contents of the context when a workflow is executed, that is the messages sent between the different nodes. The Dashboard panel is used for the creation of the dashboards. This Dashboard panel appears when the Node-RED dashboard nodes palette is installed.

[18] https://nodered.org/docs/user-guide/nodes



Node	"799a5f66.d2a5b"
Type	inject

Figure 7: Node-RED sidebar

3.1 iHelp Nodes

Two different node palettes have been designed and implemented to satisfy the iHelp platform requirements: the *iHelp Storage* and *AI Models* palettes. The creation of custom Node-RED palettes was done according to the node-RED documentation[19]. Three types of files are needed in the folder of each palette: one JSON file named package.json, and as many JavaScript and HTML files as nodes in the palette. The package.json file provides the information of the palette: the name and the description of the palette and the nodes that belongs to that concrete palette. The JavaScript files stores the logic of the nodes, and the HTML files provide the configuration dialog implementation. In the following section the different palettes and nodes created within the context of the iHelp project are described.

3.1.1 iHelp Storage Nodes Palette

The iHelp storage palette contains the nodes used for the creation of the SQL queries by dragging and dropping the database operators in the Node-RED workspace in order the DSS to be integrated with the iHelp's Big Data Platform. That is, the creation of queries following the low code model of Node-RED. At this stage of the project the database nodes available in the palette are: *Select*, *Join* and *Insert* nodes as shown in Figure 8.

[19] https://nodered.org/docs/creating-nodes/

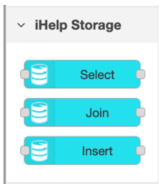
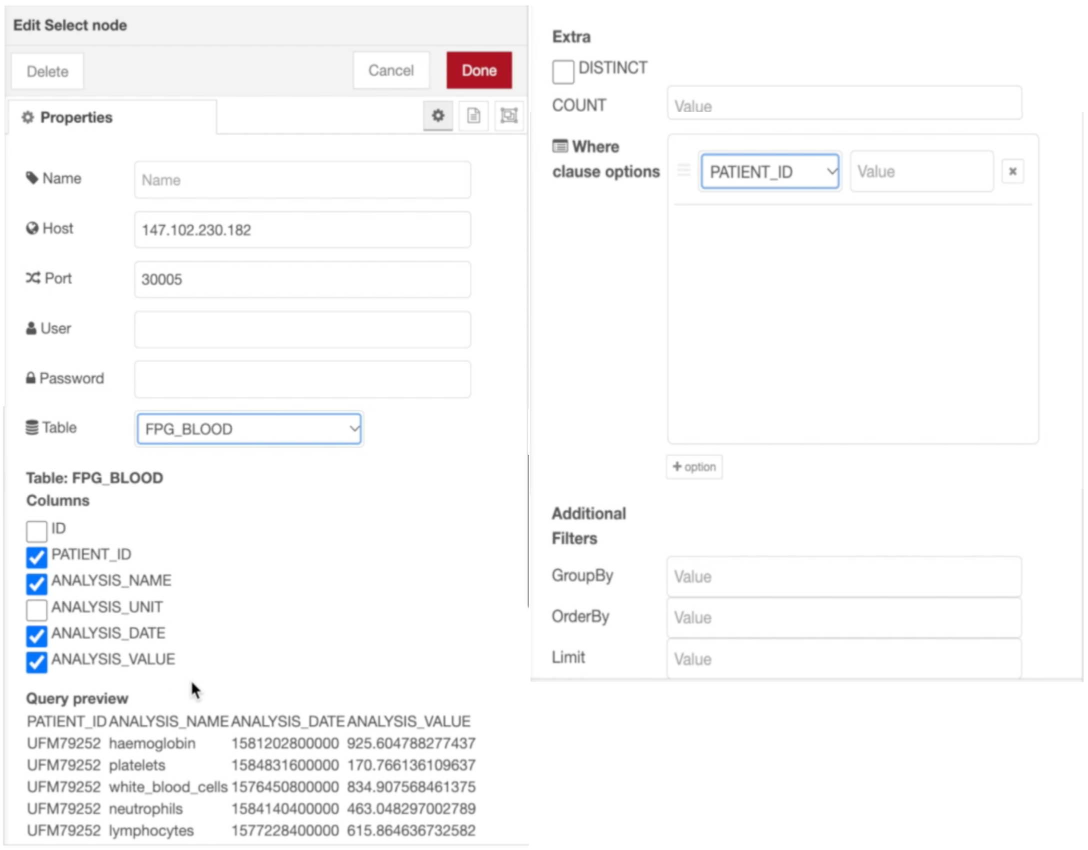


Figure 8: iHelp storage palette

The *select node* is used for choosing the relevant columns of the configured table and applying filters over the table. Internally, a *select* SQL statement is created. Figure 9 shows the configuration edit dialog of the *select* node. The first part shows the connection to the expected database that must be defined by filling in the information in the fields *Host, Port, User* and *Password*. Once these fields are filled in, *Table* drop down shows the tables in that database. Once a table is selected, FPG_BLOOD table in this case, a checkbox with the columns of the table and a previsualization of the first five rows of the table are depicted. The checkbox allows to select columns whose data is going to be used. At the time the columns are selected their values are shown in order to help the data scientist in the query building process. In this case the columns *PATIENT_ID* and *ANALYSIS_NAME, ANALYSIS_DATE* and *ANALYSIS_VALUE* are selected. Additionally, there is a checkbox and a field to include the *distinct* and *count* options in the *select* SQL statement.

There is another menu for configuring the *where* clause in a *select* statement (right part of Figure 9). For each *where* clause a new line appears to concatenate different predicates by means of "AND" and "OR" logical operators. Each line has two fields, the first one is a dropdown menu that shows the columns of the table and the second one is a text box to define the predicate. *Group by, Order by* and *limit* clauses are included to indicate how to sort the results and the maximum number of rows to be returned.



Name	Name
Host	147.102.230.182
Port	30005
User	
Password	
Table	FPG_BLOOD

Table: FPG_BLOOD
Columns

- [ ] ID
- [x] PATIENT_ID
- [x] ANALYSIS_NAME
- [ ] ANALYSIS_UNIT
- [x] ANALYSIS_DATE
- [x] ANALYSIS_VALUE

Query preview

PATIENT_ID	ANALYSIS_NAME	ANALYSIS_DATE	ANALYSIS_VALUE
UFM79252	haemoglobin	1581202800000	925.604788277437
UFM79252	platelets	1584831600000	170.766136109637
UFM79252	white_blood_cells	1576450800000	834.907568461375
UFM79252	neutrophils	1584140400000	463.048297002789
UFM79252	lymphocytes	1577228400000	615.864636732582

Extra

- [ ] DISTINCT

COUNT	Value
Where clause options	PATIENT_ID, Value

+ option

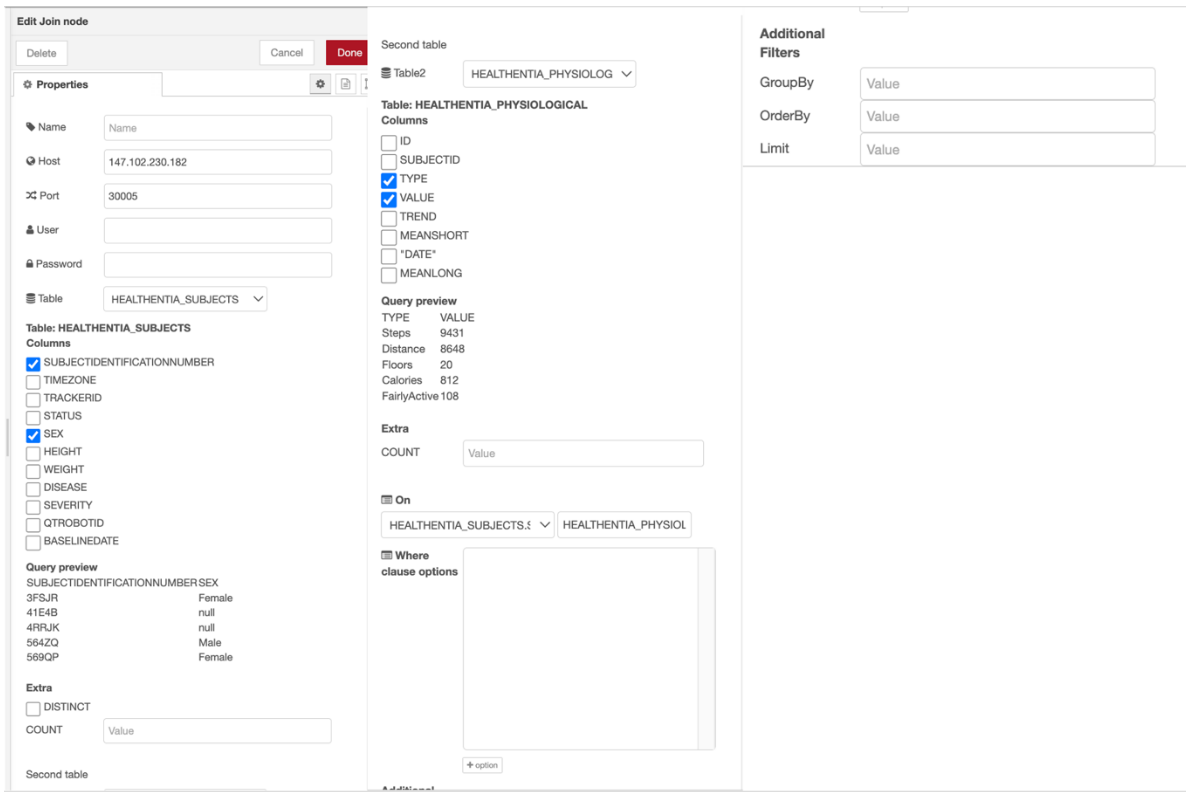
Additional Filters

GroupBy	Value
OrderBy	Value
Limit	Value

Figure 9: Select node configuration edit dialog

A similar menu and procedure are used for the definition of *join* statements being implemented by the *join* node. In this case the configuration node edit dialog (Figure 10Figure 10) needs data from the two tables to be joined, HEALTHENTIA_SUBJECTS and HEALTHENTIA_PHYSIOLOGICAL tables in the figure. The left table to be joined, the HEALTHENTIA_SUBJECTS table, is configured by selecting the columns of interest (*SUBJECTIDENTIFICATIONNUMBER and SEX*). At this point the values of up to five rows are displayed. A second table is also selected to be joined with the previous one, the HEALTHENTIA_PHYSIOLOGICAL table. Columns *TYPE* and *VALUE* are selected, and the corresponding rows are then displayed. The *On* menu is used to indicate the column to join. The internal code generated for the example join SQL statement that in Figure 10 is:

```
SELECT HEALTHENTIA_SUBJECTS .SUBJECTIDENTIFICATIONNUMBER, HEALTHENTIA_SUBJECTS .SEX,
HEALTHENTIA_PHYSIOLOGICAL.TYPE, HEALTHENTIA_PHYSIOLOGICAL.VALUE
FROM HEALTHENTIA_SUBJECTS
JOIN HEALTHENTIA_PHYSIOLOGICAL ON HEALTHENTIA_SUBJECTS.SUBJECTIDENTIFICATIONNUMBER =
HEALTHENTIA_PHYSIOLOGICAL.SUBJECTID;
```



Edit Join node

Name: Name
Host: 147.102.230.182
Port: 30005
User:
Password:
Table: HEALTHENTIA_SUBJECTS

Table: HEALTHENTIA_SUBJECTS
Columns
- [x] SUBJECTIDENTIFICATIONNUMBER
- [ ] TIMEZONE
- [ ] TRACKERID
- [ ] STATUS
- [x] SEX
- [ ] HEIGHT
- [ ] WEIGHT
- [ ] DISEASE
- [ ] SEVERITY
- [ ] QTROBOTID
- [ ] BASELINEDATE

Query preview

SUBJECTIDENTIFICATIONNUMBER	SEX
3FSJR	Female
41E4B	null
4RRJK	null
564ZQ	Male
569QP	Female

Extra
- [ ] DISTINCT

COUNT: Value

Second table
Table2: HEALTHENTIA_PHYSIOLOG

Table: HEALTHENTIA_PHYSIOLOGICAL
Columns
- [ ] ID
- [ ] SUBJECTID
- [x] TYPE
- [x] VALUE
- [ ] TREND
- [ ] MEANSHORT
- [ ] "DATE"
- [ ] MEANLONG

Query preview

TYPE	VALUE
Steps	9431
Distance	8648
Floors	20
Calories	812
FairlyActive	108

Extra
COUNT: Value

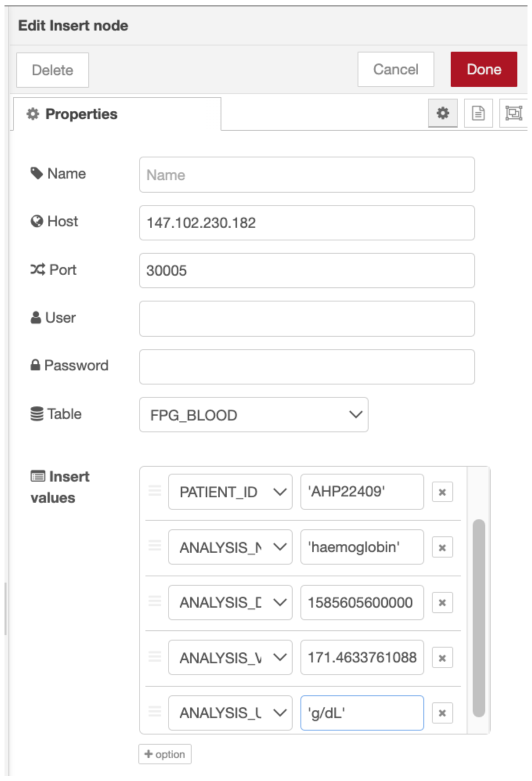
On: HEALTHENTIA_SUBJECTS.S | HEALTHENTIA_PHYSIOL

Where clause options

Additional Filters
GroupBy: Value
OrderBy: Value
Limit: Value

Figure 10: *Join* node configuration edit dialog

The *insert* node is used to insert new data into a table within the Big Data Platform database. Figure 11, shows the insert node configuration edit dialog. In the first part of the edit dialog panel the user can configure the connection with the database then, the Table dropdown is used to select a table from the database, in this case FPG_BLOOD. Next, the insert values panel allows to configure the values for each column of the selected table. Clicking on the add button, a new row appears with a dropdown menu and a text input field; within the dropdown the user must select the column from the table. Then, the values in the text input field must be provided. When all columns have been configured the user can click on done to finish with the configuration of the insert node.


Edit Insert node

Delete | Cancel | Done

Properties

Name	Name
Host	147.102.230.182
Port	30005
User	
Password	
Table	FPG_BLOOD

Insert values

PATIENT_ID	'AHP22409'
ANALYSIS_N	'haemoglobin'
ANALYSIS_D	1585605600000
ANALYSIS_V	171.4633761088
ANALYSIS_U	'g/dL'

+ option

Figure 11: Insert node configuration edit log

The *iHelp Storage* palette is implemented by adding a folder with seven files: the *package.json* file, three JavaScript files and three HTML files, two per node. Figure 11 shows the content of the package.json file where the palette and the nodes are defined. In the package.json file one can configure the name of the package that contains the nodes, the version of the nodes, the description of the palette, the script to test the nodes in case there is any test, the keywords that defines the palette, the author, the type of license and the nodes. Each node is added to the *nodes* label, being the key, the name of the node and the value is the code that is executed by that node. In the figure the nodes are *Select, Join* and *Insert.* The code for each node is in the files: *connection.js*, *select.js* and *join.js*.

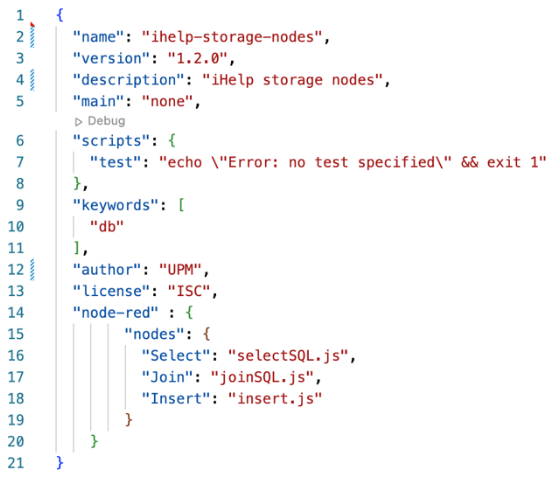

```
{
  "name": "ihelp-storage-nodes",
  "version": "1.2.0",
  "description": "iHelp storage nodes",
  "main": "none",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "db"
  ],
  "author": "UPM",
  "license": "ISC",
  "node-red" : {
      "nodes": {
        "Select": "selectSQL.js",
        "Join": "joinSQL.js",
        "Insert": "insert.js"
      }
    }
}
```

Figure 12: iHelp storage palette package.json file

3.1.2 iHelp AI Models Palette

The Analytics Workbench platform is based on DRyICE (short name for ICE Knowledge Discovery)[20]. Several clustering algorithms and AI models are available for processing the iHelp data. DRyICE is designed to allow the deployment of models defined as one or more Docker containers, instantiated via a Docker compose file. The algorithms run inside these containers, receive data, process, and forward the result to the next component. DRyICE uses Kafka environment variables to specify how messages are passed down the chain. The messages being passed can either be data or signals that data is ready.

