ESSnet Smart Surveys

Grant Agreement Number: 899365 - 2019-DE-SmartStat

https://webgate.ec.europa.eu/fpfis/wikis/display/EstatBigData/ESSnet+Smart+Surveys+2020-2021

Workpackage 3

Development of a conceptual framework, reference architecture and technical specifications for the European platform for Trusted Smart Surveys

Deliverable 3.2 Report on the Proof-of-Concept

Final version, 31-01-2022

Prepared by:

Massimo De Cubellis , Fabrizio De Fausti, Claudia De Vitiis, Alessio Guandalini, Francesca Inglese, Mauro Bruno, Giuseppina Ruocco, Raffaella Maria Aracri (ISTAT, Italy)

Nils Meise (DESTATIS, Germany)

Joeri van Etten (CBS, The Netherlands)

Franck Cotton (INSEE, France)

Work package Leader:
Claudia De Vitiis (ISTAT, Italy)
devitiis@istat.it; +390647737401

Table of contents

Introduction:

Overview on the Proof-of-Concept

Background

The Work Package 3 (WP3) is carrying out preparatory work to create a European-wide platform, to share and re-use smart survey solutions and components on the TSS_u platform. The main goal of the Work Package 3 is the development of a conceptual framework, a reference architecture and technical specifications for a European platform for trusted smart surveys, including support for secure private computing to avoid data concentration (e.g., secure multiparty computation), full transparency and public auditability. The first phase of the work consisted in the conceptualisation of a general platform for trusted smart surveys for collecting data for official statistics, following a top-down design approach from reference frameworks to detailed technical specifications. The outcome of the work carried out in this first phase in 2020 was presented in deliverable D.3.1 "Report on the Preliminary Framework".

The second phase consists in the development of Proofs-of-Concept, in the form of modular prototype elements for essential aspects of the architecture such as:

- active and passive data collection and the use of machine learning for the identification of activities, identification of parallel tasks , missing data, etc.;
- privacy-preserving computation solutions;
- full transparency and auditability of processing algorithms;
- integration of incentive schemes into the platform.

The activities concerning the Proof-of-Concept are organized in Task 3.2 of WP3, according to the following list of sub-tasks and corresponding involved partners:

3.2	Development of Proof-of-Concept	**Istat**
3.2.1	Data collection and survey methodology	**Istat** Destatis Cbs
3.2.2	Use of machine learning for evaluating collected data	**Istat** Destatis Cbs
3.2.3	Privacy-preserving computation solutions	**Cbs** Istat Insee
3.2.4	Process auditability solutions	**Insee** Istat
3.2.5	Integrating of incentive schemes into the platform	**Destatis**

The planning of the PoCs was described in Deliverable 3.1 where the experimental activities were grouped into two sets, according to the perspective of the issues addressed: one group has a methodological point of view, while the second subset concerns more technical and architectural aspects. Furthermore, this distinction follows the structure of the two working groups set up within ESSNet with the dual purpose of linking the activity of Work Package 2 and Work Package 3, and linking the activities of the sub-tasks within WP3.

At the end of the second year and considering the results of the Proof-of-Concept, the framework will take a step towards the definition of the platform specifications and requirements. This final part of the activity will be described in Deliverable 3.3.

Objectives of the Proof-of-Concept and main results

The general purpose of the PoC is to subject the main aspects of the platform described in the Preliminary Framework to operational verification, to assess whether the characteristics outlined or modeled are applicable in the practice of smart surveys from different perspectives, and to provide evidences and inputs for the definition of the enhanced framework and the specification of the methodological and technical requirements of the platform.

In particular: i) the PoC on methodology and machine learning aims at designing and developing a G*eneralized Machine Learning Component* (GMLC) for data provided by the same type of sensor (e.g. accelerometer, gyroscope, thermometer), divided into different software modules that will allow it to be applied to different contexts and survey needs; ii) the PoC on Application of Privacy Enhancing Technologies (PET) tests the applicability of various privacy techniques to the Trusted Smart Survey platform, and discusses the practical feasibility of the proposed architecture components; iii) the PoC on incentive schemes models how an incentive in form of gamification can be included into a Trusted Smart Survey (TSSu) and highlights which gamified elements are suitable for an overall implementation and how they can be used in conjunction with a targeted focus on respondents.

Methods, incentives and privacy preservation are the dimensions explored in these tasks impacting not only on the deployment of the application services to be provided by the platform and on the technical requirements, but, more in general, on the quality of the results and therefore on the (smart) survey total error (measurement and representation errors). In these PoCs, important issues arise related to the active and passive collection of data and their processing, such as the trade-off between data quality (model accuracy, response rate and respondent's behaviour) and burden and the trade-off between quality control and privacy.

The PoC on architecture and metadata tests the assumptions made in the previous tasks (3.1.3 and 3.1.6) of Deliverable 3.1, and models the business and application layers, to propose a set of application services to be provided by the platform. Furthermore, the PoC describes the architectural aspects and the metadata related to smart data processed in the Machine Learning (ML) PoC.

Summary of the deliverable

The deliverable is divided into four chapters. In the following, the executive summaries of the chapters are reported.

Chapter 1: Treatment of sensor data for Smart Surveys through Machine Learning: a Generalized Module "PoC on methodologies"

The application of Machine Learning (ML) techniques is a central aspect of smart surveys. The PoC of sub-tasks 3.2.1 and 3.2.2, which aims at the definition and implementation of a generalized machine learning component, wants to contribute to the development of the European platform.

In a Smart Survey platform, it is fundamental to use software modules that can be applied to different contexts and surveys. Machine Learning is an essential technique for sensor data processing and the fact that its implementation is agnostic draws great interest.

The goal of the PoC is to design and develop a G*eneralized Machine Learning Component* (GMLC), divided into different software modules that will allow it to be applied to different contexts and survey needs. In addition, sensors do not often measure directly variables of interest and the PoC will explore ways to assist. A machine learning module can be generalized to different surveys if it deals with data provided by the same type of sensor (e.g. accelerometer, gyroscope, thermometer).

The GMLC is developed on a cross-survey component performing multi-class supervised classification tasks, although the extension to regression tasks can be implemented. The generalization of the process is realised considering different modules that perform a specific function in the pipeline. Modules can be interposed into the process in a different order and new modules can be added to fit new surveys.

ActivPAL and iLog datasets constitute a baseline for developing a component in terms of modularity. Both data collected from a diary and sensors by Health pilot (Work Package 2.3) and in SmartUnitn(Two) surveys are used to build a more generalised pipeline. The use of data collected through different devices (wearables for Health pilot and smartphones for SmartUnitn (Two) surveys) have implications on the results. In fact, while for the activPAL use case, the performance of the physical activity classifier is good (accuracy of 87.6%), for the iLog use case, the performance of the "mean of transport" classifier is lower (accuracy of 61.6%).

These differences are due to the good quality of the data collected in a controlled environment, such as the laboratory for the annotation of the labels, and also to the wearable instrument activPAL that measures the acceleration with a good sampling rate, and a systematic collection of the associated acceleration signal. In SmartUnitn(Two) surveys, for some classes, the performance is very high when the associated signal is very characteristic (for example because of the presence of high-frequency vibrations). For other classes, for which the acceleration profile is very similar, the performance is lower. The worse performance of this use case compared to the activPAL can be explained by the difference in the types of activities we want to classify in the two cases, and by the differences in the device position, constant in one case and variable in the other. In fact, signals from smartphones present a great variability of the acceleration profile for the same activity, due to the position in which the device is placed.

Chapter 2: - Application of Privacy Enhancing Technologies to the TSSu platform

The main objective of the proof-of-concept is to test the applicability of various privacy-enhancing technologies to the Trusted Smart Survey platform, with the purpose of enabling production of statistics without access to respondent level microdata. The first part of this 'test' described is theoretical. It aims to answer the question: "given the supposed possibilities and shortcomings of PET's, how can we design a platform architecture that utilizes these technologies to the best of their abilities?." The second part aims to answer questions regarding the practical feasibility of the proposed architecture components. Given limited resources, we chose to focus on the component regarding statistical analysis using secret sharing based MPC.

The result of the first part is a description of an example platform architecture that allows for the implementation of the PET's required for production of aggregate statistics without compromising respondent privacy. Combined with the other architectural results presented in this deliverable, these results can serve as a basis for the next iteration in development of a platform architecture where all these aspects are combined. Finally, from the architectural part of the PoC, the chapter highlights: that for a well-designed and well-defined survey, the architecture required for oblivious data-analysis is not much more complicated than one would expect for a platform without; quality control, logging and hence survey maintenance without compromising privacy is difficult; design and development of surveys is incredibly difficult given the restrictions of privacy enhancing technologies. These difficulties are inherent to the goal of applying these technologies and can hence not be overcome by technological advancement.

In the second part, the process of adapting the existing PET to the statistical use case is described by developing a toy model application that enables aggregate statistical analysis using multi-party computation for a general (smart) survey. This showcase should not be interpreted as a demonstration of how to develop a production-level application for privacy preserving statistical analysis. Rather, it presents some very basic ideas and considerations that can be used in the development of such an application. The final conclusions with respect to this part of the PoC is that while it seems possible to adapt existing open-source technology to statistical use-cases typical for the TSSu platform, the process is highly involved. As such, proper implementation embedded within the TSSu platform likely requires experts in the fields of statistics, software engineering and cryptography. Alternatively, one might consider commercial offerings.

Chapter 3 – Full transparency and auditability of processing algorithms. "PoC on architecture and metadata"

The main goal of this PoC is to test the assumptions and benchmark the results concerning:

- The previous inventory of official statistical standards and ongoing projects, released by Task 3.1.3[1] dealing with the preparatory work for the design of a TSSu platform and integration in existing architectural framework
- The approach adopted by task 3.1.6 Metadata and Process auditability, leading to the design of a Metadata Repository (MR) composed of several subsets, based on existing frameworks and ontologies.

The chapter is structured in two main parts, describing respectively the architectural analysis and the metadata captured for the Machine Learning (ML) PoC.

In the architectural use case, the main steps of a generic TSSu are modelled. More in detail, starting from the description of the statistical process in terms of GSBPM phases and BREAL business functions, the architectural PoC has modelled the business and application layers, to propose a set of application services to be provided by the platform. The analysis of operational models and architectural scenarios has complemented the theoretical assumptions related to the architectural layers. The specification of the dimensions impacting on the deployment of the application services, such as data storage and processing environment or adoption of Privacy preserving techniques is essential to define the requirements of the technical infrastructure of the platform.

The second part deals with the description of process metadata related to the tasks performed by the ML generalized module, developed to process accelerometer data. Starting from the conceptual subsets of the MR, the PoC has allowed to describe the process steps executed, to track data transformations, to document and assess the methods applied in each step.

The last paragraph summarizes the main results achieved by the PoCs, the main lessons learnt and some open issues to be addressed, to achieve an enhancement of the reference architecture delivered at the end of the first year of the project. One of the main open issues is to highlight the relationship between the dimensions explored by the single tasks, such as: methods, incentives, privacy preservation, process auditability, assessment of smart data quality, technical requirements.

[1] Deliverable 3.1 Report on the Preliminary Framework, available from: https://ec.europa.eu/eurostat/cros/content/wp-3-conceptual-framework-european-platform_en

Concerning the main results achieved, the architectural PoC has demonstrated the feasibility of process standardization, to foster the integration between traditional and smart survey tasks. This analysis supports the specification of the main technical requirements of the TSSu platform, and is essential to plan the development activities. The metadata PoC has confirmed the initial assumptions, based on a central repository for the collection and management of the process metadata related to specific tasks of sensor data processing.

Chapter 4: – Integrating of incentive schemes into the platform

The activity of task 3.2.5 explores incentives schemes in a Proof of Concept to understand how gamification could be utilized for a gamified (smart) surveys. It asks which steps are required to design a gamified survey by looking at different survey phases, and use patterns by understand these phases as the respondents or player journey respectively. In addition, prospective respondents are handled with an abstract concept of user types. The implementation of gamification results in a trade-off: It eases the hard task of completing a longer running survey for respondents, and at the same time requires more and new forms development by a survey agency. These additional development activities are mainly due to the different mode of app-bases surveys and design requirements that result from that decision.

The chapter consists of the following parts. First, an introduction that gives an insight why gamification was chosen and under what assumptions the PoC was conducted. Second, which assumptions are useful when designing a gamified survey for a specific group of respondents (2. Desired User Types). Third, the actual design of a gamified survey, following design principles for apps and gamified apps in particular (3. User Journey). Finally, a conclusion discusses the results and identifies open issues for further investigation (4. Results).

Main general conclusions and future steps

The four PoCs succeded in verifing the feasibility of specifics parts of the conceptual framework descripted in deliverable 3.1. This constutes a step foward in the direction of a more practical definition of the European platform, because experimenting with the different aspects of a smart survey produced a more concrete vision of what can be achieved and with which problems. What will also contribute to the description of the structure and the components of the platform are the inputs coming from the pilots carried out by Work Package 2 (WP2), in particular those descripted in deliverable 2.8 and 2.5.

A further step, that will be part of the definition of the enhanced framework, is the activity concerning the technological infrastructure: technical specifications wil be provided to support the generalized ML component, the private data analysis and the metadata management, in light of the other pocs and after them. This activity, currently ongoing, is taking into account also the inputs provided by the results of the WP2 pilots (deliverale 2.8 and 2.5 in draft)

Finally, some important open issues, to which the work currently underway is trying to provide answers or discussions, are:

- WP2 field tests will be used to benchmark the architectural and metadata frameworks resulting from the PoCs, to complement the analysis with the process steps performed, as well as the service functionalities and the methods actually implemented.
- In order to integrate smart data pipeline and traditional survey tasks, some aspects, such as logistics and data collection monitoring, need to be adjusted according to the type of data sources.
- Distinguish in architecture and quality metadata the main features of a smart survey with respect to the source of the data.

- The machine learning should be widen in terms also of online learning and active learning, including the respondent involvement.
- Regarding privacy preservation, the active-passive trade-off (also in-device versus in-house processing) on data quality should be taken into account.
- Also the trade-off among quality, burden and privacy preserving aspects of data collection in the smart surveys should be discussed.

Task 3.2
Sub-tasks 3.2.1 and 3.2.2

Treatment of sensor data for Smart Surveys through Machine Learning: a Generalized Module "PoC on methodologies"

Prepared by:

E. Cerasti, M. De Cubellis, F. De Fausti, A. Guandalini, F. Inglese, A. Pappagallo, F. Pugliese, M. D. Terribili

Task leader: Fabrizio De Fausti

Outline

Executive summary

The application of Machine Learning (ML) techniques is a central aspect of smart surveys. The PoC of sub-tasks 3.2.1 and 3.2.2, which aims at the definition and implementation of a generalized machine learning component, wants to contribute to the development of the European platform.

In a Smart Survey platform, it is fundamental to use software modules that can be applied to different contexts and surveys. Machine Learning is an essential technique for sensor data processing and the fact that its implementation is agnostic draws great interest.

Our goal is to design and develop a G*eneralized Machine Learning Component* (GMLC), divided into different software modules that will allow it to be applied to different contexts and survey needs. In addition, sensors do not often measure directly variables of interest and the PoC will explore ways to assist. A machine learning module can be generalized to different surveys if it deals with data provided by the same type of sensor (e.g. accelerometer, gyroscope, thermometer).

The GMLC is developed on a cross-survey component performing multi-class supervised classification tasks, although the extension to regression tasks can be implemented. The generalization of the process is realised considering different modules that perform a specific function in the pipeline. Modules can be interposed into the process in a different order and new modules can be added to fit new surveys.

activPAL and iLog datasets constitute a baseline for developing a component in terms of modularity. Both data collected from a diary and sensors by Health pilot (Work Package 2.3) and in SmartUnitn(Two) surveys are used to build a more generalised pipeline.

The use of data collected through different devices (wearables for Health pilot and smartphones for SmartUnitn (Two) surveys) had implications on the results. In fact, while for the activPAL use case, the performance of the physical activity classifier is good (accuracy of 87.6%), for the iLog use case, the performance of the “mean of transport” classifier it's not as good (accuracy of 61.6%).

These differences are due to the good quality of the data collected in a controlled environment such as the laboratory for the annotation of the labels, and also to the wearable instrument activPAL that measures the acceleration with a good sampling rate, and a systematic collection of the associated acceleration signal. In SmartUnitn(Two) surveys, for some classes, the performance is very high when the associated signal is very characteristic (for example because of the presence of high-frequency vibrations). For other classes, for which the acceleration profile is very similar the performance is lower. The worse performance of this use case compared to the activPAL can be explained by the difference in the types of activities we want to classify in the two cases and by the differences in the device position, constant in one case and variable in the other. In fact, signals from smartphones present a great variability of the acceleration profile for the same activity, due to the position in which the device is placed.

The report is organized as follow. In section 1, we explain the reasons behind the idea to develop a GMLC and its potential applications in the context of smart surveys. In section 2, we give a brief description of the sources provided by Work Package 2.3 Health pilot (activPAL) and the SmartUnitn(Two) (iLog) survey. Section 3 describes the GMLV, focusing on methodological and technological issues and relative solutions adopted during the development of the pipeline. Sections 4 and 5 present two use cases and describe the processes used to treat the sensor data collected employing both a wearable device (activPAL - accelerometer experiment) and a smartphone device (iLog app - SmartUnitn(Two) survey).

1. A Generalized Machine Learning Component: Motivation and Potential Applications

One of the main characteristics of the TSS is the use of data collected through intelligent sensors (see ESSNet Smart Survey - Deliverable 3.1 Report on the Preliminary Framework).

These data are generally unstructured (e.g., texts, images, signals), thus, they cannot be processed with traditional statistical analysis tools. However, ML techniques are particularly useful for this scope since they can discover hidden relationships between data.

In the smart survey's production, the application of ML techniques can take place in different stages of the process such as data integration, classify and coding, review and validate, edit and imputation (see ESSNet Smart Survey - Deliverable 3.1 Report on the Preliminary Framework - paragraph "6.3 GSBPM Process Phase for Sensor Data and ML Algorithms"). Here, we will see how to build an ML component to support the collection phase.

The basic and general idea of the Smart Survey project is to design a platform for the development of TSS. This implies the design of TSS production pipelines with the use of modular and generalized components.

For the generalization of our component, we have considered the following requirements:

- it must be designed in a modular way, in the sense that the component is made of modules or programs implementing well-defined process functions with inputs and outputs (such as the pre-processing module);
- it can be inserted in different stages of the production process, depending on how the survey is designed;
- It must be agnostic concerning the survey.

We know that achieving each of the above requirements within a single algorithm is challenging, nonetheless the goal of the PoC is to investigate the aspects necessary for our component to perform the same task on different surveys.

The main requirement for our component is to be able, to carry out multi-label supervised classification of a variable of interest for the survey, starting from signal data.

The component created within the ML PoC will classify a variable of interest of a specific survey, for example, the physical activity a respondent is performing or the means of transport he is using, the input being the signals coming from devices used by the respondent (i.e., wearable or mobile phones).

The GML component can be applied to several contexts. We are going to present a couple of possible use cases:

- to train the component on data from users participating in a pilot-survey, to create an ML model to embed into an App that will be distributed to carry out the real smart survey.

- insert the untrained component in the device where it learns from the single user's behaviour how to classify the variable of interest for the survey.

In the first case, the GML component is used to make inference on a group of respondents while in the second case the GML works with a single respondent.

2. Data Sources

In this section we are going to describe the dataset used for the methodological PoC, coming from two different sources: data provided by WP2.3 for the Health pilot, used to perform an accelerometer experiment by the activPAL device; data provided by the University of Trento, which are the result of a smart survey carried out in 2018 within the SmartUnitn(Two) Project (iLog).

A descriptive data analysis has been carried out both on iLog and activPAL data sources. These analysis aims to understand the data we are dealing with, highlighting characteristics (size, structure, observation period) of the sources and problems to face, such as outliers or missing data.

2.1 Accelerometer Experiment (activPAL)

The accelerometer experiment involved two phases. The first phase was executed in a laboratory, where participants performed various exercises, under supervision, while wearing the activPAL device by placing it on their thighs. Each participant performed the following activities for about 5 minutes: cycling (slowly or hard), sitting, standing, walking and running. The climbing stairs activity was performed twice, and the jumping activity was performed 5 times.

After the lab session, each participant wore an activPAL device for one week without supervision. During this time, respondents were asked to keep a diary in which they wrote down what physical activities they performed while wearing activPAL.

Tab. 1 – Activity in the lab and weekly dates from respondents

Accelerometer experiment	
Activities in the lab	**Weekly dates from respondents**
sitting standing walking running climbing stairs jumping cycling (light and heavy)	when they: - worked - cycled - sported - slept

The observed population in the experiment consisted of 27 people (at the start there were 39, but after early phases of the process 12 were eliminated for invalid or unavailable data), which the following characteristics: sex, weight, height and body mass index.

(For more details on objectives and methodology adopted in the accelerometer experiment see WP2 Del. 2.3).

2.1.1 Data Description

activPal data considered in this PoC refer only to the activities carried out in the laboratory. The complete observation period lasts 10 days, from 30th September 2019 to 10th October 2019, and each activity was performed within 1 day. We matched data collected in the laboratory with data inferred from activPAL accelerometer, to build a labelled dataset. After this matching, we picked the 27 users with the highest quality information. We put the resulting matches in an opportune data structure.

Data from activPAL sensor are available, for each user, as a ".PAL datx" file. Such a file need to be opened in the "PAL Analysis" framework (v8.11.5.64), where a dashboard shows the activities performed by the user, together with a set of information about the sensor. This information can be exported as .csv or .pdf files. In particular, we exported: the "15 Seconds Epoch" report, containing the activities detected from the activPAL sensor every 15 seconds; the "Uncompressed acceleration data" report, containing accelerometer data collected along the entire period.

activPal data cover all the observation period without any stop (no missing data), by generating samples with a constant frequency of 20 Hz, thus with a period of 50 milliseconds. For example, for 15 seconds of activity 300 observations (samples) are generated. More specifically, as each activity duration in the laboratory was of 5 minutes, around 6000 rows in data for each activity were generated.

Each row of an activPAL sample contains X, Y and Z-axis values of the acceleration. *Figure 1* below (Kuster, R.P. et al.,2020) shows which direction of human body movement the X, Y and Z axis correspond to.

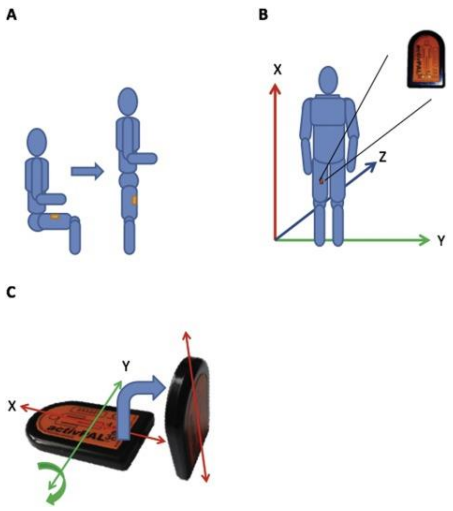


Figure 1

Note that the components of the acceleration *a* along the X, Y, Z axes are expressed in numerical values from 0 to 255 in *g* units (gravity acceleration) according to the formula *G= (a-127)/63.*

In the following frequency analysis, we are going to show X, Y and Z axis acceleration values for some specific activities, that is: "cycling light", as an example of an activity where the same movement is repeated at each time instant (Figure 2a, 2b and 2c); "sitting", as an activity such that when the user is seated, the acceleration value in the Z component corresponds to the gravitational acceleration (Figure 3c, where Z = 200 it holds that G is around 1g, that corresponds to 9,81 m/s^2).

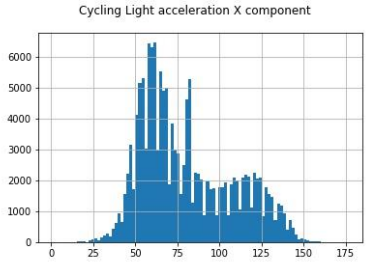


Figure 2a

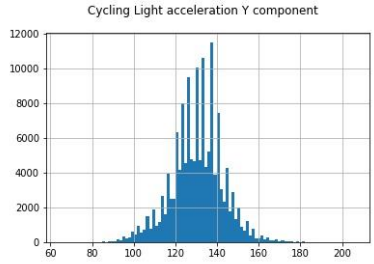


Figure 2b

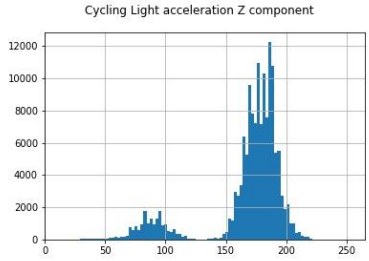


Figure 2c

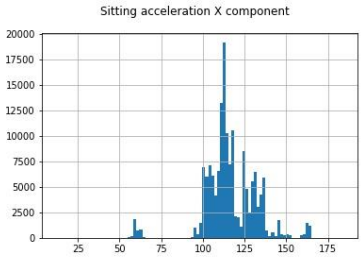


Figure 3a

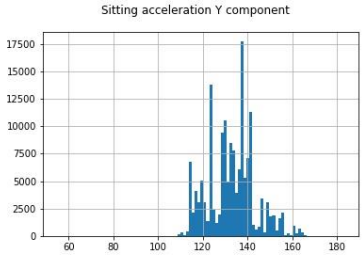


Figure 3b

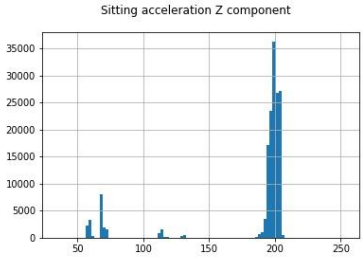


Figure 3c

2.2 SmartUnitn(Two) Project (iLog)

The goal of the SmartUnitn(Two) project was to conduct a survey to understand how the spatio-temporal organization of the students of different degrees affects their academic performances. In the survey design, many aspects were considered: a sample of university students defined by considering the comparability between departments and the generalizability of the results; a set of combined data acquired from a smartphone (all sensor data in accordance with the smartphones' specifications), survey (psychological questionnaires) and demographics obtained from the students and third-party (administrative sources held by the University of Trento); a recruitment strategy; tools and measurement (iLog app installed on each student`s mobile phone).

The recruitment strategy covered the following aspects: connecting with students (website developed to show the information on the project and experiment and setting up an email account as the main means of communication for the helpdesk); preparing the on-boarding process (an extended handbook and a synthetic version have been created and provided to the students, in order to explain the purpose of the research, the questions and sensors used and all the procedures to download, install and use iLog); incentive (the way to incentivize students was designed to be monetary: a reward of 20 euro every two weeks was to be given to each participant. In addition, three cash bonuses of 100 and three of 150 euros for the three most deserving students, respectively of the firsts and seconds two weeks, were set. These three subjects were identified based on three criteria: i) how long the application had remained in operation, ii) how often did the GPS, Bluetooth and Wi-Fi sensors turn off, and iii) how many answers had been provided through the questionnaire.

For this experiment, the technological tool used was iLog which provided two functionalities: data collection and time diaries. iLog is designed to collect data from multiple sensors simultaneously, both hardware (e.g., GPS, accelerometer, gyroscope, etc.) and software (e.g., applications running on the device). A dedicated

backend infrastructure manages the synchronizing and storing activities of data streams from smartphones. Time diaries are logs where respondents are asked to detail how they allocated their time during the day. The app can administer the time diary starting from a question consisting of three sub-questions about students' activities, locations and social relations every 30 minutes and one sub-question about the mood. Every triple of questions can be answered within 150/360 minutes from its notification, with a maximum of 5/12 questions stacked in a queue; otherwise, it expires and is treated as null. Questions appear as silent notifications, in order to avoid bothering students and disrupting their activities too much (Busso, Bison and Giunchiglia, 2019).

The dual activity of the app can therefore be defined as divided into two components. The first collects data from the internal sensors in the phone in a completely non-intrusive manner, without the need for any interaction from the user, but keeping him informed of the current collection through an always visible notification. Each phone is equipped with several sensors, whose usage is monitored by the app, in full compliance with the rules agreed for privacy. Therefore, no sensitive content such as message texts, internet searches or call content will be recorded.

The second component requires, instead, the active collaboration of the user, as it involves the administration of a questionnaire. This will appear directly on the smartphone and will be comprised of two groups of questions: the first, concerning the expectations on the day and whether these will be expected or not, will appear twice a day, in the morning and the evening; the second, which consists of 4 short queries, will appear at intervals of 30 minutes in 24 hours. The 4 questions will concern, in order: what you are doing, where you are, who you are, and finally, how you judge your mood (Busso, 2017). A further question is how you are moving (on foot, by car, by bicycle, by train.)

Tab. 2 - Questionnaire/time diaries and sensor data

iLog data	
Questionnaire and Time diaries	**Sensor data**
General information *[Morning] How did you sleep?* *[Morning] How do you expect your day to be?* *[Evening] How was your day?* *[Every half an hour questions]* *What are you doing?* *Where are you?* *With whom are you?* *What is your mood?* *How are you moving?*	Acceleration, Linear Acceleration, Gyroscope, Gravity, Rotation Vector, Magnetic Field, Orientation, Temperature, Atmospheric Pressure, Humidity, Proximity, Position. WIFI Network Connected to, WIFI Networks Available, Running Application, Screen Status, Flight Mode, Battery Charge, Battery Level, Doze Modality, Headset plugged in, Audio mode, Music Playback, Audio from the internal microphone, Notifications received, Touch event, Cellular network info, Activity Performed (Google Data)

The duration of the experiment was two weeks for two phases. The first phase started on 07/05/2018. In these weeks, the students responded to the time diary while iLog collected data from the device. Due to some problems of server overload, as well as errors in downloading and access procedures by students, the effective date of the experiment beginning was 09/05/2018. The second phase started on 21/05/2018. In this week, the students had just to leave the app running - 5 times per day.

Various problems concerning the iLog server, as well as malfunctions of the app itself, occurred during the first phase and, in part, during the whole experiment. This caused the loss of both the consistency of some data and students' participation (273 on 307). Of these, respondents who replied at least once to the questionnaire are 240. In other words, 35 subjects have just downloaded the app, without answering any questions. 3 other subjects abandoned the experiment during the app test between 7 and 8 May, hence 237 subjects started compiling. The total events recorded between May 9 and June 7 were 130,489 (just over half of those expected). Of these, 110,702 were detected in the first two weeks and 19,787 in the second two weeks. Since not all the students performed the task correctly, the observations produced by the 237 participants were further selected. The *perfect* participants (i.e., those who had a complete set of observations for each variable) were:

a) about 22 for the first period
b) about 23 for the second period

Relaxing the requirements, for the first two weeks the subjects were considered valid if they:

a) failed to reply at most 7 times for a continuous period of more than 10 hours
b) completed the questionnaire for at least 13 days
c) provided a number of valid answers greater than 300.

