UNIVERSITY OF TRIESTE

DATA SCIENCE & SCIENTIFIC COMPUTING

SYNTHETIC DATA PLATFORM

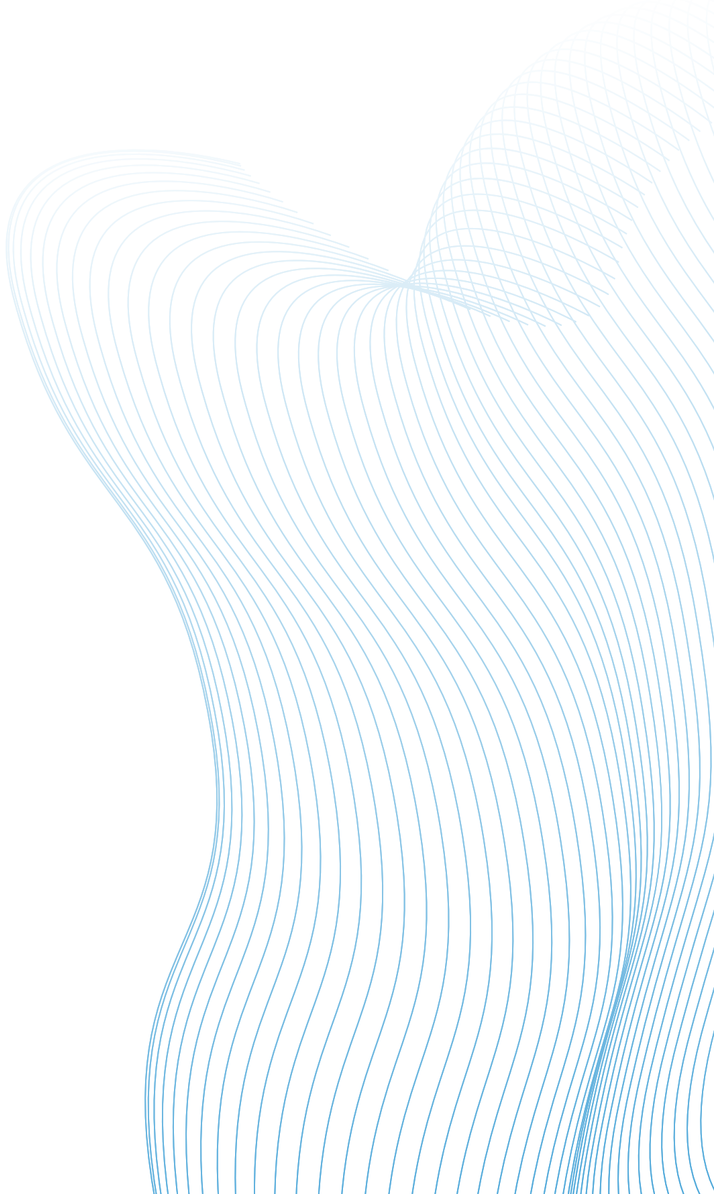
Final Project for the course on Open Data Management & the Cloud
AY 2022/2023

Francesco Ortu

github.com/Francesc0rtu/SyntheticDataPlatform

AGENDA

- INTRODUCTION
- DATA LIFE CYCLE
- CLOUD USAGE
- IMPLEMENTATION
- CONCLUSIONS



INTRODUCTION

PROJECT **GOALS**

Implement a platform to produce open synthetic data

Since there are many data that could not be published, why do not reproduce it!

Discuss data lifecycle

How handle data, from the row source to the synthetic reproduction.

Challenge, implementation and scalability

Because data generation is quite a new technique, there are a lot of challenge to discuss.

The **idea** is to implement an application where users could load a private or public tabular data and get an open synthetic copy of the original dataset.

The users will be **private** or **public** institutions that wants to share private date (i.e. protected by the *GDPR*) or simply want to have more data to train an AI system.

APP **FUNCTIONALITY**

INPUT DATA

Users load a private/public dataset

TRANSFORMATION

It is created the synthetic copy of the input dataset

OUTPUT DATA

The output data are saved.

SHARING DATA

Output data are sharing with a open license

LINK DATA

The output data are liked(if possible) to other open data

KEY **CONCEPT**

The output data will be always open

The main idea is to have a service to publish confident data, like healthy or financial data in order to increase the openess in the society.

The input data could be open or not

Obviously the input data could be not open, so we need to implement users and handle private data for security reason. If the input data is open, it will be linked to the output data

The service support tabular and relational data

Ideally, the application will support any kind of data with an open standard, anyway for implementation reasons support only tabular data.