The DSS hides the implementation and configuration of the Analytics Workbench executed on DRyICE by providing different analytics nodes available in the iHelp AI Models palette. These nodes are configured so that a call to the REST API provided under the scope of T4.2 – “Model Library: Implementation and Recalibration of Adaptive Models” is executed. The *Model List* node sends a request to the API asking for the list of all available AI Models. The models are listed in the configuration menu of the *Model List* node (Figure 14). The *Model Parameters* node sends a request to the REST API asking for the parameters required to execute the AI Model selected in the previous *Model List* node. The *Model Execution* node sends a request to execute the model. When a workflow triggers the execution of a model, the corresponding invocation is sent to the workbench. Figure 13 shows the *analytics* nodes available in the iHelp AI Models palette.

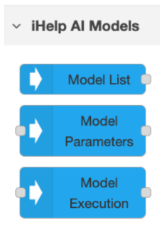
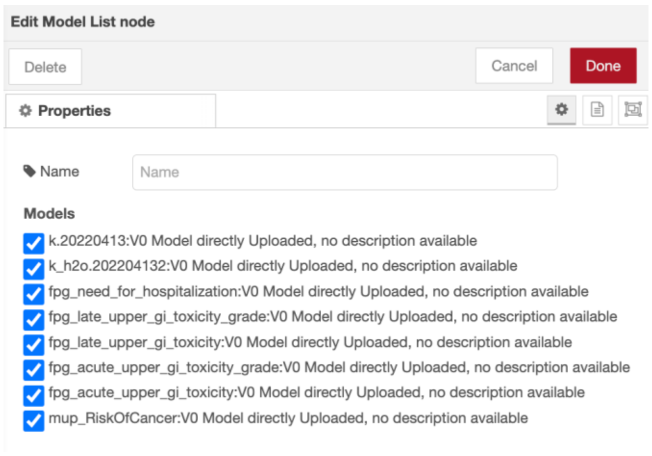


Figure 13: iHelp AI Models palette



Name: Name

Models

- ✓ k.20220413:V0 Model directly Uploaded, no description available
- ✓ k_h2o.202204132:V0 Model directly Uploaded, no description available
- ✓ fpg_need_for_hospitalization:V0 Model directly Uploaded, no description available
- ✓ fpg_late_upper_gi_toxicity_grade:V0 Model directly Uploaded, no description available
- ✓ fpg_late_upper_gi_toxicity:V0 Model directly Uploaded, no description available
- ✓ fpg_acute_upper_gi_toxicity_grade:V0 Model directly Uploaded, no description available
- ✓ fpg_acute_upper_gi_toxicity:V0 Model directly Uploaded, no description available
- ✓ mup_RiskOfCancer:V0 Model directly Uploaded, no description available

Figure 14: *Model List* node configuration panel

[20] https://drydev.icelab.cloud/docs/#!index.md

3.1.3 Generation of Custom Dashboards

The workflow editor is used to create workflows that connect custom nodes with iHelp Storage nodes and iHelp AI Model nodes and visualize the results in Custom Dashboards. Figure 15 shows the Workflow Editor with a set of iHelp Storage and iHelp AI Model nodes. These nodes are connected using wires and other Node-Red nodes for creating of dashboards. There are two different flows in the editor. The first one uses three iHelp *select* nodes to retrieve information about a patient's heartbeats, sleep time and activity. The second flow uses the three available AI models nodes and the dropdown, form and iframe Node-Red dashboard nodes. The dropdown node will list the available AI Models in the custom dashboard, the form node will present the parameters of the model in a form and the iframe node will prompt the result of the AI model in the custom dashboard.

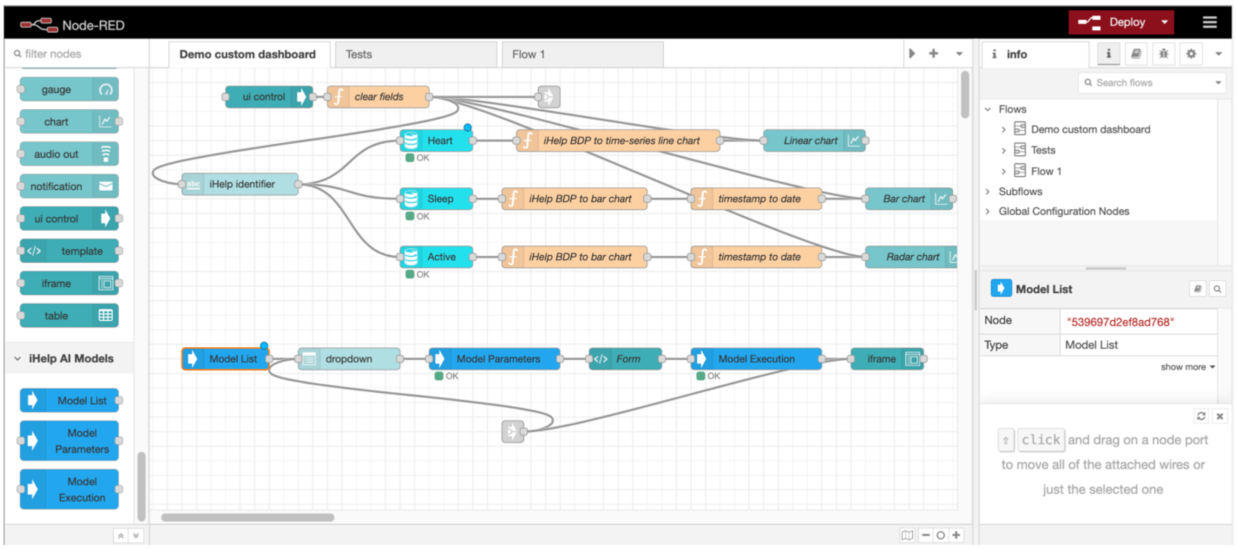


Figure 15: Workflow Editor Interface

All flows are stored in the Node-RED internal storage and can be reused.

4 Custom Dashboards Interface

The dashboard interface is used for presenting the results of workflows and queries to different types of users. The Node-RED Dashboard module[21] allows the creation of live dashboards using different nodes from its palette. The dashboard panel of the sidebar in the Node-RED editor is used for the definition of the layout of the dashboards. Figure 16 shows a screenshot of the dashboard panel where a dashboard entitled Healthentia patient is defined. This dashboard is split into two sections called groups, Patient data and AI Model respectively. Groups are sections of the dashboard where the different visualization widgets (table, charts, buttons, dropdown lists, etc.) can be set. Figure 17 shows the layout editor of the Healthentia patient data dashboard, this editor appears when the Healthentia patient data dashboard is selected from the Dashboard panel of the sidebar. In the example of in Figure 17, the two layouts are shown as grids defining the area that each layout is going to occupy in the dashboard. The group named *Patient data* contains a text input widget and three charts (linear, bar and radar), while the *AI Model* group contains a dropdown widget, a form and an iFrame.

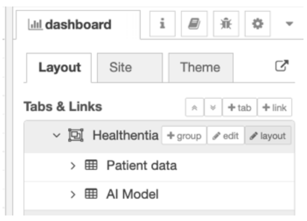


Figure 16: Dashboard panel of the sidebar of the Node-RED editor

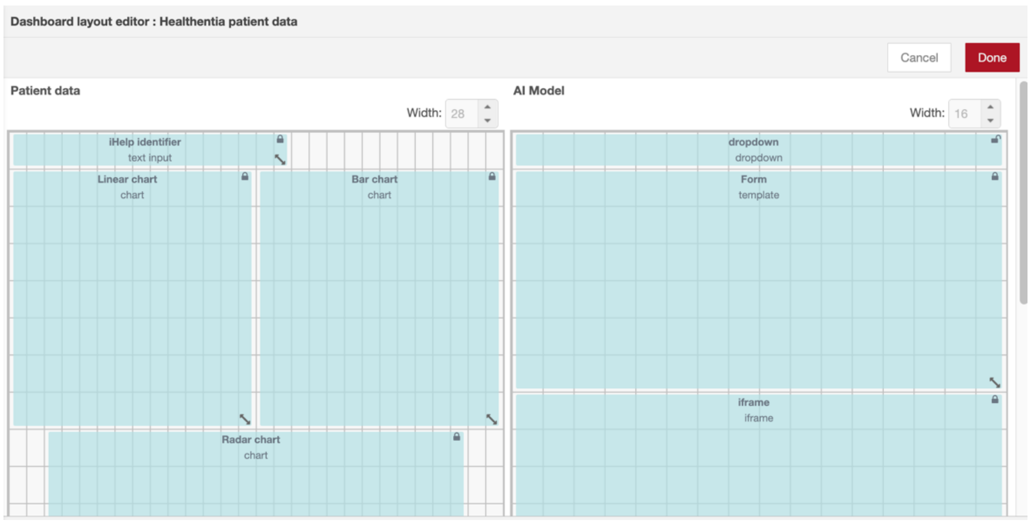


Figure 17: Dashboard layout editor

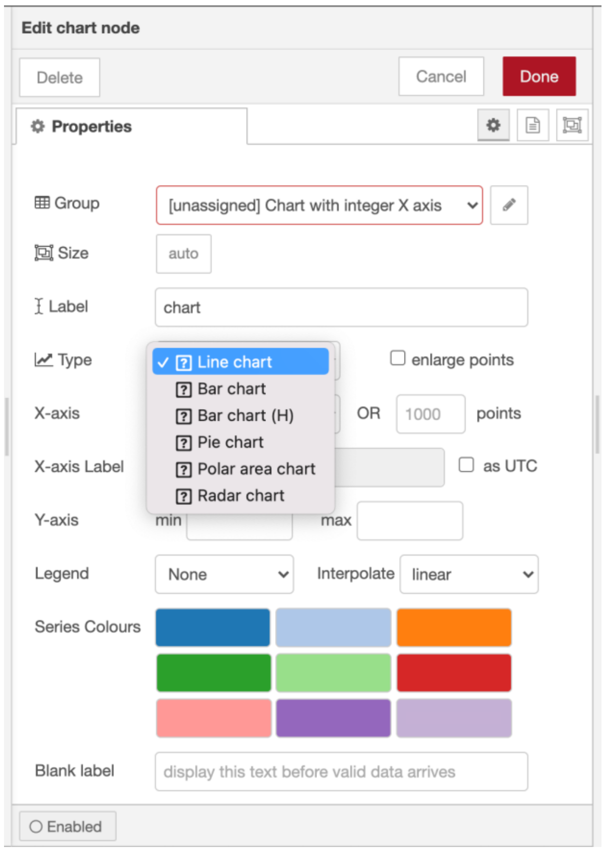
Some of the Dashboard palette available nodes are:

- ui_button: Adds a button widget to the layout.
- ui_dropdown: Adds a dropdown list widget to select an option.
- ui_switch: Adds a switch widget to the layout.
- ui-slider: Adds a slider widget where several options can be added to select.

[21] https://flows.nodered.org/node/node-red-dashboard

- ui_numeric: Shows a number in the dashboard.
- ui_text_input: Adds text to the payload message in the flow.
- ui_date_picker: Shows a calendar and allows to pick a date or range of dates.
- ui_colour_picker: Color selection.
- ui_form: Adds a customized form to the dashboard.
- ui_text: Adds unmodifiable text to the dashboard.
- ui_gauge: Adds a gauge widget to the dashboard.
- ui_chart: Adds a chart to the dashboard.
- ui_audio: Allows to include audio to the dashboard

The ui_form, ui_text_input and ui_date_picker nodes are used to provide information to the Node-RED editor to be used in the workflow creation. The rest of nodes are used for presenting the results produced by SQL queries and analytical workflows in the dashboard. The ui_chart node configuration edit dialog is used for selecting the type of chart for displaying the results. The charts currently available are: line chart, bar chart, pie chart, polar area chart and radar chart. Figure 18 shows the UI-Chart configuration node edit dialog with the available chart types listed.



Edit chart node

Group: [unassigned] Chart with integer X axis

Size: auto

Label: chart

Type: [✓ Line chart] / Bar chart / Bar chart (H) / Pie chart / Polar area chart / Radar chart

enlarge points

X-axis OR 1000 points

X-axis Label as UTC

Y-axis min max

Legend: None

Interpolate: linear

Series Colours: 🟦 🟦 🟧 🟩 🟩 🟥 🟥 🟪 🟪

Blank label: display this text before valid data arrives

Enabled

Figure 18: UI-Chart node configuration edit dialog

Figure 19 shows the Healthentia patient data Custom Dashboard designed in Figure 15, Figure 16 and Figure 17, where data for a particular patient obtained from the iHelp Big Data Platform using the iHelp Storage nodes is being retrieved and represented using linear, bar and radar charts in a custom dashboards that can be accessed by the model builder and physician/health care professional users.

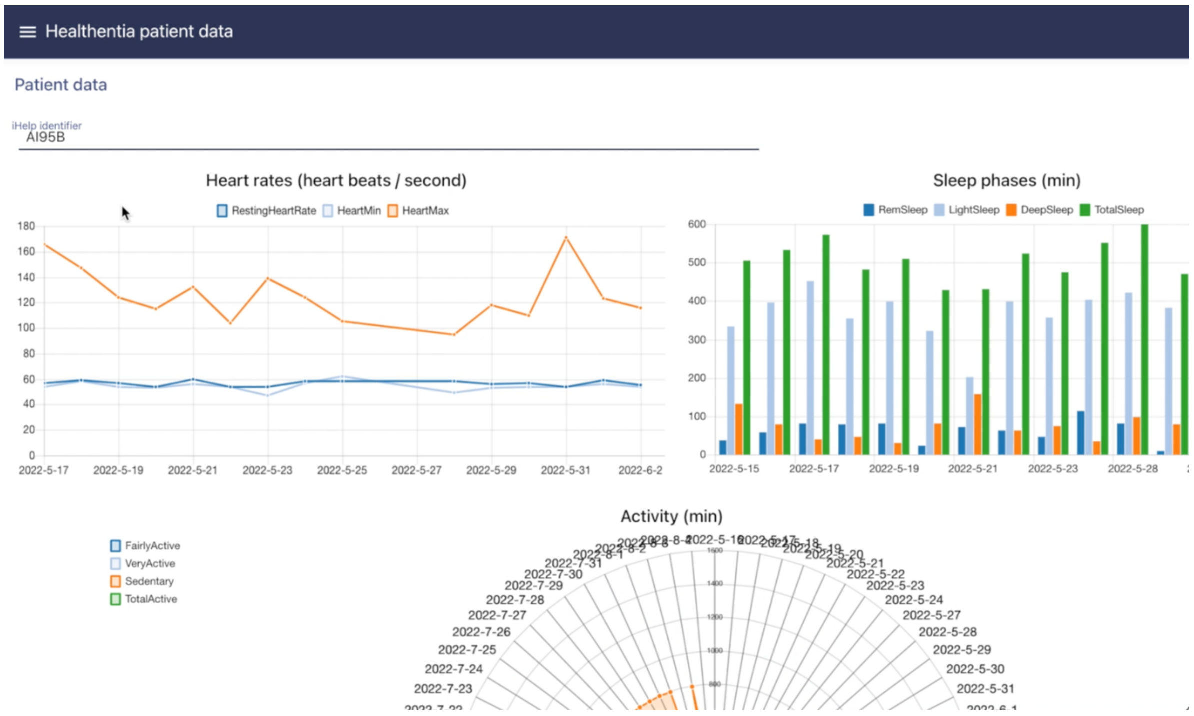


Figure 19: Healthentia patient data custom dashboard example

4.1 Model builder Interaction

This section presents the interaction diagrams for creating dashboards and workflows and the visualization of results within a visualization widget in a previously created dashboard.

When the model builder creates a dashboard for a physician/health care professional or policy maker, she starts creating a panel tab in the Dashboard panel of the sidebar of the Node-Red workflow interface and creates a group in the new panel tab. The group is used to put widgets together and define their location in the dashboard. Once the group is created the user can drag and drop a button widget node into the Node-RED editor, access the configuration node edit dialog and assign the button to the created group, selecting the group name from the dropdown list that appears, as shown in Figure 21. Finally, the user clicks the deploy button of the Node-RED editor and the new dashboard is deployed and accessible from the web browser (Figure 22). The whole interaction is depicted in Figure 20.