For the second two weeks, the subjects were considered valid if, in addition to having decided to participate in the second part of the survey, they:

a) failed to respond more than 6 times for a continuous period of more than 13 hours
b) completed the questionnaire for at least 11 days
c) provided a number of valid answers greater than 100

Then, the total number of valid subjects are 184 for the first period and 113 for the second period.

2.2.1 Data Description

The data considered refer to people who participated in the survey for at least 29 days was 134.

iLog data structure is organized as a file system containing 149 directories, one per respondent; in each directory, there are other nested directories containing sensors and questionnaire (diary) datasets.

Conversely, to activPal data, the iLog sensor data do not cover all the observation periods without any stop. Sensors' data flow stops and restarts often. It means that the sampling frame is not constant, because the lag between two observations is not constant neither. Furthermore, iLog data show also some outliers: the analysis of mean, median, max and min values of the three variables shows outlier values. The units containing these outlier values are just a minority, but it is important to detect and treat them in some way, correcting or dropping these values. The frequency distributions of the three acceleration components X, Y,

Z reflect the specific characteristics of their domain: for instance, the histogram below shows a higher frequency for the Z component around 10 and -10 values, which can be interpreted as the gravitational acceleration (9.81 m/s^2)

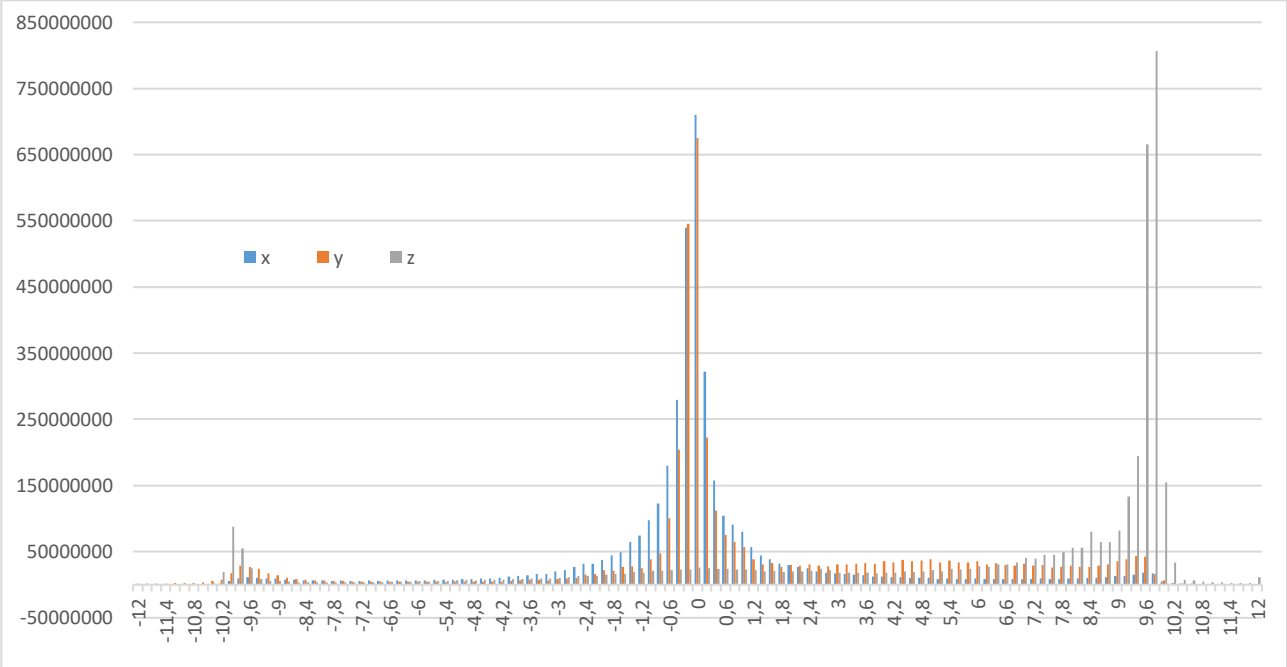


Figure 4

3. Development of the Generalized ML Component

Generalized Smart Survey Component

Different ML techniques can be particularly useful for designing a Trust Smart Survey. In this Proof of Concept, we try to address issues that may arise by designing a generalized component suitable for different surveys and we suggest some solutions.

We decided to focus on a cross-survey component performing multi-class supervised classification task, although the extension to regression tasks can be implemented.

Another important aspect is modularity. The entire process is subsumed into modules that perform a specific function in the pipeline. Modularity is helpful to the generalization process as modules can be interposed into the process in a different order and new modules can be added to fit new surveys.

How we proceeded

To design the pipeline of the process and implement the different modules we thought of two use cases:

- Phase 1 -use case 1activPAL: We train a model on a dataset collected during a pilot phase and relative to different users. The model is then deployed in the real survey during the data collection phase to infer the variable “human activity” performed by the user.
- Phase 2- use case 2 iLog: we train a model within a dataset collected during a first phase from only one user, who is asked for the used means of transport. In a second moment of the data collection,

the model is trained with a labelled dataset and used to infer the mean of transport for the next period.

These ML models aim to decrease the statistical burden during the survey.

As shown in the figure below, in Phase 1 we design a draft our pipeline and apply the lesson learnt to phase 2 to extend a more generalised component.

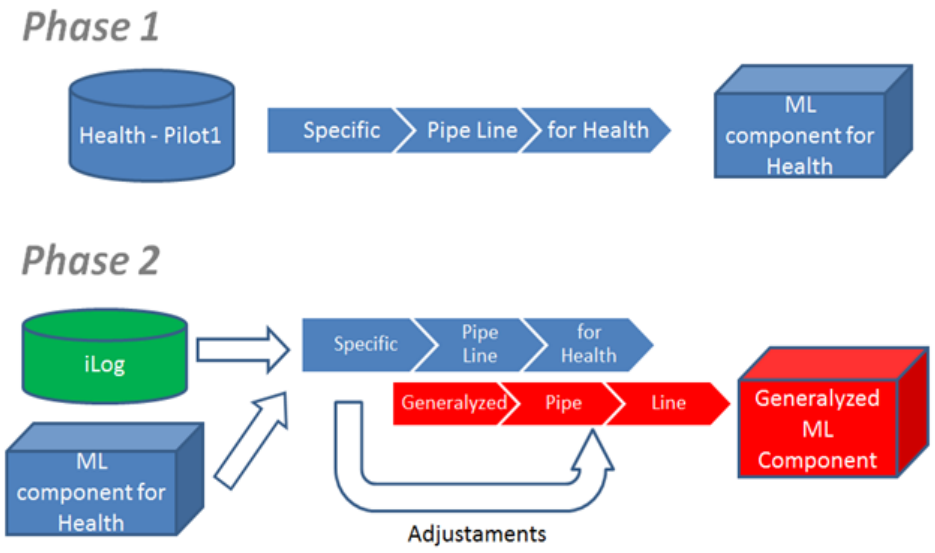


Figure 5

4. Use case 1: activPAL

The first use case we envisioned uses the data collected for pilot 1 of WP2.3 Health, specifically the laboratory phase. In the inference phase, the trained algorithm will need to infer the activity performed by a generic user wearing the wearable device.

The type of task performed by the ML component is therefore a multi-class supervised classification of user activity through acceleration data collected by a wearable tool. In the pilot of WP3.2 for each respondent 8 different activities were annotated during the lab data collection: cycling heavy, cycling light, jump, run, sitting, stairs, stand, walk. Another characteristic aspect of this use case is the permanent position of the wearable device on the body during the whole time period. All these quality aspects may have a good impact on the performance of the ML component in the inference task.

4.1 Process Description

In the following paragraphs, we describe the process necessary for training and measuring the quality of the component ML starting from the data collected in the data collection phase of WP2.3. The pipeline that describes the process consists of modules, each performing a specific function. The modules generate outputs that serve as inputs of subsequent modules (the detailed description of the process metadata is a part of the WP3.metadata Proof of Concept). The modules are designed to be generalized for other surveys by setting some parameters. In figure 6 we show the pipeline schema of the use case 1 activPAL.

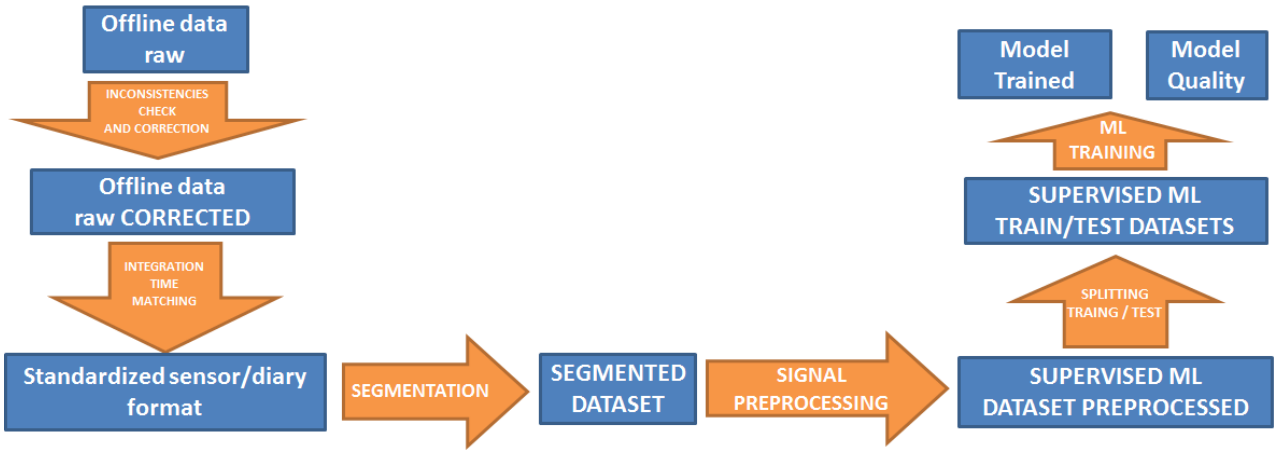


Figure 6

4.1.1 Inconsistencies check and correction

Similarly to the editing and imputation step of GSBPM framework or data wrangling of BREAL framework, this phase, deals with analyzing the provided dataset to check for inconsistencies in both accelerometer data and diary data. In the case of errors, the operations to be made are either to delete the inconsistent data or try to correct it. While the accelerometer data collected through activPAL presented a great consistency, as described in paragraph 2.1.1, those deriving from the diary annotation of the activities show anomalies and inconsistencies about dates and times of start and end of an activity.

The corrections carried out consist of eliminating overlapping time periods between the end of an activity and the start of the successive one. After this step, all lab activities of all 39 respondents were considered.

4.1.2 Integration and time matching

The integration and time matching phases integrate information from two different processes:

- the data collected by devices, in particular, the acceleration values on the three spatial axes X, Y, Z
- the data coming from the annotation phase, in particular the laboratory activity.

This process - details the business function *data wrangling* in the *BREAL* framework and it has the purpose of associating the target variable to the input data. To this scope, a matching is performed on the temporal variable coming from the two datasets. A standardization step is needed since the temporal variable is generally written in two different formats, a human-readable format related to the annotations and the other coming from machine-readable (*Unix-like*) devices, with high temporal resolution more representative of time-varying signals.

At the end of this process, a table will be produced, where rows represent a single sample, (i.e. for activPAL wearable the frequency is 20 samples per second) and the columns contain the time variable in *Unix-like* format, the acceleration values Ax, Ay, Az, the target variable (activity), and the *ActivityBlockIndex*, an index that identifies a block of signals associated with a single annotated activity. The *ActivityBlockIndex* will be used in the next segmentation phase.

This final format is quite generalizable to other surveys with a different target label such as the transport mean or even the numeric variable (ML regression task). In addition, if other physical variables are collected at the same time, other columns, such as gyroscope, can be added.

These last two steps are closely related to surveys. Building a generalized module being able to deal with every kind of missing data or error of annotation is not possible. Part of the generalization process consists of a deep preliminary visualization and analysis of the data and a good description of the metadata, enabling the integration process.

4.1.3 Overview of the next steps

The developed idea to recognize the activity performed by the respondent in a survey with machine learning algorithms is quite simple. It leverages the ability of CNN convolutional neural networks to recognize and classify images. Deep learning techniques today provide a set of architectures of different complexity that are state of the art for the realization of machine learning tasks such as classification, speech recognition and other typical artificial intelligence tasks.

In view of a generalized algorithm, the plasticity of the neural network allows to treat in the same framework signals from different sensors (gyroscope, accelerometer, etc.) in a joint fashion. It is possible to extend the set of variables in input to the neural network, similarly to how to pass from an image in black and white to a colour image helping in this way the algorithm in the recognition task. Furthermore, the substitution of a parameter, such as the loss function, allow this architecture to be extended from a classification task to a regression task.

In our generalizable framework, the technique for transforming constant length pieces of signal (segments) into images is provided by wavelets. In the following sections, we describe these steps in the Segmentation and Signal Preprocessing process.

4.1.4 Segmentation

In the previous steps, we transformed the dataset into a standardized format for supervised machine learning. The signal associated to an annotated activity can be variable in time depending on the type of activity (e.g. jumping is a shorter activity than sitting).

The segmentation phase has the main purpose of extracting chunks of signals of the same length (segments) from the dataset. The length of these segments depends on a trade-off: they have to be not too short because the number of samples needs to be sufficient to characterize the activity, and not too long otherwise it would lead to computational problems.

Each segment consists of a sequence of 128 samples, equal to 6.4 seconds for activePAL. The extraction takes place within each *ActivityBlockIndex,* characterized by the same label related to the activity of the respondent. The extraction starts from a random point of a given *ActivityBlockIndex,* with a probability varying for each activity in order to create a final dataset balanced against the number of segments for each activity.

In this way, the segmentation process leads to the creation of a balanced dataset of the required size.

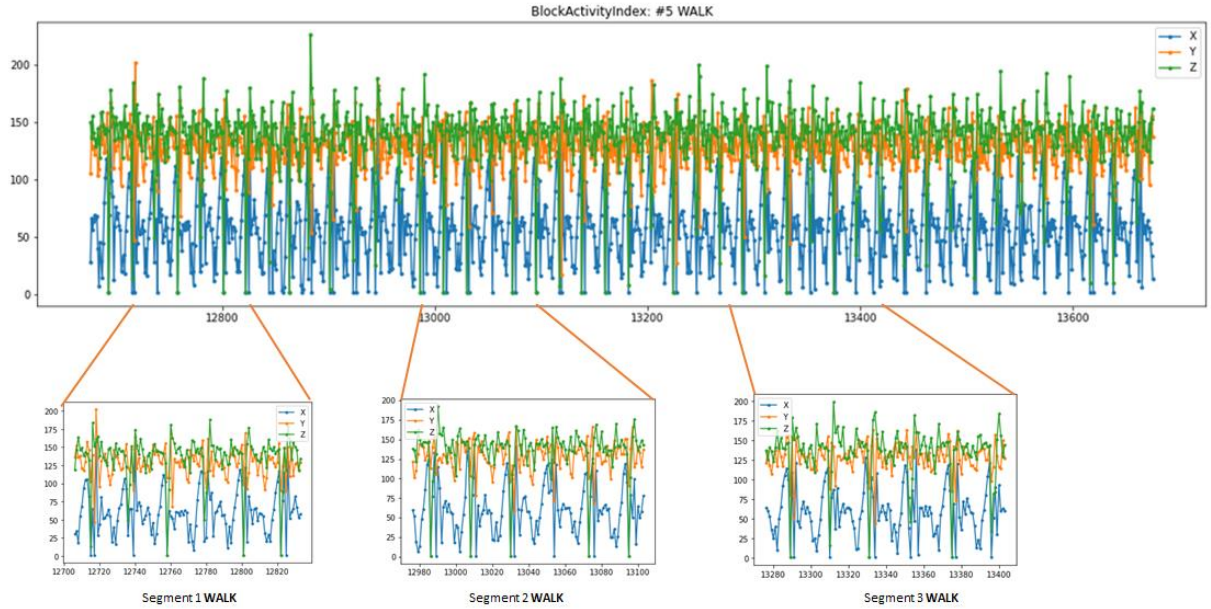


Figure 7

In figure 7, we show the segmentation process for the *BlockActivityIndex*=5 that generates 3 segments of length 128 samples with the label WALK.

4.1.5 Signal Preprocessing

For the recognition of human activities (HAR) a technique often used is wavelets together with CNNs. In our pipeline, we use wavelets to preprocess the 3 components of the 128 samples of the activPAL accelerometer signal coming from the segmentation phase. The preprocessing aims to transform the input data in a format suitable for the machine learning algorithm; in particular, each component of the signal will be transformed into a 2-dimensional object while performing features extraction.

There are two types of wavelet transformation: DWT Discrete Wavelet Transform and CWT Continuous Wavelet Transform (Polikar R., 2006)**.** Although CWT and DWT have a lot in common, they are usually used for different purposes: while DWT is a perfect tool for encoding, like image compression, CWT is mostly applied for signal analysis tasks (Rioul O. and Duhamel P., 1992).

CWT will be implemented at this stage-leveraging the ability of CWTs to distinctly locate features in the time domain. CWT provides an excellent opportunity to extract complex spectral features of a signal (Sadowsky J., 2017).

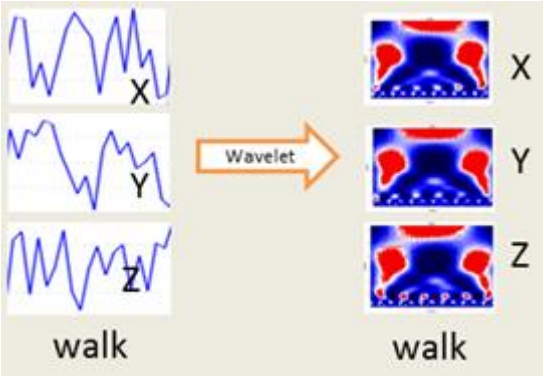


Figure 8

4.1.6 Splitting Train and Test

The dataset created in the previous phase of preprocessing consists of 15000 trials, divided in 10000 samples for training and 5000 samples for testing. This is an essential step in Machine Learning, leading to having an estimate of the error of generalization. In this case study, the subdivision is performed on the ensemble of all respondents, in order to simulate the phase of investigation in which the model is trained on all the customers and distributed on the same ones.

4.1.7 ML Training

The convolutional neural network implemented in our pipeline is inceptionV3, a network showing as a good trade-off between complexity (number of parameters used) and predictive ability. InceptionV3, described in more detail in appendix 1, typically treats images composed of 3 channels, Red, Green and Blue, but in our approach, the channels are replaced by the X, Y, Z components of the acceleration signal, after being transformed into wavelet in the previous phase.

The choice of the hyper-parameters and the choice of the number of epochs in the learning phase come as a result of the analysis of the network performance on a validation set, obtained by a further splitting of the training set.

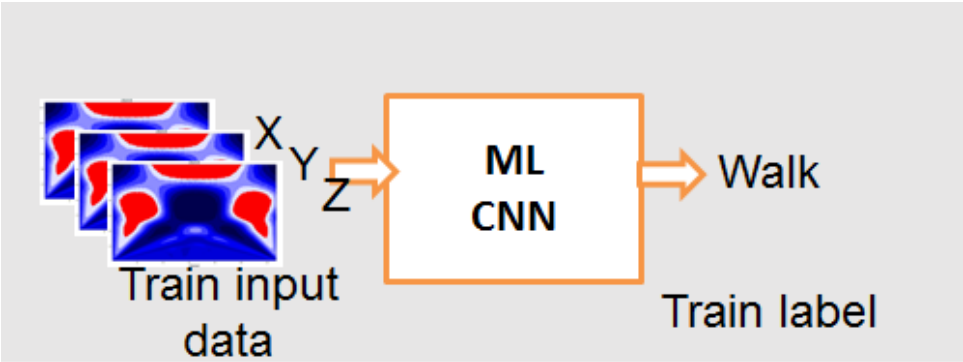
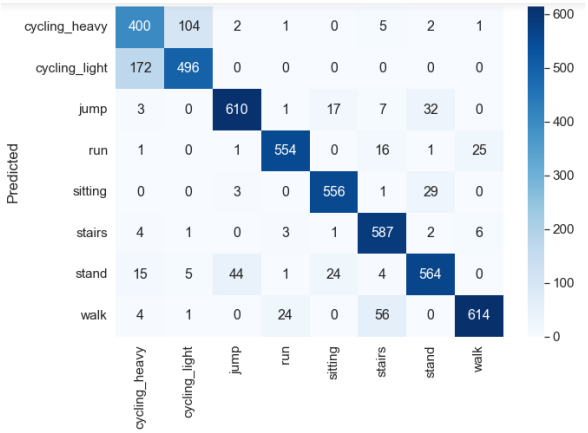


Figure 9

4.2 Output Evaluation

Typical metrics for measuring the quality of a supervised multi-class classification task are the confusion matrix, precision, recall, f1-score and accuracy. The confusion matrix gives information about how each class is classified by the network in relation to other classes. Other metrics such as precision, recall and f1 score are specific to each class. Finally, accuracy is a more global metric. These metrics give an estimate of the performance the model would have in the inference task and therefore its ability to generalize. This esteem is reliable as far as the training sample are similar to the ones presented as input in the inference task. In the activPAL use case the underlying figures show the results obtained on the test set. The values of precision and recall on the 5000 trials divided into balanced classes are quite similar within each activity. Some activities are recognized better than others, for example, stairs, run, jump, sitting while cycling is confused between heavy and light. Globally we can say that with an accuracy of 87.6% the performance of the classifier is good. This is due to the good quality of the data collected, a controlled environment such as the laboratory for the annotation of the labels, and also to the wearable instrument activPAL that measures the acceleration with a good sampling rate, and a systematic collection of the associated acceleration signal.



Activity	precision	recall	f1-score	support
cycling_heavy	0.78	0.67	0.72	599.0
cycling_light	0.74	0.82	0.78	607.0
jump	0.91	0.92	0.92	660.0
run	0.93	0.95	0.94	584.0
sitting	0.94	0.93	0.94	598.0
stairs	0.97	0.87	0.92	676.0
stand	0.86	0.90	0.88	630.0
walk	0.88	0.95	0.91	646.0

Figure 10

5. Use case 2: iLog

The second use case, iLog, uses data from the SmartUnitn(Two) survey described in the previous paragraphs. Our goal here is to train a machine learning component with the data coming from a single user using and extending the pipeline built for the first activPAL use case.

The data taken into account are those collected with the user's smartphone together with the diary data through the iLog application. The variable that we want our component to predict - is the mean of transport used by the user for travelling. The task to be performed as in the first use case is a supervised multi-class classification of the variable “mean of transport” with the following modes: on foot, by car, by motorcycle, by bicycle, by train.

5.1 Process Description

Preliminary analysis performed on the iLog data showed some differences from the activPAL data. Data from the diary present a good standard without specific manual annotation errors, due to the annotation mode implemented in the iLog app. Regarding the data from sensors, these are affected by problems of standardization, such as the non-constant sampling rate due to the heterogeneity of the used devices and problems due to the presence of anomalous and missing data (outliers).

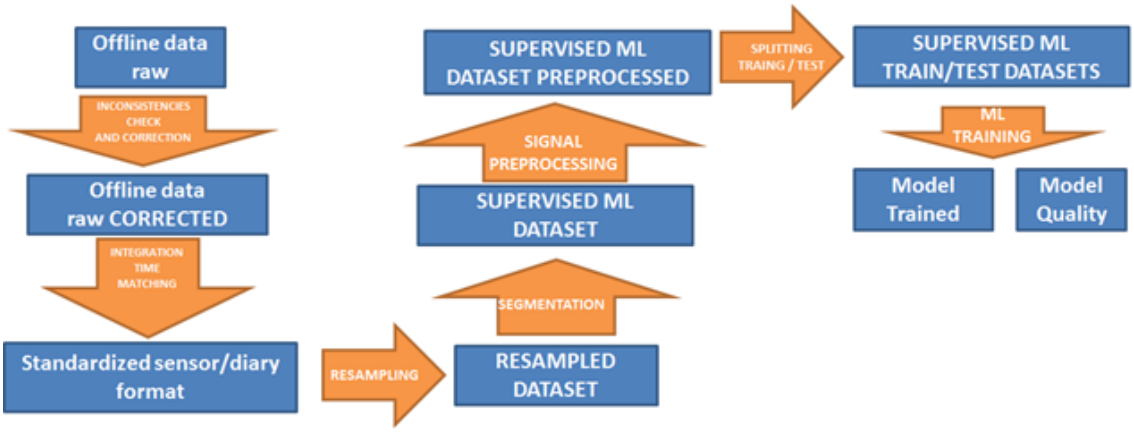


Figure 11

In the inconsistencies check and corrections phase, we limited ourselves to eliminate missing data and outliers in acceleration data. In the integration and time matching phase, we created a table in the standardized format, like the one described in paragraph 4.4.1 in the activPAL use case: on each row, we have a sample and on the columns the time instant, the acceleration values along the three axes, the mean of transport used for the movement and the variable "*BlockActivityIndex*" that identifies the signal block labelled by the user.

The main difference of this data structure from the activPAL one is that here the sampling rate is not constant. The temporal distance between one sample and the next can vary by an order of magnitude, from 1ms to 60ms even for the same user. This variability requires the insertion of an additional resampling module that unifies the frequencies within the data collected by the user. The resampling phase performs linear interpolation of the signal samples, spacing them at the same time distance of 50ms (20Hz), by performing either a down-sampling or an over-sampling of the original signal. Within each *BlockActivityIndex* the time distance between two consecutive samples can be much greater than the 20Hz sampling period, having samples separate by even seconds or minutes. This is due to interruptions in the device data collection caused by external factors. In those cases, and in all cases in which the temporal distance between one sample and the other is greater than a threshold value of 100ms, the samples are considered as belonging to different blocks of a continuous signal, which means associated to a different *BlockActivityIndex*. This is, in fact, a re-indexing of *BlockActivityIndex*.

If we want to add more input signals coming from different device sensors (like accelerometer and gyroscope) we need to manage their asynchronous sampling times. Hence the resampling phase needs to include a resampling that makes the time instants of all considered sensors coincide by performing a synchronization.

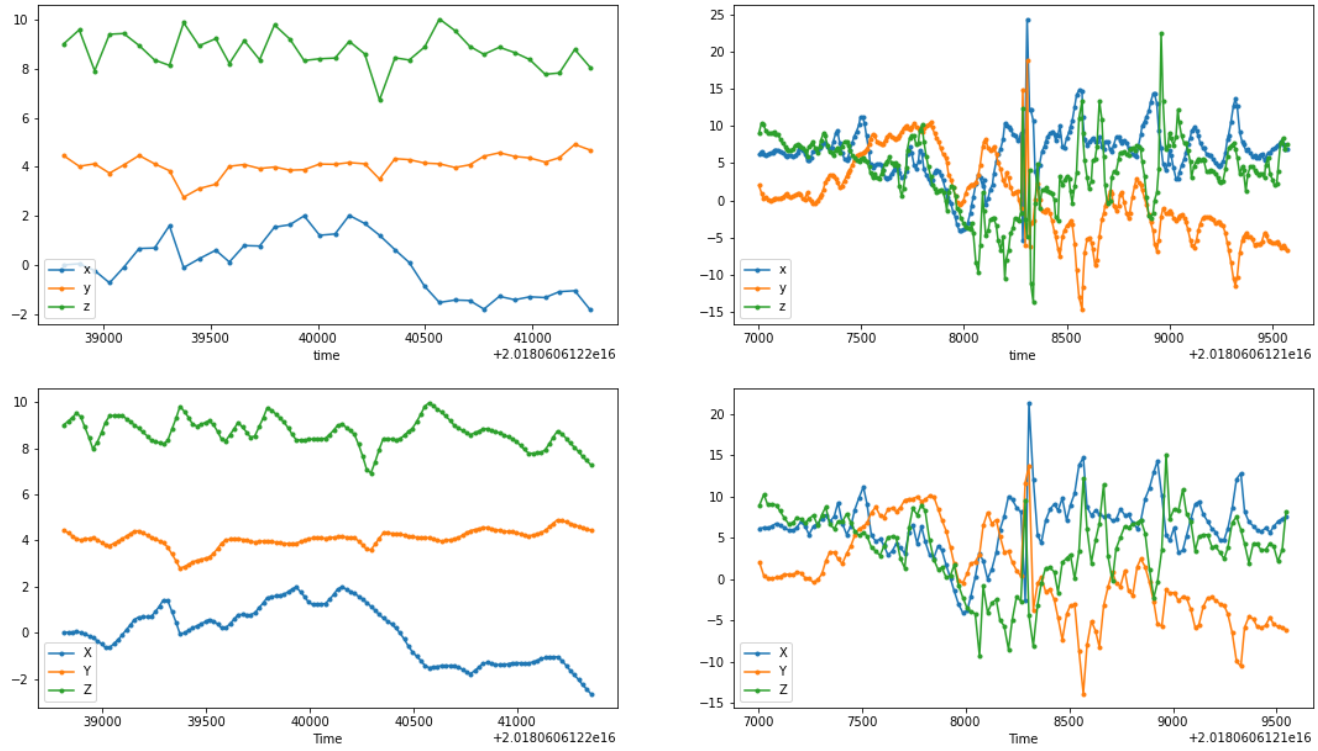


Figure 12

In figure 12, we show how resampling performed over the three axes x, y, z of the acceleration signal. The upper graphs are two different original signals with respectively 70ms and 10ms time sampling. The lower graphs are what we get if we resampled them to the same time sampling of 20ms. On the left we show an over-sampling while on the right we have a down-sampling with loss of information about the high frequencies.

5.2 Output Evaluation

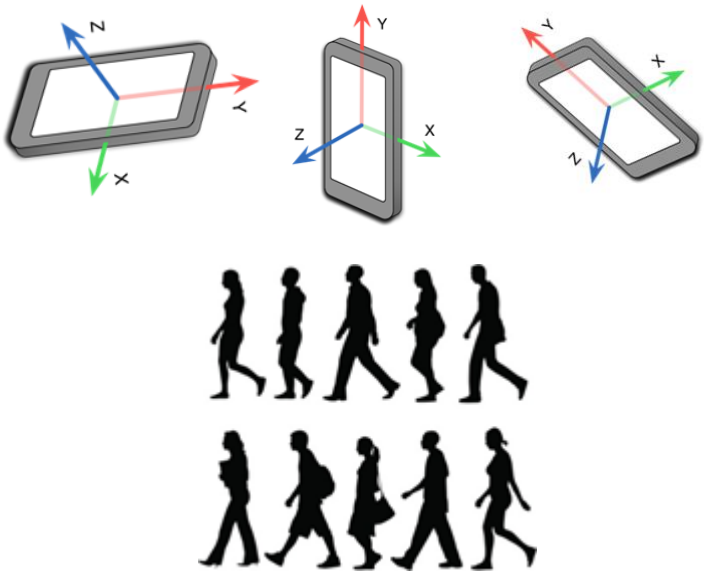
In figure 13 we show the performance of the "mean of transports" classifier inferred from the acceleration signal collected from a respondent's device. The metrics are computed on a balanced test consisting of 100 trials for each class. We show the confusion matrix, the precision-recall and f1 score metrics, and the overall accuracy of 61.6%.

