Research Practical AI-NET PROTECT 4 health WS 2022/2023

Emilian Cikalleshi, Resmin Hossain, Bini Mariam John, Bhargavi Kannan, Fabian Maxeiner, Hisham Parveez, Abhinav Ralhan, Fabian Ring, and Varnana Vijay

University Koblenz, Koblenz Germany

Abstract. We describe a scenario which handles simulated medical data of patients and exchanges data using Data Spaces and Dataspace Connectors to ensure data sovereignty. This is done by synthesizing vital parameters of patients, using Machine Learning to predict a patients health condition locally on their own devices and exchanging the hyperparameters used for the Machine Learning with an external application. This external application collects the hyperparameters from multiple patients and aggregates them into one set of hyperparameters, which is then sent back to all patients. In this scenario, no personal data of the patients is exchanged and all exchanges are visible to the patients through the agreements for the exchange in the Dataspace.

1 Introduction

This paper explores data exchange in a medical domain using the concept of International Data Spaces and Dataspace Connectors as described by the International Data Spaces Association.

In today's time, more and more people want be in control of their own data. This includes the information of what their data is used for, and more importantly, have the ability to allow or disallow that data usage. This is especially important in the medical domain, where on the one side highly personal data of the patients is handled, but on the other side more and more patients would enjoy a better exchange of their medical data between their doctors. But the patient should be able to know what data is exchanged and only necessary data is exchanged. In an ideal case, the patient has the data sovereignty over his own data.

Figure 1 is a Rich Picture which describes a scenario, where the goal is to use the International Data Spaces (IDS) idea of Data Spaces to achieve data exchange in the medical domain with a purpose of the data exchange, and using Dataspace Connectors to access the Data Spaces. The red box describes activities happening locally on a patients device, like collecting his vital health data and using Machine Learning to predict the current health state, which can be displayed on an application on the patients device. Here, the patient is in control of his data. The yellow box shows, that the idea described in the red box is

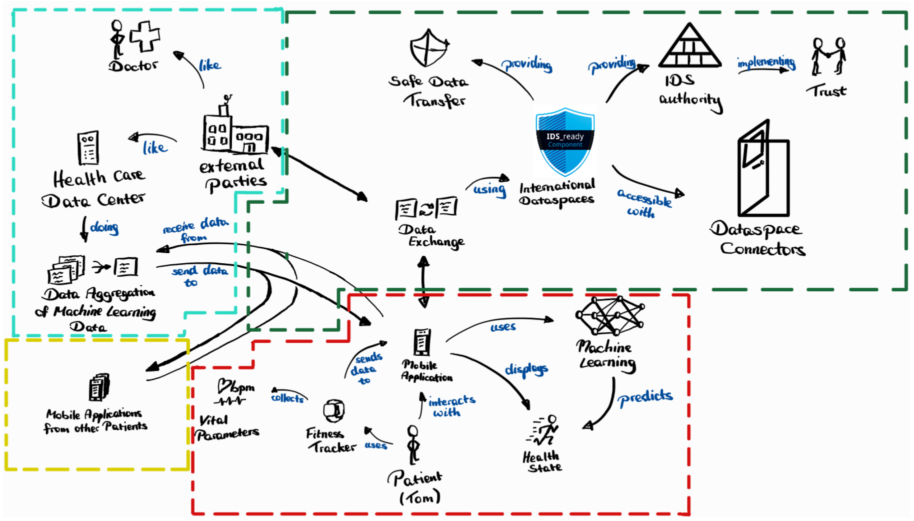


Fig. 1: Big Picture

happening simultaneously on multiple devices from different patients. The blue box shows different people or organizations that would like to have access to personal or non-personal data of the patients for different purposes. These can be from the before mentioned doctors, who can benefit from a fast update of their patients health condition or automatic alarms if the health condition worsens, to organizations like a patients health care company, which would like to collect personal or non-personal data, like parameters used on the local Machine Learning, in order to combine that knowledge in a meaningful way to improve the application or the Machine Learning algorithm for all patients. The green box is the last box of the Rich Picture and shows that the idea is to use the IDS and Dataspace Connectors as the basis for all exchanges regarding a patients data, as its goals are to provide a safe data transfer and to include an IDS authority in the Data Space to ensure trust for all participants.

While this is bigger scenario, the work of this paper will be about implementing the basics of this scenario with less functionality. However this could be used as the foundation for future work on the subject.

1.1 Goals

To achieve an implementation of the basics described in the scenario in Figure 1, a definition of what actually has to be implemented is needed. The goal of work for this paper is to implement a reduced scenario, where patients can use an application which collects and handles personal vital data and exchanges non-personal data with an external party. To justify the necessity of the data exchange, a purpose for the exchange has to be defined. While this scenario does not exchange personal data of patients and therefore does not require the necessity to provide secured and trusted data exchange, a future extension might. Because of that, Data Spaces and Dataspace Connectors will be used for the data exchange to implement a basis that allows self-determination of personal data for the patient.