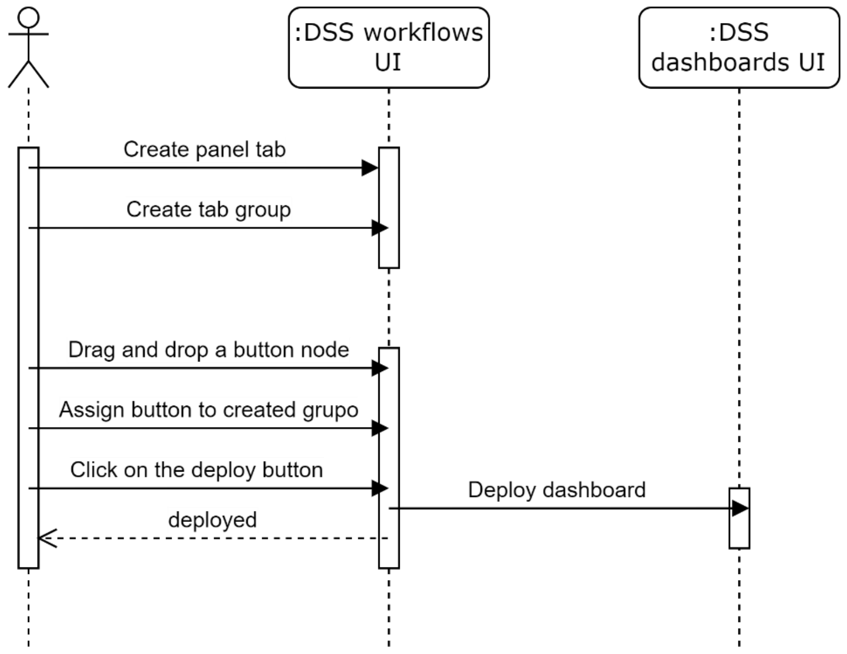
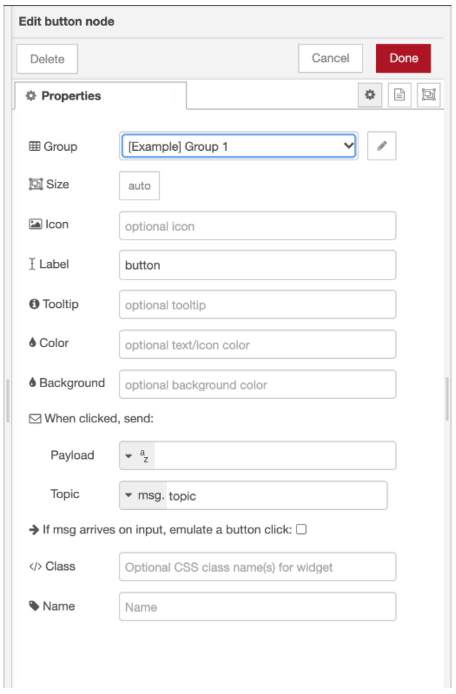


Figure 20: Dashboard definition and button widget addition example



Group	[Example] Group 1
Size	auto
Icon	optional icon
Label	button
Tooltip	optional tooltip
Color	optional text/icon color
Background	optional background color

When clicked, send:

Payload	ᵃz
Topic	msg. topic

If msg arrives on input, emulate a button click: ☐

Class	Optional CSS class name(s) for widget
Name	Name

Figure 21: Button widget node configuration edit dialog

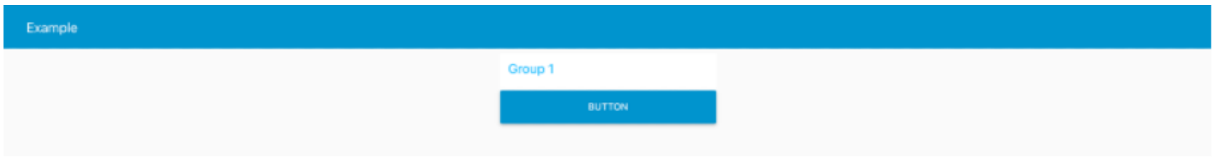


Figure 22: Dashboard and widget creation example

5 Models Explainable Dashboard Hub (EDH)

5.1 Introduction

The goal of eXplainable Artificial Intelligence (XAI) approach is to help users understand AI, learn from it, and utilize it in new ways. It improves AI systems' readability, interpretability, and accountability. Hence, in order humans to comprehend an AI system and its judgments, more explicable models must be developed. Furthermore, it helps health care professionals make better decisions by enhancing their understanding and confidence in the data they receive and analyse. In this respect, HCPs can improve their confidence in the decisions made by AI models, because XAI enables others to observe what occurs within the "black box" and understand how derived decisions were reached. A most difficult aspect is to generate new insights, but the most crucial aspect is organizing these insights in a way that is edible, repeatable, updatable, and would record the impact of change.

To this end, the design and implementation of Explainable Dashboards play a vital role on the enhancement of the interpretability and explainability of the developed AI models. To this extend, an Explainable Dashboard Hub (EDH) is introduced and utilized as part of the whole DSS suite. Several dashboards are used to monitor metrics and outcomes of the AI models, and they provide easy-to-understand approaches of visualizing both the data and the insights derived from the utilization of different AI models. More specifically, HCPs would be able to select multiple models for numerous cases, to view numerous different visualizations and model comparisons. This will help the HCPs to enhance their knowledge and understanding on models' outcomes. In that context, the present document introduces and details the design and implementation of the EDH, that contains many different Explainable Dashboards for each one of the models that are being provided by the iHelp Platform, as well as comparisons of these different models for different cases. It is an interactive framework with user friendly interface that the HCP can gain knowledge with an organized way, through various plots, diagrams etc. In the below subsections the technical approach that is followed, and the overall workflow of the EDH are detailed, as well as the first software prototype and the main components of the EDH. Finally, the baseline technologies that have been used are also presented in more details.

5.2 Technical Approach

The overall technical approach is presented in Figure 23, where an EDH is depicted that contains several different Explainable Dashboards with model performance metrics, insights, plots, and various charts for explaining results along with the integrations and the overall proposed workflow. It is integrated with either external AI models that are provided through pickle files or with the Analytic Workbench through related APIs. All the Explainable Dashboards will be organized and summarized in one single place, the EDH that provides a user-friendly interface so that HCPs may interact with the different available dashboards. In addition, it is going to be embedded into the DSS suite to be presented in a holistic and integrated way. It follows a dockerized and containerized approach, integrated with Kubernetes, so that it is aligned with the overall infrastructure of the iHelp project. More specifically, the technical approach is consisted by the following four parts that are presented in the below subsections.

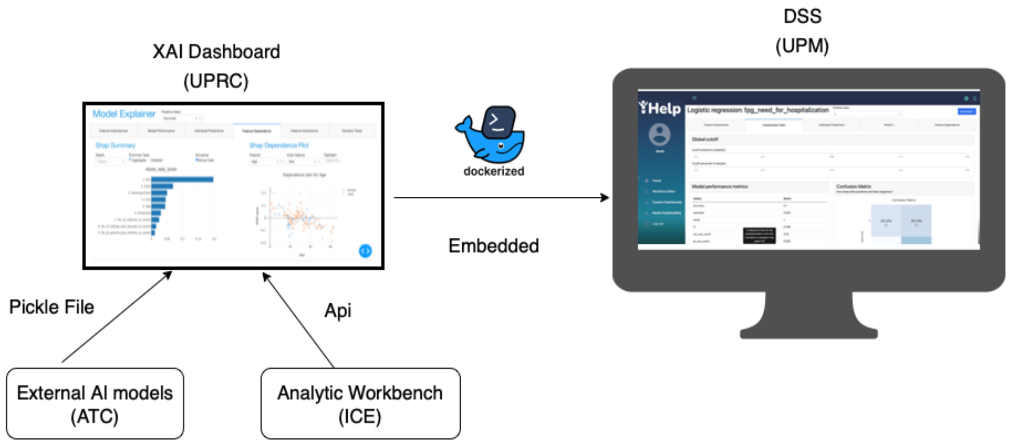


Figure 23: EDH technical approach overview

5.2.1 Explainers from AI models

The data analytics models implemented in the context of the project are provided along with the train and test datasets via the Analytic Workbench component. At this phase of the project, the EDH obtains models either from static pickle files or via a corresponding API, in case these models have been incorporated and are available through the Analytic Workbench. The EDH then fits the models with the datasets into a “explainer file”, a file that maintains the information for a particular model, including the diagrams, analytics, and plots that will be applied in the corresponding Explainable Dashboard. Hence, the EDH is an HTML interactive representation of every explainer file.

5.2.2 Explainable Dashboards

Each model's analytics after being translated into an explainer file, is displayed in a dashboard that constitutes the Explainable Dashboard. Every single Explainable Dashboard delivers statistics and graphics that are tailored to the model's requirements. For each model type, such as logistic regression, decision trees, xgboost etc., the dashboard generates graphical representations of their various different outcomes. Through the dashboard, HCPs can easily select the models which they wish to analyse, as well as to perceive visually the comparisons between models for specific cases. Respective examples of the visualization of such dashboards are presented in Figure 24.

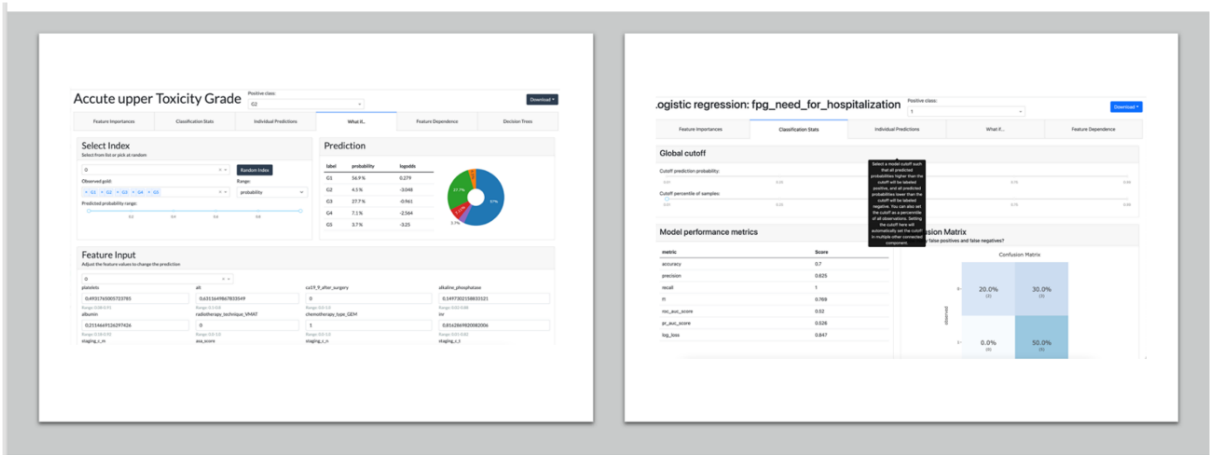

Accute upper Toxicity Grade
Select Index
Prediction
Feature Input
ogistic regression: fpg_need_for_hospitalization
Global cutoff
Model performance metrics
Confusion Matrix

Figure 24: Examples of Explainable Dashboards

5.2.3 EDH embedded into the DSS

The implementation of the EDH contains one main dashboard, which is the EDH that summarizes and organizes many different Explainable Dashboards for various AI models in the form of cards with related information, model types and origin, in one single place, as depicted in Figure 26. EDH also includes an easy-to-select filter for HCPs to be redirected to a model comparison page. The model comparison page may compare two different models, at a time, and present their corresponding diagrams side by side. The interface and implementation for the EDH will be incorporated into the DSS suite via a corresponding iframe and the respective URL endpoint under which the EDH would be available. These two components, i.e, EDH and DSS, will be dockerized via the same docker compose file and a specific pod will be available for each one of these components, as is being described in the next subsection.

5.2.4 Deployment of EDH

The process of containerizing an application with Docker starts with the creation of a Docker Image including both DSS code and EDH components. The dockerized and containerized method was selected because it simplifies the creation, deployment, and execution of applications by using containers. And containers will allow the DSS suite and explainable hub application to be packaged with all of its necessary components, including as libraries and dependencies, and shipped as a single package. By doing so, it is guaranteed that the apps will work on any other system, independent of any specific settings the machine may have that vary from the one used for coding and testing. Docker containers make it possible to version-control Docker images without the need to transport them across systems individually. Additionally, Docker reduces deployment time to seconds. It generates a container for each process and does not boot an operating system. It is possible to produce and delete data without fear that the expense of recovering it would be prohibitive. Finally, the EDH and DSS containers are going to be integrated with the project's Kubernetes cluster to be align with the overall infrastructure of the iHelp project.

5.3 Prototype Overview

The prototype of the EDH is presented and discussed in this section. For every model provided by ATC and validated on FPG data, a dashboard that is either automatically generated or customized to meet the demands of each model has been developed. EDH is a collection of tools for rapidly creating interactive dashboards with several visualizations for evaluating and presenting the forecasts and processes of (scikit-learn compatible) ML models, such as xgboost, catboost and lightgbm. All these dashboards are analysed and organized on a single page, the EDH, a web-based interface that will be incorporated into the DSS suite. Moreover, the EDH includes cards with info (Title and type) of many Explainable Dashboards which are organized and summarized in one place, as depicted in Figure 25. In addition, this first software prototype provides the option to select and compare two different models in an interpretable way. More specifically, the EDH incorporates "comparison pages" that present two models' statistics and diagrams, so the HCPs can easily choose the models they want to view and analyze. This enables them to have an overview of the importance of specific features, allowing them to reach faster conclusions about the most significant factors in pancreatic cancer, based on the comparison of the diagrams. The main functionalities of the EDH are:

- Explainable hub: Assessing Explainer Dashboards for all models and frameworks through a single location/dashboard.
- Filtering through the models.
- Provision of different modified and integrated visualizations.

- Various statistics and diagrams for every model, which are presented in an interactive way.
- What-If (in case it is turned on, when starting the dashboard) to help understand the changes in the model behavior, if the features or parts of the data are modified. It also allows for the comparison of different models.

Every single Explainable Dashboard interface contains different visualization tabs. Each tab includes different visualization categories, such as feature importance, classification stats and what-if analysis. More specifically, each Explainable Dashboard consists of the below tabs/visualizations:

- SHAP Values that illustrate how each factor individually influences the forecast.
- Feature importance that enables HCPs to go deeper into observing how model performance alters as a feature is shuffled.
- In the case of a Regression model utilizing XGBoost or RandomForestRegressor it makes possible to visualize the individual decision trees, whereas in the case of Classifier models, it could provide confusion matrices, ROC-AUC curves, etc., to better comprehend the models' decisions.
- What-If analysis (enabled before launching the dashboard) can help in understanding how the model's behavior varies when characteristics or portions of the data are altered. Additionally, it enables HCPs to compare various models.

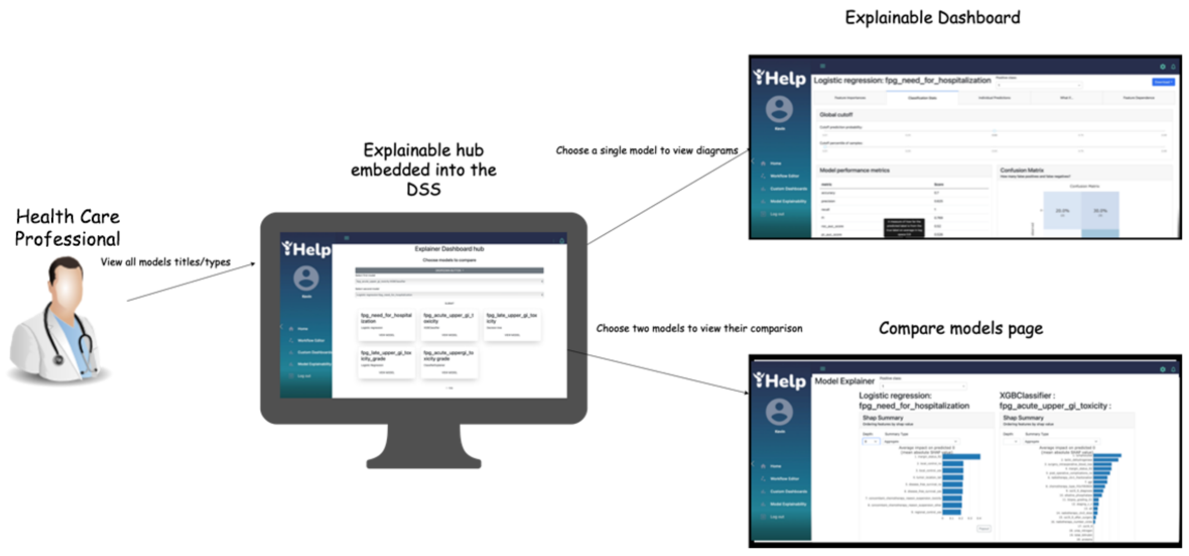
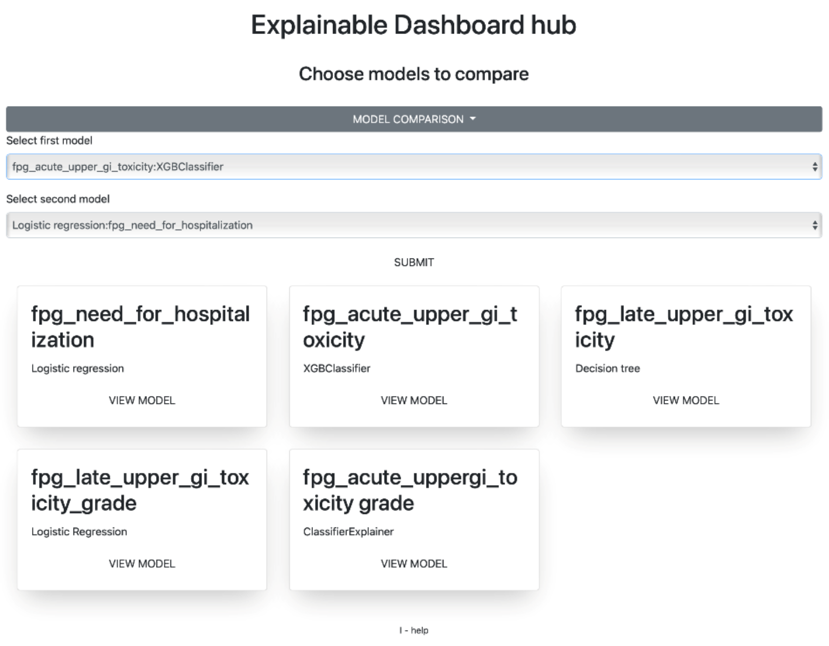

Explainable Dashboard
Explainable hub embedded into the DSS
Choose a single model to view diagrams
Health Care Professional
View all models titles/types
Choose two models to view their comparison
Compare models page
Logistic regression: fpg_need_for_hospitalization
Model Explainer
XGBClassifier : fpg_acute_upper_gi_toxicity
Explainer Dashboard hub

Figure 25: Explainable Dashboard Hub prototype overview

5.4 Interfaces

This section describes the interfaces included on the Explainable Dashboard Hub. The interfaces, as already mentioned, are embedded into the DSS suite. The main page of the EDH is a hub that contains cards with the info of each model. Specifically, the basic interface presents a model including its title and description, as well as a link that leads to the model's single Explainable Dashboard page, as depicted in Figure 26. In addition, Figure 27 depicts the Model Comparison tool that allows the HCPs to compare two different models to identify, for instance, the importance and impact of the different data attributes in those models.