The performance in some classes is very high (f1 score 94% for the motorcycle class); this indicates that the associated signal for this activity is very characteristic (for example because of the presence of high-frequency vibrations). For other classes, such as car, we have a lower f1-score of 51% instead, having the class "car" often classified as "train" by the algorithm. This is since the acceleration profile for the two means of transport is very similar.

Labels	precision	recall	f1-score	support
On Foot	0.63	0.53	0.58	100
Auto	0.49	0.54	0.51	100
Bicycle	0.64	0.32	0.43	100
Moto	0.96	0.92	0.94	100
Train	0.48	0.77	0.59	100

Figure 13

Another aspect that characterizes signals from smartphones is the great variability of the acceleration profile for the same activity, due to the position in which the device is placed. The smartphone in fact can be held in hand, be in the pocket, in the jacket or the backpack. Even when placed in the same place, it can be rotated in different positions. The use of additional information such as the one coming from gyroscope data could help make better predictions. The worse performance compared to the activPAL performance can be explained by the difference in the types of activities we want to classify in the two cases and by the differences in the device position, constant in one case and variable in the other. Further studies to increase the quality of the prediction should be done.



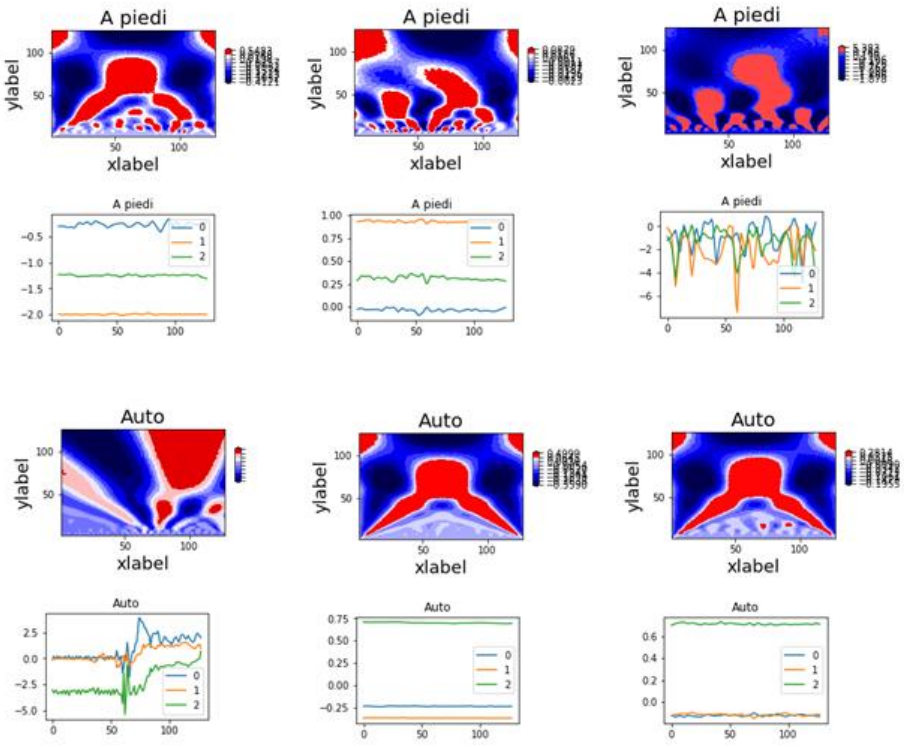


Figure 14

6. Conclusions and Lessons Learned

Typically, in the treatment of sensor data through ML algorithms, a domain-specific pipeline is designed. The collected data are in a first step pre-processed and then used to train a predictive ML model for a given variable of interest that represents a component-specific for that survey. Viceversa in our PoC, we showed how it is possible to design a survey-agnostic ML component, i.e., a component that can process the same types of input data (in our case signal data) to build a single ML component that can be used for different types of surveys.

In our PoC we applied the same pipeline to data collected through different devices (wearables for Health pilot and smartphones for SmartUnitn (Two) surveys) with different results. In fact, while for the activPAL use case, the performance of the physical activity classifier is good (accuracy of 87.6%), for the iLog use case, the performance of the "means of transport" classifier it's not as good (accuracy of 61.6%).

These differences are due to the good quality of the data collected in a controlled environment such as the laboratory for the annotation of the labels, and also to the wearable instrument activPAL that measures the acceleration with a good sampling rate, and a systematic collection of the associated acceleration signal. In SmartUnitn(Two) survey, the performance is low lower. The worse performance of this use case compared to the activPAL can be explained by the difference in the types of activities we want to classify in the two cases and by the differences in the device position, constant in one case and variable in the other.

From our point of view, the major strengths underlying the generalized ML component we proposed are:

- the ability to use signals from different sensors (accelerometer, gyroscope, thermometer) even jointly in a single framework thanks to the use of wavelets and CNN;

- the possibility to change the task of machine learning algorithm from classification to regression by using algorithms such as CNN -
- the modularity, that allows adding other modules in the pipeline to perform new functions;
- the modularity of CNN, allowing to add in its architecture categorical input variables like screen status or flight mode that can help the prediction.

The weaknesses of this component are:

- it does not guarantee the same levels of accuracy of an ad-hoc component;
- the training phase can be expensive in terms of time and computing power.

Although it is an absolutely frontier field explored, the results achieved are very interesting and promising. The flexibility/modularity of the generalized component allows further refinements and improvements: other approaches can be explored taking into account the trade-off between the ability to adapt to a wider range of signals data and the best accuracy; a system of accuracy control mechanisms - for individual learning (real-time) - can be implemented during the training and the inference phase to carry out a new update training.

References

Busso, M., Bison, I., Giunchiglia, F. (2019). Experiment Design SmartUnitn(Two). Report Università di Trento.

Busso, Matteo (2017). SmartUnit Project - i-Log Handbook.

Kuster, R.P., Grooten, W.J.A., Blom, V., Baumgartner, D., Hagströmer, M., Ekblom, Ö. (2020). Is Sitting Always Inactive and Standing Always Active? A Simultaneous Free-Living activPal and ActiGraph Analysis. *Int. J. Environ. Res. Public Health* **2020**, *17*, 8864. https://doi.org/10.3390/ijerph17238864

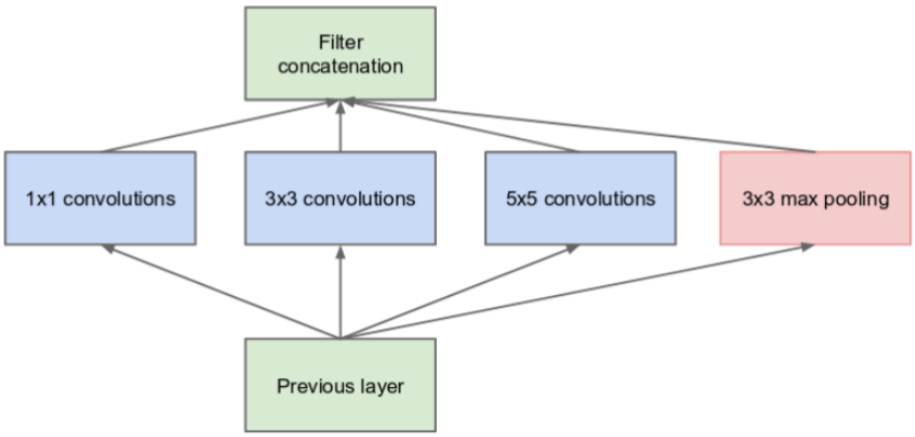
Polikar R. (2006). The Wavelet Tutorial. [Online]. Available: http://users.rowan.edu/~polikar/WTtutorial.html

Rioul, O. and Duhamel, P. (1992). "Fast algorithms for discrete and continuous wavelet transforms, "in IEEE Transactions on Information Theory, vol. 38, no. 2, pp. 569-586, March 1992, doi:10.1109/18.119724. Copy

Sadowsky, J. (2017). Investigation of signal characteristics using the continuous wavelet transform Johns Hopkins APL technical digest, 1996 - jhuapl.edu Neural Computation Volume 29 | Issue 9 | September 2017 p.2352-2449).

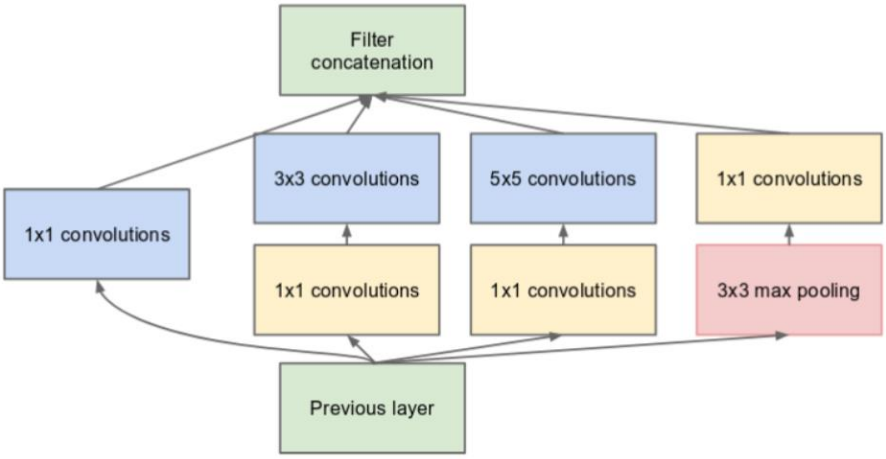
Appendix 1 - The Convolution Neural Network: Inception V3 Model

In our generalized ML pipeline, we have developed an approach Wavelet + CNN. The name of the chosen CNN is **GoogleNet** (designed by Google) or **Inception** V3 and is made of a stack of Convolutional Layers arranged in a very smart way. Inception's aim is to reduce the number of model's free parameters and improve the top-1 accuracy *(Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826).* Before the introduction of Inception Architecture in 2016, the most traditional CNNs just stacked convolution layers deeper and deeper, with the hope of achieving better performance. Unfortunately, it does not work since there is a problem of accuracy saturation in stacking multiple layers of a neural network. Both Microsoft and Google excogitated different strategies to stack more layers: **residual** connections and **inception** layers. The most popular versions of Inception are v1, v2, v3, v4 and Inception-ResNet. The insight of Inception comes from the following fact: features, within each input picture, may have extremely different sizes, which means the **data sparsity** of each image is different. Namely, in an image representing a dog the area occupied by the actual dog can be very different from one image to another. Because of this huge variation of data sparsity for the same object inside different images, selecting the proper kernel size for the convolution is quite difficult. A larger kernel is preferred for information that is distributed more globally, and a smaller kernel is preferred for information that is distributed more locally. Naively stacking large convolution operations is computationally expensive and produces the vanishing gradient problem. The idea behind Inception is the implementation of filters with **multiple sizes** operating on the **same layer.** In the following picture, there is a basic inception module. It performs **convolution** on input, with **3 different sizes of filters** (1x1, 3x3, 5x5). The outputs are **concatenated** and sent to the next inception module. The CNN architecture becomes **wider** other than deeper.



Inception Module.

An issue with GoogleNet, such as any Deep Neural Network (DNN), is that they are computationally very expensive. It is possible to limit the number of input channels by adding an extra 1x1 convolution before the 3x3 and 5x5 convolutions. Even though the choice of extra operations may not appear sensible, 1x1 convolutions are cheaper than 5x5 convolutions. Regarding the max-pooling layer, because of its down-sampling nature, there is not the reduction upside as in the convolutional layers, so the 1x1 convolution is introduced after the max-pooling layer. All this is illustrated in the following picture.



Inception Module with 1x1 convolutions.

By using the 1x1 convolutions, Google built a reduced inception neural network known as GoogLeNet (Inception v1). GoogleNet was made of 9 inception layers.

Appendix 2 - Standardization of data sensors from different smartphone

In the latest years, the interest in human activity recognition (HAR) research has significantly grown, matching the spread of motion sensors on users' personal mobile devices and mobile sensing applications. The accelerometer is one of the most used sensors for this scope, as it allows the recognition of several human activities with good performances and low energy consumption.

However, performance can be significantly affected when HAR is implemented at a large scale in the real world, because of variations across device manufacturers, models, OS types, and CPU load conditions that cause several heterogeneities. Moreover, due to the fast growth in smartphone technology, those heterogeneities can change significantly over years. The most common type of heterogeneities are sensor bias, sampling rate heterogeneity and sampling rate instability, and they are described below.

Sensor Biases (SB): it refers to the sensor internal bias and depend on the sensor quality. Accelerometers differ in precision, resolution and range and low-cost sensors can be poorly calibrated, inaccurate and supporting a limited range. Moreover, biases can be introduced by the mobile final assembly process or by shocks experienced by the device during usage, causing calibration errors.

Sampling Rate Heterogeneity (SRH): the default and supported sampling frequency for sensors vary a lot across different smartphones model. Usually, it varies between 25 Hz to 200 Hz (for common smartphone model released in 2010-1014). Ref.Tab1

Sampling Rate Instability (SRI): it refers to the regularity of the timespan between successive sensor measurements and it depends on the management of the sensor application by the device OS. Hence SRI is a feature specific to a device. Sampling rate stability can be affected by several factors related to the device workload and OS performances, such as multitasking and I/O load on mobile devices, which can result in delays in OS level timestamp attachment to sensor measurements.

Several methods can be implemented in order to try to reduce the impact of heterogeneities on classification activity. Here we report two methods:

- data resampling
- clustering

The resampling method requires adding a data preprocessing step that results in a resampling of the sensor data coming from different devices in order to have equidistantly spaced samples for all devices (i.e., choose a common sampling frequency for all the devices). Down-sampling seems to yield higher gains than up-sampling since it preserves data characteristics while up-sampling can introduce artifacts.

The clustering method requires grouping devices in different clusters according to feature similarities and to train a classifier for each identified cluster. Training for each device model would be too costly and it would not take into account sensor bias that can vary across devices of the same model. It would be useful instead to create device clusters according to median sensor bias and standard deviation of sampling frequency.

Finally, we mention the further source of heterogeneities, that are not device-dependent: they come from device orientation and on-body placement. These variables need to be considered too to evaluate HAR performance.

References:

“Smart Devices are Different: Assessing and Mitigating Mobile Sensing Heterogeneities for Activity Recognition“ A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. Conference Paper. November 2015. DOI:10.1145/2809695.2809718

ESSNET SMART SURVEYS WP3

Task 3.2
Sub-tasks 3.2.3

Application of Privacy Enhancing Technologies to the TSSu platform

Version: 1.0

Prepared by:

Joeri van Etten (CBS)

Sub-task leader: Rob Warmerdam (CBS)

Outline

Executive summary

The main objective of the proof-of-concept described in the following part of the deliverable was to test the applicability of various privacy-enhancing technologies to the Trusted Smart Survey platform, with the purpose of enabling production of statistics without access to respondent level microdata. The first part of this 'test' described was theoretical. It aims to answer the question: "given the supposed possibilities and shortcomings of PET's, how can we design a platform architecture that utilizes these technologies to the best of their abilities?." The second part aims to answer questions regarding the practical feasibility of the proposed architecture components. Given limited resources, we chose to focus on the component regarding statistical analysis using secret sharing based MPC.

The result of the first part is a description of an example platform architecture that allows for the implementation of the PET's required for production of aggregate statistics without compromising respondent privacy. Combined with the other architectural results presented in this deliverable, these results can serve as a basis for the next iteration in development of a platform architecture where all these aspects are combined. Finally, from the architectural part of the PoC, we conclude that for a well-designed and well-defined survey, the architecture required for oblivious data-analysis is not much more complicated than one would expect for a platform without. However, quality control, logging and hence survey maintenance is without compromising privacy is difficult. Also, design and development of surveys is incredibly difficult given the restrictions of privacy enhancing technologies. These difficulties are inherent to the goal of applying these technologies and can hence not be overcome by technological advancement.

In the second part, we showcase the process of adapting existing PET to the statistical use-case. We do this by means of developing a toy model application that enables aggregate statistical analysis using multi-party computation for a general (smart) survey. This showcase should not be interpreted as a demonstration of how to develop an production-level application for privacy preserving statistical analysis. Rather, it presents some very basic ideas and considerations that can be used in the development of such an application. The final conclusions with respect to this part of the PoC is that while it seems possible to adapt existing open-source technology to statistical use-cases typical for the TSSu platform, the process is highly involved. As such, proper implementation embedded within the TSSu platform likely requires experts in the fields of statistics, software engineering and cryptography. Alternatively, one might consider commercial offerings.

1. Introduction

A vital part of the Trusted Smart Survey platform is the relationship of trust between respondents and NSI's. Among other things, establishing this relationship requires the platform to treat respondent data with as much care as possible. What this means is dependent on the current state of technology with regards to protection of data and privacy. Encryption of data *in transit* and *at rest,* governance of data access while data are *in use* and manual statistical disclosure control when output is published used to be the state-of-the-art. However, with the advent of numerous privacy enhancing technologies this is not the case anymore. Very simply put, these technologies either allow for data to remain encrypted while *in use* or for analysis to be performed at the site of data collection, doing away with the need to share data in the first place. In current times, careful treatment of data and privacy to nurture the relation of trust between NSI's and the respondent will hence inevitably entail application of these technologies as well. The following chapter will investigate *how.*

More specifically, to understand the intricacies of applying these new technologies to an actual survey platform, we have divided this proof-of-concept into two parts. The first relates to the development of a

platform architecture that enables the application of the technologies required to achieve the privacy protection mentioned in the previous paragraph. For the second part, the goal was to understand, through experimentation, what it takes to adapt existing technologies to the aggregate statistics use-case of TSSu. Particularly, we chose to investigate how secure multi-party computation techniques can be used to privately compute aggregated statistics given data from two typical smart surveys use-cases: a mobility survey and an indoor air-quality survey based on sensor data. The approach we took was to learn the considerations that go into adapting current PET through the development of a toy model application. This process is described in the second part of this document.

2. Architecture

In the following chapter we discuss the development of a cohesive architecture for the application of PET's to the TSSu platform, to enable the production of aggregate statistics without the need for anyone or any system to have access to private information. For this purpose, we describe a survey in terms of distinct phases. On each of these phases, different restrictions are imposed by privacy considerations. Dealing with these restrictions is achieved by applying the appropriate PET's. These PET's then lead to requirements for the platform architecture for the different phases of a survey. Finally, based on these requirements an architecture is presented that allows for the application of all these technologies along the survey process.

2.1 Phases of (Smart) Survey

To determine what privacy measures should be taken at any given time during a smart survey, it's important to understand the different phases of the survey. Of course, there are different criteria on which the distinction of phases can be based, but for the investigation of the application of PET, distinguishing phases based on the state of the data seems to be the most useful. Then, the first phase of the survey is its design phase. During this phase, data is not collected and processed with the same purpose as during the actual survey. In fact, for the design of a traditional questionnaire-based survey, no data has to be collected from respondents whatsoever. In this regard, a smart survey which is—at least in part—based on data passively collected data from smart devices, is different. Namely, to determine what information can be extracted from this kind of data, the data itself should be investigated. During this phase, use of the data has to be flexible and there should be access to more data than is strictly necessary for production of the final statistic to help understand the data. When machine learning is required for analysis of the data, then we should train these models during the design phase, as the output of the model is required for design of subsequent steps of the survey.

When design of the survey is finished, the next phase is collection of the actual data. The Smart Survey platform should allow for various forms of data collection ranging from questionnaires collected over the phone or filled in on PC's or smartphones, to sensor data collected from smart devices. Barring over-the-phone questionnaires, all these data collection methods have in common that collection can be split up into two phases: (1) transfer from source of information(*e.g.* respondent when filling in questionnaire, physical information collected by sensor) to the respondent's personal device and (2) transfer from the personal device to the NSI('s)/TSSu Platform. Courtesy of this phasing is that privacy matters can be considered separately as well. Namely, during the first phase of data collection, data remains on the personal device of the respondent, so as long as the data is shielded from the outside, there is no need to hide the data. During the second phase of collection, data will be transferred from the personal domain of the respondent to the TSSu platform, where more people will have access. Hence, to preserve privacy of the respondent, additional privacy preserving measures should be applied during this phase.

Having collected the data from individual respondents, aggregation should be performed. A successful aggregation phase depends on using the right aggregation functions, weighting samples properly and pre-processing non-aggregated data such that aggregation produces proper results. For example, if the statistician wants to know the mean for a variable for all respondents, the appropriate aggregation operation is to sum all (weighted) values for this variable and dividing by the total number of samples. However, if no pre-processing has been done, *e.g.* missing values have not been imputed, then the results could be skewed due to data collection problems. Hence, it is important that pre-processing is done with the final aggregation in mind.

2.2 Applicable Technologies

In general, whenever private data sources need to be shared the use of privacy enhancing technologies is warranted. In the case of official statistics, we can be a bit more precise, as the interest lies specifically in aggregated data originating from various sources. Hence, data from a single source is shared for official statistics with the purpose of being combined with other sources. The use of privacy enhancing technologies for official statistics therefore applies to the act of combining private data sources for aggregation. The use of technologies that enable privacy preserved analysis of single data-sources are hence less useful, as they will inevitably require additional technologies for private aggregation. Hence, for the rest of this document we will focus on technologies that allow for privacy preserved aggregation.

Based on this consideration, the most applicable technologies that have been identified for the TSSu platform are:

- *Federated learning* for machine learning tasks, using *multi-party computation* or *homomorphic encryption* for secure aggregation
- Multi-party computation using *TEE* or *Secret sharing*

For more background on these technologies, see appendix A.1

2.3 Impositions on TSSu platform

The application of privacy enhancing technologies puts additional constraints on the architecture of a process that incorporates them. *E.g.* federated learning requires an infrastructure with sufficient computing power where the data is located, use of TEE requires dedicated hardware, secure aggregation of secret shared data prohibits branching operations during analysis and so on.

2.3.1 Infrastructure

Federated Learning

A federated learning infrastructure needs at least a few important components:

- Central coordinating server
- Clients who will perform training on local data
- Networking between central server and clients

In typical use cases, clients are smart devices (*e.g.* smartphones, smartwatches, smart utility meters). The use of such devices in federated learning poses a few additional challenges.

- Varying compute capabilities

For modern smartphones this should not be a problem, however, smart utility meters typically have low compute capabilities(security and privacy in smart grid, page 4)

- Unstable network

A typical problem with devices connected through mobile networks(*e.g.* smartphones, smartwatches by proxy, cars). These devices can enter regions with limited network coverage at any time, which should be taken into account when implementing federated learning on such devices

- Limited data-transfer allowance

Again, this problem pertains to devices connected to mobile networks, which typically have a limited allowance for the transfer of data. When trained models contain lots of parameters, or if many iterations of training are required, this could become a limiting factor for federated learning using such devices.

If these challenges are dealt with appropriately, the components described above suffice to enable distributed learning. However, recall from section 2.2.1 that the updated weights sent from the clients still contains some private information. Hence, we need additional measures to provide complete input privacy.

Possible candidates are technologies like (1) secure multi-party computation and (2) homomorphic encryption. In the case of (1), FL clients secret share weights across multiple servers who jointly aggregate the weights according to an appropriate MPC protocol. For (2), weights are encrypted homomorphically on the client side, such that they can be aggregated by the central server without decrypting. Then, the aggregated encrypted weights are returned to the client, where they can be decrypted.

Secret Sharing

An infrastructure that allows for secret sharing as a means of secure storage and aggregation needs, at least, the following components, as shown in figure :

- Multiple compute servers for joint secure aggregation based on MPC protocols

These perform the actual computation on the respondent data shares. They do so following MPC protocols which, depending on the type of computation, can require many rounds of communication. Hence, a stable, low-latency and high-bandwidth network between them will greatly speed up computations.

- Multiple separate storage servers for distributed storage of shares

After data has been collected and (optionally) pre-processed on respondent devices, shares of these data are distributed across the storage servers. In principle, these databases can be running anywhere, but it makes sense to have these running in the same datacenter/on the same hardware as the computing servers, to minimize communication overhead. When the data is queried, computing servers retrieve their respective shares given they have been authorized by the coordinating server.

- Central coordinating server

When NSI's query secret shared data, this component (1) authorizes the computing servers to access the required shares and (2) instructs them on which functions to perform. Then, shares of the computation result can be sent to the coordinating server to be reconstructed and sent to the NSI. Second, when an NSI queries aggregate results, the coordinating server instructs MPC servers to gather their respective shares of the input data.

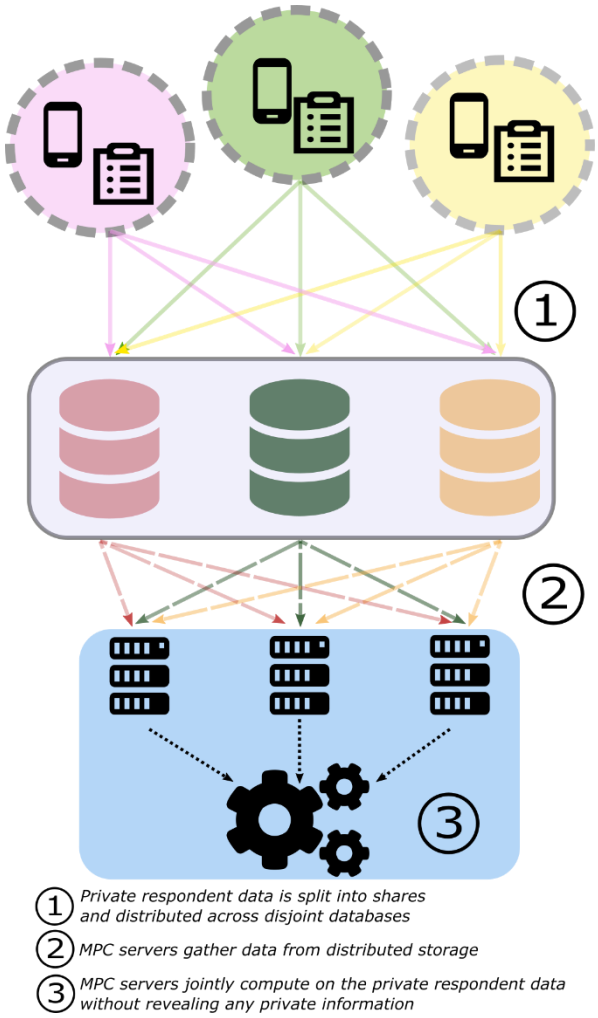


Figure 1. Actions performed by secret sharing architecture components

Secret sharing as a secure aggregation strategy requires a distributed computing environment with networking between the two. As complex analysis requires a lot of communication between compute servers, reliable networking is paramount for good performance. For storage of shared secrets, storage across distributed databases is required as well. It makes sense to have these databases running on the same hardware, or at least in the same datacenter as the compute servers to minimize communication overhead. However, in principle these are separate entities, so they can run in different places if that is required for different reasons.

Trusted Execution Environment

Use of TEE's for secure aggregation requires either (1) supporting hardware like Intel SGX chips, or (2) an infrastructure that allows for software-based TEE, as seen on AWS Nitro. Additionally, infrastructure is required for storing private respondent data in accordance with the distributed trust paradigm. For data only to be used in TEE's, secret sharing of the actual data is not necessary, but secret sharing of encryption keys is. A requirement is hence an distributed key storage facility.

2.3.2 Survey Design

In general, the higher the level of granularity required during data analysis, the more difficult it is to do this in a privacy-preserving way. For example, it is quite simple to implement a multi-party computation protocol to compute the mean for a variable given a collection of private datasets containing information about that

variable. On the other hand, it would be very difficult for app developers to debug a travel data collection app without being able to see the actual data that produces the error.

Before we can investigate how PET's relate to survey design, let us first distinguish a few different stages of the survey design process. Typical survey design starts with the question: "what data do we need to answer question x?" However, when considering big data sources, survey design could also start with the question; "what questions could we answer given dataset X?".

In the first case, a survey is designed specifically to obtain the data necessary to answer the question x. For example, in a traditional questionnaire, it is possible to ask precisely those questions that when answered truthfully will provide the information necessary.

Considering the second case of statistics based on big data and other sources that have not been collected specifically for the purpose of statistics production, the situation is quite different. Design of 'surveys' based on such data has to begin with studying the data-source to learn about the peculiarities of the data, what information can be extracted from them and how to do so. This process requires considerably more granular data access. That is, it might not suffice to only have access to aggregate values.

After exploration, models have to be constructed to extract the information necessary for outputting relevant aggregate statistics. These models can be rule-based algorithms, or probabilistic models like neural networks. Constructing the former requires a proper understanding of the data, its information content and thorough testing. Without access to the microdata, in some form or another, it might be very difficult to achieve this. The latter, however, only requires sufficient data for training the models to make classifications that will allow for producing relevant statistics. This requirement can be fulfilled in a privacy-friendly way using federated learning.

For now, we hence propose that survey design of the big data-based surveys be carried out in two phases. The first of which is a small-scale pilot to collect microdata necessary for exploratory data analysis, determining pre-processing requirements and constructing aforementioned rule-based models. Next, a larger scale phase can start, which is a hybrid smart survey, collecting both diary/questionnaire based data, as well as smart data for the purpose of training required ML models in a federated way.

2.3.3 Statistical Analysis

In official statistics, we are not interested in results at the level of individual respondents, but rather at the population level. These population level statistics can in principle be privacy-friendly (though one has to take care that aggregated datasets don't reveal anything about individuals, or groups of individuals), so if the input privacy is taken care of and disclosure control is sound, release of private information through official statistics should be minimal.

Naively, one could propose anonymization—removing all personally identifiable information—as a means of realizing input privacy. After all, we are only interested in aggregated results, which should not contain or depend on personal identifiers. However, to properly aggregate samples in a way that accurately represents the entire population, individual samples are subjected to a weighting model. These models depend on information about the samples that might be external to the actual data.

Hence, to allow for weighting using external data, samples should be identifiable. Typically, NSI's would thus pseudonymize the data, *i.e.* personal identifiers as social security numbers are replaced by random identifiers

which can be used to join data about the respondent with data relevant to the weighting model. To protect privacy of the samples, we thus need other input privacy preserving measures as SMPC or TEE.

Thankfully, it is reasonable to expect that the desired outputs of the statistical analysis phase could be well-defined in advance of conducting the survey. That is, surveys can be designed in such a way that the input to the statistical analysis phase is as low-dimensional as possible and the format of the data is as convenient as possible given the aggregate functions to be performed.