WHAT WE **NEED**

CORE SYSTEM

We need the core system (deep learning based) for generate synthetic data.

DATABASE

We have to handle user, input data and output data and link them together.

ARCHIVE

We need a method to store large files and make it public.

DATA LIFE CYCLE

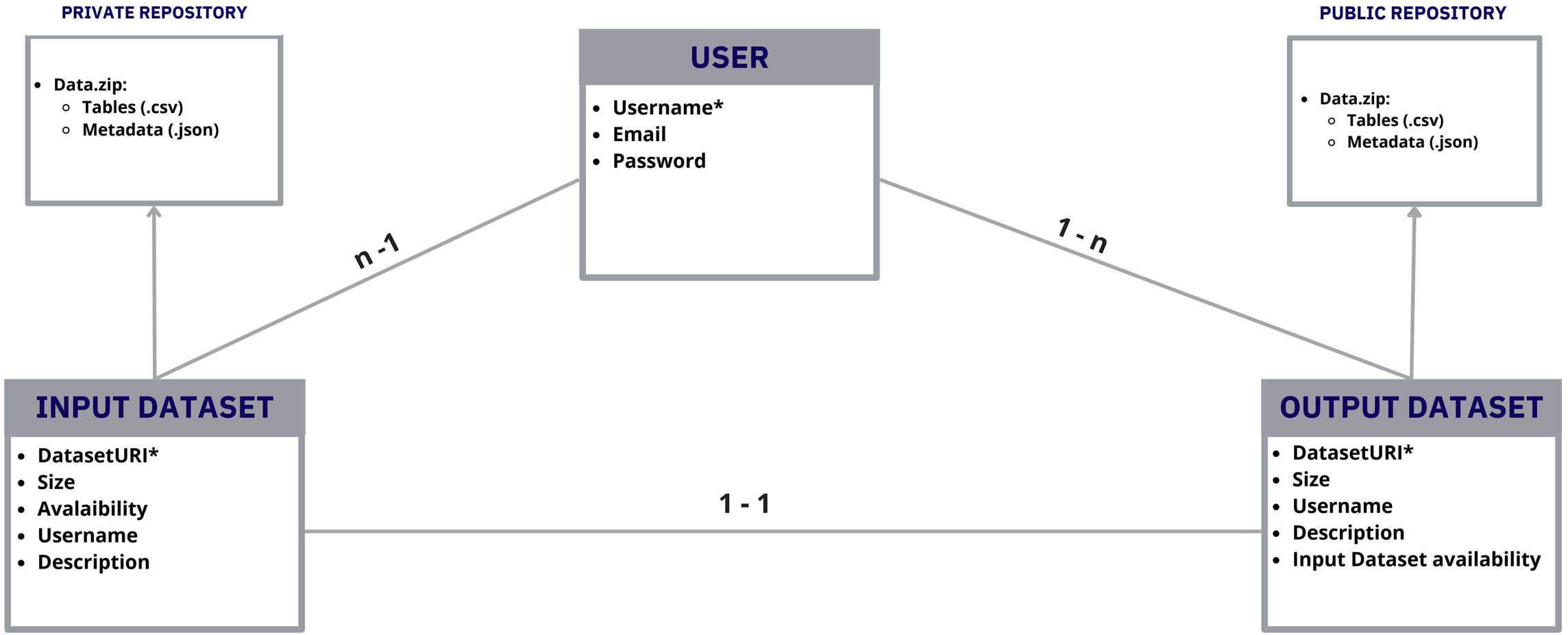
INPUT **DATA**

- Input data are uploaded from the user.
- Input data must have a 3 star openess. (based on the 5-star scheme proposed by Tim Berners-Lee), since the application needs open standard file extensions.
- In addition, data must have the extension supported from the application(.csv for now).
- Input data must be accompanied by a specific metadata file (JSON format) that describe the data itself and the relations between tables, in order to select the right technique to apply.
- Input data are stored in a private repository, along their json metadata description and linked with output data and user.

OUTPUT **DATA**

- Output data will be generated from the application.
- Output data will have at least 4 star openess.
- Output data will inherit the metadata description of the corresponding input data, adding that is a synthetic reproduction.
- If the corresponding input data is free available the data is linked.
- Output data will be put into and archive and will be free accessible.