Explainable Dashboard hub

Choose models to compare

Select first model

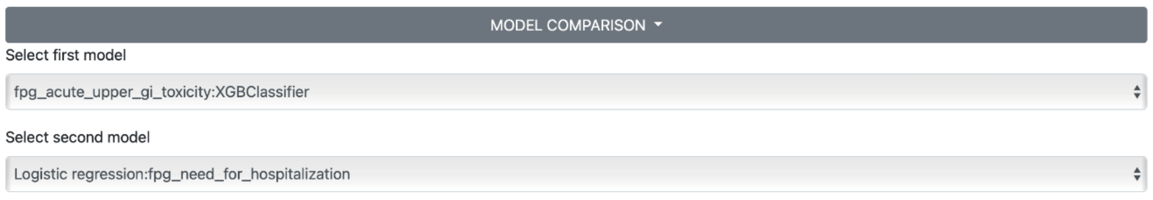
fpg_acute_upper_gi_toxicity:XGBClassifier

Select second model

Logistic regression:fpg_need_for_hospitalization

fpg_need_for_hospitalization Logistic regression VIEW MODEL	fpg_acute_upper_gi_toxicity XGBClassifier VIEW MODEL	fpg_late_upper_gi_toxicity Decision tree VIEW MODEL
fpg_late_upper_gi_toxicity_grade Logistic Regression VIEW MODEL	fpg_acute_uppergi_toxicity grade ClassifierExplainer VIEW MODEL	

Figure 26: Explainable Dashboard Hub (EDH)



Select first model

fpg_acute_upper_gi_toxicity:XGBClassifier

Select second model

Logistic regression:fpg_need_for_hospitalization

Figure 27: Model comparison tool

Every single Explainable Dashboard is an interactive web page application that extends the EDH and is stored in a specific link. The EDH, additionally, has pages with two separate model visualization diagrams. This enhancement enables the HCPs to compare different outcomes based on distinct attributes. The user can select the models to view the comparisons through dropdown buttons, as depicted in Figure 28.

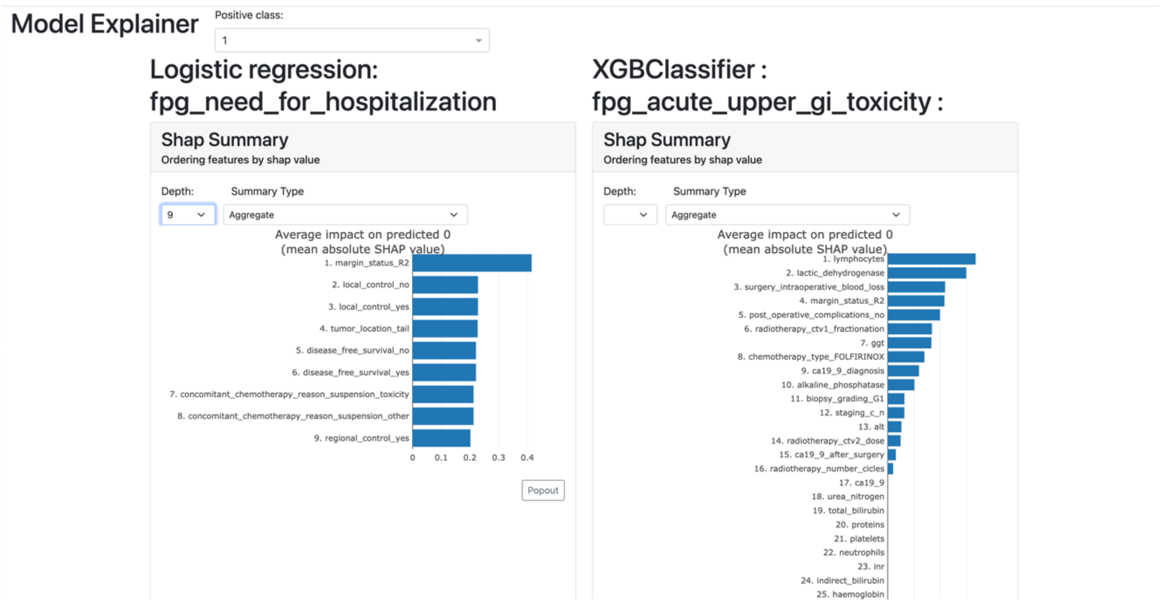

Model Explainer
Positive class:
1
Logistic regression: fpg_need_for_hospitalization
XGBClassifier : fpg_acute_upper_gi_toxicity :
Shap Summary
Ordering features by shap value
Depth:
Summary Type
9
Aggregate
Average impact on predicted 0 (mean absolute SHAP value)
1. margin_status_R2
2. local_control_no
3. local_control_yes
4. tumor_location_tail
5. disease_free_survival_no
6. disease_free_survival_yes
7. concomitant_chemotherapy_reason_suspension_toxicity
8. concomitant_chemotherapy_reason_suspension_other
9. regional_control_yes
0
0.1
0.2
0.3
0.4
Popout
Shap Summary
Ordering features by shap value
Depth:
Summary Type
Aggregate
Average impact on predicted 0 (mean absolute SHAP value)
1. lymphocytes
2. lactic_dehydrogenase
3. surgery_intraoperative_blood_loss
4. margin_status_R2
5. post_operative_complications_no
6. radiotherapy_ctv1_fractionation
7. ggt
8. chemotherapy_type_FOLFIRINOX
9. ca19_9_diagnosis
10. alkaline_phosphatase
11. biopsy_grading_G1
12. staging_c_n
13. alt
14. radiotherapy_ctv2_dose
15. ca19_9_after_surgery
16. radiotherapy_number_cicles
17. ca19_9
18. urea_nitrogen
19. total_bilirubin
20. proteins
21. platelets
22. neutrophils
23. inr
24. indirect_bilirubin
25. haemoglobin

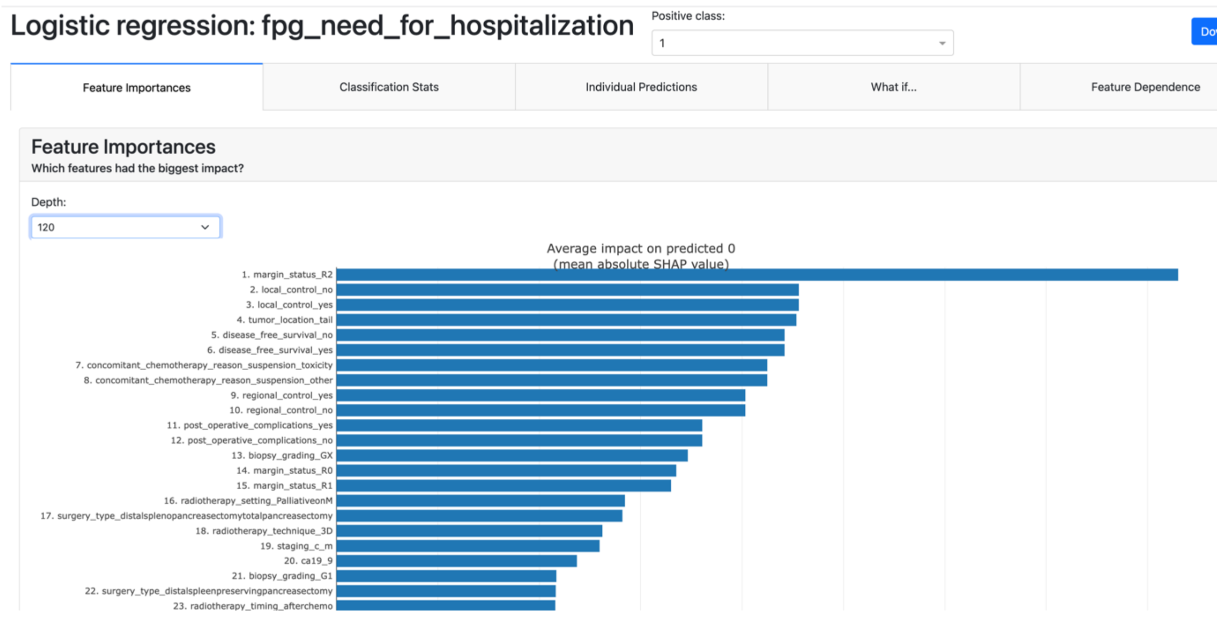
Figure 28: Model comparison page

5.5 Main Components

As mentioned also previously, the EDH is a collection of tools for rapidly creating interactive dashboards with several visualizations for evaluating and presenting the forecasts and processes of ML models, such as xgboost, catboost and lightgbm etc. Depending on the type of the model, every single Explainable Dashboard has a single link and has tabs with specific main components. By main components is meant the visualization categories of the diagrams. Every main component category can be accessed by the HCPs through corresponding tabs in an interactive way. The main components are varied to fit the needs of each model. For example, in the case of the Decision tree models, one more tab may appear.

5.5.1 Feature importance

The term “Feature Importance” is used to characterize methods that provide a value to each of a model's input characteristics, with those values effectively representing the "importance" of those features to the outcome. More weight is given to the feature in the model's prediction that has a higher score. The feature importance visualization, as is presented in Figure 29, showcases with each blue bar the importance of each feature to the outcome. For instance, in the case of the logistic regression model of “fpg_need_for_hospitalization” the feature with the biggest impact is “margin_status R2”, as depicted in Figure 29. Additionally, in this case, the presence of margin status r2-specific values indicates that the prediction about whether a patient requires hospitalization is impacted more than any other feature.


Logistic regression: fpg_need_for_hospitalization

Positive class: 1

Feature Importances

Which features had the biggest impact?

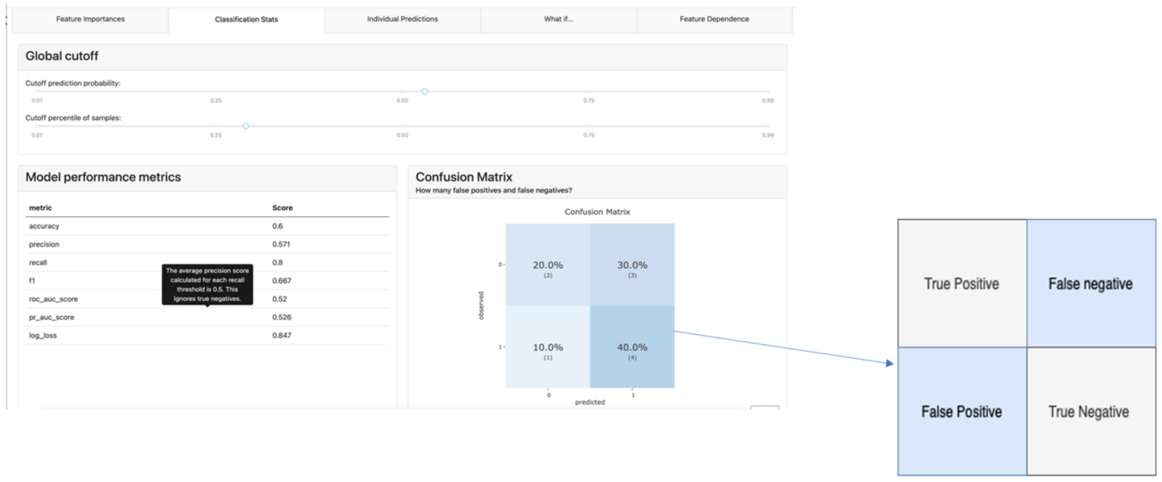
Depth: 120

Feature Importances
Classification Stats
Individual Predictions
What if...
Feature Dependence
Average impact on predicted 0
(mean absolute SHAP value)
1. margin_status_R2
2. local_control_no
3. local_control_yes
4. tumor_location_tail
5. disease_free_survival_no
6. disease_free_survival_yes
7. concomitant_chemotherapy_reason_suspension_toxicity
8. concomitant_chemotherapy_reason_suspension_other
9. regional_control_yes
10. regional_control_no
11. post_operative_complications_yes
12. post_operative_complications_no
13. biopsy_grading_GX
14. margin_status_R0
15. margin_status_R1
16. radiotherapy_setting_PalliativeonM
17. surgery_type_distalsplenopancreasectomytotalpancreasectomy
18. radiotherapy_technique_3D
19. staging_c_m
20. ca19_9
21. biopsy_grading_G1
22. surgery_type_distalspleenpreservingpancreasectomy
23. radiotherapy_timing_afterchemo

Figure 29: Feature Importance visualization

5.5.2 Classification Status

This Explainable Dashboard contains various diagrams and metrics visualizations that describe the model value, as it is presented in Figure 30. Model performance metrics table contains Accuracy score, precision, Recall, F1, Roc AUC score, PR AUC score and Log loss. It also presents a confusion matrix figure, that shows the percentage of true positive, false negative, false positive and true negative predictions of the classification model.


Global cutoff

Cutoff prediction probability:

Cutoff percentile of samples:

Model performance metrics

metric	Score
accuracy	0.6
precision	0.571
recall	0.8
f1	0.667
roc_auc_score	0.52
pr_auc_score	0.526
log_loss	0.847

The average precision score calculated for each recall threshold is 0.5. This ignores true negatives.

Confusion Matrix

How many false positives and false negatives?

Figure 30: Classification Status visualization

Furthermore, it contains a precision plot and a classification plot as depicted in Figure 31. Precision is one metric of the performance of a ML model that indicates the quality of a model's accurate prediction sand is the ratio of the number of genuine positives to the overall number of positive forecasts (i.e., the number of true positives plus the number of false positives).

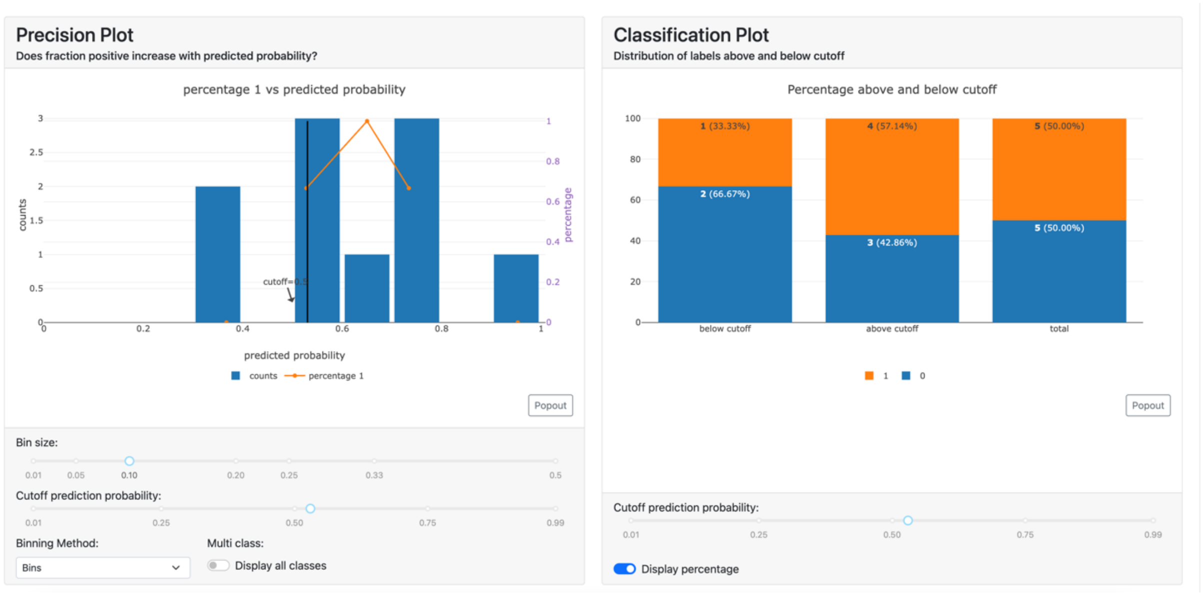


Figure 31: Precision and Classification plots

There is also a Lift curve that demonstrates the relationship between the number of instances that were predicted to be positive, and the actual number of positive examples as indicated in Figure 32. The latter also enables the comparison of the performance of a selected classifier to that of a random classifier and, finally, a cumulative precision diagram is presented.