2.3.4 Example Architecture

Having discussed all the phases of a survey, applicable technologies and the various impositions put upon the platform by them, we can start looking at an example architecture that allows for the application of all these technologies. For context, we will use a time-use survey as an example use-case (which will return as an example in the next section). Typically, these surveys are done by asking respondents to keep track of their activities in a diary. Such surveys are extremely time-consuming for respondents, as well as very privacy sensitive and hence a prime candidate for the TSS platform, which will leverage both new technologies to aid in data collection and PET to improve privacy.

Remembering the phases of a survey, let's now discuss an architecture(shown in Figure 2) that suits the tasks to be performed in these phases in a privacy protected way. The first phase is the survey design phase, however, it turns out to be more convenient to save this one for the end. Hence, we'll start with data collection.

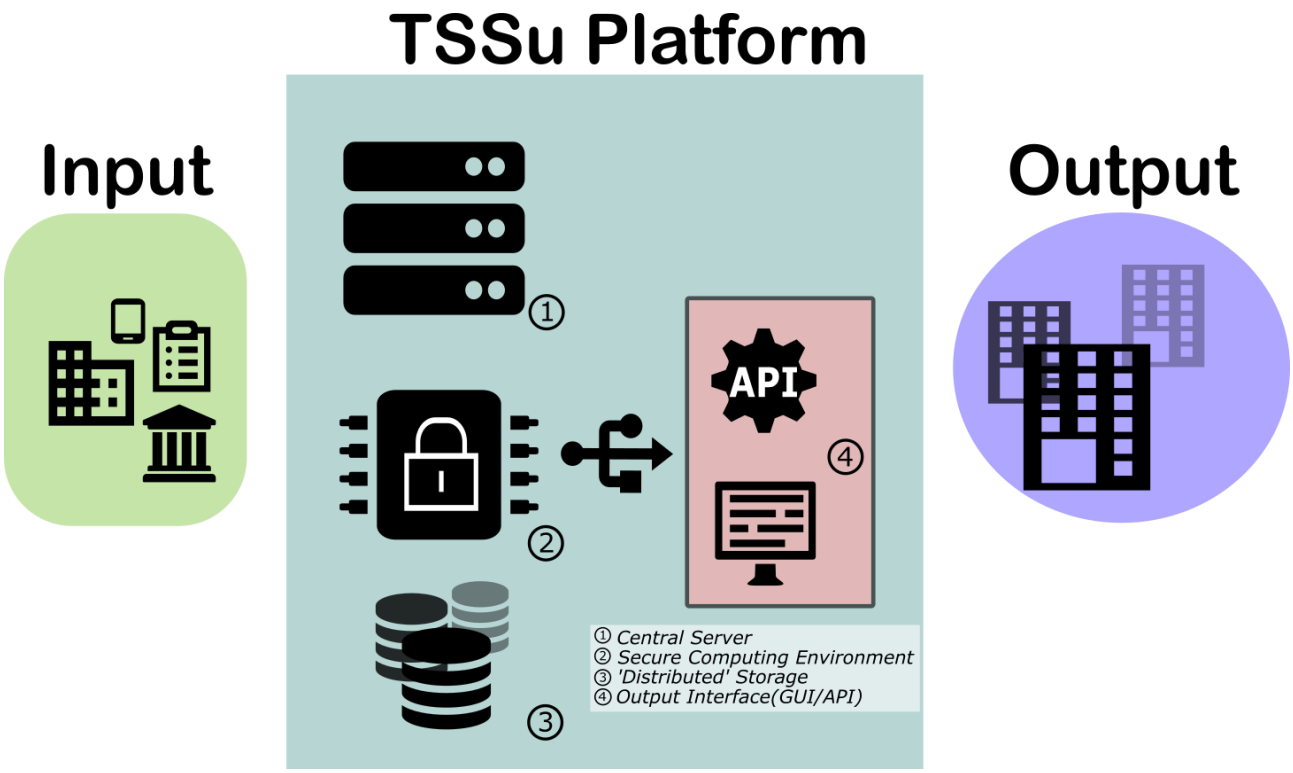


Figure 2. Platform architecture for application of PET

Data Collection

In a questionnaire-based version of the survey, this step is simply a matter of respondents filling out the questionnaire and sending the results to the NSI. In a 'smart' version of the survey though, this step is slightly more involved. In this case, we should—if possible—leverage passively collected data from the respondent's

smartphone to either replace the diary, augment it, or aid the respondent in completing the it. For example, a combination of accelerometer and GPS data could be used to determine that a respondent is riding a bike, and where the respondent is travelling to. If these classifications can be made with very high accuracy, they can be used directly as survey input. If the accuracy is not high enough, such classifications can still be used to provide a pre-filled diary to the respondent, which they will later verify. In both cases, the use of the respondent's device can reduce the burden placed on the respondent.

The methods used for classification based on the passive data collected by smart devices will often require machine learning as these general-purpose data are not specific enough to do direct rule-based/algorithmic classification. Hence, machine learning should be used to discover patterns in these data that allow making the classifications of interest. Training these models requires labelled data, preferably as much as possible. One way to obtain these data is to use the TSSu platform: *e.g.* respondents install an application on their smartphone and allow it to access their smartphones sensor This application further asks the respondent to fill the traditional diary. Combined with the sensor data, this provides a labelled dataset.

Of course, with this dataset come the usual privacy considerations. Hence, PET should be applied in the use of these data. For the training of ML models using the local labelled datasets distributed across the respondent's devices, federated learning is the technique of choice. Using this approach, as handled by the application on the respondent devices, the TSSu platform's coordinating server would send the configuration of a ML model to these devices. Subsequently, the local model instances are trained and the updates to the model are returned to the central server. Recall from appendix A.1.2 that secure aggregation requires either a homomorphic encryption or multi-party computation solution. Since we are dealing with smartphones, limited data transfer allowance is a serious concern. Hence, MPC is probably a better option, given the availability of a distributed computing environment. Using this option, most of the data transfer and computation overhead is transferred to the TSSu platform.

Pre-processing

The next phase of the smart-survey is pre-processing. The amount of pre-processing required is highly dependent on the way the data is collected. Namely, in the case of a questionnaire based survey, very minimal pre-processing should be required. However, in a smart survey based on passively collected sensor-data, high pre-processing requirements should be expected. Among these requirements we have, for example, classification using the previously trained ML models. In general, we consider pre-processing any processing that is required to get the data in a format that serves as direct input to the final aggregation functions. As mentioned in section 2.1, this means that no privacy enhancing technologies have to be applied during this phase. However, proper implementation is vital to successful application of PET during the aggregation phase, as the format of the data and necessary operations determine the feasibility of application of PET. Hence, we can consider pre-processing a vital component in a privacy preserving architecture.

Sharing and Storage

As shown in Figure 3, having pre-processed the data on the respondent's devices, the data has to be shared with the platform. From this point on, data should remain hidden and only after sufficient aggregation of all (or a sufficiently large subset of) the respondent data can the results be revealed. In fact, for the process of sharing and storage this is quite often common practice already. Namely, hiding the data during these stages

is simply a matter of applying plain encryption. New in our case is that data should be 'encrypted' in a way that allows for subsequent aggregation, without prior decryption.

If secret sharing based MPC is the secure aggregation method of choice, this means that a respondent's dataset D should leave his or her device in the form $[D] = (D_1, \dots, D_n)$, where D_i denote shares of the dataset to be shared with n parties. The parties in this case are a distributed storage facility provided by the TSSu platform, comprised of n disjoint servers. These servers store the shares of the respondent data, to be used in aggregation at a later stage.

In terms of data storage and transfer, the use of this approach provides the same security as regular encryption does. However, distributed storage has the added benefit of removing a single location of trust (*e.g.* a single NSI). Hence, for an attack on the data to be successful, multiple servers have to be compromised.

If TEE's are chosen for secure aggregation, it is not strictly necessary to secret share respondent data. However, considering the added benefit of removing a single point of trust during transfer and storage, it might be reasonable to implement another means to remove this single point of trust. For example, a possibility is to secret share the decryption keys. Then, these shares can be put together inside the TEE to decrypt the data prior to aggregation. Effectively, this entails that the data can only be used when a sufficient number of parties agree on this. Of course, there is still a single point of trust during the aggregation inside the TEE, which is inherent to the use of this technology.

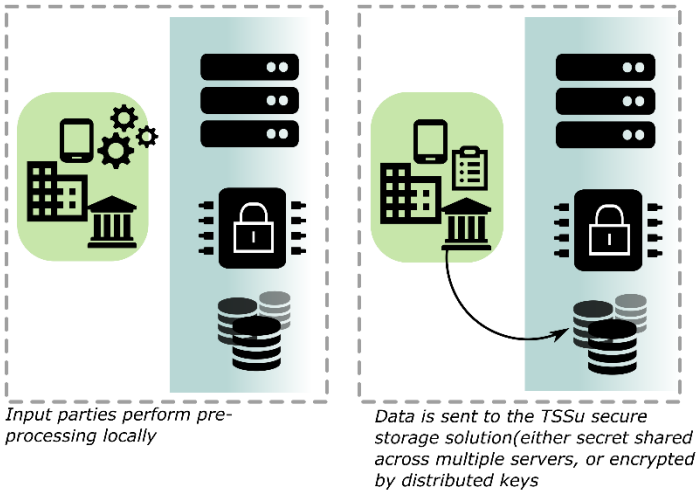
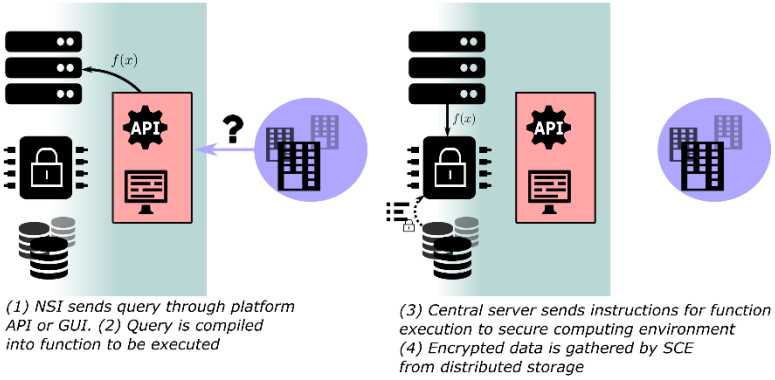


Figure 3. Pre-processing and sharing of data

Statistical Analysis

Once the data has been properly shared with the platform and stored in a way described above, it is ready to be used in aggregation. That is, NSI's should now be able to query aggregate functions of the data to output statistical results. However, use of the data is still bound by authorization. Namely, when respondents agree to share their data, they do so under certain terms and conditions. For example, in some surveys, respondents might only authorize their country's NSI to use the data, while in others, they could agree for their data to be used by all European NSI's connected to the platform. Proper authentication of the querying NSI and authorization to use the data should hence be the first step when the data is used.



When the querier has been authorized to use the data, they will able to view descriptions of the data in terms of metadata, view descriptive statistics (*e.g.* mean, variance, etc) and perform aggregate functions. These aggregate functions often require combination with external data sources for weighting. Remember, this is precisely the reason why can't just anonymize respondent data; we need to be able to couple external data on a record by record basis. The possibility of combining external data with the private respondent data is hence a required functional component of the architecture.

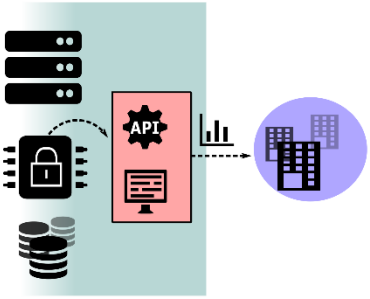


Figure 4. Statistical analysis

To perform the aggregate functions and view the results, four actions need to be performed:

- Verify that the function provides sufficient level of aggregation
- All required inputs need to be brought to the secure aggregation environment of choice.
- The actual function needs to be run against the data
- The results need to be reconstructed/decrypted and returned to the queries

A graphical representation of these steps, as well as the required components, is shown in Figure 4. The first of these actions might need a little more clarification. Remember that we consider the outputs of aggregated functions not to be privacy sensitive, as they are not relatable to a single individual. However, for this to be true, a sufficient number of records need to be aggregated. Hence, if we construct a query that returns an aggregate based on only a few records, high probability inferences can be made about the individuals that make up the aggregate. There should henceforth be a verification step as part of the execution of the aggregate function which checks that the output is sufficiently aggregated.

Survey Development

The final phase to be considered is the development phase of the survey. As we discussed in section 2.3.2, this is the most complicated phase with regards to the application of PET as development requires granular access to the data. The proposed solution was to first conduct small pilot surveys with respondents who consent to the use of their data for development of surveys without the use of PET (of course, their data will still be treated as diligently as possible). In terms of the PET architecture, the initial survey development phase is hence not that interesting.

An exception is when the use of machine learning is required for constructing models for classification based on passively collected data. Training such models is part of the development phase, but should be considered separately as much more data is required and federated learning can enable privacy-friendly training of these models. For performing federated learning within the platform architecture as shown in Figure 5, no extra components are necessary per se.

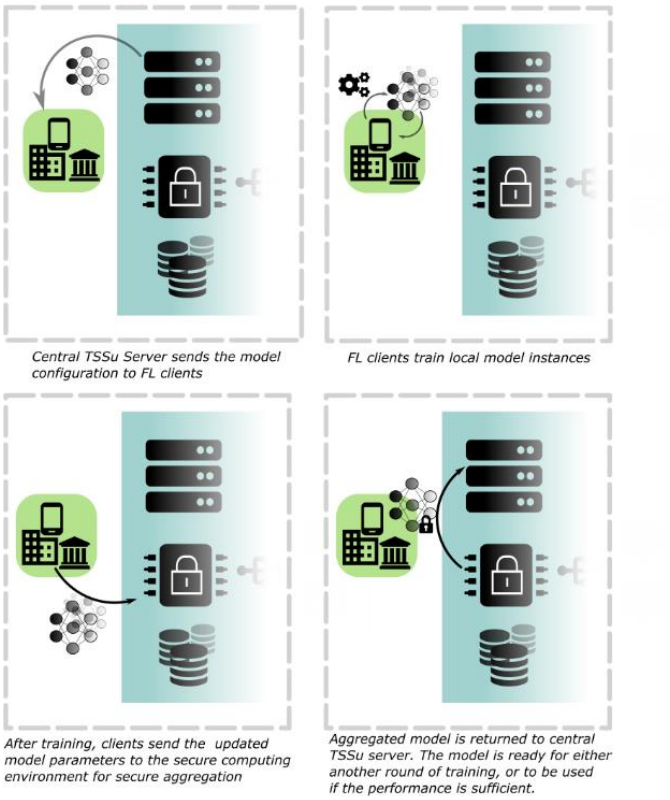


Figure 5. Federated Learning on TSSu platform

3 Aggregate Statistics using MPC

To examine the feasibility of doing aggregate statistics with secret shared data, a very basic prototype of a statistical analysis tool was built using secret sharing functionality provided by *MPYC* (1), a python package that enables *m*-party computation based on the BGW protocol, *i.e.* Shamir secret sharing with protocols for addition and multiplication. However, the results should be mostly independent of the type of secret sharing used, as well as the specific implementation thereof. In fact, initially we used another python secret sharing implementation called *SyMPC* (2) as a secret sharing backend. However, due to a few limitations introduced by the version of SyMPC available at the time of writing, the application was ported to MPYC. Further, it should be stressed that this PoC is not meant as an evaluation of the MPC capabilities of either MPYC or SyMPC, but to investigate the functional feasibility of statistical analysis on secret shared data. The repository for the toy model can be found on (3).

3.1 Example use-cases

Two specific use-cases were chosen to inform which functionalities are essential in a tool for aggregate statistical analysis, based on their relevance to the TSSu platform. The first was the analysis of air-quality data

collected by CBS, for the TSSu pilot done in the context of WP2. The data consists of measurements of various air-quality indicators (*e.g.* CO_2, NO, humidity) taken at fixed time intervals, taken by the uHoo air quality sensor (4). A sample of the data is shown in table 1. Note that these data are very privacy sensitive as changes in air quality parameters could, for example, reveal a person's sleep schedule. For the prototype development, a month's worth of data from 30 respondents was used. However, not all datasets contained actual data, because of non-functioning uHoo devices.

Table 1: uHoo data sample

	Time	Space	CO2	Temp-erature	Humidity	Sound-Level	Particulate Matter	VOC	NO2	Ozon	CO
0	1/6/2021 0:00		452	24.6	34.5		4	366	25.1	7.6	0
1	1/6/2021 0:10		441	24.5	34.6		10	353	25.9	7.7	0
2	1/6/2021 0:20		425	24.6	34.5		6	353	26.3	7.7	0
3	1/6/2021 0:30		412	24.4	35		4	336	25.8	7.7	0

Since the air quality data is a new data-source, no well-defined statistical analysis exist yet. We hence chose to attempt some general aggregate functions on the data as sum, mean, (co)variance on the data and conditional subsets thereof(*e.g.* find mean concentration CO_2 on days with average temperature above 18^o C).

The second use-case was a typical mobility survey used to describe population level mobility patterns. These surveys typically collect data in the form labelled *journeys,* which are in general represented by a start location, end location, distance and mode of transport. These data would traditionally collected by questionnaire-based methods, however, a smart version using GPS and other sensors would yield the same data as an intermediate step. Namely, GPS coordinates would algorithmically be transformed into starts and stops, distances would be calculated and mode of transport would be inferred using machine learning. Note that this is in line with the architectural model of local pre-processing before privacy enhanced aggregation.

We chose to generate data because of privacy considerations involved in using actual survey data. Since the statistical characteristics of the data should not matter for our purpose, data was simply generated by random sampling within some parameters. A sample of a generated dataset for a single respondent is shown inTable 2. Note that based on the respondent identifier, these data could be enriched by combining with other sources.

Table 2: Sample of synthesized mobility data

ID	Start	End	Distance	MOT
1	Castricum	Groningen	147.3055	Bicycle
2	Zwartewaterland	Bunschoten	62.47778	Boat
3	Middelburg	Oosterhout	86.92164	Car

Typical queries on these data involve aggregate functions on subsets of the data where certain conditions are met, *e.g.* find the most used mode of transport for journeys from Amsterdam to Rotterdam. It turns out that such queries are a bit more involved when working with secret shared data. Namely, selecting certain values based on conditions is a type of branching operation. Such operations are prohibited at the respondent level, as they require reconstruction. We will discuss this issue in more detail in the next section.

3.2 Technology and Algorithms used

To enable private computation on the data from the use-cases described in the previous section, the multi-party computation protocols based on Shamir secret sharing were used. Implementations of these protocols were taken from the Python library MPYC. Since the goal of the research was to test the feasibility of user-friendly analysis of aggregate statistics, we did not include any infrastructural components into the scope. Hence, we did not use an actual MPC environment for development, but rather emulated one locally.

Using MPYC as a secret sharing backend for the toy model application, we gain access to an overloaded versions of basic python functions for our secret shared data . Of course, not all python functions have been overloaded with an MPC version. However, MPYC does support all functions that were required during this PoC. For the full suite of functions provided by MPYC, see (5)

Note that in the mock application described in the next section, we have implemented some functions that are provided by MPYC ourselves. The reason for this is to allow for some additional requirements imposed by the statistical analysis use-cases. The need for this leads to quite an important conclusion drawn from the exercise presented here. Namely, given the specific requirements of aggregate statistics use-cases, implementation of functions provided by MPC packages might need to be adapted to fit these requirements.

Sharing Categorical Data

Since MPYC is based on Shamir secret sharing, where—as explained in appendix A.1.3 — secrets are zeroes of polynomials over a finite field $\mathbb{F}$, typically $\mathbb{Z}_{2^k}$; the set of integers modulo 2^k. Of course, categorical data does not generally satisfy this requirement. Luckily, this is achieved simply by assigning a unique element from $\mathbb{F}$ to every possible category. If were sharing over $\mathbb{Z}_{2^k}$, we could then simply use integers up 2^k for this purpose. In fact, this is already common data-storage practice as integers take much less space than arbitrary length strings.

Encoding categorical variables as integers allows the full suite of private operations to be applied, however, most of these don't make any sense. In fact, the only primitive operator that has value is the equality operator, which we can use to select certain rows based on their values as will be described below.

Selecting from private columns

Selection operations play a vital role in data analysis, since they allow for computations on subsets of the data satisfying certain conditions. For example, we have collected a list of records with two columns, *age* and *income*. If we want to compute the mean income for the group with *age > 25,* we select the records that satisfy this condition and compute the mean income for this subset of records. In an MPC setting, this is not quite that simple. Namely, technically speaking the above example involves a branching operation which are forbidden in private computation. See, selecting a record based on the condition *age > 25* requires us to know the value of *age*, which might be private.

However, as discussed in appendix A.1.1, it is possible to do bitwise operations on our shared secrets. Hence, we have access to the usual selection of comparison operators ($=, <, >, \leq, \geq$). Statements involving these operators will return 1 when true and 0 when false. If, however, we apply these to a shared secret, the result (either 0 or 1) will be a shared secret as well. This means we can do comparisons privately. Now, say we have a secret shared column $[X] = ([x_1], [x_2], \dots, [x_n])$, where square braces denote a sharing, *i.e.* $[X]$ denotes the element-wise sharing of array X. We want to compute *e.g.* the mean for all values x_i with $x_i > a$ without revealing them. What we can then do is create a Boolean array $[m] = ([m_0], \dots, [m_n])$ with $m_i = 1$ if $x_i > a$ and $\mathrm{m_i} = 0$ otherwise. Then, if we perform elementwise multiplication (Hadamard product) of $[m]$ and $[X]$, we obtain $[X'] = ([x'_1], [x'_2], \dots [x'_n])$ with $x'_i = x_i$ if $x_i > a$ and $x'_i = 0$ otherwise. Hence, we now have an array containing the secret values we wanted to select, padded with 0's for the values that didn't satisfy our condition. To compute the sum of this array, we simply compute $\sum_i [x'_i]$. However, to obtain the mean, we should not divide by the number of elements in the array, but rather the number of non-zero entries. Quick inspection shows that this can conveniently be obtained privately by computing $\sum_i [m_i]$. It turns out that this method works for all functions that we will consider in this document.

Hiding Input Size

In MPYC, the size of the input array representing a secret shared column $[X] = ([x_1], [x_2], \dots, [x_n])$ is considered a public value. One of the reasons for this is that parties need to be able to iterate over elements of the array, which requires them to know the number of elements. In some cases, this might not be a problem. For example, for the air-quality use-case, we are dealing with time-series data. The size of the individual respondent datasets is hence not related to private information belonging to the respondent, but to a survey-wide sampling frequency for the air-quality measuring device. However, in the mobility survey use-case, the size of the respondent dataset equates to the number of journeys made by the respondent. This is private information and needs to be hidden.

A (partial) solution is to use the masking construction described above. That is, prior to sharing, we pad X with random values to construct $\hat{X} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_{n+k})$ along with a mask $M = (m_1, m_2, \dots, m_{n+k})$ for a integer $k \in \mathbb{Z}^{0+}$and $m_i = 1$ for $i \leq n$ and $m_i = 0$ for $i > n$. Choosing a fixed value for $n + k$, the servers will only learn from the sharing of $\hat{X}$ that the size of X is between 0 and $n + k$. This is a lot better than before, but the server still learns without a doubt that the size of X is smaller than $n + k$, which violates the concept of perfect privacy. However, since a maximum input size will be revealed when we analyze counts of the aggregated data anyway, we will consider the above acceptable.

Note that we have to pick $n + k$ sufficiently large to accommodate for all possible values of n as we require ≥ 0 . For datasets with a very large possible values of n, this can introduce considerable overhead, especially for operations that are not vectorized. Another possibility is to sample k randomly. However, considering for example the case $n = 0$ and random sampling of k sets $k = 0$, the server learns without a doubt that $n = 0$ as $k \geq 0$. This is unacceptable from a privacy perspective. Hence, we don't use this option and only consider the 'brute-force' solution from the previous paragraph.

Aggregate Functions

General

Consider again a private input represented by an array $X = (x_1, x_2, \dots x_n)$. We hide the input size as described above, hence we have a padded array $[\hat{X}]$ with $\hat{X} \in \mathbb{Z}_p^{n+k}$ and a mask $[M]$ with $M \in \mathbb{Z}_2^{n+k}$. The most basic

aggregate function we can apply is the `count()` function. This should return the number of records in a column, which in our case is obtained by simply computing the sum of all elements in $[M]$, *i.e.*

$$[count(X)] = \sum_i [m_i] \equiv [n_X]$$

where once more, we use notation $[s]$ to denote a sharing of s.

Continuous Variables

Given that all the basic arithmetic operators are supported, computation of common aggregate functions for continuous variables like mean, variance, covariance is a straightforward endeavor in principle. Here we'll review the functions that have been implemented in the toy model application, along with some additional remarks.

For a the padded array $[\hat{X}]$ described above, we compute the mean as

$$[\mu_X] = \frac{1}{[n_X]} \sum_i [m_i][\hat{x}_i]$$

Note that division is a relatively costly operation, so when optimizing for performance it might be a good idea to rewrite equations such that the number of required division operations is reduced (if possible).

Private sample variance of X is computed as

$$[var(X)] = \frac{1}{[n_X] - 1} \sum_{i=1}^{n} [m_i]([\hat{x}_i] - [\mu_X])^2$$

Private standard deviation $[\sigma_X]$ is just the square root of the variance. However, computation of inverse powers is quite costly to do privately. Hence, if the standard deviation is the final result we are after, we should reconstruct the private variance and take square root publicly.

Given another padded array $[\hat{Y}] = ([\hat{y}_1], [\hat{y}_2], \dots, [\hat{y}_{n+k}])$ representing a column from the same dataset as X (which thus uses the same mask M), we compute the private covariance as

$$[cov(X,Y)] = \frac{1}{[n_X] - 1} \sum_{i=1}^{n} [m_i]([x_i] - [\mu_X])([y_1] - [\mu_Y])$$

Finally, we consider the `min()`, `max()` and `range()` functions. Given the existence of comparison operators, finding these is a matter of sorting the input array. However, note that these functions do not involve a sum over all elements of the input. Hence, our padding construction for hiding the input size complicates matters. Namely, if we pad with random values, the minima and maxima might be one of these values. A possibility is to restrict the range of these values $\hat{x}_i$ to $min(X) < \hat{x}_i < max(X)$, such that the minima and maxima remain unchanged. This can be done before the input is secret shared. However, a more convenient solution is to just secret share the minimum and maximum separately, in addition to the full array X. Then, if we want to privately compute the maximum of a collection of arrays $X_1, X_2, \dots X_m$ we simply

construct $A = ([max(X_1)], [max(X_2)], \dots, [max(X_m)])$, sort ascendingly and return the element at the m-th index.

Weighting

As should be clear by now, an essential part of statistics is weighting of aggregate functions. Weighting is necessary because we would like to use a subset of a population *(read:* survey respondents) to make inferences about the entire population. Of course, some types of people are more likely to respond to surveys than others, hence the need for a corrective factor. In practice, this means we relate to a set of data $X = x_1, x_2, \dots, x_n$ a set of weights $W = \{w_1, w_2, \dots, w_n\}$, which can be used to increase or decrease the contribution of any of the x_i to the output of an aggregate function. For example, we can compute the weighted mean as

$$\mu_X = \frac{\sum_i w_i x_i}{\sum_i w_i}$$

From an MPC perspective, this is a simple operation if we can relate the right weight w_i to its corresponding datapoint x_i. Of course, if we are dealing with a padded array $[\hat{X}]$ and a mask $[M]$, the equation for the (secret shared) mean becomes

$$[\mu_X] = \sum_i \frac{w_i[m_i][\hat{x_i}]}{w_i[m_i]}$$

For now, we only considered the weighted mean. One can of course also consider weighted versions of more complicated functions (*e.g.* variance of the weighted mean). However, from an MPC perspective these functions are not very different, so we leave

Aggregation

For the description of the functions in this section we considered a general input array $X = (x_1, \dots, x_n)$. This could be a column in a dataset for a single respondent. However, aggregate functions computed on single respondent datasets are not very useful (we are not even allowed to reconstruct these from a privacy perspective).

Rather, we are interested in aggregated results for the entire sample of respondents in a survey. As such, let $X_1, \dots X_p$ be a collection of arrays describing data collected from p respondents about variable X. We simply append these arrays to obtain $X_{p+1} = \frown_{i=1} X_i^p$, where we use notation $\mathrm{X} \frown \mathrm{Y} = (\mathrm{x}_1, \dots \mathrm{x}_\mathrm{n}, \mathrm{y}_1, \dots \mathrm{y}_\mathrm{n})$. All of the functions described above generalize trivially to X_{p+1} and hence allow us to compute on an aggregate of datasets.

3.3 Application Design

For our toy model application we are using *MPYC* (1) as the secret sharing back-end. MPYC is written in *Python*, so it made sense to choose this for the application as well. For Python, the most popular data analysis library is Pandas (6), hence, we chose to model the application after the Pandas API. That is, data is stored in dataframes, which are data-structures for storing tabular data, containing labelled rows and columns. In our

case these contain private data which is secret shared. We call the resulting `DataFrame` analog a `PrivateDataFrame`. This object class provides only limited methods for analysis as the object contains mostly private data. Rather, to use the data one first needs to construct an `Aggregate` out of various `PrivateDataFrame's`. This object provides methods for computing aggregate functions on the whole aggregate dataset or parts thereof.

The workflow for using the application is as follows:

- Data is pre-processed to ensure similar formats across all datasets and stored in `pandas.DataFrame's`
- Respondent datasets are turned in `PrivateDataFrame` *(i.e.* columns are turned into tensors which are secret shared)
- (Optionally) Based on external data, weights are added to the `PrivateDataFrame` to enable weighting aggregate functions.
- Data analyst creates `Aggregate` object out of various `PrivateDataFrame`'s
- Data analyst performs aggregate functions provided by `Aggregate` object on the private data.
- Aggregate results are reconstructed if sufficiently aggregated.