With that in mind, to achieve that goal we want to create an application which collects personal medical data (vital sign parameters) of a patient. This application trains a Machine Learning model using the individual data of the patient and predicts the health state of the patient. Furthermore, the applications of multiple patients share their Machine Learning hyperparameters with an external organization, which could represent the health care company in the blue box from Figure 1, that collects the hyperparameters and aggregates them into one aggregated set of hyperparameters. This aggregated set of hyperparameters gets shared back to all the applications of the different patients and are then used by each application individually to retrain their local Machine Learning models. After that, the process of predicting, sharing, aggregating and retraining repeats.

More remarks regarding the implementation are:

– No personal data of a patient leaves their own device.
– During the implementation of this scenario, data will be synthesized to simulate usage of the application by a patient.
– Since the hyperparameters are not modified locally, there won't be any difference after the first aggregation. However since this is supposed to be a basis for future work where either other data will be exchanged or the hyperparameters are influenced locally by the training of the Machine Learning, the aggregation will still happen repeatedly.

1.2 IDS and Dataspace Connectors

The International Data Spaces (IDS) [1] is an initiative that aims to create a secure and trusted data exchange infrastructure for the digital economy. The IDS is based on the concept of data sovereignty, which means that individuals and organizations have control over their data and can decide who has access to it.The IDS provides a framework for secure data exchange that allows data to be shared and used in a way that is transparent, trustworthy, and compliant with legal and regulatory requirements. The IDS architecture is based on a set of standards and protocols that enable data exchange between different organizations and domains. The IDS standards cover areas such as data privacy, security, data usage control, and governance.

The International Data Spaces Connector is a key component of the IDS architecture. It is a software component that enables secure and trusted data exchange between different organizations. The IDS Connector acts as a gateway between different data spaces and provides a standardized interface for data exchange. It is designed to support different use cases and scenarios, including data exchange between different industries and across different countries. The IDS Connector uses a range of security and privacy technologies to ensure that data is protected throughout the data exchange process. These include encryption, authentication, access control, and auditing. The IDS Connector also supports the use of data usage policies, which can be used to define the conditions under which data can be accessed and used. The IDS Connector is designed to be flexible as well as scalable, and it can be deployed in different types of environments, such as cloud-based, on-premises, or hybrid.

One of the key benefits of the IDS architecture is which allows organizations to share and use data in a way that is transparent and trustworthy. This is important for enabling new business models and innovation in the digital economy. For example, the IDS can be used to support data-driven services and applications that rely on the exchange of data between different organizations. The IDS can also help to reduce the costs and complexity of data exchange by providing a standardized interface and a common set of rules and protocols.

Overall, the International Data Spaces is an important initiative in the development of a secure and trusted data exchange infrastructure for the digital economy. It provides a framework for organizations to share and use data in a way that is transparent, trustworthy, and compliant with legal and regulatory requirements. The IDS is a collaborative effort that involves partners from industry, research, and government organizations, and it has the potential to support innovation and growth in the digital economy.

2 Implementation

Now that the goal of work for this paper is defined, and the IDS and its purpose and functionalities are explained, the next step is to show not only what will be implemented in this paper, but also how it will be implemented and why it is implemented the way it is, before the results are shown.

2.1 Architecture

To implement the described scenario, the decision was made to create new and use existing applications running in Docker, which will in combination represent the mobile applications for the patients, as well as a Training Coordinator application, which represents an application running on the side of the health care company and aggregates the hyperparameters. Components that realize one running system will be started in one Docker stack.

As shown in Figure 2, there will be three Docker stacks running for mobile applications. That way, it is easier to test if the Training Coordinator communicates with all mobile applications, and see if the aggregation works as intended. For the Training Coordinator, only one running Docker stack is needed.

Each mobile application will consist of the following components:

- Data Synthesizer
- Storage
- Machine Learning

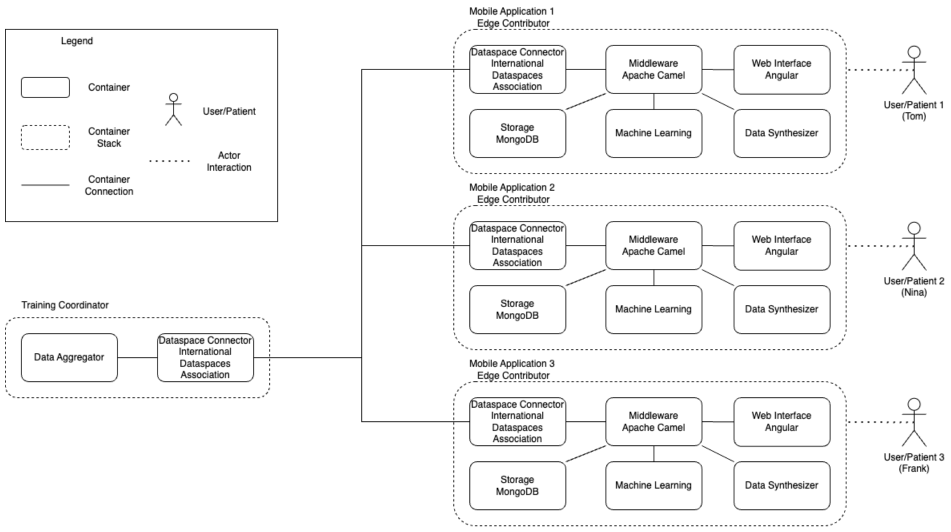


Fig. 2: Big Picture

- Middleware
- Web Interface
- Dataspace Connector

While these will be described in more detail in the following sections, the basic idea is the following: The Data Synthesizer will simulate the collection of vital data from a patient, which gets stored in the Storage. From the Storage, the data on the one side will be used in the Machine Learning and written back to the Storage with an added predicted health state, as well as being accessed by the Web Interface to display relevant data for the patient. The Dataspace Connector communicates with the Dataspace Connector from the Training Coordinator to offer and request hyperparameters and aggregated hyperparameters, which it gets and sets from and in the Machine Learning. All communication between components of the mobile application will be handled by the Middleware.

The Training Coordinator consists of a Dataspace Connector and a Data Aggregator, which will aggregate the incoming hyperparameters from multiple mobile application, and offers the aggregated set for them to request.

Due to limitations in available hardware, all Docker stacks, which means the Training Coordinator, as well as the three mobile applications, will run in one Docker instance on one computer. However each Docker stack will communicate internally in its own Docker network, to ensure that no data is available to the other stacks without the intended one.

2.2 Data Synthesizer

The key objective of Data Synthesizer is to generate artificial health data and vital indicators of a human body. The Data Synthesizer is required to replicate the behaviour of actual users using the application. It is developed based on research about health statistics [3] and indicators [4], and broadly categorises medical data into two components: vitals indicator signs and general health statistics. Vital signs are a set of measurements used to monitor the body's functions and overall health. Particularly for this research, we consider vital signs including heart rate, systolic blood pressure, diastolic blood pressure, body temperature, breathing frequency. For general health statistics we include human attributes like age, weight, step count, stress level, blood oxygen level, sleep hours, rapid eye movement (REM) sleep hours and calories burnt. The attribute values follow normal distribution ranges for all individuals. The attribute values and ranges may vary for individuals based on factors such as age, time of the day, and other health conditions. For example, body temperature should normally be between 97 and 103 degrees Fahrenheit. Further description about the health dataset has been provided in the Appendix.

For the Data Synthesizer, we have developed a Docker microservice which offers multiple endpoints for data exchange. These offerings include generating data for training, data for prediction and dummy data only for the visualisations. Training data includes bulk raw user data and the target variable. Prediction data includes a single record of medical data and no target variable. The dummy data generated for visualisation helps in seeing the type of artificial data is generated. The microservice is developed using Python based libraries such as random, regex, faker [5]. With the help of these libraries, artificial health data can be generated for the application.

The training data simulates a real life scenario of an individual user. This means that health attributes such as step count, calories, et al, change over the course of the day. These data points are recorded at random moments over the course of the day and may vary from user to user when they choose to record it. To imitate this behaviour, the concept of random intervals is introduced. Random intervals are generated for each user in periods selected randomly over the day. Within these random intervals, the Data Synthesizer generates random timestamps. For these timestamps, medical health data is generated and for each medical record user information, health stats and the vitals are generated. The Data Synthesizer incorporates the fact that some of these attributes are cumulative for the day (step count, calories) while other attributes are independent of the time of day (heart rate, blood pressure). It also accounts for the fact that the weight and height of a person is generated according to the age of a user. The label or the target variable generated here is called the "health state". This label is generated for each health record based on a threshold parameter for each attribute. This threshold parameter ensures that the lowest 1st percentile or the top 99th percentile of the attribute values from each attribute indicate whether a person is in a risky or a healthy health condition. For example, body temperature above 102.5 degrees tells us that a user is in a risky health condition.

2.3 Storage

Storing health data comes with many challenges and difficulties [6]. In this research project, we use MongoDB as the storage technology as it offers a document-based database with a flexible schema design. This means that we can add more attributes about health records for the patient if needed. MongoDB also offers native support for time series data with Time Series collections. With this, we can use various features particularly suited to time series datasets such as having an expiry date for each record. Currently, datasets are created and used from a patients perspective. This means that the data provided to the application can be of various data types and values. MongoDB is very adept at dealing with flexible data types and medical health records [7].