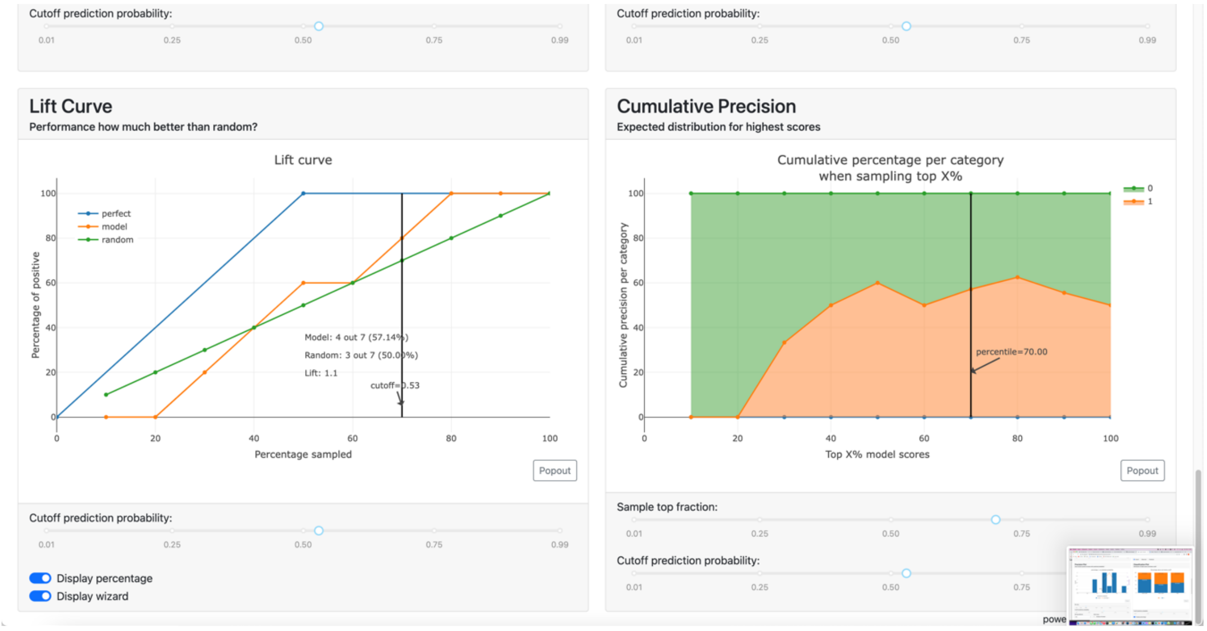
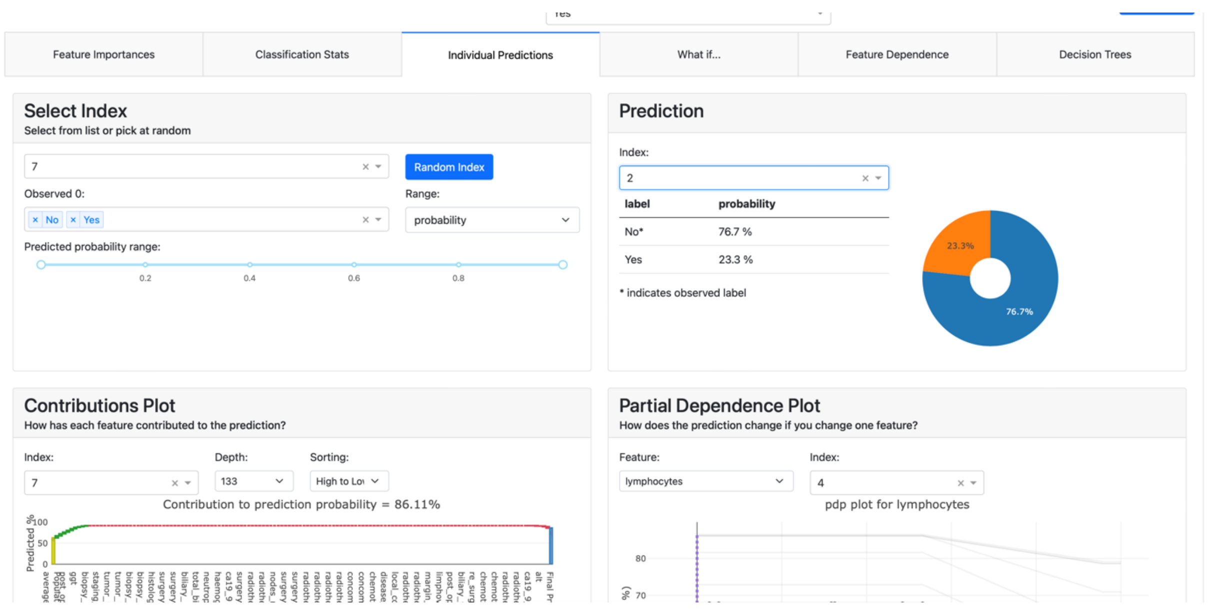


Figure 32: Lift cure and Cumulative Precision plots

5.5.3 Individual Predictions

This type of visualization contains interactive forms, pies, and diagrams, as presented in Figure 33. The HCPs could choose to see the influence of each feature in the model outcomes or combinations of many different inputs. Moreover, the individual predictions visualization includes a pie that presents the probability of each class align with the index of the dataset.



Select Index
Select from list or pick at random

7

Observed 0: No, Yes

Range: probability

Predicted probability range:

Prediction

Index: 2

label	probability
No*	76.7 %
Yes	23.3 %

* indicates observed label

Contributions Plot
How has each feature contributed to the prediction?

Index: 7
Depth: 133
Sorting: High to Low

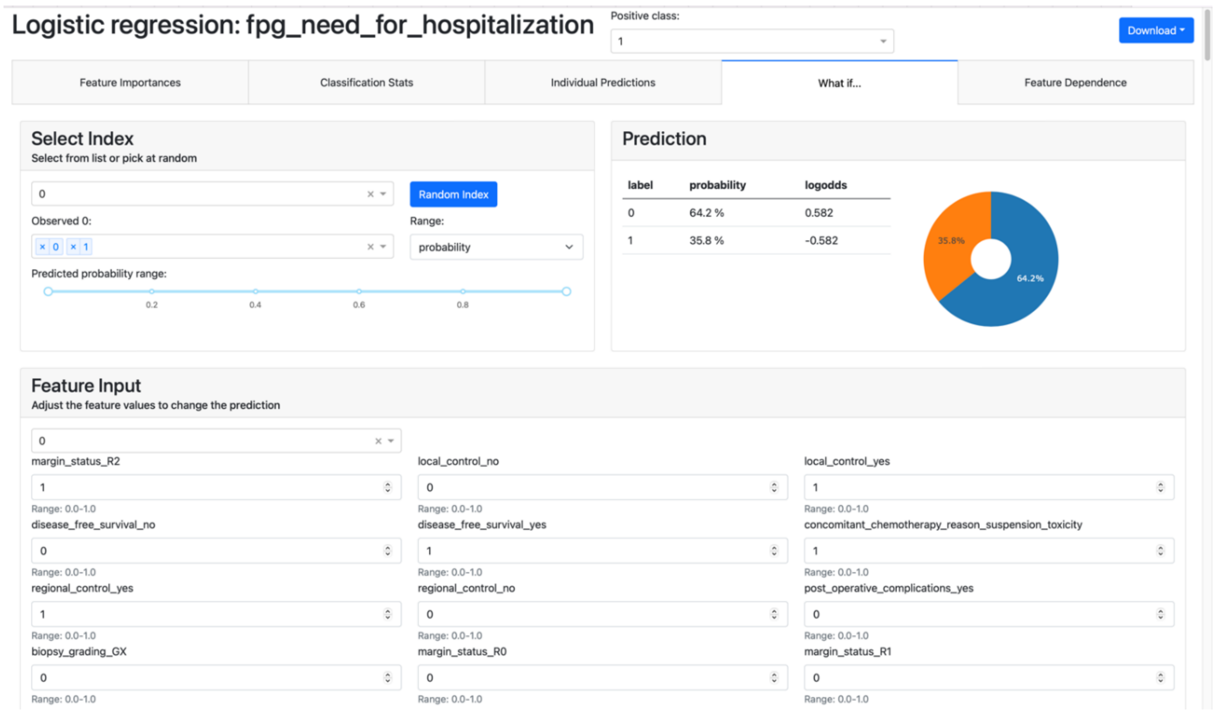
Partial Dependence Plot
How does the prediction change if you change one feature?

Feature: lymphocytes
Index: 4

Figure 33: Individual prediction visualization

5.5.4 What-if

What-If visualizations, depicted in Figure 34, (in case turned on while starting the dashboard) help HCPs to understand the changes in the model behaviour if they modify the specific features or attributes of the data. It also allows them to compare different models.



Logistic regression: fpg_need_for_hospitalization

Positive class: 1

Select Index
Select from list or pick at random

0

Observed 0: 0, 1

Range: probability

Predicted probability range:

Prediction

label	probability	logodds
0	64.2 %	0.582
1	35.8 %	-0.582

Feature Input
Adjust the feature values to change the prediction

0

margin_status_R2: 1 (Range: 0.0-1.0)	local_control_no: 0 (Range: 0.0-1.0)	local_control_yes: 1 (Range: 0.0-1.0)
disease_free_survival_no: 0 (Range: 0.0-1.0)	disease_free_survival_yes: 1 (Range: 0.0-1.0)	concomitant_chemotherapy_reason_suspension_toxicity: 1 (Range: 0.0-1.0)
regional_control_yes: 1 (Range: 0.0-1.0)	regional_control_no: 0 (Range: 0.0-1.0)	post_operative_complications_yes: 0 (Range: 0.0-1.0)
biopsy_grading_GX: 0 (Range: 0.0-1.0)	margin_status_R0: 0 (Range: 0.0-1.0)	margin_status_R1: 0 (Range: 0.0-1.0)

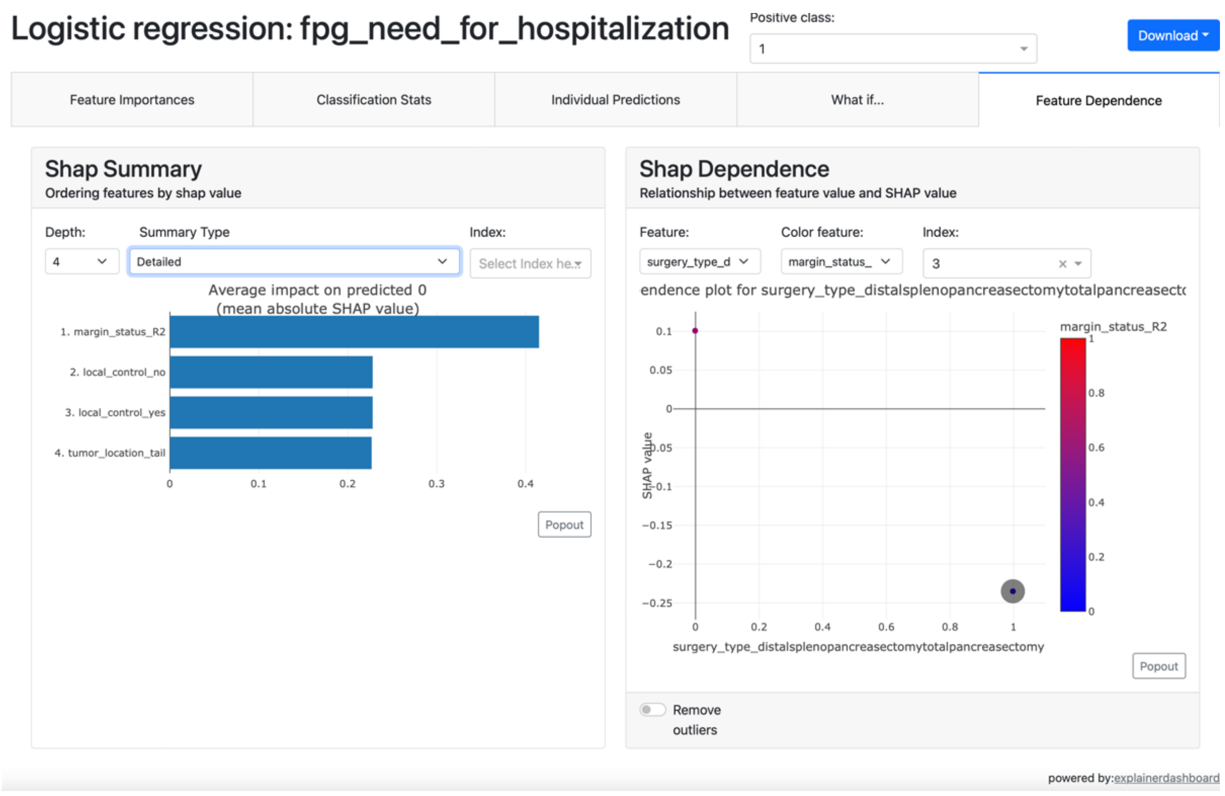
Figure 34: What-if visualization

5.5.5 Feature Dependence

Feature Dependence Explanations are model-agnostic global explanation methods that evaluate the relationship between feature values and model target predictions, hence they offer better explainability and interpretability in the examined analytical results in dependence with the different examined features.

SHAP (SHapley Additive exPlanations) framework provides dependence plots with interaction visualization. SHAP feature dependence might be the simplest global interpretation plot, as it offers the ability to the end-user to pick a feature and interpret the analytical outcomes based on this feature. For each data instance, a point is plotted with the feature value on the x-axis and the corresponding Shapley value on the y-axis.

SHAP values indicate how much a feature altered the outcome (compared to if we made that prediction at some baseline value of that feature). More specifically, SHAP is a game-theoretic method for explaining the results of any ML model. It integrates optimum credit allocation with local explanations by using the standard Shapley values from game theory and associated expansions. Moreover, the SHAP values of a model's output describe the influence of attributes on the outcome. The summary plot combines the importance and the effect of features, while each point on the summary plot represents a Shapley value for an instance and a feature. In the case of the characteristic tumour location tail there are low Shapley values, since this is a least essential feature. More specifically, the bar depicts the features' value from low to high and the characteristics are listed in significance ordering, as depicted in Figure 35 below.



Logistic regression: fpg_need_for_hospitalization

Positive class: 1

Shap Summary
Ordering features by shap value

Depth: 4
Summary Type: Detailed
Index: Select Index he...