DATA **MODEL**


PRIVATE REPOSITORY
Data.zip:
Tables (.csv)
Metadata (.json)
USER
Username*
Email
Password
PUBLIC REPOSITORY
Data.zip:
Tables (.csv)
Metadata (.json)
n - 1
1 - n
INPUT DATASET
DatasetURI*
Size
Avalaibility
Username
Description
1 - 1
OUTPUT DATASET
DatasetURI*
Size
Username
Description
Input Dataset availability

METADATA

```
{   "dataset name": "example",
    "dataset description": "Example dataset for testing",
    "license": "CC0",
    "numeber of tables": 3,
    "doi citations": ".."
    "Other metadata": ...
    "tables": {
        "table1": {
            "fields": {
                "user_id": {
                    "type": "id",
                    "subtype": "integer"
                },
                "country": {
                    "type": "categorical"
                },
                ...
            },
            "primary_key": "user_id"
        },
        "table2": {
            "fields": {
                ...
                }
            },

        },
        ...
    }
}
```

Metadata are used not only to provide context and additional description about the data itself, but also to supply to the core system the relation among the tables.

For this reason, we store metadata in JSON file that must have a fixed structure (Metadata Vocabulary)

Ideally, the metadata respect the standard of W3C *"Metadata Vocabulary for Tabular data"[1].*

[1]https://w3c.github.io/csvw/metadata/

CHALLENGES

LONG-TERM REPOSITORY	The output datasets should have been stored for a long-time period without any updating, so it is important to use an infrastructure that could cope possibly faults.
ACCESSIBILITY	Output data should been accessible multiple time in a *read-only* mode. For this reason the repository have to deal with many read query of a large files. MongoDB ATLAS offers GridFS tho handle BSON files with size > 16 MB.
SECURITY	The implementation of the private repository for the input data must be take care of the security of the data, since the idea is that the input data could be strictly private.

FAIR **PRINCIPLES**

FINDABILITY

Output data are registered in a searchable resource and the metadata describe it. The metadata are also human-readable.

ACCESSIBILITY

Output data are retrievable by their identifier through an open protocol (web, API).

INTEROPERABILITY

Output data will (ideally) use a standard metadata description and use open format. Output data will be linked with the original one, where it is possible.

REUSABILITY

Output data are released with an open license, rich metadata and respect community standard.

CLOUD USAGE

WHY USE **CLOUD**

DATABASE & STORAGE	Since the application is data-intensive and is meant to store large files, the usage of some Cloud service (Google storage, AWS S3..) could avoid the implementation of the storage system.
COMPUTATIONAL COST	The application is also highly demanding in term of computational resources and a Cloud service could help to handle resources without efforts.
SIMPLER IMPLEMENTATION	Using a Cloud provider probably will simplify the implementation and the scale-up.

HOW TO USE **CLOUD**

FOR STORAGE

Implement the database and the storage using a cloud platform.
Depending from the traffic different choice of service can be made, based also from the number of input/output operations.

FOR COMPUTATION

Running the application in a virtual machine on the cloud (PaaS).
The resources requested could be adjusted based on the computational demanding. (Most of the time the application manages only the download page, so no GPU needed!)

FOR SIMPLIFY

All the major cloud providers are starting to supply software service to generate synthetic data (SaaS)[5], that could be used as core system without build and maintain directly the ML/DL model.

IMPLEMENTATION

USED **FRAMEWORKS**

WEB FRAMEWORK

A python micro-framework to create web application in easy way.

DATABASE & STORAGE

A cloud service of MongoDB noSQL database. Atlas has the possibility to be linked to AWS, Google Cloud and Azure.

CORE MODELS

ML library to generate synthetic data based on PyTorch

CONTAINER

Container service to run and distribute the application.

LIVE DEMO

SOURCE: https://github.com/Francesc0rtu/SyntheticDataPlatform
IMAGE: https://hub.docker.com/r/francescortu/sdp

CONCLUSION

POSSIBLE **IMPROVEMENTS**

ADD MULTIPLE FILE EXTENSION

In order to enable the synthetic generation of the most common open standard, the application will have to support multiple file extension like: JSON, XML, images...

STANDARD JSON SCHEMA

To achieve more interoperability and to obtain better user experience the application should support the standard JSON vocabulary for tabular data of the W3C. In addition the app should havea procedure to validate the input metadata.

API

Providing API, the user could embedded the system in his software, improving the accessibility of the application.

MULTIPLE DATASET VERSION

The application should support the updating of the input dataset (and consequently the output dataset), storing the different version and managed it in a efficient way.

REFERENCES

[1] https://w3c.github.io/csvw/metadata/

[2] https://flask.palletsprojects.com/en/2.2.x/

[3] https://www.mongodb.com/

[4] https://sdv.dev/SDV/

[5] https://www.microsoft.com/en-us/ai/ai-lab-synthetic-data-showcase

THANKS FOR YOUR ATTENTION