2.4 Machine Learning

The scenario requires the creation of a Machine Learning model [8] to classify the health state of the patient according to the generated health data. The model takes the health parameters of the patient generated by the Data Synthesizer as the input and provides the health state of the patient as the output. In this scenario there are only two possibilities how the health state of a patient can be classified: "Healthy" or "Risky". After an elaborated research on several Machine Learning algorithms, the conclusion was to employ a decision tree algorithm for the model, as a decision tree is an algorithm with less complexity along with a satisfying result.

Decision tree is a supervised Machine Learning algorithm used for classification and regression. The algorithm breaks down the training dataset into smaller datasets (for binary decision tree into two smaller datasets). Since there are only two possible health states for each patient, a binary Decision Tree Algorithm [9] has been used to design the model.

At the initial stage, the model was trained with some generated training dataset. After that, the model was implemented on the test dataset. At the current state of the project, the training of the model can be triggered when needed. Also, the classification is triggered via the web interface.

The most important factor of the model is the hyperparameters as only the hyperparameters are shared with the training coordinator. Hyperparameters are those parameters of a model which we define exclusively to control the learning process of the model. The performance of a model depends thoroughly on the hyperparameters. If the values of the hyperparameters are changed, the performance, as well as the output of the model, will switch accordingly.

2.5 Middleware (Apache Camel)

Apache Camel [10] works as the Middleware of a mobile application (as seen in Figure 2) and sits between the different components of the system, providing a layer of abstraction that allows them to interact without needing to know the details of each other's implementation. This makes it possible to create a loosely coupled system that can be easily modified and extended over time. Apache Camel is an open-source integration framework that provides a powerful and flexible solution for integrating different systems and applications. It acts as a mediator between different systems, allowing them to communicate with each other using various protocols and data formats. One of the key benefits of using Apache Camel as the Middleware is its extensive set of connectors and components. It also provides a powerful routing engine based on Enterprise Integration Patterns [11], which can be used to create complex message routing scenarios. These allow developers to easily connect and route messages between different systems using a wide range of protocols and data formats.

So in the implementation Apache Camel provides the routes to connect the components of the mobile application with each other. These routes allow different components to communicate and exchange data with each other. For instance, one route can use an HTTP request from the Web Framework to retrieve data from the MongoDB and transform it into the appropriate format for display. Another route can take data from the Data Synthesizer and store it in the MongoDB. By using this Middleware approach, the components of the mobile application are decoupled and its flexibility is increased. If the Storage component needs to be changed in the future, only the Middleware layer has to be modified and not the Web Framework or the Data Synthesizer. This advantage in flexibility is the reason for using a Middleware instead of connecting each component directly. This approach also had the advantage that each component of the mobile application could be developed independently without knowing the functionality of a different component. Only the interfaces of each component needed to be fixed so that the Middleware could connect with them.

2.6 Web Interface

Patients can engage with the data in a meaningful way by utilising a web application, which allows them to view what is happening in the back-end services in real-time. The Web Interface component used in the research project is developed using the Angular framework.

In the given scenario, the goal is to provide patients with the ability to view their collected vital data, how their health state is labeled by the Machine Learning and view when data was last exchanged, as well as who it was exchanged with. Therefore the web application interacts with the Middleware and gets data, which is displayed in a graph as well as in a table in the front end. It presents the vital health data, vital health status, data exchange and a "dev mode" in 4 different tabs. The layouts are designed in such a way that they are easy to understand. The loaders, filters, etc. are included to give a more appealing feel to the user. This view of web application gives the patient an idea on his/her health details, and also helps in comparing the current health status with the past status.

The first tab, which is for vital health data, shows a graph with the values of vital health parameters upon the user's requested time interval. The time interval can either be selected by choosing two specific dates, or by using a predefined button for the last 15 minutes, last 1 hour, last 12 hours or last 48 hours. The second tab provides a table with the vital health parameters of the patient along with the health status. The third tab presents the data exchange, so it displays what data is shared with whom to the patient, the purpose of the data exchange, when the agreement for exchange was first initialized and when it was last used. To be more specific, the tab provides all the data exchange agreement details like agreement modification date, agreement link, artifact modification date, artifact description etc. The final tab, the "dev mode" is not specifically for the patient, instead it offers the developers more information. The tab has 3 buttons. The first is to trigger the Data Synthesizer. The next one is to trigger the training for the Machine Learning and the final is to display the hyperparameters currently used in the Machine Learning. The editing function for hyperparameters is not provided in the front end for now. The loaders are used along with buttons to let the user know whether the button press was successful or not.