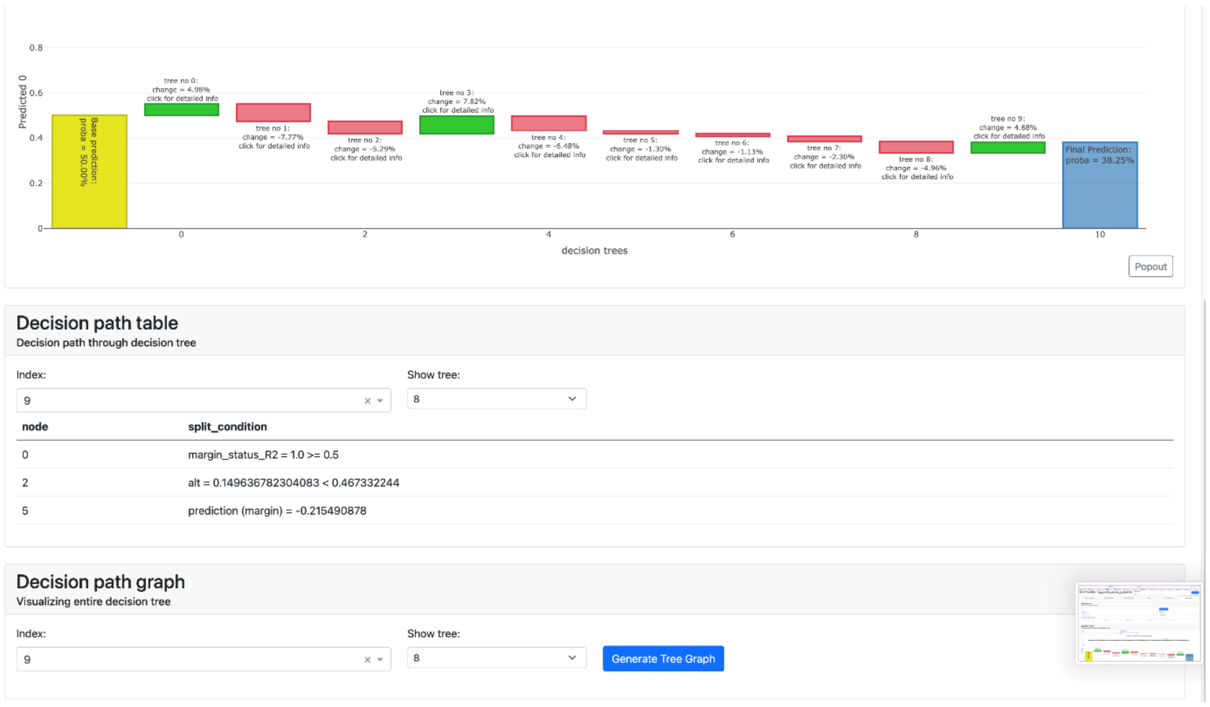
Shap Dependence
Relationship between feature value and SHAP value

Feature: surgery_type_d
Color feature: margin_status_
Index: 3

Figure 35: Feature significance visualization

5.5.6 Decision Tree visualizations

For decision tree-based models, there is the decision tree visualization, as presented in Figure 36, that contains the decision path table. It is a table that shows for every node of the tree the split condition and threshold. A decision tree path is a table depicting the probable consequences of a set of interconnected decisions. It enables a specific list to compare alternative actions based on their costs, probabilities, and benefits. They may be used for, either casual conversation or the development of an algorithm that statistically predicts the optimum option.


0.8
0.6
0.4
0.2
0
Predicted 0
Base prediction: proba = 50.00%
tree no 0: change = 4.98% click for detailed info
tree no 1: change = -7.77% click for detailed info
tree no 2: change = -5.29% click for detailed info
tree no 3: change = 7.82% click for detailed info
tree no 4: change = -6.48% click for detailed info
tree no 5: change = -1.30% click for detailed info
tree no 6: change = -1.13% click for detailed info
tree no 7: change = -2.30% click for detailed info
tree no 8: change = -4.96% click for detailed info
tree no 9: change = 4.68% click for detailed info
Final Prediction: proba = 38.25%
0
2
4
6
8
10
decision trees
Popout

Decision path table

Decision path through decision tree

Index: 9

Show tree: 8

node	split_condition
0	margin_status_R2 = 1.0 >= 0.5
2	alt = 0.149636782304083 < 0.467332244
5	prediction (margin) = -0.215490878

Decision path graph

Visualizing entire decision tree

Index: 9

Show tree: 8

Figure 36: Decision tree visualization

5.6 Baseline Technologies and Tools

This section describes the technologies that have been utilized for the design and implementation of the core technical components of the EDH. The overall workflow and integration between the different technologies and tools is also presented in Figure 37. The EDH has been implemented utilizing Python programming language and its widely used libraries, such as scikit-learn [22] , plotly [23] and explainerdashboard[24]. Other programming languages like HTML, JS and CSS were also utilized to provide a more user-friendly frontend environment to the HCPs. To make the visualization process more efficient, the implementations are divided into two python scripts: generateexplainers.py and flaskapi.py. The first script creates for each model an explainer file with information about the plots, diagrams, and tables that are going to be embedded into the specific explainer dashboard. More specifically it decodes the pickle files that contains the models along with the Xtest, Ytest and save the info of the visualization in the explainer file. The second Python script (flaskapi.py) creates an API that interacts with the both the HCP and Model Builder and embeds the explainers into the presentation level to avoid calculate the results at the same time and be more optimized. Furthermore, it retrieves the explainers' files and generates dashboards, whenever it receives a request. In addition, flaskapi.py has user-requested custom methods that compare models. The information of the Explainable Dashboards is shown in cards, on the main html page (the Explainable Dashboard Hub), which is called from the Flask API using the "render template" method. In the case that an HCP clicks on a certain Explainable Dashboard, a request is sent to the Flask API and the HCP or the Model Builder is redirected to the appropriate link and model visualization.

[22] https://scikit-learn.org/stable/
[23] https://plotly.com/
[24] https://explainerdashboard.readthedocs.io/en/latest/

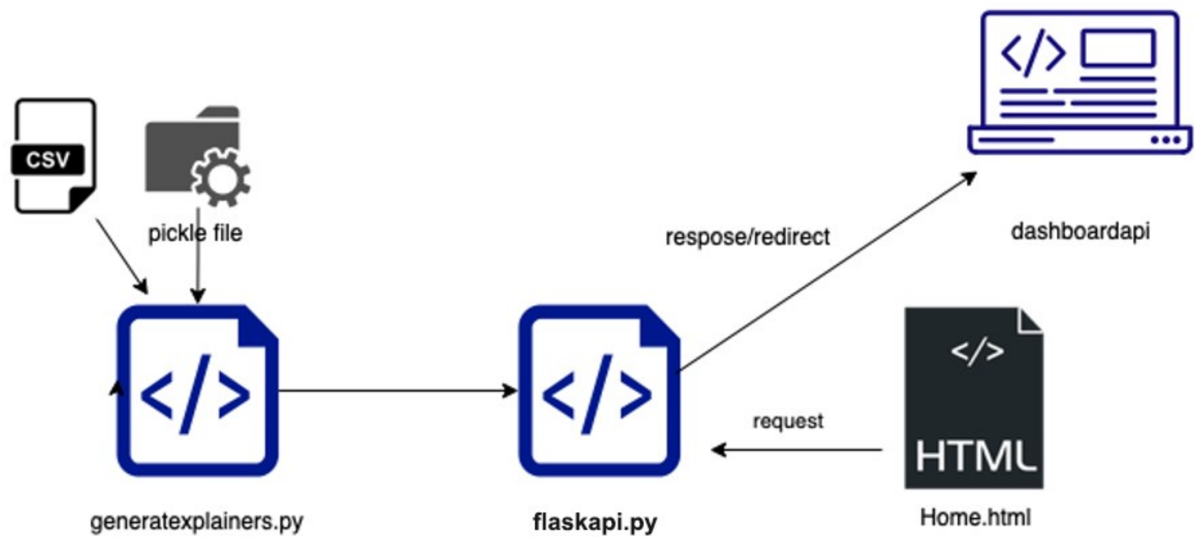


Figure 37: Baseline technologies and tools

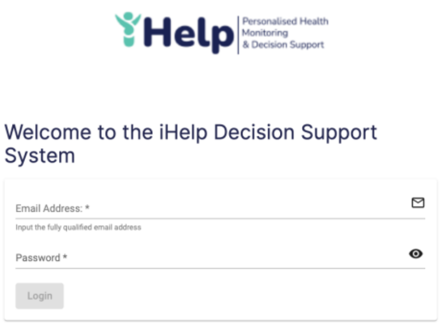
6 Web Interface

The web interface sub-component provides the main interface of the DSS where all DSS sub-components and other iHelp components are integrated. The web interface is based on Angular and Angular materials. A set of interfaces described in D2.7 – "Functional and Non-Functional Specifications II" have been implemented. In particular, the interfaces corresponding to the following sequence diagrams have been implemented:

- Patient enrolment using DSS Dashboard
- Data visualisation
- Risk assessment
- Risk mitigation/treatment planning
- Advice review

6.1 Log in interface

The log in interface, presented in Figure 38, allows the iHelp users to authenticate into the iHelp platform. At this point of the project, two different user roles are contemplated, the model builder and the health care professional. The log in interface will be integrated with the iHelp's Identity Manager component, as introduced in the context of D2.5 – "Conceptual model and reference architecture II".



Welcome to the iHelp Decision Support System

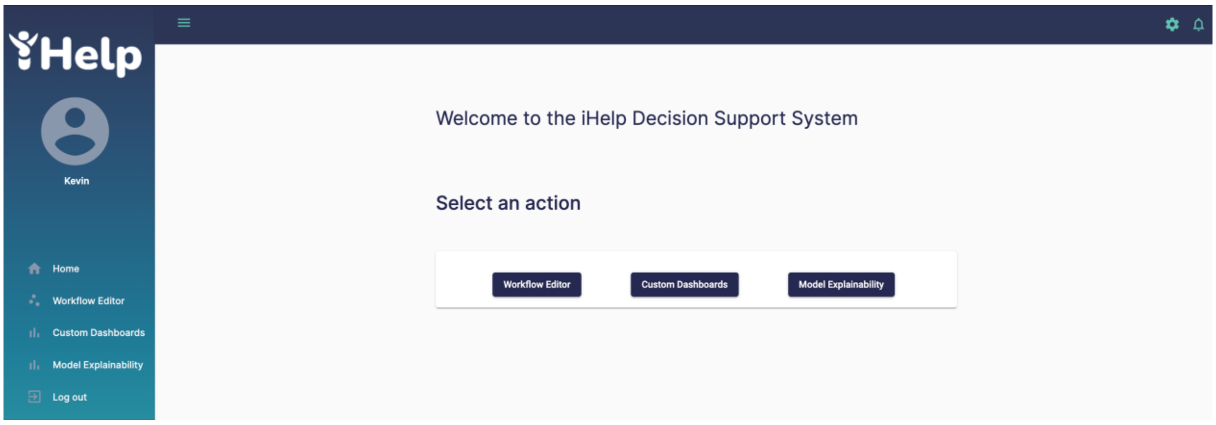
Figure 38: Login page

6.2 Model builder user interfaces

The model builder user interacts with the DSS to study different AI models provided by the model library (Analytic Workbench, Personalized Predictor and Predictor and Risk Identifier components), and analyses the results obtained through the Model Explainability interface (Section 6.4). The model builder user can also query data stored in the Big Data Platform, represent the data using different types of charts and implement custom dashboards that can be accessed by other model builder users and health care professionals (Section 6.4). This last functionality differentiates the DSS from other tools available on the market. During this section, the Menu and the Workflow Editor interfaces are presented.

6.2.1 Menu interface

Once the model builder user has accessed the system, he/she can see three functionalities: the Workflow Editor, the Custom Dashboards and the Model Explainability. Figure 39 shows the appearance of the Menu interface where the three functionalities can be accessed from the buttons in the central part of the screen and the menu on the left part of the screen.



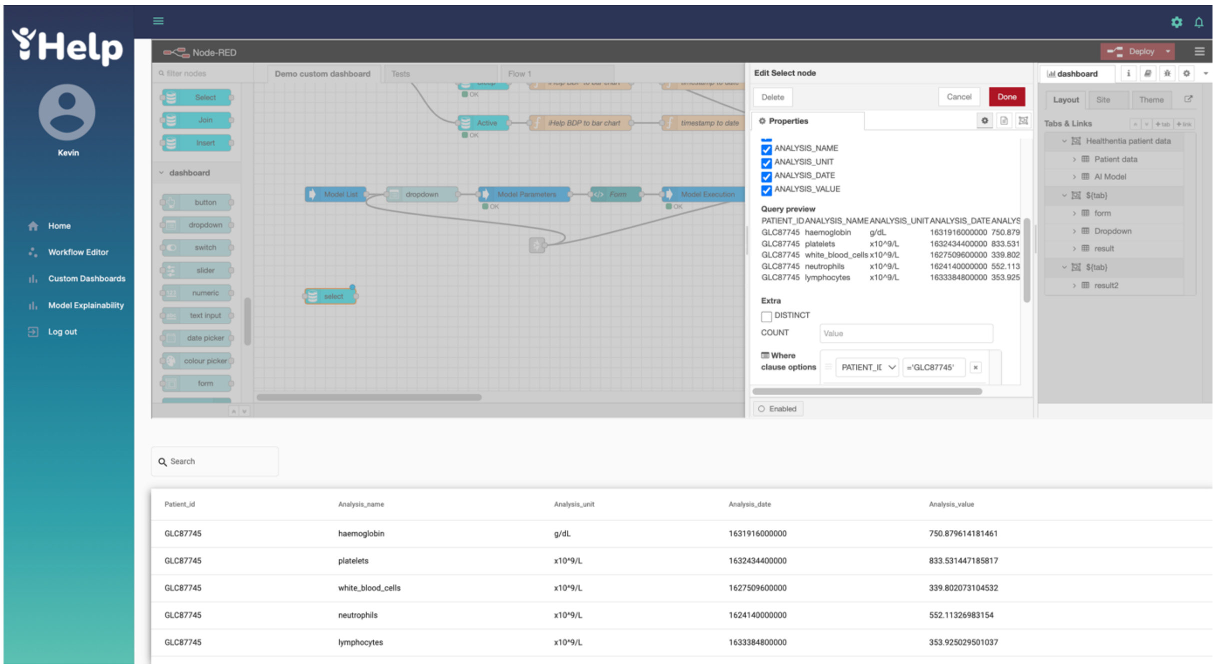
Welcome to the iHelp Decision Support System

Select an action

Figure 39: Model builder user menu interface

6.2.2 Workflow Editor interface

The Node-RED editor interface is embedded into the DSS and connected with it as shown in Figure 40. This interface allows the model builder to observe the data stored in the iHelp storage by creating SQL like query flows with the database palette nodes and preview the results of each node in the table presented below the Node-RED editor interface. This table changes dynamically, this means that when a new database node is added to the flow, the table preview the result of the query configured on it.



Patient_id	Analysis_name	Analysis_unit	Analysis_date	Analysis_value
GLC87745	haemoglobin	g/dL	1631916000000	750.879614181461
GLC87745	platelets	x10*9/L	1632434400000	833.531447185817
GLC87745	white_blood_cells	x10*9/L	1627509600000	339.802073104532
GLC87745	neutrophils	x10*9/L	1624140000000	552.11326983154
GLC87745	lymphocytes	x10*9/L	1633384800000	353.925029501037

Figure 40: Model builder Workflow Editor interface

6.3 Health care professional user interface

The health care professional user interacts with the DSS by adding and editing new patients into the iHelp program. Visualizing the individuals' information, analysing the results obtained from the AI models and accessing the custom dashboards interface. Within this section the menu, patient enrolment, patient edit and patient visualization interfaces are going to be presented.

6.3.1 Menu interface

The health care professional has access to a set of functionalities: the patient visualization, the patient enrolment, patient edit, model explainability and custom dashboards. The access can be done through the buttons in the centre of the screen and the menu on the left as depicted in Figure 41.