3.4 Demo

To demonstrate the user experience of a data scientist/statistical analyst using the secret shared data through the TSSu platform, as well as some demonstrations of the considerations presented in the previous section, we will now look at some examples of the toy model application in action. Starting with the air quality use-case, we start by loading the data into `pd.DataFrame's` and take a look

```
>>>import pandas as pd
>>>import os
>>>datasets = [pd.read_csv(os.path.join("uHoo_juni", file), sep=r"\t")
               for file in os.listdir("uHoo_juni")]

>>>datasets[0].head()
```

	Tijd	Ruimte	CO2	Temperatuur	Luchtvochtigheid	Geluidsniveau	Fijnstof	VOC	NO2	Ozon	CO
0	1-06-21 00:00	CBS1	452	24.6	34.5	NaN	4	366	25.1	7.6	0.0
1	1-06-21 00:10	CBS1	441	24.5	34.6	NaN	10	353	25.9	7.7	0.0
2	1-06-21 00:20	CBS1	425	24.6	34.5	NaN	6	353	26.3	7.7	0.0
3	1-06-21 00:30	CBS1	412	24.4	35.0	NaN	4	336	25.8	7.7	0.0
4	1-06-21 00:40	CBS1	405	24.4	34.9	NaN	4	324	25.8	7.7	0.0

It turns out that some respondent datasets are empty, so let's remove those

```
>>>non_empty = []
>>>for df in datasets:
       if any([df[key].empty for key in df.keys()]):
           pass
       else:
           non_empty.append(df)
```

If we wouldn't care about privacy we could combine the datasets to construct an aggregate and perform some aggregate statistics as follows

```
>>>aggregate = pd.concat(non_empty)
>>>aggregate.describe()
```

	CO2	Temperatuur	Luchtvochtigheid	Geluidsniveau	Fijnstof	VOC	NO2	Ozon	CO
count	77835.000000	77835.000000	77835.000000	0.0	77835.000000	77835.000000	77835.000000	77835.000000	77835.0
mean	656.869108	23.347389	49.807450	NaN	5.135916	390.929672	5.071116	5.168805	0.0
std	413.905487	2.177549	7.906439	NaN	11.083555	424.179799	16.985880	2.056434	0.0
min	400.000000	14.400000	22.300000	NaN	1.000000	0.000000	0.000000	2.300000	0.0
25%	449.000000	22.000000	44.600000	NaN	2.000000	66.000000	0.300000	3.800000	0.0
50%	518.000000	23.400000	49.700000	NaN	4.000000	190.000000	0.600000	4.700000	0.0
75%	670.000000	24.700000	54.600000	NaN	6.000000	610.000000	0.900000	5.700000	0.0
max	5579.000000	34.600000	80.200000	NaN	200.000000	1200.000000	177.300000	18.900000	0.0

```
>>>aggregate["CO2"].var()
171317.75206811333
>>>aggregate["CO2"].cov(aggregate["NO2"])
-941.6460103811469
```

If we want to weigh our aggregate function, we can do for example

```
#random weights
>>>weights = np.random.random((aggregate.shape[0]))

#weighted average
>>>w_avg = (aggregate["CO2"] * weights).sum() / weights.sum()

>>>w_avg
656.4393493000837
```

This is quite convenient, but of course, input privacy of the respondent is lost. Hence, we start again using our toy model application to illustrate how this could be done with preserved input privacy. First, we need two important objects: the `PrivateDataFrame` and the `Aggregate`.

```
>>>from psatss import PrivateDataFrame, Aggregate
```

Using the `PrivateDataFrame` object, we turn the air quality datasets into their secret shared form

```
#Create 'private dataframes', these are versions of the pandas dataframes where
columns containing private data are secret shared. Turns out some datasets were
empty, so we've removed those
>>>private_dfs = [PrivateDataFrame(df)
                  for df in non_empty]
```

What happens we construct this object is that individual values in the dataset are secret shared and the references to these secret shared values are organized in way that allows for aggregation and reconstruction later on. Of course, in practice, the construction of this object would go a bit differently. Namely, dataset values would be secret shared by the respondent device with the MPC servers of the platform according to

the specifications provided by the central coordinating server, who then organizes the pointers to the shares . The principle is, however, quite similar.

A data scientist can now have complete access to the object, but when they try to access the data, only metadata is returned

```
>>>private_dfs[0]

Private Dataframe containing data regarding variables ['CO2', 'Temperatuur',
'Luchtvochtigheid', 'Fijnstof', 'VOC', 'NO2', 'Ozon', 'CO']

Variables ['Tijd', 'Ruimte'] were skipped because the contents were non
numerical. These have to be encoded as numbers for secret sharing.

Columns ['Geluidsniveau'] contained missing values. Consider imputing these
before sharing.
```

This is because the data has not been sufficiently aggregated yet and will hence reveal private information. Also, note that a few variables of the original dataset have not been processed. If these are required for statistical analysis, one needs to find a way to encode these into a format that can be secret shared(*i.e.* elements of a finite field, though integers will do as MPYC will encode these as such).

To access the statistical information in the data, a data scientist has to aggregate the datasets. The secret shared data in the `PrivateDataFrame`'s will then be combined into an object that provides methods for analysing the data using MPC protocols.

However, if we want to weigh the data according some pre-determined weights, we need to add these to the `PrivateDataFrame`'s first.

```
#Add random weights to the dataframes

>>>np.random.seed(10)

weights = []

>>>for pdf in private_dfs:
    pdf.add_public_weight("random_weight", np.random.random())

#Create aggregate from private dataframes
>>>priv_agg = Aggregate(private_dfs)
```

Let's first take a look at a description of the `Aggregate` object

```
>>>priv_agg
Aggregate of 20 datasets
Private Variables: ['CO2', 'Temperatuur', 'Luchtvochtigheid', 'Fijnstof', 'VOC',
'NO2', 'Ozon', 'CO']
```

Selecting a certain column, for example 'CO_2', returns an `AggregatedColumn()`, which contains the columns for this variable from all respondent `PrivateDataFrame()`'s. This object provides methods for performing aggregate functions, like

```
>>>priv_agg["CO2"].mean(reconstruct = True)
656.8642142834142

#If we don't set reconstruct = True, a SecureNumber is returned
>>>priv_agg["CO2"].mean()
<mpyc.sectypes.SecFxp64:32 at 0x1d3c51b2f40>

#We can compute variance similarly
>>>priv_agg["NO2"].variance(reconstruct = True)
287.51426232862286

#Or covariance like this
priv_agg["NO2"].cov(priv_agg["CO2"], reconstruct = True)
-942.6268973771948
```

We can also weight these functions using the respondent-specific weights that were added before we constructed the `Aggregate()`. The `Aggregate()` ensures that the right weights are applied to the right values.

```
>>>priv_agg["CO2"].mean(weight = priv_agg.weights["random weight"],
                        reconstruct = True)
683.3180708731525
```

Unsurprisingly, the result is pretty close to the unweighted mean since we sampled the normalized weights randomly from the normal distribution.

Analogous to Pandas, we can select certain parts of the data based on conditions in a pythonic fashion like[2]

```
>>>private_agg[private_agg["CO2"] > 200]
Aggregate of 20 datasets.
Columns: ['Tijd', 'Ruimte', 'CO2', 'Temperatuur', 'Luchtvochtigheid',
'Geluidsniveau', 'Fijnstof', 'VOC', 'NO2', 'Ozon', 'CO']
```

The object returned is again an `Aggregate`, where the masks of the columns have been updated to reflect the selection condition, such that when we perform aggregate functions only the values from the selected records are used. Note that the aggregate still contains 20 datasets. This is because apparently all datasets had at least one record where 'CO2' was higher than 200 ppm. If our selection condition is very specific though, we might run into the following problem

[2] A reader who tries the source code themselves will find that these selection operations are very slow. The problem is that MPYC does not yet provide a vectorized version of the comparison operators. This means that for every row in the aggregate, all the rounds of communication for one comparison are needed. Number of rounds hence scales linearly with the dataset size in this case. If the operation had been vectorized, all rows could be processed in parallel, requiring only the number of communication rounds necessary for one comparison, leading to constant scaling with dataset size

```
>>>temp_g_34 = priv_agg[priv_agg["Temperatuur"] > 34]
>>>temp_g_34["CO2"].mean(reconstruct = True)
---------------------------------------------------------------------------
ReconstructionError            Traceback (most recent call last)
<ipython-input-95-3e2a6be0b14e> in <module>
----> 1 temp_g_34["CO2"].mean(reconstruct = True)
<ipython-input-85-05f36c68b0a3> in mean(self, weight, reconstruct)
      205 mean = s / self.N
      206 if reconstruct:
-->   207 return self.reconstruct(mean)
      208 return mean
      209

<ipython-input-85-05f36c68b0a3> in reconstruct(self, r)
      249 return mpc.run(mpc.output(r))
      250 else:
-->   251 raise ReconstructionError('Aggregation level insufficient')
      252
      253 class ReconstructionError(Exception):

ReconstructionError: Aggregation level insufficient
```

The system is preventing us form reconstructing the result because our selection criteria have selected data from an insufficient number of respondents (there was only one record with temperature greater than 34).

If the aggregation level is sufficient, we can query aggregate results like before

```
>>>co2_le_600 = priv_agg[priv_agg["CO2"] <= 600]
>>>co2_le_600["NO2"].cov(co2_le_600["Temperatuur"], reconstruct = True)
-7.263699083356187
```

This is the basic functionality of the application. However, let's also look at the mobility use-case to see some more specific considerations. We load the data as before and take a look

```
>>>datasets = [
               pd.read_csv(
                           os.path.join("RespondentData", file), sep=","
                           ) for file in os.listdir("RespondentData")
               ]
>>>datasets[3].head()
```

	ID	Start	End	Distance	MOT
0	0	Borger-Odoorn	Voorschoten	183.792347	Walking
1	1	Sint Anthonis	Aalsmeer	105.228433	Train
2	2	Voorst	Montfoort	81.952900	Boat
3	3	Neder-Betuwe	Steenbergen	98.206437	Bicycle
4	4	Gooise Meren	Waalwijk	65.504997	Train

The data contains *journeys* described by a start and end location in terms of a municipality name, covered distance in kilometers and a mode of transport. In practice, this information could have been filled in by a respondent in a diary, deduced from passively collected GPS data in conjunction with machine learning to classify mode of transport or a combination of both.

With the exception of the ID and distance columns, the data is categorical. For sharing the secrets, we need to turn them into numerical values. We can simply do this by assigning an integer to every category as follows

```
>>>import csv

#Construct municipality code dictionary
>>>munic_codes = {}

>>>with open('Gemeenten_alfabetisch_2020.csv') as read_csv:
       reader = csv.reader(read_csv, delimiter="\t")
       header = next(reader)
       for row in reader:
           munic_name = row[2]
           munic_code = row[0]
           munic_codes.update({munic_name: int(munic_code)})

>>>#Construct dictionary with codes for different modes of transport
>>>MODES_OF_TRANSPORT = ["Walking", "Car", "Train", "Boat", "Bicycle"]

>>>mot_codes = {}

>>>for index, mot in enumerate(MODES_OF_TRANSPORT):
       mot_codes.update({mot: index})

#Encoding functions
>>>def encode(input, codes = None):
       encoded = codes[input]
       return encoded

>>>def decode(input, codes = None):
       decoded = list(codes.keys())[list(codes.values()).index(code)]
       return decoded

#Apply encoding
>>>for dataset in datasets:
       dataset["Start"] = dataset["Start"].apply(encode, codes = munic_codes)
       dataset["End"] = dataset["End"].apply(encode, codes = munic_codes)
       dataset["MOT"] = dataset["MOT"].apply(encode, codes = mot_codes)
>>>datasets[3].head()
```

	ID	Start	End	Distance	MOT
0	0	1681	626	183.792347	0
1	1	1702	358	105.228433	2
2	2	285	335	81.952900	3
3	3	1740	851	98.206437	4
4	4	1942	867	65.504997	2

This is all we need to do to proceed as before. Let's check some more examples. First, we aggregate the private respondent data as before

```
#Create private dataframes
>>>priv_dfs = [PrivateDataFrame(df) for df in datasets]

#Join them together in aggregate
>>>private_agg = Aggregate(priv_dfs)

#Check description
>>>private_agg
Aggregate of 31 datasets
Private Variables: ['ID', 'Start', 'End', 'Distance', 'MOT']
```

Then, we can do some computations

```
#Compute number of journeys starting in amsterdam and the mean distance
>>>amsterdam_start = private_agg[
                                private_agg["Start"] == munic_codes["Amsterdam"]
                                ]
>>>amsterdam_start["Distance"].count(reconstruct = True)
3.0
>>>amsterdam_start["Distance"].mean(reconstruct = True)
59.551676059374586
```

Notes on performance

All tests were done locally, in an emulated MPC environment. That is, all computations were performed according to the appropriate MPC protocols, but no actual transfer of data between parties took place. Since communication between parties is the greatest contributor to overhead involved in MPC, not much can be said about performance related topics. The only exception is with regards to the comparison operations: even without network communication, just running the functions took a considerable amount of time. The reason for this, as was already mentioned in a footnote, is that MPYC did not support vectorized execution of comparison operations, leading to $O(n)$ scaling with the number of rows. In principle, nothing prevents these operations from being vectorized, so this should be a simple performance optimization.

4 Conclusions and Lessons Learned

4.1 Architecture

In the first section of this chapter, we looked at the design of an architecture that would allow for the application of various privacy enhancing technologies to the data collected in the context of the TSSu. The purpose of these technologies is simple: enable the production of aggregate statistics without the need for anyone, or any system to have access to the actual data. To aid in the design of this architecture, we distinguished between different phases of the survey process based on the functional and privacy enhancing requirements.

From this section, we can draw a few key conclusions:

- Given well-defined and well-designed surveys, application of PET for oblivious data-analysis requires a few additional architectural components. However, the resulting architecture is not much more complicated than one would expect for a platform without PET.

- Quality control, logging and hence survey maintenance is difficult without compromising privacy. It is hence likely that trade-offs have to be made in this regard.
- Design and development of surveys is incredibly difficult given the restrictions of privacy enhancing technologies. These difficulties are inherent to the goal of applying these technologies and can hence not be overcome by technological advancement. Hence, it is probably required to have a non-privacy-enhanced pilot phase for new surveys, in order for these to be designed properly.

4.2 Aggregate Statistics using MPC

The second section of this chapter was dedicated to a more practical exercise, namely the development of a toy model application enabling aggregate statistics using secret shared data. The purpose of this exercise was to showcase some considerations that go into performing aggregate statistics in an MPC setting, as well as to demonstrate what is possible using an open-source MPC library. Note however that many functionalities are missing from the application for it to translate directly into a production setting that matches the architecture described in the first part of this chapter.

The main conclusions and lessons learned from the exercise were:

- It can be expected that, at the time of writing, when using open-source MPC software, much implementation has to be done on top of this software. Some of these implementations go beyond typical software engineering and require expert knowledge of cryptography or statistics for these to be done properly (in contrast to the sometimes haphazard work presented in this document).
- Private statistical analysis requirements set for the PoC were mostly met by the designed application. However, statistical analysis included just very basic aggregate statistics. More research would be necessary if such a design would satisfy all the needs of modern statistical analysis.
- Implementing this design in a toy model by adapting existing technology was pretty straightforward. However, note that the scope was limited to an emulated environment. Implementing an actual MPC environment is much more involved.
- Production level implementation should be considered a serious software engineering/cryptography/methodological endeavor requiring experts in these fields.

Finally, given the high complexity of implementing technologies in a production platform it is highly recommended that NSI's do not attempt to do this themselves, but in collaboration with a community of experts or a commercial party.

5 References

1. MPYC. [Online] https://github.com/lschoe/mpyc.

2. SyMPC. [Online] https://github.com/OpenMined/SyMPC .

3. [Online] https://github.com/javanetten/psatss.

4. [Online] https://getuhoo.com/.

5. MPYC documentation. [Online] https://mpyc.readthedocs.io/en/latest/.

6. Pandas. [Online] https://pandas.pydata.org/.

7. *Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution.* Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. 2018, 27th USENIX Security Symposium, SEC'18, pp. 991–1008.

8. S. V. Schaik, Andrew Kwong, Daniel Genkin. SGAxe: How SGX Fails in Practice. *sgaxe.com.* [Online] 2020. https://sgaxe.com/files/SGAxe.pdf.

9. JASON. Secure Computation for Business Data. [Online] 2020. https://irp.fas.org/agency/dod/jason/secure-comp.pdf.

10. Fabio Ricciato, Triin Siil, Riivo Talviste, Baldur Kubo, Albrecht Wirthmann. A proof-of-concept solution for the secure. *ec.europa.eu.* [Online] 2021. https://ec.europa.eu/eurostat/cros/sites/default/files/unece2021_estat_cybernetica_v6.pdf.

11. *How to share a secret.* Shamir, Adi. 1979, Communications of the ACM, pp. 612–613 .

12. Jeffreys, H., Jeffreys, B. S. Lagrange's Interpolation Formula. *Methods of Mathematical Physics 3rd ed.* Cambridge, England : Cambridge University Press, 1988, p. p. 260.

13. *Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation.* Ben-Or, M., S. Goldwasser, and A. Wigderson. 1988, 20th Annual ACM Symposium on Theory of Computing. ACM Press. 1–10, p. 4.

14. Cramer, R., Damgård, I., & Nielsen, J. *Secure Multiparty Computation and Secret Sharing.* Cambridge : Cambridge University Press.

Appendix

A.1.1 Federated Learning

Collection of sensor data from smart devices is an important part of the TSSu platform. These data often require classification by machine learning models to be converted into useful input aggregated statistics. Invariably, training of these machine learning models requires data, ideally as much as possible. The data for training these models belongs to various owners, hence training of these models should be considered a case of combining private sources, necessitating the use of PET.

Perhaps the most efficient technique for privacy preserved machine-learning is federated learning. Contrary to the conventional machine-learning process for training models on data from various sources, which starts with collecting these data in a central location and training a single instance of the model, the federated approach leaves data in its place and brings the model to the data. Several instances of the model are trained for a given number of epochs, the updated weights are sent to the central server and aggregated to construct an updated model. This process, shown in Figure 6, is repeated for a pre-defined number of times, or whenever model accuracy is satisfactory.

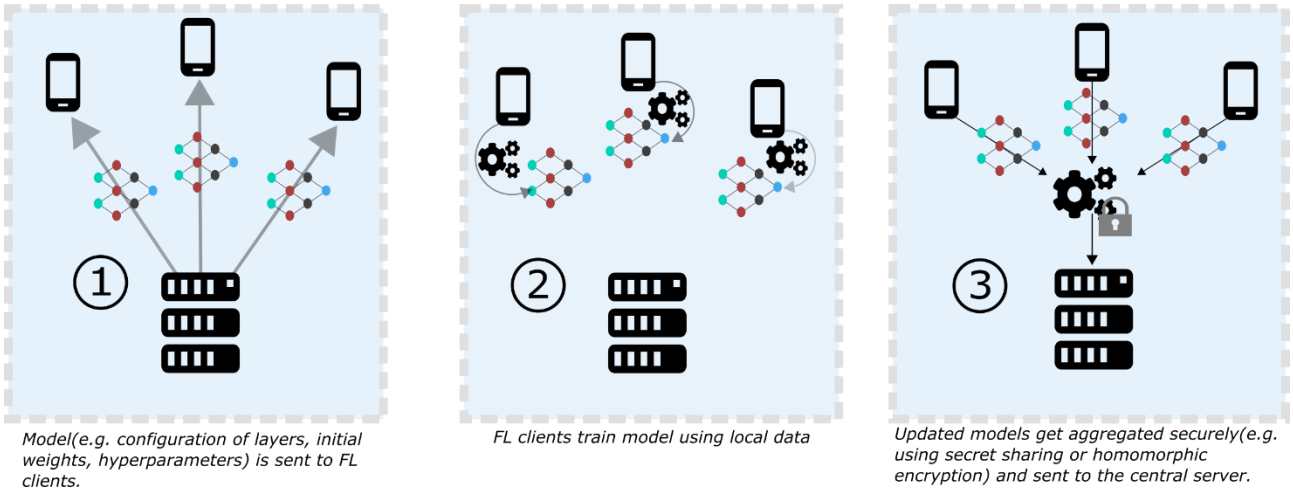


Figure 6. The federated learning process

By itself federated learning is however not enough to provide absolute input privacy. Namely, updated weights do in principle reveal information about the local training data. Of course, it is not possible to fully reconstruct the local data based on these weights. However, some leakage may occur and is, in fact, to be expected. For example, consider training an activity recognition model across multiple devices carried by individuals. If one of these individuals predominantly walks, then his local model will only develop predicting capabilities with respect to walking. This information is contained in the weights that are sent to the central server, and hence leak private information about the individual. A proper federated learning implementation therefore requires a strategy for secure aggregation of the models. For this, we can use techniques like homomorphic encryption, trusted execution environments or secret sharing. More on these techniques in the following sections.

A.1.2 Trusted Execution Environment

After collection of respondent-level microdata in the personal domain of the respondent, these data need to be combined with other sources for aggregation. A possibility for doing so is provided by so-called *trusted*

execution environment(TEE) or *secure enclaves*. These are either software (*e.g.* AWS Nitro) or hardware (*e.g.* Intel SGX) based techniques for running pre-agreed programs in isolated environments with very limited IO, located within another computing environment.

In practice, when a TEE is used for combining private data sources, the following workflow is carried out:

- Input parties (*e.g.* NSI's) agree on a program (*e.g* a set of aggregate functions) to be executed. A cryptographic hash of the program and its parameters is created, which the TEE will use to prove that the agreed upon program was run.
- Program, as well as any public parameters are uploaded to the TEE.
- Input parties send their data in encrypted form to the TEE.
- Pre-agreed program(including decryption of inputs) is run with provided parameters. The TEE makes sure that only pre-agreed functions are performed.
- Results are encrypted and returned to the input partie(s)
- Via the attestation process, the cryptographic hash created in step (1) allows anyone who has access to the source code of the program to verify that the output was indeed created by said program

Even though in principle any function can easily run in a TEE and as such is quite readily adapted to complicated functions, aggregation using this technology is still less flexible than without. Namely, all functions to be executed inside the TEE have to be pre-agreed upon. Hence, if a specific function cannot be composed of functions allowed inside the TEE, it has to go through the authorization process and has to be added to the allowed functions. Of course, this is not necessarily a bad thing, as it adds to the security of the platform. It does however limit the flexibility experienced by the user of the platform.

Another point worth considering is that trust is localized in a single entity, namely the hardware/software implementation of the secure enclave. If there turns out to be a security issue with this implementation, an attacker only has to compromise the host system. As shown by various instances of security issues with Intel SGX (7), (8).

Based on these vulnerabilities, in (9) it was argued that trusted execution environments are not yet ready to be used for purposes like ours. On the other hand, in a recent Eurostat project (10) TEE's were implemented with reported success. Given these varying reports, we will still consider TEE's as a viable candidate for secure computation, but with a footnote regarding the vulnerabilities.

A.1.3 Secret Sharing

As described in the previous section, the major downside of using trusted execution environments is the localization of trust in a single entity (the hard- or software solution used). A multi-party computation technique that does not share this problem is *secret sharing*. In its simplest form, secret sharing involves breaking up a secret value into multiple shares that seem random by themselves, but given enough of them, uniquely define the original value. These shares can hence be shared with other parties without them gaining any knowledge of the secret, if they can be trusted to not collude to reconstruct the original value.

For example, say we want to share $s = 5$ with two parties, Alice and Bob. Then we randomly pick s_A, define $s_B = s - s_A$ and send s_A to Alice and s_B to Bob. To them, their shares seem random, but if we add them back together, we retrieve the secret value s. Such a secret sharing scheme is called *additive*, as the sum of shares

equates to the shared secret. Typically, for additive sharing schemes, we need all parties to combine their shared to reconstruct the secret. We call such a scheme an *n*-out-of-*n* secret sharing scheme, referring to the fact that secrets can be reconstructed if we combine $t = n$ of the n shares, where t refers to the *threshold* t and n to the number of parties. A simple example of how we can do computations using additively secret shared numbers is shown in Figure 7

In contrast, another common method, proposed by Adi Shamir Shamir (11), provides *t*-out-of-*n* secret sharing with $\mathrm{t} \geq (\mathrm{n} - 1)$ based on the following theorem:

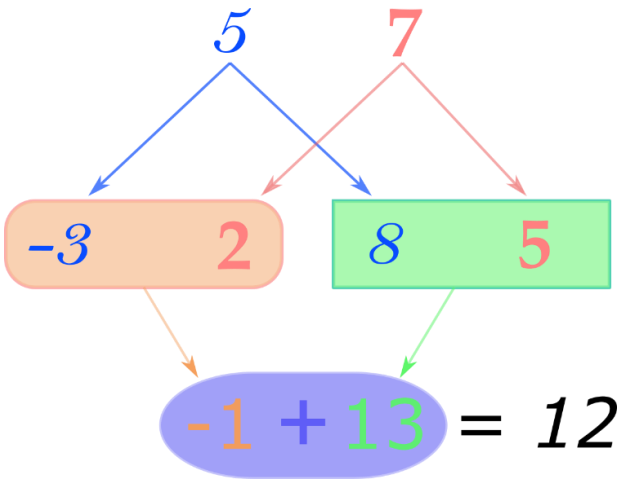


Figure 7. Two numbers are secret shared additively. Then, their sum is computed by adding the individual shares.

Intuitively, we can view this as generalization of the fact that two points uniquely define a straight line, but see (12) for proof.

Now, given this theorem, let's take a secret[3] $s \in \mathbb{Z}_p$ that we want to share with n parties, with $p > max(s, n)$. Let the desired threshold be $t \leq n$. Then,

- Construct random degree $(t-1)$ polynomial $\mathrm{p}(x) = a_{t-1}x^{t-1} + \ldots + a_1\mathrm{x} + a_0$ over $\mathbb{Z}_p$ by randomly choosing $t-1$ coeffients $a_{t-1}, \ldots, a_1 \in Z_p$
- Set $a_0 = s$ such that $p(0) = s$
- Evaluate $p(x)$ at n points $x_1, \ldots x_n$ to construct the sharing $[s] = \{(p(x_1), x_1), \ldots, (p(x_n), x_n)\}$, where square braces denote that s has been secret shared.

[3] Note that our secret has to be an element of the finite field $\mathbb{Z}_p$, *i.e.* it should be in the set of integers modulo p. This is a typical requirement in cryptography, but let us take a moment to see why this is necessary for our case at hand. First, taking our secret $s \in \mathbb{R}$ leads to problems in practice as computers use approximate floating-point representations for decimal values rendering it impossible to reconstruct a secret accurately. If we have integer secrets $s \in \mathbb{Z}$, this problem is solved, however, we run into another. To see this, let's consider an example: say we have a secret $s = 1$ to be shared with Alice and Bob. We then sample randomly $a_1 = 3$ to construct the polynomial $p(x) = x + 1$ and send to Alice $s_A = (p(3), 3) = (4,3)$ and to Bob $s_B = (p(1), 2) = (2,2)$. Now, let's say Alice knows computers can only deal with integers accurately, so she knows that the coefficients of the polynomial have to be integers. Then, if she tries $s = p(0) = 2$ combined with the knowledge that $p(3) = 4$, then she should have $p(x) = \frac{2}{3}x + 2$. However, since Alice knows that the polynomial coefficients can only be integer valued, she knows that the secret cannot be $s = 2$. This means our secret sharing scheme over $\mathbb{Z}$ does not provide perfect secrecy. The solution is to take our secret—and hence all coefficients of the polynomial—to be elements of a finite field, for example $\mathbb{Z}_p$ with p a large prime. This way, no matter Alice's share $(p(x_a), x_a)$, all guesses $p(0)$ have equal probability of being the secret.

- Send to every *i-th* party the share $s_i = (p(x_i), x_i)$
- Based on the theorem above, t out of the n shares now uniquely define the polynomial $p(x)$
- Hence, the secret can be reconstructed if t parties combine their shares, find $p(x)$ and evaluate at $x = 0$

In the context of secure multiparty computation, the above construction might not seem that useful yet as we have only considered sharing of secrets, no actual computation. However, it turns out that shared secrets can readily be computed upon given the right protocols. For example, say we want to compute the sum of two secret shared values $[s] = (p(x_A), x_A), (p(x_B), x_B)$ and $[s'] = (p'(x_A), x_A), (p'(x_B), x_B)$ shared with two parties A and B using polynomials $p(x)$and $p'(x)$ of degree $t - 1$. This can easily be achieved by realizing that $q(x) = p(x) + p'(x)$ defines a new polynomial defined uniquely by the sum of the shares held by the parties A and B as

- $q(x)$ is of the same degree as $p(x)$ and $p'(x)$
- $q(x_A) = p(x_A) + p'(x_A)$ and $q(x_B) = p(x_B) + p'(x_B)$
- $q(0) = p(0) + p'(0) = s + s'$

Hence, together parties A and B can reconstruct $s + s'$.

Multiplication of secrets is a bit harder and we will not look into the execution in detail here, but the curious reader is referred to (13). Typical for MPC protocols is that operations involving multiplication require communication between the parties. This fact is generally the cause for much of the overhead involved in secret-sharing based MPC (in the current case we have $O(n^2)$ per multiplication for n party computations.

Given the existence of an addition and multiplication operator, we can now compute any arithmetic circuit on our secret shared values. However, for practical applications, we often need comparison operators as well. Unfortunately, trying to represent these as arithmetic circuits over $\mathbb{Z}_p$ leads to very large circuits. However, there exist efficient techniques for converting secrets into their bitwise representations and performing comparisons in a bitwise manner, see for example section 9.1.6 of (14)

Task 3.2
Sub-tasks 3.2.4

Full transparency and auditability of processing algorithms
“PoC on architecture and metadata”

Prepared by:

Francesco Amato (ISTAT, Italy)

Raffaella Maria Aracri (ISTAT, Italy)

Fabrizio De Fausti (ISTAT, Italy)

Francesca Inglese (ISTAT, Italy)

Mauro Bruno (ISTAT, Italy)

Giuseppina Ruocco (ISTAT, Italy)

Task leaders: Mauro Bruno - ISTAT (Architectural task)

Franck Cotton - INSEE (Metadata task)

Outline

Executive summary

The main goal of this deliverable is to test the assumptions and benchmark the results concerning:

- The previous inventory of official statistical standards and ongoing projects, released by Task 3.1.3[4] dealing with the preparatory work for the design of a TSSu platform and integration in existing architectural framework
- The approach adopted by task 3.1.6 Metadata and Process auditability, leading to the design of a Metadata Repository (MR) composed of several subsets, based on existing frameworks and ontologies.

The document is structured in four chapters and two main parts, describing respectively the architectural analysis and the metadata captured for the Machine Learning (ML) PoC. More in detail, the first chapter provides an overview of the specific objectives of both, the Architectural and the Metadata PoC, as well as the expected results.

The second chapter concerns the architectural use case, modelling the main steps of a generic TSSu. More in detail, starting from the description of the statistical process in terms of GSBPM phases and BREAL business functions, the architectural PoC has modelled the business and application layers, to propose a set of application services to be provided by the platform. The analysis of operational models and architectural scenarios has complemented the theoretical assumptions related to the architectural layers. The specification of the dimensions impacting on the deployment of the application services, such as data storage and processing environment, adoption of Privacy preserving techniques is essential to define the requirements of the technical infrastructure of the platform.

The third chapter deals with the description of process metadata related to the tasks performed by the ML generalized module, developed to process accelerometer data. Starting from the conceptual subsets of the MR, the PoC has allowed to describe the process steps executed, to track data transformations, to document and assess the methods applied in each step.

The fourth chapter summarizes the main results achieved by the PoCs, the main lessons learnt and some open issues to be addressed, to achieve an enhancement of the reference architecture delivered at the end of the first year of the project. One of the main open issues is to highlight the relationship between the dimensions explored by the single tasks, such as: methods, incentives, privacy preservation, process auditability, assessment of smart data quality, technical requirements.