For testing purposes, it was necessary to run all the edge connectors on a single machine. However, a challenge arose while simultaneously running the web frameworks on the browser in terms of the configuration of the connections from one container to another (between Middleware and Web Framework specifically). While it may seem simple in theory as each of the docker compose files are configured in a different network, an extra step was needed for Angular to receive the data offered by Camel routes. Exposing Camel ports and making routes accessible in the host machine was not an option for security reasons. This is where the use of an NGINX [12] reverse proxy proved to be useful. The Angular application is contained within NGINX and exposed in the browser. Angular requests the necessary data through NGINX, which forwards it to the appropriate Camel route, eventually receiving the data as a response that is then forwarded back to the Angular application. NGINX acts as a reverse proxy, enabling communication between Angular and Camel components while also providing security for this communication.

2.7 Training Coordinator and Data Aggregator

In our proposed architecture the Training Coordinator is an independent server consisting of Dataspace Connector and an aggregation module. In a real-word the Training Coordinator represents a health organisation where the data is exchange with the different edge-nodes representing mobile devices of patients via their respective Dataspace Connectors. Since the communication between the edge-nodes and the Training Coordinator is bi-directional, it consists of two sub components in the connector, namely the provider [13] and the consumer [14], to handle communication. The provider is used to add data and make it available to other connectors of the edge-nodes while the purpose of consumer is to fetch the data from various connectors of the edge-nodes. The Data Aggregator is a separate module integrated inside training coordinator. It's purpose is to aggregate the data collected from various Dataspace Connectors. In our proposed solution the data exchanged between Dataspace Connectors are the Machine Learning model hyperparameters. Linear JSON structures are implemented to represent model hyperparameters content where keys indicate the hyperparameters name along with its corresponding numeric value used for training model. The Data Aggregator creates a new structure consisting of the inputs along with newly computed aggregated numeric values.

The implementation of the Training Coordinator consists of three major components namely the Data consumer, the Data Aggregator and the Data provider.

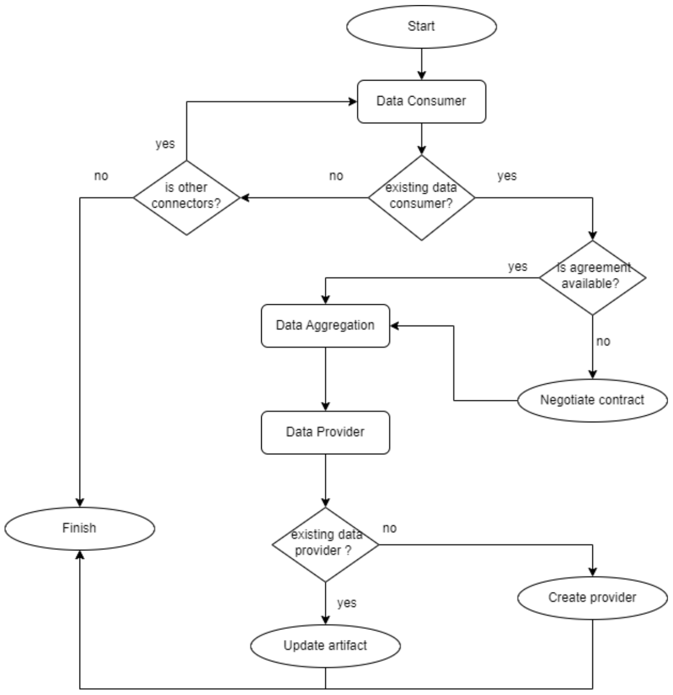
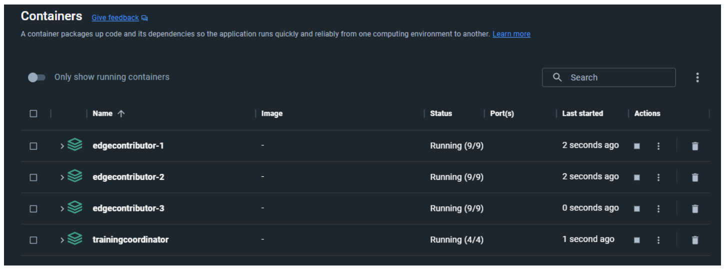


Fig. 3: Training Coordinator Process flow

The Data consumer fetches data providers from the various IDS connectors of the edge-nodes sequentially. Any node without a provider would be skipped. An internal lookup table exists which consists of the connectors id along with their corresponding agreement id with the Training Coordinator. the data providers are consumed directly if there is an existing agreement, otherwise a new agreement is created and added to the internal lookup table for successive use. The Data Aggregator performs aggregation on the data by the consumers. The mean value average method is used to compute the aggregation of the collected Machine Learning hyperparameters. The newly created aggregated data structure from the aggregation is added to the data provider. The purpose of the provider is to make the data available to all other IDS connectors in the network for consuming. The structure of the data provider is quite crucial, in the given scenario. If the structure of the provider is invalid, then the data is not available at other connectors. The provider structure is created once for each connectors of the nodes. For every successive update only the core data of the provider is updated.