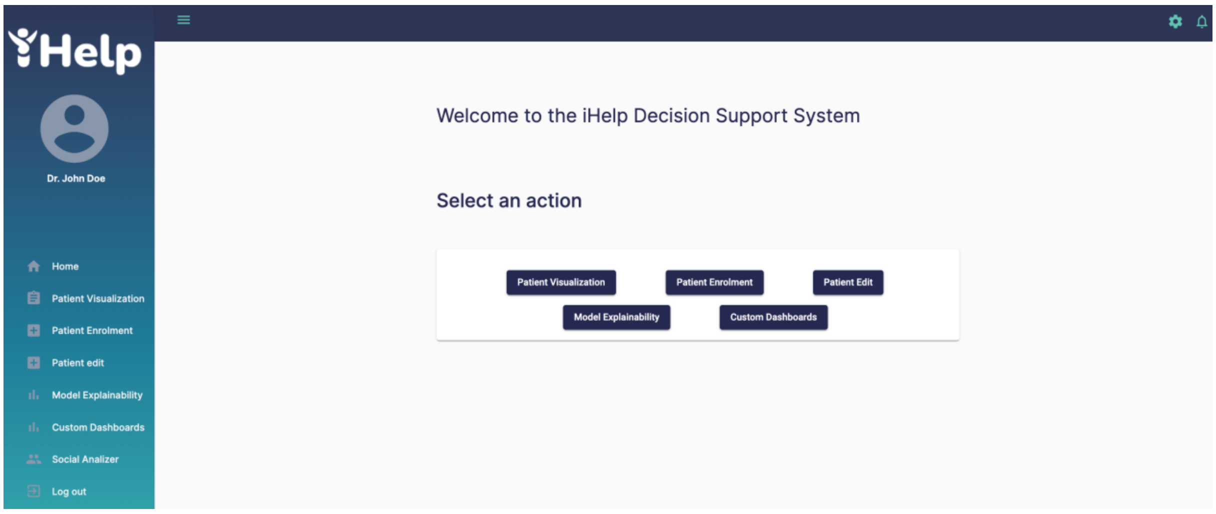
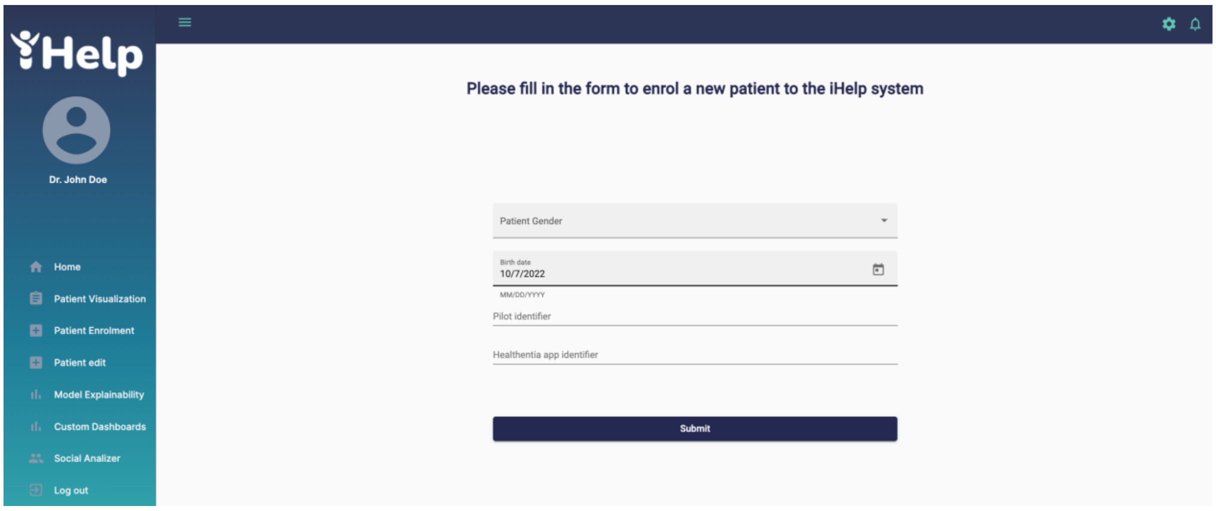


Figure 41: Health care processional user menu interface

6.3.2 Patient Enrolment interface

Figure 42 shows the patient enrolment interface that allows the health care professional to register new patients into the iHelp program. Within the form, the health care professional has to provide the gender, birth date, the pilot and the Healthentia app identifiers. The pilot identifier is the patient identifier within the hospital database. The Healthentia app identifier is the identifier that the mobile application has assigned to the patient once he/she has registered. In this way, the data provided by the hospital is aggregated with the data provided by the mobile application without registering the identity of the patients within the platform. If the patient is not registered in the mobile application at the time of the enrolment this field can be added later through the patient edit interface. When the patient is registered an iHelp identifier is provided and this will be the unique identifier used for the patient within the iHelp platform.



Please fill in the form to enrol a new patient to the iHelp system

Patient Gender

Birth date 10/7/2022
MM/DD/YYYY

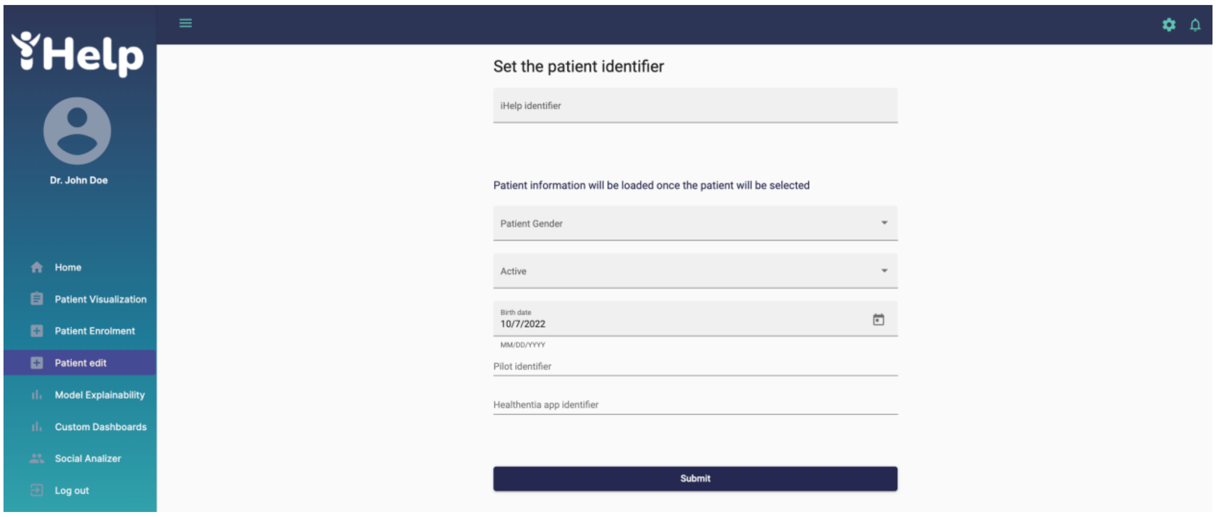
Pilot identifier

Healthentia app identifier

Figure 42: Patient enrolment interface

6.3.3 Patient Edit interface

The patient edit interface shown in Figure 43 allows the health care professional update the information registered by the patient in the iHelp platform. The health care professional has to enter the iHelp identifier of the patient and automatically the fill is loaded with the data stored in the Big Data Platform. The gender, birth date, pilot identifier and Healthentia app identifiers can be updated. Moreover, an extra field appears, this Active field is set to true if the patient continues in the iHelp program or false if the patient as left the program.



Set the patient identifier

iHelp identifier

Patient information will be loaded once the patient will be selected

Patient Gender

Active

Birth date 10/7/2022
MM/DD/YYYY

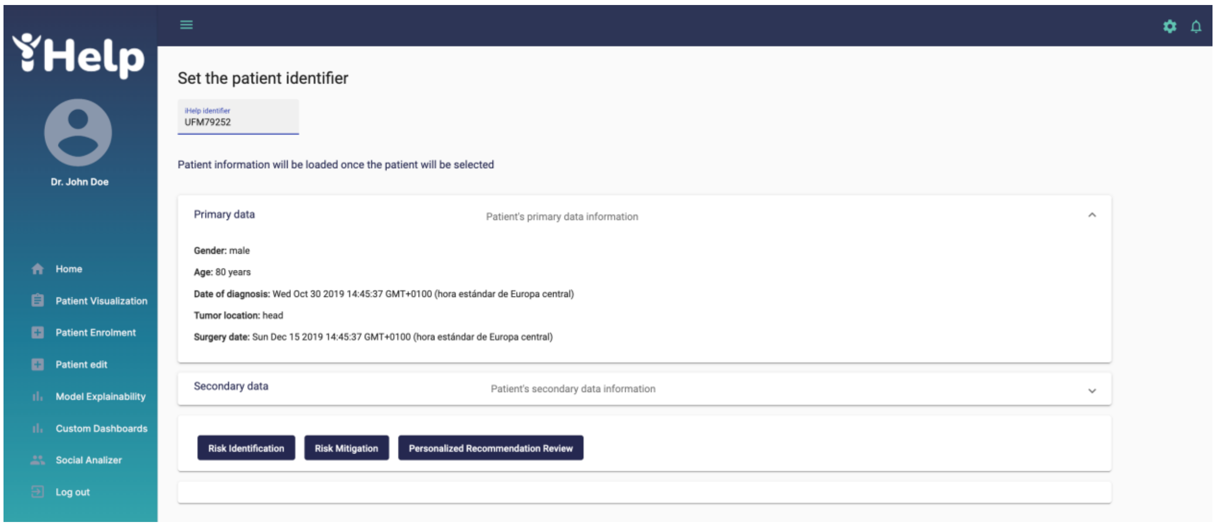
Pilot identifier

Healthentia app identifier

Figure 43: Patient edit interface

6.3.4 Patient Visualization interface

The Patient visualization interface, depicted in Figure 44 allows the health care professional to visualize data from the patient. This data has been divided into two different types of data: the Primary Data that is the data provided by the hospital and the Secondary data that is the data provided by the Healthentia platform. At the bottom of this patient visualization component there are a set of buttons: the Risk Identification, the Risk Mitigation and the Personalized recommendation review.



Set the patient identifier

iHelp identifier: UFM79252

Patient information will be loaded once the patient will be selected

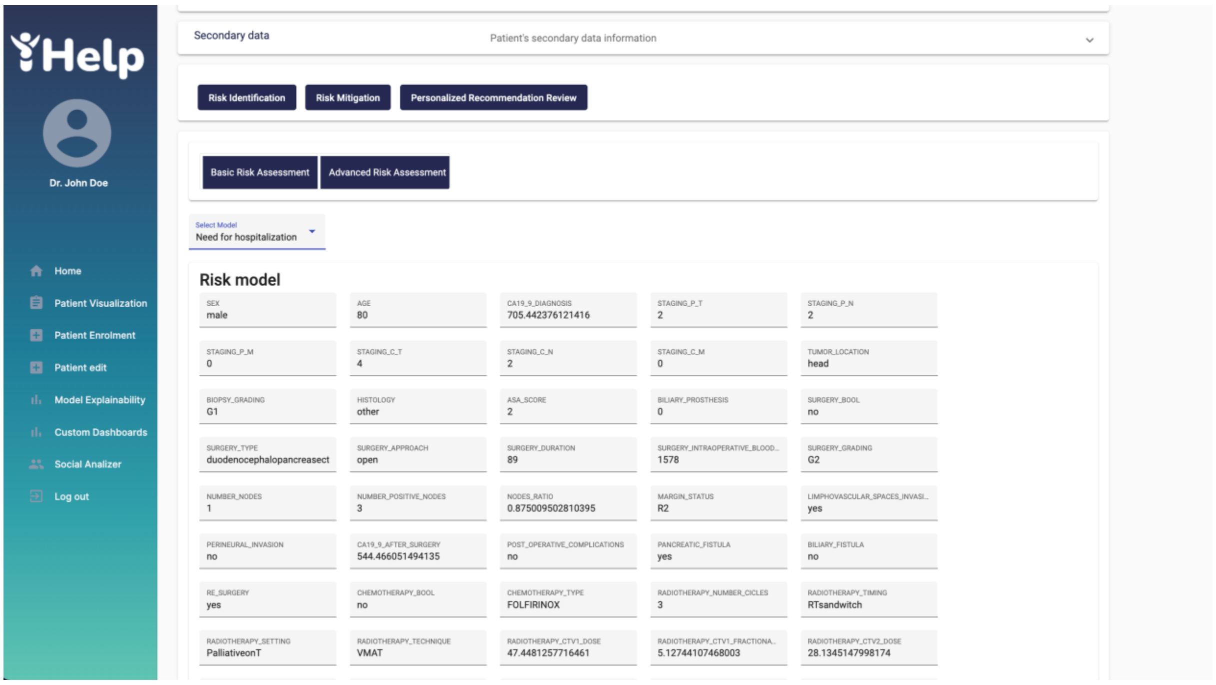
Primary data — Patient's primary data information

Gender: male
Age: 80 years
Date of diagnosis: Wed Oct 30 2019 14:45:37 GMT+0100 (hora estándar de Europa central)
Tumor location: head
Surgery date: Sun Dec 15 2019 14:45:37 GMT+0100 (hora estándar de Europa central)

Secondary data — Patient's secondary data information

Figure 44: Patient visualization interface

The risk identification allows the execution of the AI models developed by the Personalized Predictor component. The basic risk assessment uses primary data, and the advance risk assessment uses both primary and secondary data. Clicking on the basic risk assessment or advance risk assessment a drop down with the available models appears. In Figure 45, the Need for hospitalization AI model is selected. Automatically all the parameters required by the model to be executed are shown in a form. The form fields for which we have the data stored in the Big Data Platform are filled in automatically, thus avoiding that the health care professional has to fill in the model parameters. At the end for the form a submit button allows to execute the model within the Analytic Workbench component and the result is provided to the health care professional.



Secondary data — Patient's secondary data information

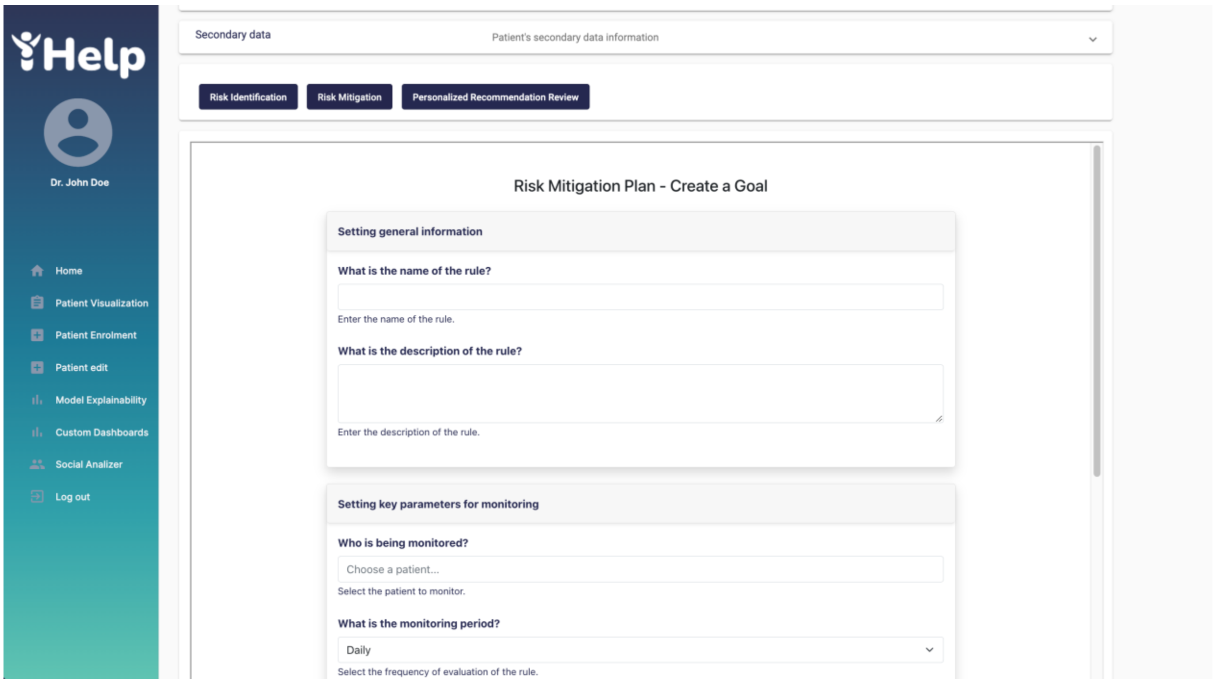
Select Model: Need for hospitalization

Risk model

SEX: male	AGE: 80	CA19_9_DIAGNOSIS: 705.442376121416	STAGING_P_T: 2	STAGING_P_N: 2
STAGING_P_M: 0	STAGING_C_T: 4	STAGING_C_N: 2	STAGING_C_M: 0	TUMOR_LOCATION: head
BIOPSY_GRADING: G1	HISTOLOGY: other	ASA_SCORE: 2	BILIARY_PROSTHESIS: 0	SURGERY_BOOL: no
SURGERY_TYPE: duodenocephalopancreasect	SURGERY_APPROACH: open	SURGERY_DURATION: 89	SURGERY_INTRAOPERATIVE_BLOOD...: 1578	SURGERY_GRADING: G2
NUMBER_NODES: 1	NUMBER_POSITIVE_NODES: 3	NODES_RATIO: 0.875009502810395	MARGIN_STATUS: R2	LIMPHOVASCULAR_SPACES_INVASI...: yes
PERINEURAL_INVASION: no	CA19_9_AFTER_SURGERY: 544.466051494135	POST_OPERATIVE_COMPLICATIONS: no	PANCREATIC_FISTULA: yes	BILIARY_FISTULA: no
RE_SURGERY: yes	CHEMOTHERAPY_BOOL: no	CHEMOTHERAPY_TYPE: FOLFIRINOX	RADIOTHERAPY_NUMBER_CICLES: 3	RADIOTHERAPY_TIMING: RTsandwitch
RADIOTHERAPY_SETTING: PalliativeonT	RADIOTHERAPY_TECHNIQUE: VMAT	RADIOTHERAPY_CTV1_DOSE: 47.4481257716461	RADIOTHERAPY_CTV1_FRACTIONA...: 5.12744107468003	RADIOTHERAPY_CTV2_DOSE: 28.1345147998174

Figure 45: Risk identification interface

The Risk Mitigation button opens the Monitoring and Alerting System component interface. This component is developed under the scope of T5.5 – “Monitoring, Alerting, Feedback and Evaluation Mechanisms”. A detailed explanation about this interface is provided in D5.11 – “Monitoring, Alerting, Feedback and Evaluation Mechanisms I”.



Risk Mitigation Plan - Create a Goal

Setting general information

What is the name of the rule?

Enter the name of the rule.

What is the description of the rule?

Enter the description of the rule.

Setting key parameters for monitoring

Who is being monitored?

Choose a patient...

Select the patient to monitor.