Concerning the main results achieved, the architectural PoC has demonstrated the feasibility of process standardization, to foster the integration between traditional and smart survey tasks. This analysis supports the specification of the main technical requirements of the TSSu platform, and is essential to plan the development activities. The metadata PoC has confirmed the initial assumptions, based on a central repository for the collection and management of the process metadata related to specific tasks of sensor data processing.

[4] Deliverable 3.1 Report on the Preliminary Framework, available from: https://ec.europa.eu/eurostat/cros/content/wp-3-conceptual-framework-european-platform_en

1. Objectives of the Poc

The main goal of the PoC is to test the assumptions and validate the outcomes resulting from the tasks within work package 3, dealing respectively with:

- Preparatory work for the design of a TSS_u platform and integration in existing architectural framework (task 3.1.3)
- Metadata and Process auditability (task 3.1.6).

In relation to architectural aspects, the PoC will allow to better analyse the TSS_u platform requirements in terms of users' needs. A deeper understanding of the tasks to perform (What) is essential to identify the main application components to implement (How), and the technical solutions for a common infrastructure. More precisely, the objectives of the architectural PoC are:

- Model the platform business layer, particularly data collection activities and ML steps, to increase the degree of process standardization for a TSS_u
- Model the platform application layer to identify the main application services executing the main steps of a TSS_u
- Model the data flow throughout the process to assess different scenarios assuming different data storage environments: on platform premises, and/ or at national level, and/ or in the cloud
- Provide an overview of the different dimensions to be considered for the enhancement of the initial framework that will be delivered by WP3 at the end of the project, based on PoC results and WP2 insights.

Regarding metadata, task 3.1.6 activities have resulted in the design of a Metadata Repository (MR) composed of several subsets, modelled according to existing frameworks and ontology concepts. In order to contribute to the design of the TSS_u platform and identify specific domain requirements, the PoC is intended to:

- Benchmark the approach adopted to model MR subsets
- Classify process metadata according to the subsets of the proposed MR
- Identify the metadata related to sensor data processing through Machine learning (ML) techniques
- Document data transformations and methods applied in each step
- Provide an analysis of metadata component requirements and its relationship with the other components of the TSS_u platform.

These objectives will be accomplished by interacting with the colleagues involved in the ML PoC, and in the Poc on Technical Infrastructure.

2. Architectural use case: modelling a TSS_u

The main objective of the following analysis is to gain an overview of the main tasks performed to carry out a TSS_u. Modelling the business and application layers of a generic TSS_u will provide evidence of the main opportunities and challenges of using smart and traditional data sources. The general requirements of the TSS_u platform result from the combination of several issues, such as privacy preservation, quality management and process auditability that need to be addressed, regardless of a particular survey. More specifically, a top-down approach will highlight the relationship between the different dimensions describing the platform requirements which can be classified as follows:

Architectural requirements

- The TSS_u platform should provide several software solutions to interact with human respondents and smart devices, thus meeting the requirements of several surveys, regardless of the national peculiarities and statistical domain
- Reuse and share available smart data tools and applications
- Develop user-friendly solutions and intuitive interfaces for end users.

Methodological requirements

- Implement methods to deal with smart missing data and check suspicious input data, also relying on the interaction with respondents and/or smart devices
- Use intelligent algorithms to interact with respondents, to reduce response burden and minimise interference with his/her activities
- Use intelligent algorithms to interact with smart devices to minimise the consumption of additional resources, such as battery power, communication bandwidth
- Support the implementation of customized incentive schemes, to improve the participation rate.

Trust requirements

- Hardware and software solutions implemented in the TSS_u platform should guarantee security, confidentiality protection, quality assessment, process auditability and privacy preservation, also through secure private computation (e.g., Secure Multiparty Computation solutions).

Starting from the above requirements, the architectural PoC is intended to summarize the emerging issues and insights gained so far, to guide future development activities.

2.1 TSS_u Business layer

Modelling the Business layer of a TSS_u allows to specify the core activities and behaviours to use smart data for statistical purposes, as well as the main stakeholders involved in smart data collection and processing. Concerning the statistical process, the following exploration takes into account the insights gained from work package (WP2) pilot surveys. These experiences have highlighted the challenges related to smart data acquisition and processing, in order to reduce the respondent burden in social surveys. Particularly, the proposed business layer for a generic TSS_u results from:

- WP2 description of Pilot on activity tracker
- WP2 description of HBS pilot survey
- WP2 platform specifications
- Report of the legal-ethical Working Group
- Deliverable 3.1 Report on the Preliminary Framework. For methodology and reference frameworks.

The following questions have guided the business layer design:

- What is the impact of smart data sources on GSBPM phases and process steps?

- Which steps of the process can be standardized and executed in a common infrastructure, to identify shareable and reusable application components?
- Which activities can be executed in the TSS_u platform or in the respondent's device, and which tasks must be performed in-house by the National Statistical Institutes (NSIs)?
- What is the best approach to harmonize specific national needs and the peculiarities of the stakeholders engaged in a TSS_u process?

In relation to the stakeholders, as this analysis takes into account mainly the perspective of National Statistical Institutes (NSI_s), the key actors involved or affecting smart data acquisition and processing have been grouped in the following subsets:

- Smart Data providers
- Smart Data miners
- Smart Data users.

Smart Data providers

The first group includes both, respondents that provide (actively or passively) data collected through their devices, and third parties acting as primary data gatherers. In order to interact with respondents, NSI_s need to develop ad-hoc collection tools having user-friendly interfaces and other features to deal with smart devices. Data stored in environments managed by third parties can be acquired through the definition of communication protocols, to specify data transmission requirements. The TSSu platform could provide interoperable solutions to address the most common issues related to these types of data providers.

Smart data collectors

The second subset groups all actors actively involved in smart data collection, and more precisely in the survey fieldwork requiring human interaction to deal with the respondents, or monitor the response rates. The TSSu platform could provide several application services to facilitate the tasks performed by the staff involved in the fieldwork activities, such as interviewers or field coordinators.

Smart Data miners

The third group contains all entities involved in smart data acquisition and processing for statistical purposes. With reference to Big Data REference Architecture and Layers (BREAL[5]) [1], this group may correspond to "IT & Statistical Pipeline Actors" which cooperate to perform the main steps of the smart data pipeline, and include several roles such as:

- Computing Architects, for the definition of the data system and the infrastructure requirements
- Information Architects, dealing with smart data conversion, cleaning, validation and modelling
- Data Scientists exploring the potential of smart data in terms of statistical information
- Domain experts, contributing with their expertise to data modelling, quality assessment, as well as data validation and analysis.

[5] ESSnet on Big Data II, Work Package F, Deliverable F1. (2018-2021). https://ec.europa.eu/eurostat/cros/sites/crosportal/files/WPF_Deliverable_F1_BREAL_Big_Data_REference_Architecture_and_Layers_v.03012020.pdf

This group contains also those entities responsible for the definition of general frameworks and guidelines concerning methodological and technical aspects, whose activities impact on the quality and trust of the statistical process.

Smart Data users

The last subset includes final users and other organizations not involved in smart data processing, but using the statistical output obtained from smart data, such as:

- Governments, responsible for regulating smart data use and privacy protection issues and benefitting from statistical information to support decision making
- Medias, contributing to the dissemination of statistical results and the description of methods and data sources used for statistical purposes
- Researchers and all citizens which may use statistics obtained from smart data to describe or investigate a particular topic, and may participate actively to the statistical process by providing or sharing their data through smart devices.

In underlining the needs of new statistical outputs, final users foster the design of innovative statistical processes, based on new data sources.

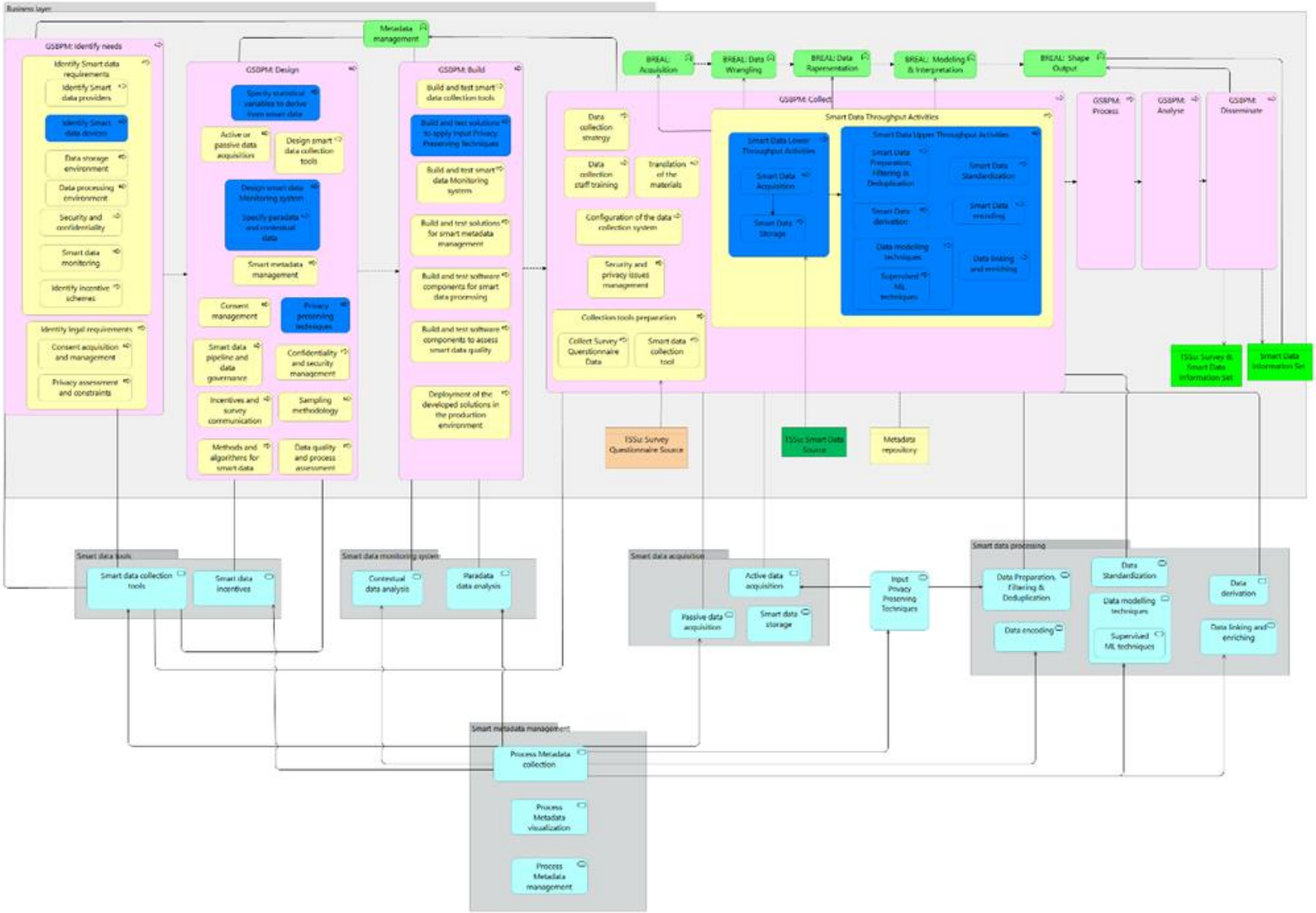
Starting from the alignment with existing frameworks, the following analysis of the main process steps for smart data processing is based on GSBPM[6] and GSIM[7] standards, as well as BREAL business functions. In order to identify a unique general pipeline for a TSS_u and combine smart and traditional data sources, the following ArchiMate[8] diagram shows for each GSBPM phase the main tasks related to smart data acquisition and processing. The yellow-coloured objects represent the activities performed in traditional surveys also, while all the tasks grouped in Smart Data Throughput activities concern only smart data sources.

[6] Generic Statistical Business Process Model, available from: https://statswiki.unece.org/display/GSBPM/GSBPM+v5.1

[7] Generic Statistical Information Model, available from: https://statswiki.unece.org/display/gsim/GSIM+v1.2+documents

[8]ArchiMate is an open and independent language used for architectural modelling, compliant with Enterprise Architecture standard and available from: https://www.archimatetool.com/

Figure 1: Business and application layer of the TSSu platform



The relationship between the first GSBPM phases and BREAL business functions is shown in the upper part of the figure. While metadata management is an overarching business function, the activities performed in the different GSBPM phases for smart data have been considered as a specialization of the related business functions. Once identified the main BREAL business functions (e.g., Acquisition and Recording, Data Wrangling, Data Representation) related to smart data activities, the following analysis will highlight, for each GSBPM phase, the main tasks to consider for the design of the application layer.

In the first GSBPM phase "Specify needs", the sub-tasks "1.5. Check data availability" and "1.6. Prepare business case" are particularly relevant to identify the connections between data requirements and the compliance with the legal framework, as well as to evaluate smart data pros and cons and other external constraints. Concerning the first issue, the legal-ethical working group has widely investigated the legal implications in terms of privacy assessment and security, depending on the type of:

- Smart data provider
- Smart data storage
- Smart data processing.

In addition to the common reference regulation, some differences in national legislations concerning privacy and security may make it more difficult to develop standard solutions. This preliminary analysis is essential to design the smart data strategy by defining:

- The data provider, respondents and/or third parties. In case of smart data provided by third parties, an initial assessment of data confidentiality is essential for the management of technical and legal solutions and may lead to the definition of protocols for smart data acquisition. Following the "push computation out" approach, these protocols may involve the adoption of input privacy preserving techniques, as well as the calculation of quality indicators to assess the quality of acquired data.
- The Type of device/ smart data source (e.g.: external sensors, data gathered through sensors embedded in mobile devices such as pictures, scanner data) generating active or passive data (primary or secondary data collection)
- Data storage environment, to be adapted to the type of smart data source (e.g.: relational, NoSQL, Json)
- Data processing environment. Both, data storage and processing environments will be analysed more in detail in the section describing the operational scenarios
- Smart data monitoring system, to check fieldwork and smart data accuracy during data collection
- Incentives to adopt, in order to motivate and reward the respondents.

Following the initial description of the business case, the GSBPM "Design phase" allows to describe more in detail the main steps of a statistical process. Focusing on the integration between smart data pipeline and traditional surveys, the main tasks to perform in this phase concern a detailed specification of:

- The statistical variables to derive from smart data
- Type of smart data acquisition, active or passive, depending on the type of smart data devices (e.g: GPS, cameras, accelerometer)
- Smart data collection tools, according to the type of smart devices to use for data acquisition. This task includes:
    - ✓ Smart data Monitoring system to check and assess data acquisition
    - ✓ Paradata and contextual data to collect during data acquisition
    - ✓ Consent management, depending on data acquisition scenarios (access to available external personal data, linkage to personal data already acquired by the NSI)

- ✓ Privacy preserving techniques to adopt according to the type of smart data acquisition

- Smart metadata management
- Smart data pipeline and data governance
- Confidentiality and security management
- Incentives and survey communication
- Sampling methodology, focusing on how to increase representativeness and prevent distortions due, for instance, to differences in the use of smart devices within particular subsets of the selected sample
- Methods and algorithms to process smart data and transform it in statistical information
- Data quality and process assessment.

Based on the preliminary design activities, the GSBPM "Build" phase concerns the development or reuse of several components executing the planned tasks in the production environment. In addition, this phase allows to test the implemented solutions and configure workflows, thus facilitating the migration from traditional pipelines to a new process combining smart and traditional data sources. Concerning smart data, the following activities are particularly relevant in this phase:

- Build and test smart data collection tools according to the type of data provider. In some cases, for sensor data, the acquisition tool may correspond to a software component allowing to retrieve or receive pre-processed data from third parties
- Build and test solutions to apply Privacy Preserving Techniques (PPT). In order to meet the privacy requirements identified in the design phase, Input privacy techniques should be implemented and tested. Several tests are essential to detail the requirements of the implementation of some Input privacy techniques during smart data acquisition, in terms of infrastructure and data collection tools. The results of the PoC dedicated to this topic, as well as the ongoing project promoted by UNECE[9] will provide useful information to identify the software solutions, as well as the hardware requirements
- Build and test smart data Monitoring system to reduce missing data and measurement errors. According to WP2 experience, capturing paradata and contextual data during data acquisition provides auxiliary information that is essential for the quality assessment of collected data
- Build and test solutions for smart metadata management
- Build and test software components for smart data processing. Among the methods for smart data processing, Machine learning techniques are particular relevant
- Build and test software components to assess smart data quality
- Deliver the developed solutions in the production environment. The pilot surveys carried out by WP2 will provide useful feedbacks about the challenges faced during the field tests. These experiences will provide useful insights for building a common infrastructure, the standardised production environment of the TSS_u platform, able to interact with several national production systems.

In the proposed model, GSBPM "Collect" phase is tightly connected to the following BREAL business functions: Acquisition and Recording, Data Wrangling, Data Representation and Modelling and Interpretation. More in detail, based on the analysis of BREAL Information layer, this phase includes Smart

[9] For more information about UNECE project Input Privacy-preserving Techniques 2021, see: https://statswiki.unece.org/display/hlgbas/Modernisation+Projects

Data Throughput Activities, a sub-process grouping several tasks to transform collected smart data in statistical data. Throughput activities can be divided in Lower and Upper activities and may vary in each survey, according to the features of the sensor device, the type of data provider, in-app, or in-house data processing. The ML PoC will allow to understand more in detail some of the Throughput Activities performed for accelerometer data.

Lower throughput activities concern mainly smart data acquisition and recording, while Upper throughput activities include:

- Data preparation, filtering & deduplication, a set of pre-processing steps to select the relevant information and create input data for the following tasks
- Data standardisation to convert data to a target format. This sub-step, as well as the following one is particularly relevant in case of passive data acquisition
- Data derivation to transform unstructured information to structured data. While the previous sub-step refers to data format, data derivation refers to the transformation of data content (e.g., creation of new variables), through the application of some rules and/or algorithms
- Data encoding to transform categorical data in binary or numeric format
- Data modelling techniques, grouping all the methods to extract statistical information from smart data, such as machine learning techniques
- Data linking and enriching, to integrate data collected through questionnaires and statistical information derived from smart data.

In addition to Smart Data Throughput Activities, other relevant tasks included in GSBPM "Collect" phase and related to the TSS_u platform are:

- Definition of a collection strategy compliant with national peculiarities
- Training of the staff involved in data collection
- Preparation of data collection tools
- Use of supervised machine learning techniques for online training or for model assessment, as suggested by WP2
- Configuration of the data collection system for data acquisition and storage
- Security and privacy issues during data collection.

At the moment, some activities performed at national level and included in this phase, such as the sample selection or arranging an agreement with third parties, have not been considered in the design of the platform pipeline.

2.2 TSS_u Application layer

Although the design of the TSS_u business layer is an on-going activity, the previous analysis allows to propose a list of the main application services to execute some process steps of a general TSS_u. These services will be described in terms of functionalities that could be offered by the TSS_u platform and will allow to define some scenarios for service deployment and sharing.

Regarding the quality, security and privacy issues, an analysis of the impact of these dimensions on the platform components will help to identify the requirements of the technical infrastructure and software solutions. While the application of data security and Input privacy preserving techniques relate mainly to the technical layer, due to the protection of data 'by design' across all application components, data quality assessment is a function that can be embedded in each software tool dealing with data processing.

The following table reports an initial inventory of services, resulting from the analysis of GSBPM phases and executing the activities related to smart data acquisition and processing described above.

Table 1: List of the main application services to implement for the TSSu platform

Smart services subsets	Main services	Description of functionalities
Smart metadata management	Process Metadata collection Process Metadata management Process Metadata reporting	This group of services realizes the overarching BREAL business function "Metadata management" and allows to acquire and visualize the process metadata used or produced by each application service and stored in the Metadata repository. In addition, within this group, ad-hoc services may allow authorised survey staff to: • check or modify the information reported in each subset of the Metadata repository, and track metadata changes • provide an overview of the metadata captured throughout the different steps for process tracking and auditability
Smart data tools	Smart data collection tools Smart data incentives	This subset of services provides a set of functionalities to facilitate the design and the implementation of tools for collecting smart data within or without survey questionnaires, depending on the type of smart data provider and the data collection technique. A relevant subset of functionalities offered by a specific service concerns the creation of incentives to foster the respondents' collaboration
Smart data monitoring system	Contextual data analysis Paradata analysis	The main goal of this group of services is to monitor smart data acquisition, by enriching the traditional indicators with the analysis of paradata and contextual data to check the fieldwork. This analysis is intended to assess the quality of collected data (in terms of representation and measurement errors) through the study of: • the technical issues arose during data collection • the respondents' behaviour
Smart data acquisition	Active data acquisition Passive data acquisition Data storage	This subset provides three main services for smart data gathering. The first service allows to gather smart data actively sent from respondents, while the second one realizes data acquisition from smart devices or third parties' platforms. The third service complements data acquisition by providing data storage capacities (relational

		DB, noSQL DB) and optimising data recording performances
Smart data processing	Data preparation, filtering & deduplication Data standardisation Data derivation Data encoding Data modelling techniques Data linking and enriching	The services grouped in this subset provide several functionalities to perform the main tasks to transform raw smart data (signals, images, texts) in statistical output. While some components need to be specialized according to the smart data source, the design of a standardized pipeline would foster the reuse of available solutions
Input Privacy Preserving Techniques	Federated learning Secure Multi-Party Computation	This subset of services relates to the algorithms and protocols to apply for Input privacy preservation. The execution of these services may require the access to a distributed computing environment

The proposed services have been designed according to a modular approach, fostering the sharing and the reuse of available software components, as suggested by the CSPA standard. Nevertheless, the functionalities reported above may vary according to the type of smart data. This approach aims at highlighting the services having a higher degree of generalization and the services that must be specialized to meet particular architectural requirements and process specific smart data sources.

2.3 TSS_u Operational models and architectural scenarios

The design of application services, to gather and process smart data, allows to identify how to deploy the implemented solutions with respect to specific dimensions. Starting from BREAL operational model, and ESS EARF standard[10], application services can be classified as follows:

- Autonomous services, developed at local level in NSI's environment, without harmonisation between countries
- Interoperable services, having a similar service interface across the NSIs, and different back-end implementations
- Replicated services, if the same application service is delivered to different NSIs
- Shared services, realized through application services available in the platform and accessed by all NSIs.

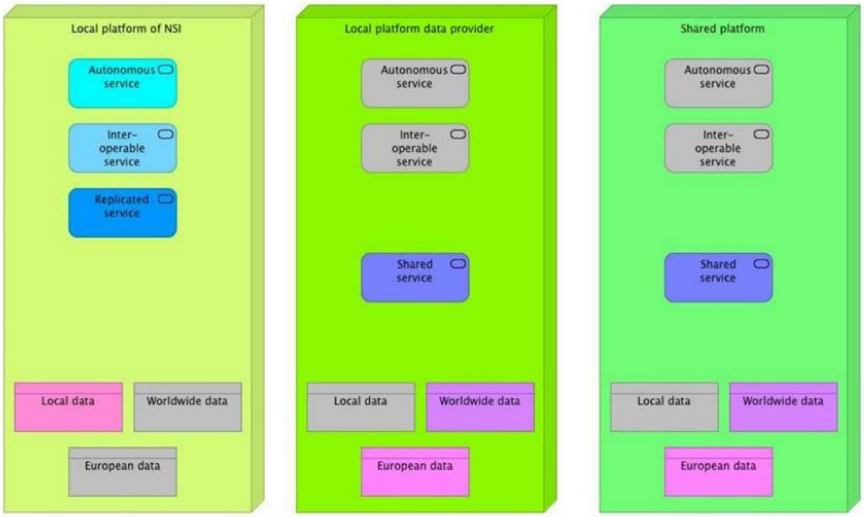
[10]The ESS Enterprise Architecture Reference Framework fosters the realisation of Vision 2020. Available from: https://ec.europa.eu/eurostat/cros/content/ess-enterprise-architecture-reference-framework_en

10

Depending on the location and owner, the infrastructures and platforms for data hosting and management can be grouped in:

- Local platforms implemented and managed by NSIs
- Local platform of external data providers accessed by NSIs to gather processed data
- Shared platform, developed by a third party to provide shared statistical services to perform smart data processing.

Figure 2: Operational model proposed by BREALI[11]



Privacy issues have a relevant impact on the design of potential operational models. The following table describes more in detail the several dimensions involved in the definition of different operational scenarios.

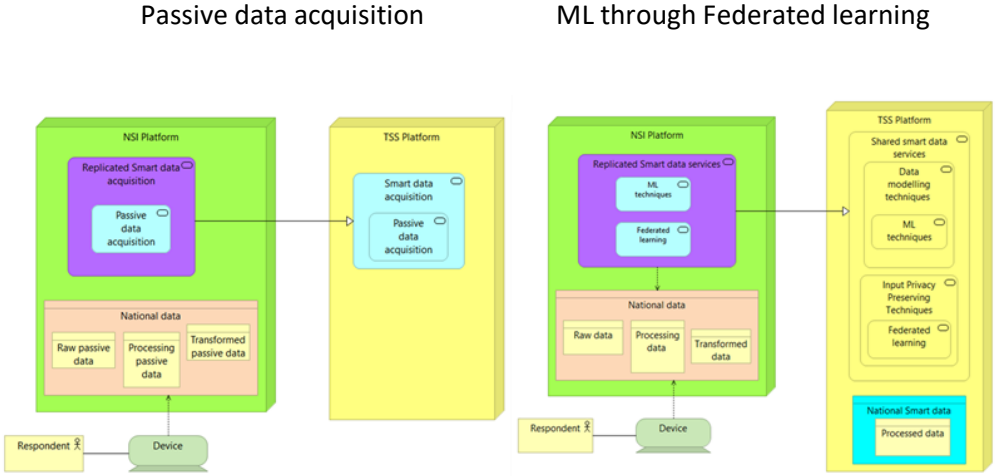
Operational model dimensions	Description
Type of data acquisition	**Passive/Active data acquisition** Application services for smart data gathering could be specialized and managed through replicated or interoperable software solutions, depending on passive or active data acquisition
Type of data provider	**Respondent /Third parties** Service deployment may vary also according to the type of data provider. Depending on the type of data acquisition, shared software solutions could be provided to meet privacy preserving requirements
Data Processing	**In-app/NSI/TSSu Platform /Third parties**

[11] Source: Scannapieco M., Bogdanovits F., Gallois F.; Fischer, Kostadin G., Paulussen R., Quaresma S. et al.: BREAL. Big Data Reference Architecture and Layers. Application layer and Information layer. ESSnet on Big Data II, Work Package F, Deliverable F2 (2021)

	Smart data can be processed in several environments. The application of Input privacy techniques requires the set-up of a distributed environment having specific hardware and software requirements
Data storage	**In-app/NSI/TSSu Platform /Third parties** As a general rule, data should be processed at rest i.e., in the environment where data is stored. In case of data acquisition from third parties, or integration between several stakeholders, Input privacy techniques should be used to guarantee privacy preservation
Service deployment	**Autonoumus/Interoperable/Shared/Replicated** Depending on national skills and capabilities, NSIs may choose to execute the shareable services provided by the TSS_u platform locally, in their own environment through replicated services, or in a common infrastructure. In case of software solutions to be customized at national level, the TSS_u platform may expose interoperable services to be configured according to national requirements. Furthermore, to gather smart data, NSIs may use services provided by third parties.

In order to complement the theoretical assumptions and identify pros and cons of potential operational models, the following use case provides an example of service deployment in compliance with privacy requirements. Supposing that a set of NSIs intend to train a ML model without sharing data, the TSSu platform could offer a distributed environment and a shared service to apply federated learning techniques. The first step of this hypothetical scenario concerns passive data acquisition of sensor data. Once data is stored in-house, by applying federated learning, NSIs can share the ML model parameters instead of national data. The figure below reports the operational model for passive data acquisition and federated learning tasks.

Figure 3: Passive data acquisition e ML model through replicated services

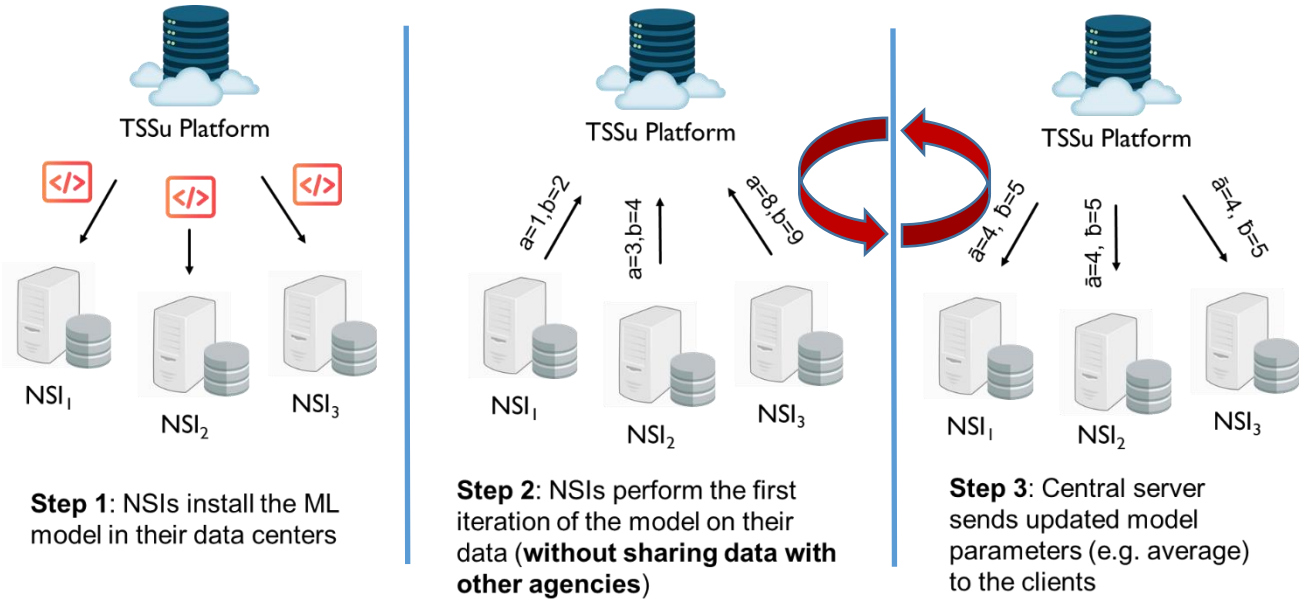


The main steps performed to combine ML training and federated learning can be summarized as follows:

- Step 1: NSIs install the ML model in their data centers through a replicated service available in the TSSu platform
- Step 2: NSIs execute a first iteration of the model without sharing data with other agencies and send the output to the central server
- Step 3: Central server collects ML results from several NSIs and sends the updated model parameters (e.g. average) to the national clients.

This approach, summarised in the following figure, allows to train the ML model on all the datasets available at European level. Each NSI can apply on national data the model trained, thus improving the accuracy of the model.