3 Results

Now that all the components used for the scenario were introduced, we can have a look on the working implementation. As seen in Figure 4 and described in Figure 2, the scenario is completely running in Docker. This contains the different instances of mobile applications to simulate patients, as well as one Training Coordinator for data exchange and data aggregation.



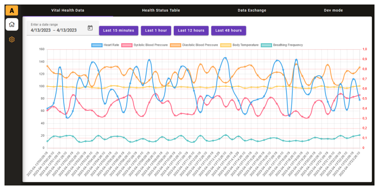
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Name	Image	Status	Port(s)	Last started	Actions
edgecontributor-1	-	Running (9/9)		2 seconds ago	
edgecontributor-2	-	Running (9/9)		2 seconds ago	
edgecontributor-3	-	Running (9/9)		0 seconds ago	
trainingcoordinator	-	Running (4/4)		1 second ago	

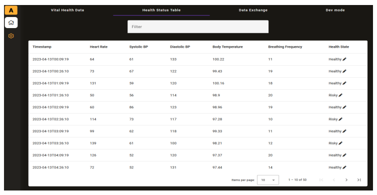
Fig. 4: Scenario running in Docker

As described in chapter 2.6, when accessing the provided Web Interface, a patient has access to his vital parameters in either a graph or a table. The selection of the time frame is either by predefined buttons or by selecting a time frame yourself. The table also displays the health states. Since the Machine Learning is automatically triggered through the Middleware, if new vital parameters are stored in the storage, it is unlikely that a patient will find newly added data in the table that hasn't been classified yet with the current health state. However, if the Machine Learning wasn't running due to different reasons or vital data is added to the storage through a way different from the route provided by the Middleware, it won't be classified by the Machine Learning. The vital data as well as health state displayed vary between each instance of the mobile application, as each one has its own Data Synthesizer.

If we take a look at the agreements in the Web Interface, there should be two agreements per mobile application displayed. One, where the consumer is the respective mobile application, which is used to receive the aggregated set of hyperparameters from the Training Coordinator, and one, where the consumer is the Training Coordinator and which is used get the hyperparameters from each mobile application to the Training Coordinator. Currently, while the agreement with the mobile application as the consumer, is once created and each time updated afterwards, each time the Training Coordinator restarts, like when Docker



(a) Graph displayed in the UI



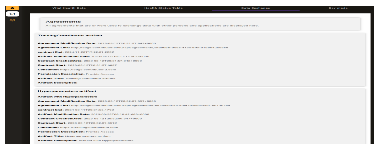
Vital Health Data | Health Status Table | Data Exchange | Dev mode

Filter

Timestamp	Heart Rate	Systolic BP	Diastolic BP	Body Temperature	Breathing Frequency	Health State
2023-04-13T00:09:19	64	61	133	100.22	11	Healthy
2023-04-13T00:26:10	73	67	122	99.43	19	Healthy
2023-04-13T01:09:19	131	59	120	100.16	18	Healthy
2023-04-13T01:26:10	50	56	114	98.9	20	Risky
2023-04-13T02:09:19	60	86	123	98.96	19	Healthy
2023-04-13T02:26:10	114	73	117	97.28	10	Risky
2023-04-13T03:09:19	99	62	118	99.33	11	Healthy
2023-04-13T03:26:10	139	61	100	98.21	12	Risky
2023-04-13T04:09:19	126	52	120	97.37	20	Healthy
2023-04-13T04:26:10	72	52	131	97.44	14	Healthy

Items per page: 10 1 – 10 of 50

(b) Table displayed in the UI



Vital Health Data | Health Status Table | Data Exchange | Dev mode

Agreements
All agreements that are or were used to exchange data with other persons and applications are displayed here.

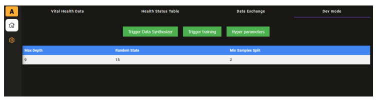
TrainingCoordinator artifact

Agreement Modification Date: 2023-03-12T20:31:57.842+0000
Agreement Link: http://edge-contributor:8080/api/agreements/afd90bff-9566-41be-8f6f-01b8042b5858
contract End: 2024-11-28T17:22:01.223Z
Artifact Modification Date: 2023-03-23T08:11:12.507+0000
Contract CreationDate: 2023-03-12T20:31:57.842+0000
Contract Start: 2023-03-12T20:31:57.683Z
Consumer: https://edge-contributor-2.com
Permission Description: Provide Access
Artifact Title: TrainingCoordinator artifact
Artifact Description:

Hyperparameters artifact

Artifact with Hyperparameters
Agreement Modification Date: 2023-03-12T20:32:09.359+0000
Agreement Link: http://edge-contributor:8080/api/agreements/e8359a9f-a52f-442d-9edc-c6b1eb1303aa
contract End: 2024-03-11T20:31:56.179Z
Artifact Modification Date: 2023-03-23T08:10:42.683+0000
Contract CreationDate: 2023-03-12T20:32:09.347+0000
Contract Start: 2023-03-12T20:32:09.351Z
Consumer: https://training-coordinator.com
Permission Description: Provide Access
Artifact Title: Hyperparameters artifact
Artifact Description: Artifact with Hyperparameters

(c) Agreements displayed in the UI

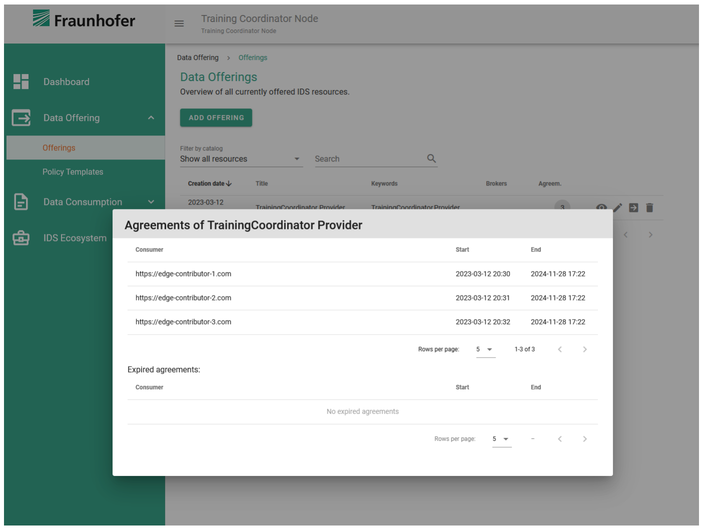


Max Depth	Random State	Min Samples Split
9	15	2

(d) Dev tools displayed in the UI

Fig. 5: Pages of the Web Interface

itself is stopped and started afterwards, a new agreement with the Training Coordinator as the consumer is created instead of reusing the existing one. This does not have any influence in the functionality, but can result in an increased number of agreements displayed in the Web Interface.



Data Offerings
Overview of all currently offered IDS resources.

Filter by catalog: Show all resources Search

Creation date	Title	Keywords	Brokers	Agreem.
2023-03-12	TrainingCoordinator Provider	TrainingCoordinatorProvider		3

Agreements of TrainingCoordinator Provider

Consumer	Start	End
https://edge-contributor-1.com	2023-03-12 20:30	2024-11-28 17:22
https://edge-contributor-2.com	2023-03-12 20:31	2024-11-28 17:22
https://edge-contributor-3.com	2023-03-12 20:32	2024-11-28 17:22

Rows per page: 5 1-3 of 3

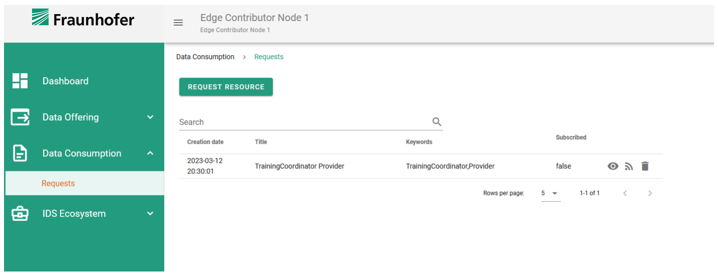
Expired agreements:

Consumer	Start	End
No expired agreements		

Rows per page: 5 –

Fig. 6: Offering of the Training Coordinator and its requester

We can also take a look in the UIs of the Dataspace Connectors of a mobile application and the Training Coordinator, to get a better idea of the offerings



Creation date	Title	Keywords	Subscribed
2023-03-12 20:30:01	TrainingCoordinator Provider	TrainingCoordinator,Provider	false

Fig. 7: Request of one mobile application

both applications provide, as well as which components have agreements which allow the use of these offerings. These can be seen in Figure 6 and Figure 7.

Lastly, we can find helpful functions for development in the "dev mode" tab in the Web Interface. Its currently two functions, triggering the training of the Machine Learning and displaying the current values of the hyperparameters, can be tested in a simple way. If the training for the Machine Learning is triggered in the UI, we can find a log message in the Docker container of the Machine Learning, which indicates that the function was called successfully. Even though the function was not yet included in a useful way for the scenario, that way we can verify that the intended function is triggered through the Middleware route. The same can be done for the function to display the hyperparameters. If it is pressed, the current hyperparameters are displayed. If the hyperparameters are modified, which could be done by starting a terminal in the container for the Machine Learning, and using the function to set hyperparameters, the newly set set of hyperparameters will be displayed instead in the Web Interface. This function also helped figuring out, that the aggregation was not working as intended. As described in chapter 1.1 the hyperparameters aren't modified later, therefore the aggregation should change the hyperparameters at most one time. And since all hyperparameters are the same for each mobile application, the aggregation wouldn't change them at all. However even after manually changing the hyperparameters for one mobile application, as soon as the hyperparameters were exchanged, only the original values were set. After watching through the logs of the Training Coordinator, we could only figure out that one point the aggregation did happen as intended, but shortly again was executed with the original values. Even though the source of the error wasn't figured out, it showed that the exchange itself, as well as the aggregation, were working.