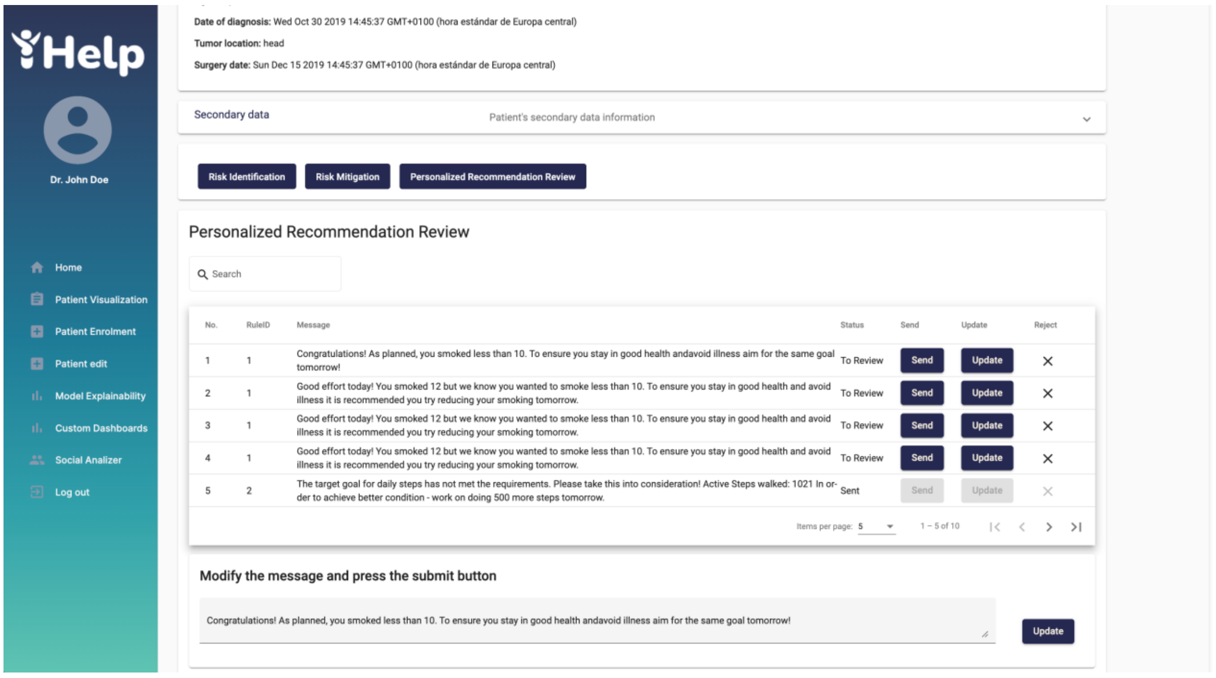
What is the monitoring period?

Daily

Select the frequency of evaluation of the rule.

Figure 46: Risk mitigation interface

The personalized recommendation review button opens the interface that allows the health care professional to review the messages produced by the Monitoring and Alerting System, update the messages, discard them or sent them to the patients through the Healthentia platform. Messages are listed within a paginated table as soon as are requested to the Big Data Platform. Each message has three buttons that allows to update, reject or sent the message. Clicking on the update button of the message, it appears at the bottom of the screen allowing the health care professional to update it, then clicking on the update button near the updated message it is stored in the Big Data Platform, and it is marked as “updated” in the table of the interface. Clicking on the reject button the message is marked as “rejected” and is removed from the table. Finally clicking on the sent button the message is forwarded to the patients Healthentia application. The integration with the Healthentia platform will be done during the second phase of the project.



Date of diagnosis: Wed Oct 30 2019 14:45:37 GMT+0100 (hora estándar de Europa central)

Tumor location: head

Surgery date: Sun Dec 15 2019 14:45:37 GMT+0100 (hora estándar de Europa central)

Personalized Recommendation Review

No.	RuleID	Message	Status	Send	Update	Reject
1	1	Congratulations! As planned, you smoked less than 10. To ensure you stay in good health andavoid illness aim for the same goal tomorrow!	To Review	Send	Update	✕
2	1	Good effort today! You smoked 12 but we know you wanted to smoke less than 10. To ensure you stay in good health and avoid illness it is recommended you try reducing your smoking tomorrow.	To Review	Send	Update	✕
3	1	Good effort today! You smoked 12 but we know you wanted to smoke less than 10. To ensure you stay in good health and avoid illness it is recommended you try reducing your smoking tomorrow.	To Review	Send	Update	✕
4	1	Good effort today! You smoked 12 but we know you wanted to smoke less than 10. To ensure you stay in good health and avoid illness it is recommended you try reducing your smoking tomorrow.	To Review	Send	Update	✕
5	2	The target goal for daily steps has not met the requirements. Please take this into consideration! Active Steps walked: 1021 In order to achieve better condition - work on doing 500 more steps tomorrow.	Sent	Send	Update	✕

Modify the message and press the submit button

Congratulations! As planned, you smoked less than 10. To ensure you stay in good health andavoid illness aim for the same goal tomorrow!

Figure 47: Personalized recommendation review interface

6.4 Common interfaces

Two interfaces are shared among both user roles, the model builders and the health care professionals, the custom dashboards and the model explainability interfaces.

Clicking on the Custom Dashboards button on the left side menu or from the menu interface, the dashboards designed from the model builder user through the Workflow Editor interface can be accessed. Figure 48 shows an example of the custom dashboard “Healthentia patient data” embedded within the web interface of the DSS.

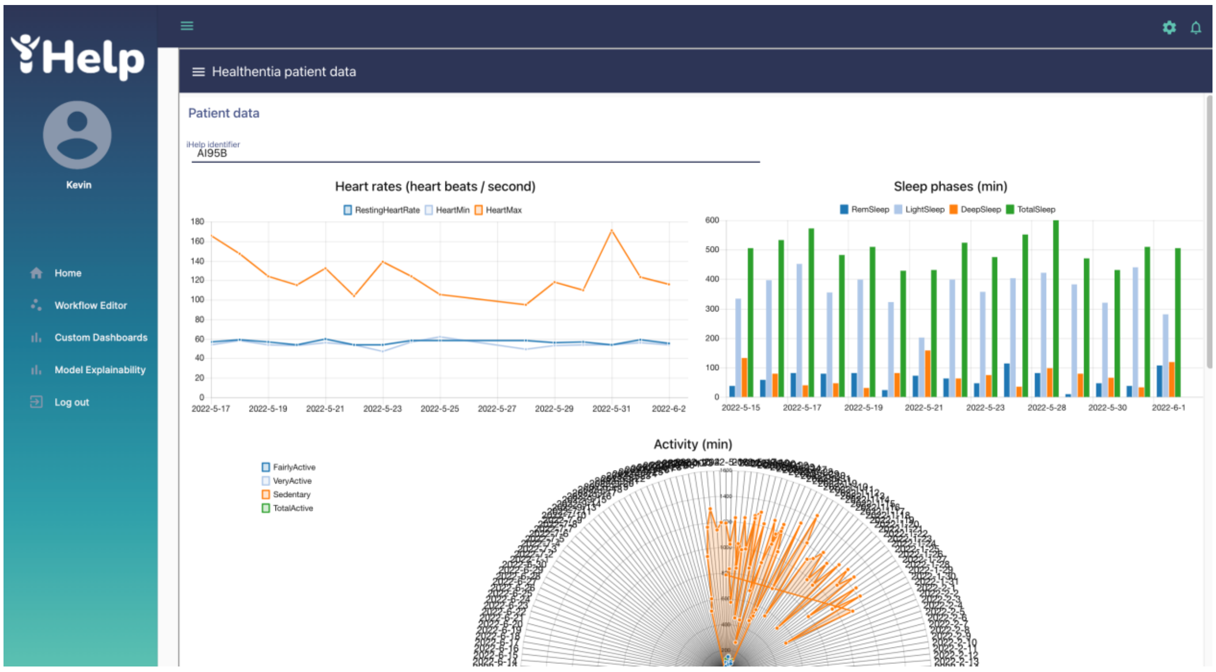
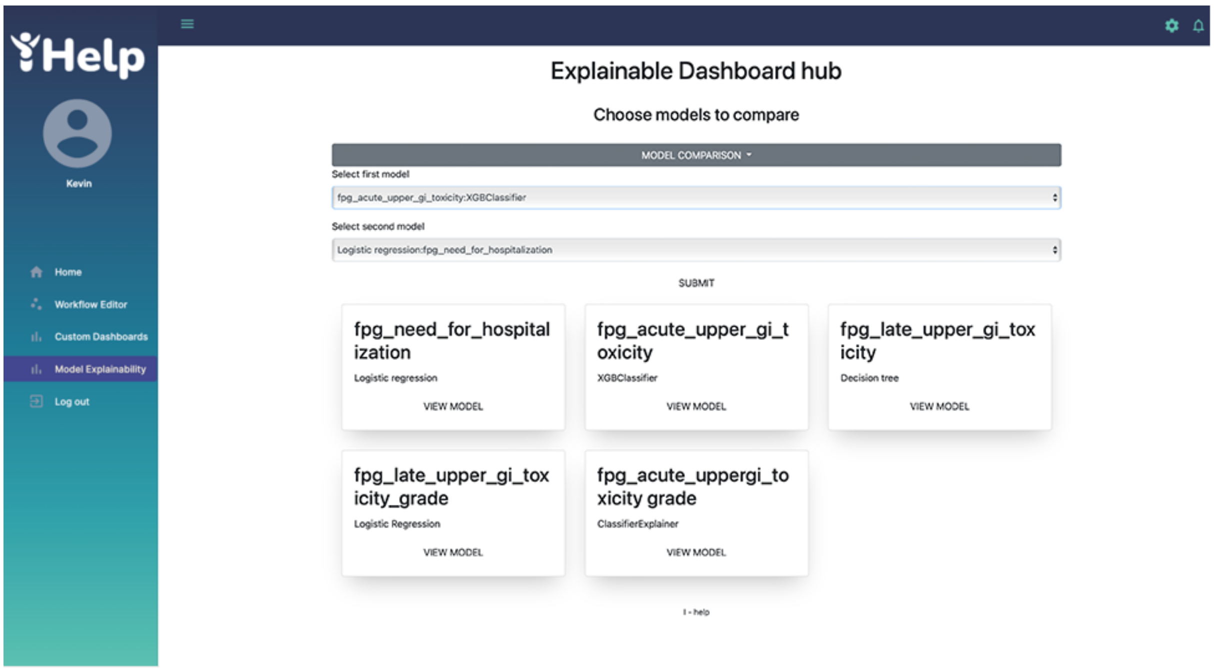


Figure 48: Custom dashboards interface

Last, by clicking on the Model Explainability button on the left side menu or from the menu interface, the EDH initial interface is accessed. Figure 49 showcases the EDH embedded in the DSS web interface. The EDH contains one main dashboard that summarizes and organizes many different Explainable Dashboards for various AI models in the form of cards with related information, model types and origin, in one single place



Select first model: fpg_acute_upper_gi_toxicity:XGBClassifier

Select second model: Logistic regression:fpg_need_for_hospitalization

fpg_need_for_hospitalization — Logistic regression

fpg_acute_upper_gi_toxicity — XGBClassifier

fpg_late_upper_gi_toxicity — Decision tree

fpg_late_upper_gi_toxicity_grade — Logistic Regression

fpg_acute_uppergi_toxicity grade — ClassifierExplainer

Figure 49: Model explainability interface

7 DSS Deployment

This section describes how the DSS is installed and deployed. The description of a local deployment for testing purposes is presented. The Kubernetes deployment with all components integrated will be delivered during the second part of the project.

7.1 Installation Using Docker

The iHelp DSS component release is available from the project Gitlab repository. All members of the consortium have access to the Gitlab repository and should be able to download the release from it with the command:

```
git clone https://gitlab.ihelp-project.eu/ainhoa/ihelp-dss.git
```

Once the distribution is downloaded, the system administrator has run the docker compose command to create the local Docker images of the Web Interface, the Workflow Editor, the Custom Dashboards and the EDH components. Three different images will be created: ihelp-dss-web, ihelp-dss-workflow and ihelp-dss-edh. These Docker images will deploy the DSS in a containerized environment with three interconnecting containers. The command to create the docker images and create the containers is:

```
cd ihelp-dss
docker compose build
docker compose up -d
```

These commands will create the images and will start the three containers the DSS is made off and exports the port 4200 to access the DSS web interface. The DSS is available in any browser by typing http://localhost:4200. This type of installation is only recommended for testing purposes. Any workflows and dashboards created are deleted when the container is stopped. In order to save locally workflows and dashboards it is possible to export them through the Node-RED editor clicking on the menu at the upper left part and clicking on the Export option as it is shown in Figure 50.This will generate a JSON file that can be stored in the user's computer. This JSON file can be imported again once the container is deployed.

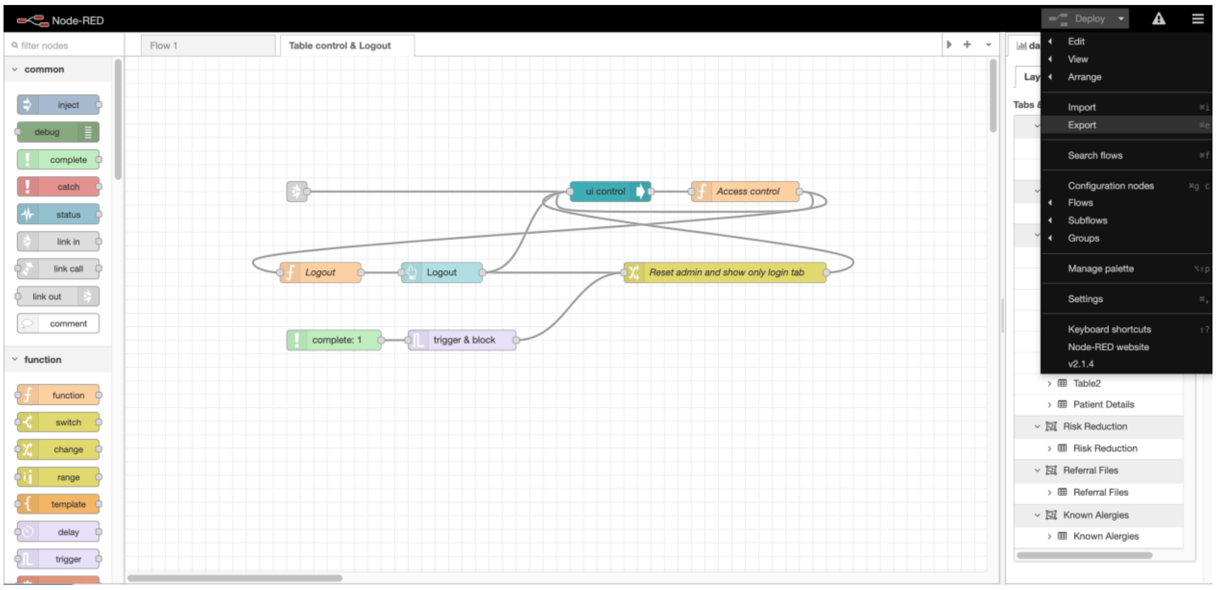


Figure 50: Export workflow or dashboard

8 Next Steps

Throughout this deliverable we have presented the DSS component, developed in the context of T4.3 – “Clinical DSS Suite with Visual Analytic Tools”. At this stage of the project, we have delivered the second prototype of the DSS, with a set of generic dashboards and integrated it with most of the iHelp platform components that have to interact with the DSS. A third version of this deliverable is foreseen to be delivered in M32 of the project, in the context of D4.8 – “iHelp DSS suite with visual analytics III”, that will include the final version and implementation of the DSS.

The integration with other components will be done during the second part of the iHelp project and a set of workshops will be prepared to show the DSS to the model builder and health care professional users to receive their feedback and to improve the DSS interfaces.

9 Conclusions

This deliverable presents the second prototype designed and developed in the context of T4.3 – "Clinical DSS Suite with Visual Analytic Tools" of the iHelp project. The DSS helps HCPs and policy makers during the decision-making process by providing customizable dashboards with different charts and other information that the analytics tools and risk identification algorithms will produce. This component utilizes the different analytical models, designed in the context of the iHelp project, to predict and identify the risk of each patient to develop Pancreatic Cancer and to decide the usage of different treatments in case of already diagnosed Pancreatic Cancer patients. Advanced and custom dashboards are provided through the DSS to the stakeholders to enhance their knowledge and understanding on the data and on the analytical insights that are being produced through the utilization of the AI models.

List of Acronyms

ATC	Athens Technology Centre
AWS	Amazon Web Services
CA	Consortium Agreement
CPU	Central Processing Unit
D	Deliverable
DB	Database
DoA	Description of Action
DSS	Decision Support System
EDH	Explainable Dashboard Hub
EU	European Union
Gb	Gigabyte
HCP	Health Care Professional
HTML	HyperText Markup Language
ICE	Information Catalyst for Enterprise
JSON	JavaScript Object Notation
LXS	LeanXcale
M	Month
MIT	Massachusetts Institute of Technology
PVC	Persistent Volume Claim
SHAP	SHapley Additive exPlanations
SQL	Structured Query Language
UPM	Universidad Politécnica de Madrid
UPRC	University of Piraeus Research Centre
XAI	eXplainable Artificial Intelligence