Figure 4: Main steps of ML training through federated learning techniques



3. Metadata use case: capturing ML metadata

The metadata PoC aims at bridging the gap between the top-down approach adopted to design a Metadata Repository (MR) and the metadata required to achieve full transparency and auditability of ML algorithms applied on sensor data. More in detail, capturing process metadata during the development activities of the ML generalized module has allowed to:

- Describe the main process steps executed
- Track data transformations and Input/output data structures
- Document the methods applied in each step
- Complement data quality assessment with process auditability
- Benchmark the process metadata classified according to the subsets of the proposed MR.

Although the following analysis is focused on smart data miners' perspective, process metadata provide useful information to the final smart data users also, fostering the understanding and the assessment of the activities performed to transform sensor data in statistical output. From this standpoint, the PoC is intended to answer the following questions:

- What type of process metadata is essential to support the survey staff dealing with ML?
- How can metadata be used during smart data processing to improve trust and quality assessment?
- Which process metadata can help final users to understand the statistical results obtained from smart data sources?

3.1 TSS_U metadata design principles

The principles that have guided the design of a metadata component for the TSS_U platform can be summarized as follows:

- Compliance and alignment with official metadata reporting[12] and existing frameworks, to promote the reuse of available metadata and reduce potential overlapping
- Use of active/passive metadata to foster process standardization and the reuse of application components
- Use-case-driven approach, to test the initial assumptions related to metadata concepts selected for the preliminary analysis of the TSS_U framework.

In the early exploration of metadata concepts, the following standards and frameworks have been considered:

- Ontologies
- BREAL Information Architecture
- European structural metadata and quality framework
- GSBPM for the process description
- GSIM concepts.

The metadata component has been conceived to provide relevant information about the following aspects:

- Statistical concepts related to the survey theme and objectives
- Collection instruments, variables and codes
- Statistical methodology used in data processing
- Process and lineage
- Quality assessment.

Based on these general requirements, the dimensions explored more in depth during the metadata PoC concern:

- Sensor data structure
- Process steps specification
- Methods applied for ML training
- Sensor data quality, traceability and trust.

[12]For an overview of the ESS quality system is available, see: https://ec.europa.eu/eurostat/web/quality/quality-monitoring/quality-reporting

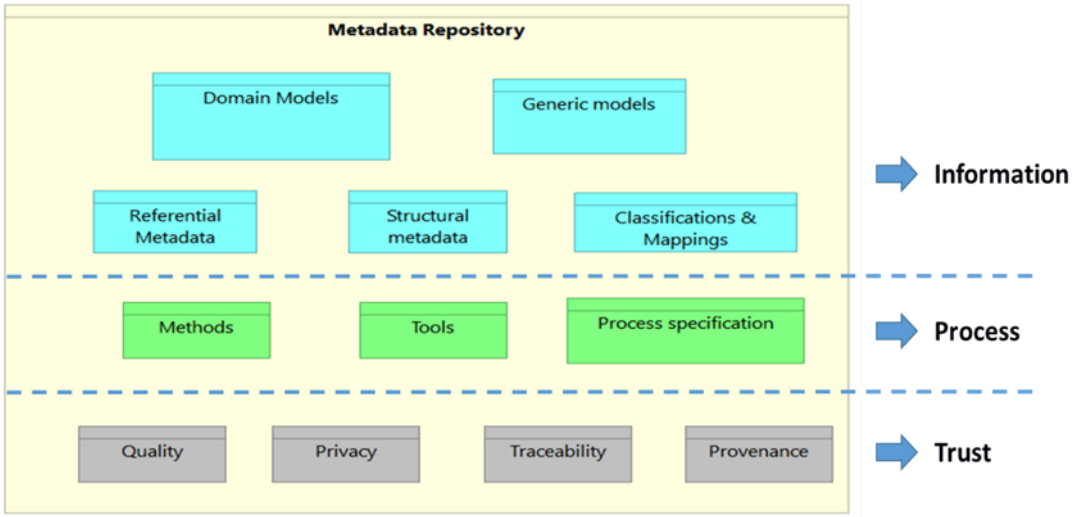
3.2 Metadata repository subsets

Adopting a bottom-up approach, the following Metadata Repository (MR) has been modelled, to meet the general requirements described above and refers to several dimensions, such as:

- Descriptions of data transformations and processing
- Tracking of the main process steps, to support process standardization, control and documentation
- Inventory of tools and methods.

The repository, depicted in the figure below, provides an overview of the metadata concepts, grouped in three main areas: Information, Process and Trust.

Figure 5: Metadata repository subsets



The first subset (Information) includes the information objects related to:

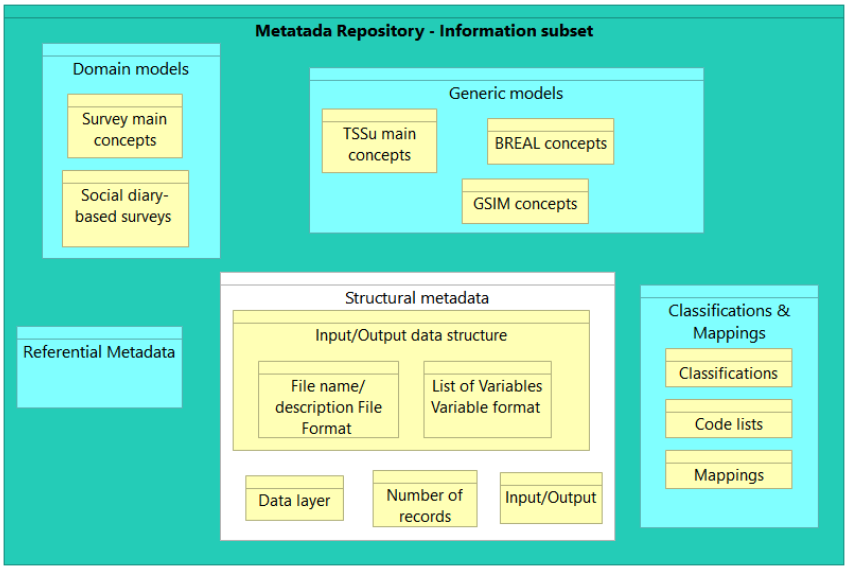
- Domain models, describing survey main concepts and objectives
- Generic models, derived from the reference frameworks (such as, BREAL information model), or providing a common representation of the main concepts describing TSS_U, such as ontologies
- The description of variables, units, data structures, classifications involved in data processing.

Some of these objects, such as Referential metadata or Classifications and mappings, are widely documented and can be modelled according to the European Statistical System (ESS) standards for reference metadata and quality indicators. For this reason, most of the elements belonging to this subset have been excluded from the following analysis, to focus on process metadata produced during sensor data processing.

The second area (Process) concerns all the tasks executed to produce a statistical output. The main elements belonging to this area provide an inventory of the methods and tools used for data processing, as well as a description of the main process steps to perform (Process specification). The third subset (Trust) includes several types of metadata, such as quality indicators and additional information to manage and monitor privacy issues, data provenance and process traceability. The elements grouped in this area are overarching, that is related to each step of the statistical process. In relation to the development of generalized software solutions, while the content of the first group of metadata (Information) is more tied to the survey peculiarities, the elements within the Process and Trust subset can be easily standardized.

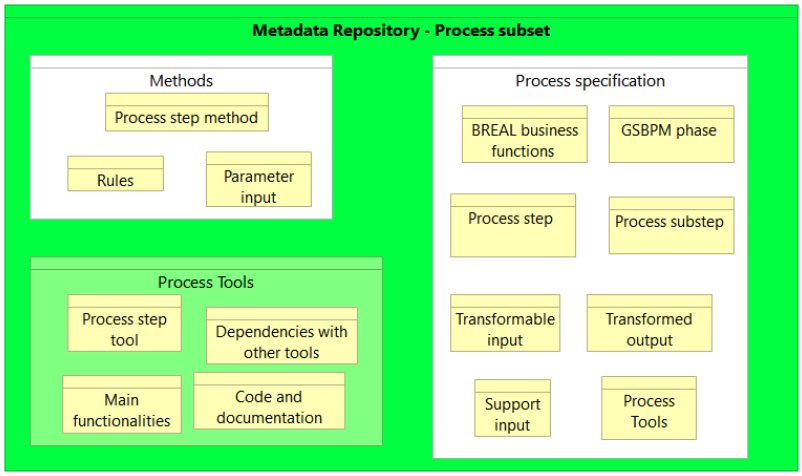
The following use case has been conceived to validate the MR subsets and report the metadata related to smart data processed through ML models. More in detail, this benchmark will result from the analysis of the metadata produced during the activities of Task 3.2, concerning the "Treatment of sensor data for Smart Surveys through Machine Learning Generalized Module." The main goal of this PoC is to prove the feasibility of applying ML algorithms on sensor data, through a generalized and shareable module serving several surveys. In order to identify the metadata to capture during the module execution, the elements of the MR subsets have been further explored. In addition, some of these elements have been defined on a more granular level, according to GSIM and ontologies concepts. The figures below show the elements detailed in each MR subset. The white coloured information objects highlight the elements examined in depth for the ML PoC. More precisely, within the Information subset, the analysis has concerned primarily the Structural metadata, due to its relevance to start any type of data treatment.

Figure 6: Main elements of the Information subset



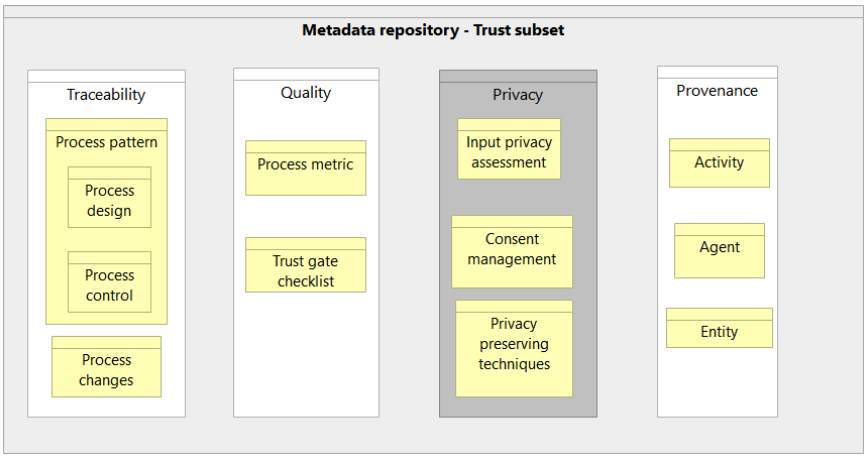
The main elements, described more in detail and belonging to the Process subset, concern the Methods and the Process specification. For each process step, the description of the methods and the related rules and parameters, as well as the specification of data input and output allow to fulfil the requirements concerning process tracking and auditability.

Figure 7: Main elements of the Process subset



The elements composing the Trust subset are detailed in the following figure. This group includes several dimensions related to process auditability, quality assessment, adoption of input privacy techniques and the provenance metadata derived from the main classes of PROV Ontology.

Figure 8: Main elements of the Trust subset



The whole set of metadata, belonging to the different areas and captured for the ML PoC, is described in the next paragraph. As already mentioned, the following analysis is the result of the cooperative work with the colleagues responsible for the ML PoC.

3.3 Capturing ML metadata for smart data miners and final users

During the development of the ML module, processing accelerometer data and based on a domain agnostic pipeline, the metadata PoC has explored the auxiliary information for:

- Documenting process steps, as well as the related data structures and methods
- Tracking data transformations due to the process execution
- Classifying the metadata according to the MR subsets.

The main goal of this analysis is to detail some of the areas composing the MR, to improve the initial conceptual design. The table below describes more in depth for each subset of metadata, the auxiliary

information captured for the ML module. The description of the single elements is complemented by the related reference standard or framework, where available.

Table 1: Description of process metadata captured by the ML module

MR Subset	Component	Subcomponent	Description	Reference standard
Structural metadata	Data structures	Process step	Work package performing a Business Process and having a "large scale" or "small scale" according to the design choices	GSIM
		File name/description	Name and/or general description of processed dataset	
		Number of records	Number of records contained in the file	
		Unit type	Class or group of objects of interest, based on a particular characteristic	GSIM
		File Format	Standard used to encode and store information	
		Variables	Characteristic of a Population to be measured	GSIM
		Variable format	Standard used to represent, read and write variables	
		Data layer	Transformations of smart information objects throughout the statistical process (Raw, Convergence, Statistical)	BREAL
		Input/Output	Specification of the main Input and Output related to the Process Step execution	GSIM
Process	Process specification	BREAL business functions	Set of behaviors concerning the organization of knowledge, resources and skills. BREAL business functions are grouped in two main subsets: Development, Production and Deployment and Support	BREAL
		GSBPM phase	Set of activities describing the main steps of the statistical process. Main GSBPM phases: Specify Needs, Design, Build, Collect, Process, Analyse, Disseminate, Evaluate	GSBPM
		Process sub-step	Composing element of a process step, executing a specific subset of activities	
		Transformable input	Type of Process Input processed and transformed by a Process Step	GSIM
		Transformed output	Result of the process step execution	GSIM

MR Subset	Component	Subcomponent	Description	Reference standard
		Support input	Type of Process Input supporting the Process Step execution, but not changed during data processing	GSIM
		Process Tools	Tools used for the process execution	
	Methods	Process step Method	Description of techniques, statistical methods and algorithms used during step execution	GSIM
		Rules	Mathematical or logical expression to be assessed for defining specific behavior	GSIM
		Parameter Input	Type of Process Input specifying the set-up of a configurable Process Step	GSIM
Trust	Traceability	Process design	Design time of a Process Step executed by a Business service	GSIM
		Process control	Specification of possible paths and decision criteria determining the actions to take after the execution of a Process step	GSIM
		Process changes	Inventory of process steps inconsistencies and main changes introduced to improve process execution and/or output	
	Quality	Process metric	Type of Process Output containing auxiliary information about a Process step execution to provide a quality assessment of the Transformed Output	GSIM
		Trust gate checklist	Checkpoints placed throughout the process to assess the compliance with quality, security and privacy requirements	
	Privacy	Input Assessment	Main privacy issues related to data acquisition, or a specific Process step (e.g. Data collection)	
		Consent Management	Specification of the process steps concerning the management of privacy consent and any following modification throughput the process	
		Input Preserving techniques	Specification of the Privacy Preserving Techniques where applied	
		Activity	Event occurring over a period of time and acting upon or with entities	PROV Ontology

MR Subset	Component	Subcomponent	Description	Reference standard
	Data provenance and lineage	Agent	Specification of the entity responsible for an activity occurring, the existence of another entity, or for the activity of another agent	
		Entity	Any kind of thing (e.g., physical, digital or conceptual) having some fixed aspects	

In the Trust subset, the element "Trust gate checklist" could facilitate data quality monitoring, by identifying for each process step the most relevant quality indicators, as well as relevant auxiliary information concerning privacy, process auditability and security management. The idea of inserting some predetermined checkpoints throughout the process derives from the Quality gates[13], proposed by the Australian Bureau of Statistics (ABS) and mentioned in the Big Data Quality Framework as an efficient measure to improve quality assessment. Similarly to Quality gates, Trust gates could be conceived according to the following principles:

- Placement in the statistical process and measures are specified in advance
- A responsible is appointed as a reference person for quality monitoring
- Tolerance thresholds and actions to undertake whether the measures do not satisfy the tolerance levels are also predetermined
- Assessment of the gate over time, to monitor its effectiveness.

Trust checkpoints could be used for a quick assessment of the tasks performed, especially of the process or sub-process steps that require several iterations and monitoring for achieving expected quality levels. For instance, during data collection, a Trust gate could support the early detection of the impact of missing sensor data on the statistical variable to derive.

The following user stories highlight the relevance of the proposed list of metadata for smart data stakeholders and particularly for:

- "Smart Data miners": in the NSI training a ML model, the suggested list of metadata allows to document data conversion, cleaning and validation performed by "Information Architects". In addition, this information supports "Data Scientists" in smart data exploration and "Domain experts" which cooperate to data modelling and analysis. In this case, captured metadata provides to each staff member relevant information about data processing and quality assessment. Relevant issues are early detected and managed, thus improving the output of each process iteration.
- "Smart Data users": a researcher intends to use the statistical output produced through a ML model. To this aim, process metadata, such as "Data Provenance and lineage" provides auxiliary information about smart data sources, thus enabling a better comprehension of the statistical output and the related process steps.

[13]For further information about Quality gates, see: https://www.abs.gov.au/ausstats/abs@.nsf/Latestproducts/1540.0Main%20Features10Dec%202010?opendocument&tabname=Summary&prodno=1540.0&issue=Dec%202010&num=&view=

Input data processed by the ML module

The first task performed for the development of a generalized ML module has concerned the selection of sensor data to process. The designed pipeline has been tested on sensor data, collected through accelerometer and provided by two main data sources:

- Accelerometer project (WP2.3 pilot 1), testing the use of ActivePAL devices to estimate and classify the physical activity and its intensity, based on the movement of the respondent. The test is composed by two main phases: i) a laboratory session, in which the physical activities performed by participants wearing the ActivPAL device have been monitored; ii) a diary based survey to collect the physical activities performed in a week while wearing ActivePAL. The activities in the lab session have been labelled as follows: cycling-heavy, cycling-light, jump, run, sitting, stairs, stand, and walk.
- The SmartUnitn(Two) project, launched by the University of Trento to analyse the impact of students' daily routine on their academic performance and test the combination of sensor data and questionnaires for data collection, as well as methods to match sensor data and respondents' descriptions. Data have been collected through an app (i-Log app), providing a diary questionnaire (active data collection) and gathering information from mobile sensors prior formal notice and without disturbing the respondent (passive data collection). The questionnaire can be divided in two main subsets, the first one concerning the expectations about the day, and the second one related to time use (some short questions: What are you doing? Where? With? How? Means of Transport?). The survey has involved more than 100 students for four weeks.

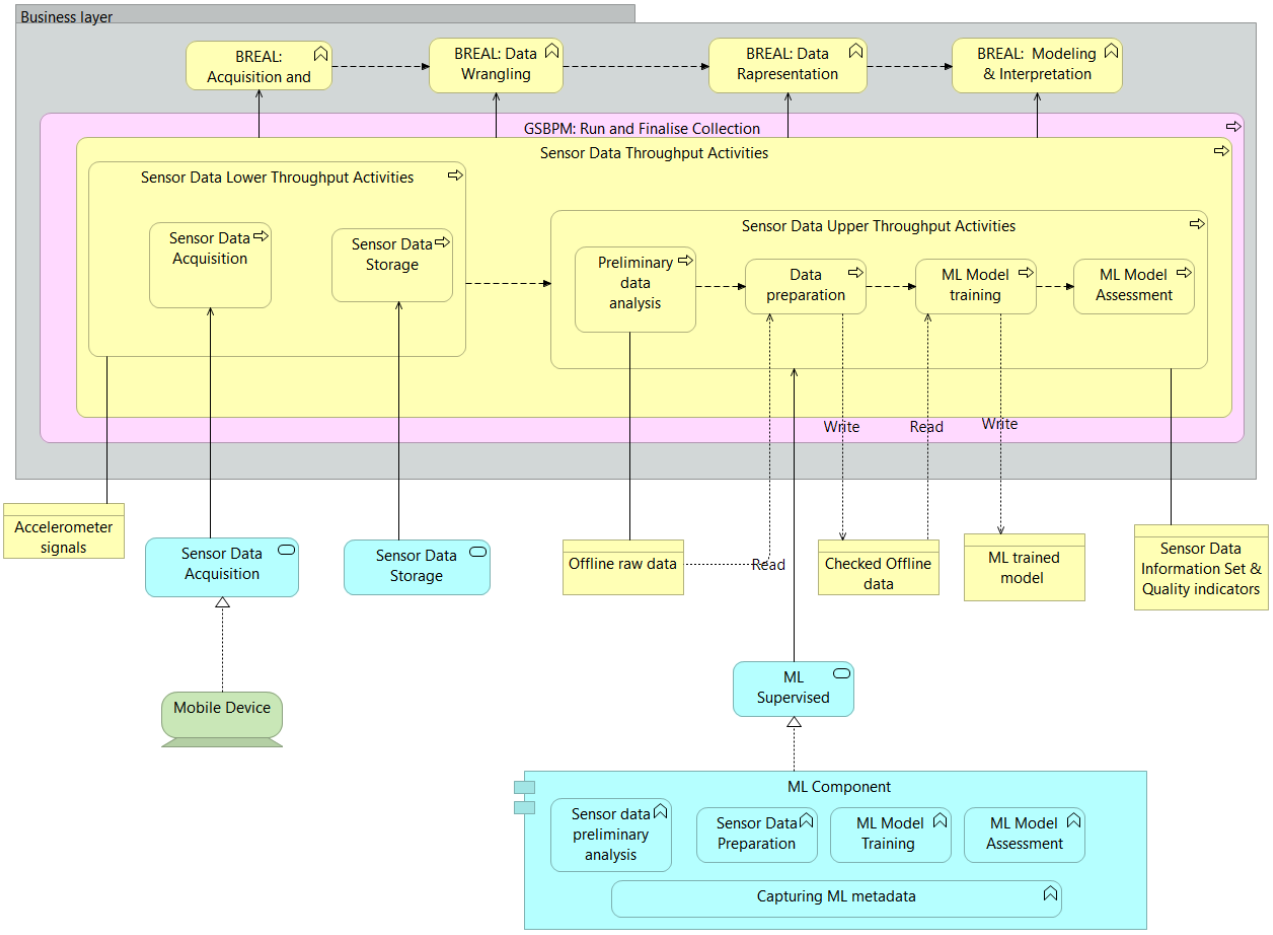
Modelling the ML generic pipeline

The pipeline conceived for the ML module aims at standardizing the main steps of a ML training model, which can be summarized as follows:

- Preliminary data analysis, providing an overview of sensor measurements and diary annotations
- Data preparation, to check data inconsistencies and integrate diary annotations and accelerometer data
- ML model training, to standardise accelerometer data, create the train and test datasets and execute the supervised ML training
- ML model assessment, to analyse and assess the quality of ML output.

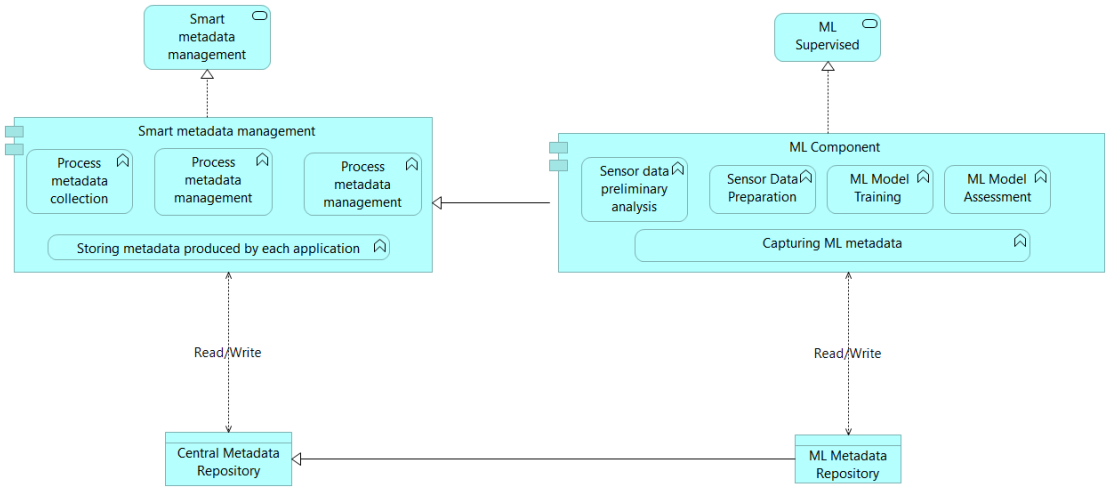
These steps and the related information objects are modelled in the figure below. The ML PoC is based on the assumption that a generalized component, executing a supervised ML algorithm on accelerometer data provided by different data sources, can be used to derive the activities performed in a particular context. This result is achieved by standardising the process steps, so they can be executed at run time on different input data.

Figure 9: Main steps of the ML pipeline



The following figure shows the interaction between the Smart metadata component, described above in the architectural section, and the ML component. The general assumption is that the centralized Smart metadata component should store and manage all the metadata produced during the different stages of data acquisition and processing. Following this design principle, the central Smart metadata component should provide available metadata, captured in the previous process steps, according to the requirements of each application component. At the end of the execution, the application components performing specific process steps may store captured metadata in the central Smart metadata component. Similarly, the latest version of process metadata produced during the ML execution should be saved and stored in the metadata repository, within the Smart metadata component. As depicted in the figure below, the interactive communication between the application components and the Smart metadata component should foster the reuse of available metadata, thus reducing overlapping between different application components using the same process metadata.

Figure 10: Interaction between the Smart metadata component and the ML component



Metadata captured during ML execution

The detailed description of the main steps of the ML pipeline is reported in the following table. This group of metadata included in the Process subset, also provides an overview of the main information objects involved in each step and a summary of the tools used for data processing. Concerning the pipeline standardization, in order to process signals from different devices, asynchronous sampling times are managed through the resampling sub-step.

Table 2: Process specification metadata captured during ML execution

Sensor data Process specification							
BREAL business functions	**GSBPM phase**	**Process step**	**Process sub-step**	**Transformable input**	**Transformed output**	**Support input**	**Process Tools**
Data Wrangling	Analyse	Preliminary data analysis		Offline raw data (Sensor and survey data)	Offline raw data (Sensor and survey data)		Data analysis procedure (Python, Pandas library)
	5.4. Edit and impute	Data preparation	Input data validation (inconcistencies, check and correction)	Offline raw data (Sensor and survey data)	Checked Offline data (Sensor and survey data)		Data validation procedures (Python)
	5.1. Integrate data		Data integration and enrichment	Checked Offline data (Sensor and survey data)	Standardized Sensor/diary format	Survey diary	Data integration procedure (Python)

Sensor data Process specification							
BREAL business functions	**GSBPM phase**	**Process step**	**Process sub-step**	**Transformable input**	**Transformed output**	**Support input**	**Process Tools**
Modelling and Interpretation	Data collection/ Data processing	ML model training	Resampling (signal transformation according to constant intervals) performed for iLog data	Standardized Sensor/diary format	Standardized Sensor/diary format		Data resampling procedure (SciPy Python)
			Segmentation	Standardized Sensor/diary format	Supervised ML dataset		Data segmentation procedure (Python)
			Signal preprocessing	Supervised ML dataset	Preprocessed Supervised ML dataset		Wavelet procedure (Python)
			Splitting training/test	Preprocessed Supervised ML dataset	Supervised ML train/test datasets		ML training procedure (Python)
			ML training	Supervised ML train/test dataset	ML trained model & Quality indicators		
Quality management	Quality management	ML model assessment	Quality assessment	Confusion Matrix, Metrics	Model assessment		ML test procedure (Python)

The Process subset, as shown in Figure 5, also includes the description of the main methods applied for data transformation and processing. The following table summarizes the techniques, statistical methods and algorithms launched in each process step, as well as the related rules and input parameters.

Table 3: Metadata captured during ML execution describing the methods applied

Methods				
Process step	**Process sub-step**	**Process Method**	**Rules**	**Parameter Input**
Preliminary data analysis		Analysis of sampling frequencies, number of missing values, analysis of		

Methods				
Process step	**Process sub-step**	**Process Method**	**Rules**	**Parameter Input**
		frequency distribution		
Data preparation	Input data validation (inconcistencies, check and correction)	Checked activity date and time provided in the diary by the respondant (survey data)		
	Data integration and enrichment	Sensor and survey data linkage based on activity time and respondent id (Deterministic record linkage) to create a labelled signal (assuming the activity as label)	Deterministic rules	Name of the variables for record linkage
ML model training	Segmentation	Probabilistic approach to extract segments of labeled signals with the same duration	Probabilistic rules	Lenght of segment
	Signal preprocessing	Wavelet algorithm to extract the main features from sensor signal	Deterministic rules	Wavelet of family
	Splitting training/test	Data are randomly splitted in two subsets: the training subset and the test subset	Constraint: the trials (combination of activity and signal) must be balanced for each activity label	Number of trials for train, number of trials for test
	ML training	Training model based on the CNN (Convolutional	Deterministic rules	Max number of epochs, learning rate, optimization

Methods				
Process step	**Process sub-step**	**Process Method**	**Rules**	**Parameter Input**
		Neural Network) classifier		algorithm, validation split
ML model assessment	Quality assessment	Process metrics computed: confusion matrix and the following indicators: accuracy, precision, recall and F1 score		

The following group of structural metadata describes the data objects transformed in each process step, in terms of file format, variables, and other essential characteristics to extract, upload and process data. In relation to the central MR, this subset of metadata belongs to the Information area. The table below reports the list of structural metadata captured for ActivePal.