4 Conclusion

We have presented a scenario, in which patients can use an application which collects and handles personal vital data to predict a patients health state and exchanges hyperparameters from the Machine Learning with an external application for data aggregation. We were successful in implementing everything in the scenario, that we intended to include into it. However that does not mean that the implementation is without flaws. The fact that the aggregation itself works, but at some point during the exchange between Training Coordinator and mobile application, the updated set of hyperparameters is not handled as intended and therefore not set correctly for the Machine Learning is unfortunate. And due to time constraints, we weren't able to find the error in the implementation.

Also some functionalities that would have been nice and could have resulted in a more well suited implementation for the scenario couldn't be added in time, like updating the hyperparameters during the execution of the application, so that the aggregation would change multiple times instead of only once. Nonetheless the resulted implementation is a good basis for the scenario and can be used for future extended scenarios, where the usage of the IDS, when handling personal data, can be fully explored.

References

1. IDS Github, https://international-data-spaces-association.github.io/DataspaceConnector. Last accessed 30 March 2023
2. Frauenhofer IDS, https://www.dataspaces.fraunhofer.de/en/InternationalDataSpaces.html. Last accessed 30 March 2023
3. Health Statistics. University of Rochester, https://www.urmc.rochester.edu/encyclopedia/content.aspx.
4. Health indicators. Kolb, S., Burchartz, A., Oriwol, D., Schmidt, S. C. E., Woll, A., & Niessner. International journal of environmental research and public health, https://doi.org/10.3390/ijerph182010711. 2021.
5. Faker Library, https://faker.readthedocs.io/en/. Last accessed 30 March 2023
6. Challenges storing Health dataset, https://winzip.com/blog/enterprise/healthcare-data-storage/
7. MongoDb, https://www.mongodb.com/industries/healthcare.
8. Machine Learning Model, https://www.javatpoint.com/machine-learning-models
9. Decision Tree Algorithm, https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html
10. Apache Camel, https://camel.apache.org/. Last accessed 30 March 2023
11. Enterprise Integration Patterns, https://www.enterpriseintegrationpatterns.com/. Last accessed 30 March 2023
12. Nginx, https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-open-source/. Last accessed 30 March 2023
13. Data Provider, https://www.international-data-spaces-association.github.io/DataspaceConnector/CommunicationGuide/v6/Provider
14. Data Consumer, https://www.international-data-spaces-association.github.io/DataspaceConnector/CommunicationGuide/v6/Consumer

Appendix

Table 1: Health Dataset Description

Attribute	Type	Description	Value Range	User at Risk Range
First Name	User Data	The attribute value is the first name of a user.	[*]	na
Last Name	User Data	The attribute value is the last name of a user.	[*]	na
Gender	User Data	The attribute value is the gender of the user.	[male, female, nonbinary]	na
Age	User Data	The attribute value is the current age of the user.	[1, *]	na
Weight (kg)	User Data	The attribute value is the current weight of the user.	[4, 120]	na
Height (cm)	User Data	The attribute value is the current height of the user.	[50, 200]	na
Step Count	Health Stats	The attribute value is the step count of the user for the day. It is calculated with regards to the hour of the day.	[0, 30000]	na
Stress Level	Health Stats	The attribute value is the current stress level of the user.	[0, 5]	na
Blood Oxygen Level	Health Stats	The attribute value is the current blood oxygen level of the user.	[90, 100]	na
Sleep Hours	Health Stats	The attribute value is the sleep hours of the user.	[4, 10]	na
REM Sleep hours	Health Stats	The attribute value is the rapid eye movement (REM) sleep hours of the user.	[0, 4]	na
Calories Burnt	Health Stats	The attribute value is the calories burnt of the user for the day. It is calculated with regards to the hour of the day.	[0, 3000]	na
Heart Rate	Vitals	The attribute value is the current heart rate of the user.	[50, 150]	<56 or >143
Systolic Blood Pressure	Vitals	The attribute value is the current systolic blood pressure level of the user.	[50, 90]	<52 or >86
Diastolic Blood Pressure	Vitals	The attribute value is the current diastolic blood pressure level of the user.	[100, 140]	<102 or >137
Body Temperature	Vitals	The attribute value is the current body temperature of the user.	[97, 103]	<97 or >102.5
Breathing Frequency	Vitals	The attribute value is the current breathing frequency rate (breath per minute) of the user.	[10, 22]	<11 or >20