Table 4: Structural metadata captured during ML execution

Structural metadata								
Process step	**File name**	**Description**	**Number of records**	**Unit type**	**File Format**	**Variables**	**Data layer**	**Input/Output**
ActivePal								
Preliminary data analysis	Offline data raw (Diary data)	Diary annotations gathered during laboratory activities to be analysed	39	Respondents	xls	ID Respondent, date, time, list of activities	Raw	Input/Output
	Offline data raw (Sensor data)	Accelerometer measurements collected through ActivePal to be analysed	Number of records varying for each respondent depending on the activities executed in a week	Accelerometer sampling measurement	datx	Spatial components of the accelerator vector (ax, ay, az)	Raw	Input/Output
Data preparation	Offline data raw (Diary data)	Diary annotations gathered during laboratory activities to be checked and validated	39	Respondents	xls	ID Respondent, date, time, list of activities	Raw	Input
	Offline data raw (Sensor data)	Accelerometer measurements collected through ActivePal to be to be checked and validated	Number of records varying for each respondent depending on the activities executed in a week	Accelerometer sampling measurement	datx	Spatial components of the accelerator vector (ax, ay, az)	Raw	Input
	Checked Offline data raw (Diary data)	Consistent Diary annotations gathered during laboratory activities	39	Respondents	xls	ID Respondent, date, time, list of activities	Raw	Input/Output

Structural metadata								
Process step	**File name**	**Description**	**Number of records**	**Unit type**	**File Format**	**Variables**	**Data layer**	**Input/Output**
	Checked Offline data raw (Sensor data)	Consistent Accelerometer measurements collected through ActivePal	Number of records varying for each respondent depending on the activities executed in a week	Accelerometer sampling measurement	datx	Spatial components of the accelerator vector (ax, ay, az)	Raw	Input/Output
	Standardized Sensor/diary format	Integration between diary annotations and accelerometer data	974247	Records for 27 Respondents	Pandas DataFrame	ID Respondent, date, time, list of activities, Spatial components of the accelerator vector (ax, ay, az)	Convergence	Output
ML model training	Standardized Sensor/diary format	Integration between diary annotations and accelerometer data	974247	Records for 27 Respondents	Pandas DataFrame	ID Respondent, date, time, list of activities, Spatial components of the accelerator vector (ax, ay, az)	Convergence	Input
	Supervised ML dataset	Clusters of segments of labeled signals with the same duration	15000	Signal segment	Numpy array for data Input and for the target variable	3-dimentional Tensor wavelet Frequency wavelet Time for Input data 1-dimensional Array for the target variable Segment index	Convergence	Input/Output
	Preprocessed Supervised ML dataset	Wavelet transformation to reduce variance within the clusters of segments	15000	Imagine like signal segment	Numpy array for data Input and for the target variable	3-dimentional Tensor wavelet Frequency wavelet Time for Input data 1-dimensional Array for the target variable Segment index	Convergence	Input

Structural metadata								
Process step	**File name**	**Description**	**Number of records**	**Unit type**	**File Format**	**Variables**	**Data layer**	**Input/Output**
	Supervised ML train dataset	Data randomly splitted in training and test subsets	10000	Imagine like signal segment	Numpy array for data Input and for the target variable	3-dimentional Tensor wavelet Frequency wavelet Time for Input data 1-dimensional Array for the target variable Segment index	Convergence	Input/Output
	Supervised ML test dataset	Data randomly splitted in training and test subsets	5000	Imagine like signal segment	Numpy array for Input data and for the target variable	3-dimentional Tensor wavelet Frequency wavelet Time for Input data 1-dimensional Array for the target variable Segment index	Convergence	Input/Output
	ML trained model	Output of ML training	24000000	model parameters	hdf5	CNN weights	Statistical	Output
	MLQuality indicators	Output of ML training		accuracy, confution matrix, precision, recall	csv		Statistical	Input/Output
ML model assessment	Confusion Matrix, Metrics	ML quality indicators to analyse for model assessment		accuracy, confution matrix, precision, recall	csv		Statistical	Input/Output

The metadata grouped in the Trust section have been analysed with respect to process traceability, quality assessment and data provenance, omitting the metadata related to Input privacy techniques, not implemented in the ML pipeline. The following table reports the metadata captured for tracking the process steps and providing a quality assessment. All the indicators computed to evaluate the output produced in each step should be included in the list of checkpoints assessing the compliance of the results with the initial quality, security and privacy requirements.

Table 5: Process traceability and quality metadata captured during ML execution

Trust					
Traceability				**Quality**	
Process design		**Process control**	**Process changes**	**Process metric**	**Trust gate checklist**
Process steps	**Process sub-steps**				
Preliminary data analysis		% of consistent records>= predetermined threshold then execute the following step		% of consistent records	√
Data preparation	Input data validation (inconcistencies, check and correction)	% of validated records>= predetermined threshold then execute the following step		% of validated records	√
	Data integration and enrichment	% of matched records>= predetermined threshold then execute the following step		% of matched records	√
ML model training	Segmentation	Number of extracted samples compliant with the signal length		% of repeated segments	√
	Signal preprocessing				
	Splitting training/test				

	ML training	Loss function measured on validation set and training set		Consistency between the loss function measured on validation set and loss function measured on training set	√
ML model assessment	Quality assessment	Confusion Matrix		Value of precision, recall, accuracy	√

In addition to quality assessment and process tracking, the Trust subset also reports the metadata describing privacy issues and data provenance. The table below provides an overview of the metadata related to privacy and data provenance captured during ML

Table 6: Privacy and data provenance metadata captured during ML execution

Trust						
Process step	**Privacy**			**Data provenance**		
	Input Assessment	**Consent Management**	**Input Preserving techniques**	**Activity**	**Agent**	**Entity**
Preliminary data analysis	Not applied	Not applied	Not applied	Acquisition and preliminary analysis of diary annotations and accelerometer data	ML module	ML application component
Data preparation				Accelerometer data editing and linkage with survey data		
ML model training				ML modelling and interpretation		
ML model assessment				Quality assessment of ML output		

4. Main results and open issues

The main result achieved through the PoCs is the design of a reference framework, based on official statistics, to guide the future developments of the TSSu platform. More in detail, the architectural PoC has delivered an overview of:

- The platform business layer, and particularly the core activities to perform for smart data collection and processing
- The platform application layer, to define the application services required for the execution of the process steps performing the main steps of a TSSu
- The different scenarios related to data storage and processing, resulting from the combination of several environments: platform premises, and/ or national infrastructures, and/ or the cloud
- The relationships between methods, privacy issues, data quality assessment and process tracking, to be considered for delivering the enhanced framework, the final deliverable of WP3 at the end of the project, based on PoC results and WP2 feedbacks.

The metadata PoC has allowed to achieve the following outcomes:

- Description of the main process steps performed
- Data transformations tracking
- Documentation of the methods applied in each step
- Combination of data quality assessment and process auditability
- Benchmark of the subsets of the Metadata Repository modelled in the previous deliverable.

Comparing the initial assumptions with the achieved results, the architectural PoC has shown the feasibility of process standardization and integration between traditional and smart survey tasks. This analysis is essential to specify more in detail the technical requirements of the TSSu platform, and start the development activities. The metadata PoC has confirmed the feasibility of a central repository for the collection and management of the process metadata produced throughout a particular task. The metadata subsets within the central repository allow to monitor a process step during its execution, providing a quick assessment of the results and fostering the adoption of a metadata driven approach.

In addition to the above results, the main lessons learned from the PoCs can be summarized as follows:

- The pipeline for collecting and processing smart data sources needs to be customized according to the type of sensors used for data gathering
- Development of interoperable services through the adoption of official statistical standards, such as GSIM, to harmonize data and metadata models
- The environment required for applying Input privacy techniques has a relevant impact on the design of the TSSu platform and further exploration is needed to assess the pros and cons and identify the best technical solution.

The main open issues concern the following aspects:

- Wp2 field tests should be used to benchmark the architectural and metadata frameworks resulting from the PoCs, to complement the analysis with the process steps performed, as well as the service functionalities and the methods actually implemented. Even if the

proposed business layer for a generic TSSu results from WP2 platform specifications and pilots, bridging the gap between WP2 and WP3 approaches requires further abstraction of WP2 tasks and more granularity of the architectural and metadata models

- In order to integrate smart data pipeline and traditional survey tasks, some aspects, such as logistics and data collection monitoring, need to be adjusted according to the type of data sources
- Although the subsets proposed for smart metadata gathering and management foster process tracking and auditability, the development of metadata-driven application components requires a detailed analysis of the functionalities provided by each solution. Even if the metadata captured throughout the ML PoC have confirmed the feasibility of process metadata standardisation, unexpected peculiarities of some tasks may highlight the need of further adjustments of the conceptual framework described in the previous paragraphs.

As a whole, the PoCs have succeeded in underlining the several issues impacting on the technical requirements of the TSSu platform. Nevertheless, further analysis is essential to increase the evidence based insights and improve the achieved outcome. The feedbacks provided by the pilot surveys will allow to improve the reference framework for the TSSu platform and guide future development activities.

References

- A Suggested Framework for the Quality of Big Data. Deliverables of the UNECE Big Data Quality Task Team, (December, 2014). Available from: https://statswiki.unece.org/display/bigdata/2014+Project?preview=%2F108102944%2F108298642%2FBig+Data+Quality+Framework+-+final-+Jan08-2015.pdf
- Aracri, Bruno et al.: Task 3.1.3 - Integration in existing architectural framework auditability - Deliverable 3.1 Report on the Preliminary Framework, available from: https://ec.europa.eu/eurostat/cros/content/wp-3-conceptual-framework-european-platform_en
- Aracri, Bruno et al.: Task 3.1.6: Metadata and Process auditability - Deliverable 3.1 Report on the Preliminary Framework, available from: https://ec.europa.eu/eurostat/cros/content/wp-3-conceptual-framework-european-platform_en
- ESS Enterprise Architecture Reference Framework (EARF), September (2015). Available from: https://ec.europa.eu/eurostat/cros/content/ess-enterprise-architecture-reference-framework_en
  10
- Generic Statistical Business Process Model (GSBPM) v. 5.1. January (2019). Available from: https://statswiki.unece.org/display/GSBPM/GSBPM+v5.1
- Generic Statistical Information Model (GSIM) v. 1.2 March (2021). Available from: https://statswiki.unece.org/display/gsim/GSIM+v1.2+documents
- Scannapieco M., Bogdanovits F., Gallois F.; Fischer, Kostadin G., Paulussen R., Quaresma S. et al.: BREAL. Big Data Reference Architecture and Layers. Application layer and Information layer. ESSnet on Big Data II, Work Package F, Deliverable F2 (2021). Available from: https://ec.europa.eu/eurostat/cros/system/files/wpf_deliverable_f2_breal_big_data_reference_architecture_and_layers_application_layer_and_information_layer_31_03_2021_final.pdf

- Scannapieco M., Bogdanovits F., Gallois F.; Fischer, Paulussen R., Quaresma S. et al.: BREAL. Big Data Reference Architecture and Layers. Business layer. ESSnet on Big Data II, Work Package F, Deliverable F1. (2018-2021). Available from: https://ec.europa.eu/eurostat/cros/sites/crosportal/files/WPF_Deliverable_F1_BREAL_Big_Data_REference_Architecture_and_Layers_v.03012020.pdf

Task 3.2
Sub-tasks 3.2.5

Integrating of incentive schemes into the platform: PoC on Incentive Schemes

Prepared by:

Nils Meise (Destatis, Germany)

Task leader: Nils Meise - Destatis

Contents

Executive Summary

The activity of task 3.2.5 explores incentives schemes in a Proof of Concept to understand how gamification could be utilized for a gamified (smart) surveys. It asks which steps are required to design a gamified survey by looking at different survey phases and use patterns by understand these phases as the respondents or player journey respectively. In addition, prospective respondents are handled with an abstract concept of user types. The main result of the PoC is that the implementation of gamification results in a trade-off: It eases the hard task of completing a longer running survey for respondents and at the same time requires more and new forms development by a survey agency. These additional development activities are mainly due to the different mode of app-bases surveys and design requirements that result from that decision. In particular, the on-boarding phase requires a well thought through implementation, because all apps suffer from quick denial by users, if they are not able to catch the user's attention and implement successful strategies for return. For example, notifications are not just reminders, but also a way to deliver meaningful feedback that nudges users to return. The PoC puts these decisions into a context and delivers design guidelines.

The report consists of the following parts: First, an *introduction* that gives an insight why gamification was chosen and under what assumptions the PoC was conducted. Second, which assumptions are useful when designing a gamified survey for a specific group of respondents (*2. Desired User Types*). Third, the actual design of a gamified survey following design principles for apps and gamified apps in particular (*3. User Journey*). Finally, a conclusion discusses the results and identifies open issues for further investigation (*4. Results*).

1. Introduction

This Proof of Concept (PoC) models how an incentive in form of gamification can be included into a Trusted Smart Survey (TSSu). It provides a conceptual overview on gamification for surveys that rely on active participation and applies it to process based model. The intention is to highlight which gamified elements are suitable for an implementation in general and how they can be used in conjunction with a targeted focus on respondents.

Gamification is not a new mechanic for interactive settings that require people to keep engaged in an activity (Chou 2016; Goethe 2019; Marczewski 2018). Also, some research on gamified surveys has already been done (Triantoro et al. 2020; Harms et al. 2015) and suggest beneficial results for response rate and data quality. The same research points out that a gamified survey is highly tailored to allow game like mechanics, which not only changes the look and feel of the survey, but at the same time asks for a higher investment in regard to design, methodological soundness, and overall more time for survey development. As always, it is a trade-off to achieve a set goal. In our case we suppose that drop-outs during longer running surveys are problematic and almost all other incentives are given before or after a survey (c.f. table 1).

Incentive	before	during	after
monetary incentive	X		X
coupon or discount	X		X
free sample	(X)	X	X
give-aways	X		X

charitable donation			X
raffle			X
access to survey results			X
gamification		X	

Table 3: Incentive Timing

Thus, we are suggesting a simplified gamification process model for use with Trusted Smart Surveys. It is simplified to allow for an easier adaption for various surveys that require respondents to deliver information over a long period of time. Although our approach to gamification is simplified it does still require extra work for implementation and due to its process-based structure only fits longer running surveys, which risk drop outs over time. To achieve this goal, the process model is applied to a mock-up survey application that requires manual input from a respondent. The level of detail is kept to conceptual features and how they could introduce game mechanics into surveys. Our particular focus is on immediate feedback during the survey period. We further assume that our gamified survey is re-enforced with a monetary incentive that is linked to survey process, overall completion and pay-out after full completion. It could also be worthwhile to consider to offer an option to donate the accumulated monetary incentive to a charity (of choice by the survey agency or list of charities curated by the survey agency). Adding different angles will, on one hand, offer more options to design experiences for different types of respondents, and on the other hand risk unintentional effects, if the experiences do not comply with the expectations of our respondents.

Motivation and engagement likely differ among respondents and may change over time. In an ideal situation we would have in-depth profiles on our respondents to target them individually with our game mechanics. However, broad profile categories – called user types – are common in game design and allow to think of mechanics that would fit certain play styles. We do the same here and also focus on desired user types that can be tied to mechanics, which help data completeness, data quality and promote citizen science.

Something new is often exciting, but tends to lose its shiny new shine quickly. To really engage with an activity, it needs to evolve over time. This is where different phases of interaction with the respondent come into play. These phases mark the player's journey through the gamified experience – or in our case the respondent's journey through the survey. Any new or unfamiliar activity needs to be explained, any game has rules to be learned and any survey has guidelines how to answer its questions. The journey starts with an on-boarding, which familiarizes a respondent with the survey application and already offers an opportunity to individualize the survey experience, which we will lay out in more detail later on. What gamification puts on top is building a habit of interaction and an urge to master the game – hence completing the survey.

First, there is a short recap on what kind of *gamification* could be suitable for surveys. Second, *desired user types* for gamified surveys are laid out and how considering them is beneficial to make respondents stick to the survey. Third, the *player's journey* frames the survey experience for respondents through three distinctive phases of their interaction with the survey. Finally, *results* are presented to offer guidelines for best practise implementation of gamified surveys or gamified elements into surveys.

2. Desired User Types

There are many ways to play a game. Trying to win a game is actually just a niche motivation to get involved in game. Successful games offer a way of engagement for a broad audience that allows for shifts in the way a game is played and enjoyed. Our model building showed that three user types align with the goals of

Trusted Smart Surveys.[14] The *philanthropist*, the *achiever* and the *player*. This does not mean that these are the only user types one could think of or the only ones that actually fit Trusted Smart Surveys. They are a suggestion for a simplified view on a target audience for a gamified survey. We will argue that they allow just enough variety to include multiple game mechanics for the targeted respondents.

In general, user types are a way to deal with uncertainty towards a target audience of a (digital) product. User types allow to narrow the scope of a product for different – even opposite – users. In a best-case scenario there are diverse mechanics that take these differences into account. However, such a strategy could also fail, if, for example, disruptive behaviour is possible, but not in line with the designer's visions or community standards. Also, not all user types are equally found in a population of users, even when considering that user types are no fixed category for a person. Hence, our goal is to make a choice right from the start and select user types that are a good fit for simple game mechanics – like *achievers* and *players* – or fit in the context of citizen science – like *philanthropists* – that Trusted Smart Surveys tries to promote.

Figure 8: User Types

Achiever

Achievers are motivated by complete mastery off all challenges thrown at them and tokens that show their accomplishment are not sufficient, though welcome (Marczewski 2018). Completion of tedious daily tasks are no problem for this user type, if the setting is right for them. Achievers are helpful to avoid missing data. They need to be provided with challenges, a certificate or trophy for their extraordinary efforts. Progression systems that show levels of expertise or overall completion also motivate this user type.

Philanthropist

Philanthropists are motivated by purpose, they want to give to other people and expect no reward (Marczewski 2018). Abstract goals of civic research, bettering society, and a general call for volunteering in surveys, which aim at giving dependable information for informed political decisions, may resonate with this type. It is about creating an experience that one is part of something greater than oneself and creating a desire to continue to keep contributing.

Player

Players are in the game for the reward (Marczewski 2018). For example, they participate in a gamified to get the maximum rewards for participation and will complete all required tasks to achieve this goal. In general, any ranking system, high-scores and outlook for rewards for participation will resonate with this type. Awarded points work both as a feedback mechanism for desired behaviour and in combination with leader boards for a competitive element amongst the players. Exchanging points for "in game items" through virtual economy mechanics also motivate this user type to keep on playing.

[14] The user types, which are discussed in this chapter, are taken from Marczewski (2018), and are explained in more detail in the first deliverable of this report.

3. User Journey

To design the user's journey through the survey we design it in three different phases. First, the *on-boarding* that is a short introductory phase that is crucial for avoiding early drop outs. Second, *immersion* the primary phase during which the user is asked to repeatedly deliver information and should embrace the activity and not be bored or repelled by repetitive action. Finally, the *completion* phase, which is meant to offer new challenges for respondent to re-immerse into the activity. Usually, such a phase would be designed to hold a user as long as possible to continually "sell" the offered services. However, surveys do not run for eternity, which in our case is a good thing, because it offers a simple "goal" or natural end of a game, which again is quite fitting for a gamified activity and also avoids negative external effects (e.g. addiction).

On-Boarding

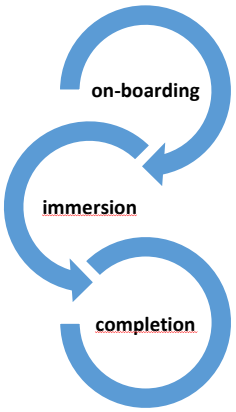


Figure 9: User Journey

The first time a user interacts with an application its functionality is unfamiliar. A typical on-boarding process will show how to interact with an application or even offer a short tutorial that involves hands-on exercise. In a more general sense, it is a tour that shows what an app has to offer to a user (Gazdecki 2018): An on-boarding could focus on the benefits of using the app (the value to the user), but no introduction to the functionality. It could also focus on the functionality of an app and give instructions on how to use it. Or it could give a full walkthrough through all screens in progressive way that requires the user to interact with each screen to learn about the functionality. During an on-boarding users are asked for permissions required by an app and for additional opt-ins like push notifications. Market studies show that one out of five users will not launch an app more than once and that opt-in for push notifications has a huge impact on using an app multiple times (Upland Software no year).

A gamified on-boarding process has the same intention of keeping users engaged and motivated to launch the survey app again. Since we already established that not every user is the same, we also have to add variety to the on-boarding process. We want to learn about our user and at the same time give her/ him the feeling that the experience individualized. For our example survey app, we can think of the following on-boarding process, which is shown in figure 1. After the initial start-up a welcoming page greets the respondent. At the same time our app will ask for consent for notifications and alerts. The next step is a short on-boarding questionnaire that is themed to ask the respondent who they are in terms of three archetypes represented by animals and their "personality." The idea is to match these on our desired user types without revealing them directly to the respondent. As a result, a suggestion to select an avatar (based on the user type) is presented to the respondent. A first variation is possible here to not follow the suggestion and explore the other avatars, which will trigger a secret token that will be relevant later in the process. Both decisions allow to explore the other avatars before making a final decision. Exploring all avatar will award a badge to symbolic pay for the invested time. If the secret token was triggered, also a special title for the avatar will be available to award curiosity. The avatar selection screen will list all explored avatars and will make the achieved title automatically available as well. The final step will trigger a pay-out on the in-app account.

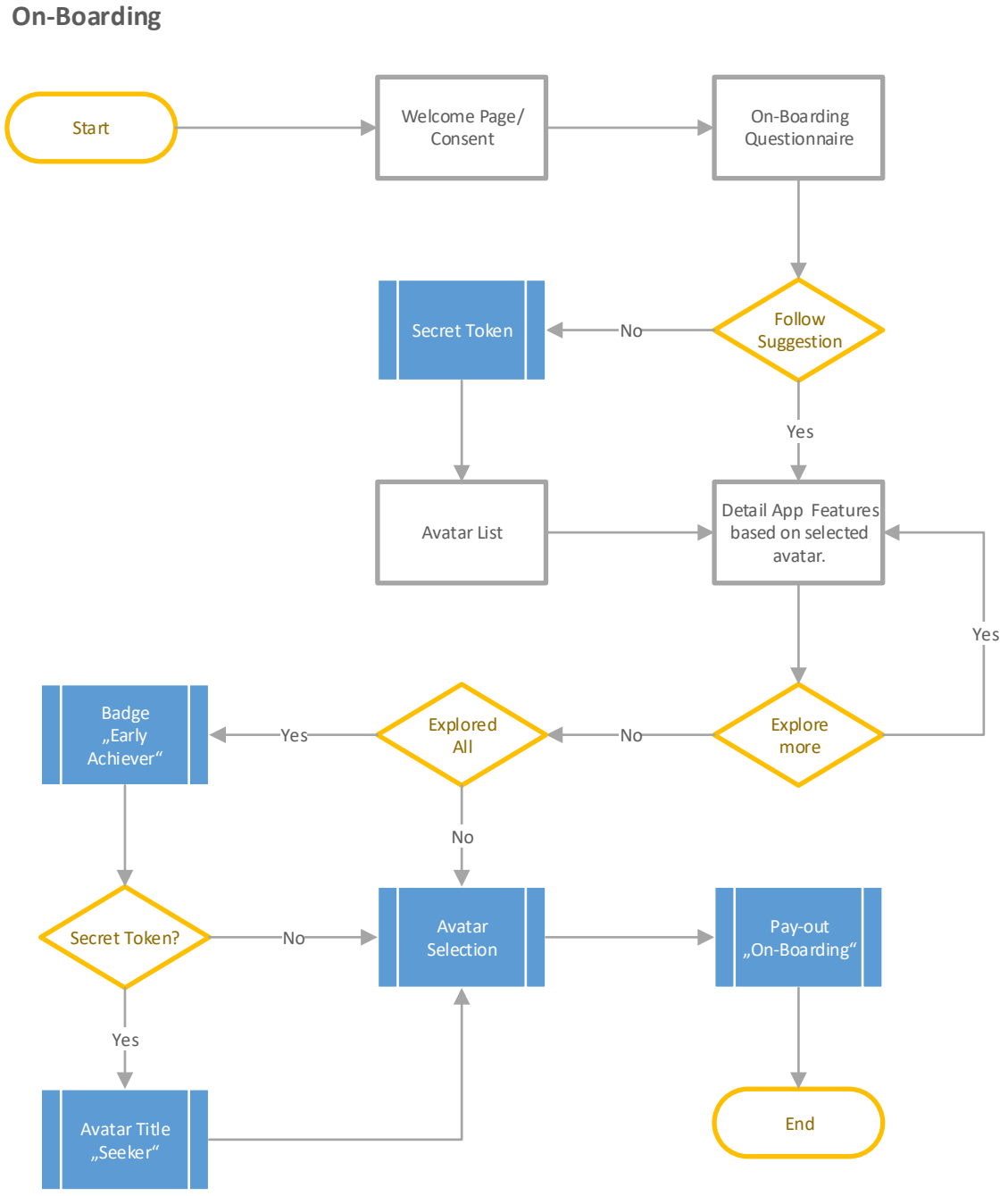


Figure 10: On-Boarding

Immersion

At this point our respondents have discovered our survey app (download and installed it) and completed the on-boarding. This part of their journey builds on immersion and flow. We will achieve this by providing meaningful feedback based on their actions. First of all, we need to answer ourselves a question: What behaviour or activities do we want to encourage? Basically, all activities that work towards completeness of data. We want our respondents to go to our survey app on a regular basis and complete relevant questionnaires on time. Basically, we need to build motivation and ideally *transform motivation into immersion*.

Motivational designs build on *relevant* feedback, which is given *in time* and *meaningful* to the respondent. For our mock-up survey app, we decided to use avatars that are initially tied to user types. So, we have a starting point on how to frame initial notifications for our respondents that inform them that action is necessary. A simple message that notifies that a step was completed is already a feedback mechanism that shows that actions do matter. Combining completed actions with points already allow multi-faceted feedbacks. Easy activities should be tied to few points and harder ones with more points that count towards progression systems in regard to survey mastery and increase the monetary incentive.

Achievers and *players* are rather easy to target with reward mechanics based on individual performance. *Philanthropists* enjoy sharing and helping, which at first seems to be difficult to implement. Completing challenges could involve access to further information on how data from these surveys is used in the statistical system and how it is relevant for political decision making. It shows that participating means having in influence in how future policies are shaped. Also helping a cause directly could be implemented by offering the option to donate earned points towards a common pool that will be transformed in a monetary donation to charity. In addition, for our mock-up survey application we assume that the survey is done on a household level with multiple participants from said household. This allows to design a role that could take over care-taking options as a household manager and aims towards the social dimension of *philanthropists*. A summary of possible options is given in table 2 below.

User Type	**Gamified Incentive Feedback Mechanisms**
Achiever	- Badges awarded for achievements - Secret achievements that award unique badges or titles - (Daily) streaks - Progression systems - Issuing certificate for survey participation and completion
Philanthropist	- Access to knowledge base on how data will be used within the statistical system. - Donating earned points (monetary incentive) to a charity. - Extra access and care-taking options as a household manager to review progress and reward earned points to household members.
Player	- Progression systems that earn ranks - In-game rewards based on ranks that earn titles (virtual economy) - Leader boards that rank most active respondents by avatar - Earning points that increase the monetary incentive

Table 4: Feedback

Completion

Winning games and completing (long) surveys share a similarity: both require to engagement in the activity. As we know, not everyone can win a game and not everyone does complete a survey. While participating in games can be already sufficient to create an enjoyable activity, quitting a (voluntary) survey has no consequence for the respondent. However, its effect is a lower response rate, less population coverage and ultimately a potential negative impact on survey quality. In short, we want all our respondents to reach the finish line. In order to do that we need to avoid the two main enemies of our desired immerse flow state: frustration and boredom. The challenge of completing a longer running survey is quite huge. So, it needs to be broken down in smaller parts that form challenges of their own.

For our mock-up application we could think of a demand to deliver data on a daily basis. This is quite a repetitive task, if it requires the *same* activity every single day. If we demand very detailed information with long questionnaires every day, the challenge of completing it could lead to frustration. Likewise, if we demand very little information and it seems too easy to deliver, it could get boring quickly, as there is no real challenge to master. Challenges are meant to be mastered and there need to be an outlook towards a reward as a step towards the next challenge. By this a continued cycle is formed as shown in figure 2.

Again, we can use game mechanics and our desired user types to find a middle ground. From the perspective of gamification, we need to provide challenges that get progressively harder to complete and offer rewards for completion. For *achievers* it is challenging not just to complete a required daily task, but also optional tasks, when they reward additional progression by going above and beyond the minimal requirements. For *philanthropists* the outlook of providing additional much needed information, which will help the cause even more, could be appealing. And for *players* mechanics like complete three additional tasks on three consecutive days to earn extra points towards the monetary incentive could be interesting. In general, the communication should be focussed on the *outlook* to a reward. In case of our mock-app survey application that means that additional notifications – based on the selected avatar – can communicate these rewards. Notifications are meant to be more than a reminder. Our design goal is that the respondents should not need a reminder to do a task, but challenged to work (or play) towards the completion of their journey.

Figure 11: User Flow

4. Results

The main result of the PoC is that gamified surveys transform the hard task of completing a longer running survey into smaller achievable goals to pave the way to survey completion. Their design becomes similar to other applications and follows their usage patterns, which are characterized by different phases from the on-boarding to a regular use, if successful, or immediate uninstall, if they fail to appeal to the user. Thus, design decisions for (gamified) app-based surveys are not solely methodological, but highly depending on expected user experience. These design decisions are crucial in regard to the goals of this activity. First, to create an incentive for a respondent to stay engaged in a survey from the start and over a longer survey period. And second, to enhance data quality with the help of the respondent.

The investigate these issues the PoC was conducted with the following assumptions. That it should focus on the conceptual implementation of gamification for smart surveys and not on respondent's behaviour or survey results; it should assume that a mobile or web-based survey app is used to provide an interactive user interface for the respondent; it should have surveys in mind that suffer from drop-out over time or low return rates; and it should use game mechanics to include other incentives (e.g. monetary incentives).

In sum, the particular outcomes – beside the main result – are that gamified smart surveys rely on mechanics that offer immersion into the activity. That they must consist of different phases and offer new challenges over time to allow for continuous immersion. In addition, they rely on in-depth knowledge or abstract assumptions of the target audience and need to fit the respondent's shifting expectations by offering relevant feedback. It was also explored how other incentives could be included via a game-based economy.

There are several open issues remaining. They will be addressed and evaluated in the report on the improved framework for the European Platform. Open issues to be addressed fall in three categories: methodological, technical and general. From a methodological perspective, the actual impact on particular surveys and the effort for implementation have to be balanced. From a technical perspective, decisions for versatility and re-usability have to be investigated to reduce the burden of developmental work for each survey. More general open issues are how gamified surveys have an impact on other design decisions for Trusted Smart Surveys on a European level or for a particular NSI. In particular, methodological open issues are the assessment of the impact on respondent's behaviour and investigations what particular surveys could benefit from gamification. Technical open issues are, missing guidelines on best practices for technical implementation (issue is partly mitigated due to the fact that gamification is mostly a methodological design question) and the technical design of game mechanics that are not broken or easily misused to produce false (survey) data. Finally, general open issues are the impact on technical Trusted Smart Survey architecture and the impact on European Platform in terms of versatility in the case of different national approaches to incentives and incentive strategies.

References

Chou, Yu-Kai (2016): Actionable gamification. Beyond points, badges, and leaderboards. Fremont, CA: Octalysis Media.

Gazdecki, Andrew (2018): The Importance of Onboarding Users in Your App. Edited by Business 2 Community. Available online at https://www.business2community.com/mobile-apps/the-importance-of-onboarding-users-in-your-app-02067234, updated on 5/29/2018.

Goethe, Ole (2019): Gamification Mindset. Cham: Springer International Publishing (Human-Computer Interaction Series).

Harms, Johannes; Biegler, Stefan; Wimmer, Christoph; Kappel, Karin; Grechenig, Thomas (2015): Gamification of Online Surveys. Design Process, Case Study, and Evaluation. In Julio Abascal, Simone Barbosa, Mirko Fetter, Tom Gross, Philippe Palanque, Marco Winckler (Eds.): Human-Computer Interaction - INTERACT 2015. Cham: Springer International Publishing, pp. 219–236.

Marczewski, Andrzej (2018): Even Ninja Monkeys Like to Play: Unicorn Edition.

Triantoro, Tamilla; Gopal, Ram; Benbunan-Fich, Raquel; Lang, Guido (2020): Personality and games. Enhancing online surveys through gamification. In *Information Technology and Management* 121 (2). DOI: 10.1007/s10799-020-00314-4.

Upland Software (no year): 21% of Users Abandon an App After One Use. Available online at https://uplandsoftware.com/localytics/resources/blog/21-percent-of-users-abandon-apps-after-one-use/.