ANITA

Anonymous big data A project funded by FFG

Virtual Data Lab

Deliverable D4.1

Author(s): Michael Platzer, Klaudius Kalcher

Reviewer: Stefan Vamosi

Document version: 0.2
Date: 07.06.2021

Federal Ministry
Republic of Austria
Climate Action, Environment, Energy, Mobility, Innovation and Technology

Programme **"ICT of the Future"** - the initiative of the Austrian Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK)

Disclaimer

This deliverable describes the work and findings of the AI-Based Privacy-Preserving Big Data Sharing for Market Research (Anonymous Big Data (ANITA)) project.

The authors of this document have made every effort to ensure that its content was accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this deliverable are responsible for any possible errors or omissions as well as for any results and actions that might occur as a result of using the content of this document.

Table of contents

1 Summary

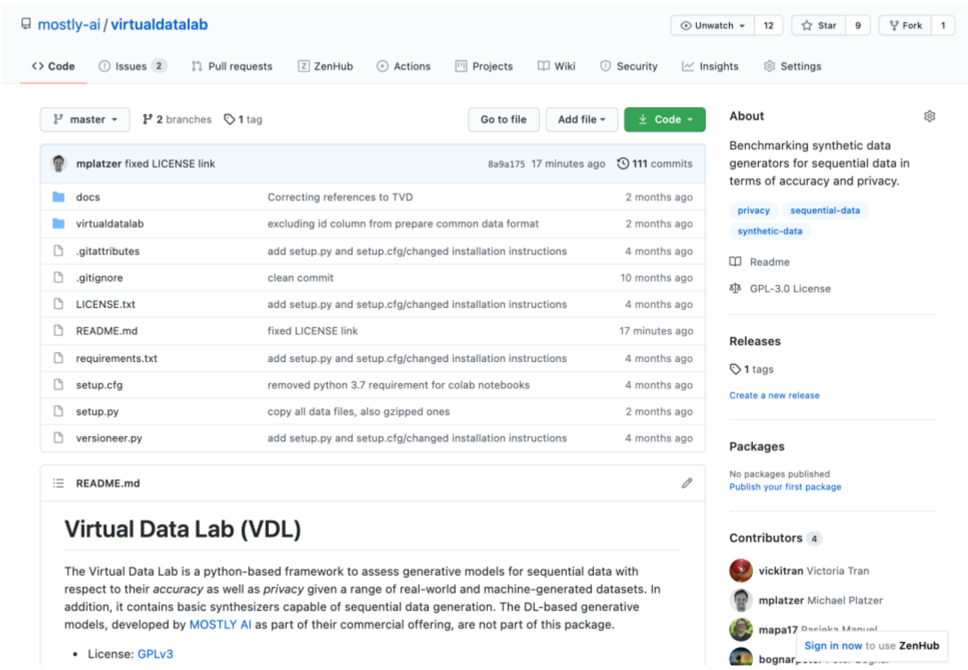
The consortium successfully developed, validated and documented an open-source Virtual Data Lab, that has been made publicly available at https://github.com/mostly-ai/virtualdatalab/. The Virtual Data Lab allows users to easily benchmark synthetic data generators for sequential data in terms of their accuracy and privacy across a range of datasets. Thus, it shall serve as an important contribution to the community to measure and track progress within the field of synthetic data.

The Virtual Data Lab is developed in Python, can be easily and instantly run on free cloud resources via Google Colab, and consists of these key components:

- Data preprocessing utilities
- Mock data generators
- Four mixed-type datasets – more can be easily added
- Three synthesizers – more can be easily added via an interface
- Accuracy metrics
- Privacy metrics
- Demo Notebooks

The Virtual Data Lab has been released under the GNU General Public License v3.

The Virtual Data Lab has been successfully used to run the simulation study (WP4.2) and benchmark the variety of generative model approaches developed within WP 5.



2 Introduction

The goal of this work package was to develop, setup and run a Virtual Data Lab, that is then used for validating the accuracy and privacy of a range of synthetic data generators across a range of datasets. The results from the preceding use case and requirement analysis directly went into the concepting of the Virtual Data Lab.

It includes

- a flexible data factory for generating a variety of artificial sequential datasets to be tested
- a range of public mixed-type sequential datasets
- a GPU cloud compute setup, capable to scale resources on-demand on Google cloud infrastructure – this is important to ensure flexibility in terms of scaling up experiments, as well as enables a high degree of parallelization which results in shorter project duration
- a range of metrics for assessing accuracy
- validation tests and measures for assessing privacy
- a range of synthesizers serving as demonstration of the interface

During the course of the project, an open-source initiative has been started by the Data to AI Lab at MIT, called SDGym. It is similar in nature, that it also aims to provide capabilities to benchmark synthetic data generation methods. However, it does not support sequential data nor includes any privacy assessments, both being key focus of ANITA. However, an in-depth analysis of SDGym served as valuable resource for the design and implementation of the Virtual Data Lab.

3 Accuracy Metrics

Accuracy is measured in terms of the statistical distances of the corresponding empirical distributions, based on the original data verses based on the synthetic data.

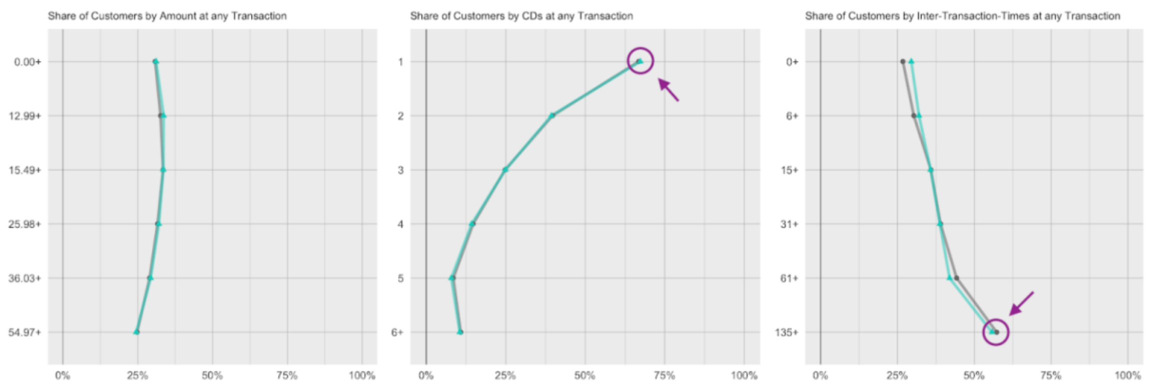
The included metrics quantify the difference between the empirical distributions of the target compared to the synthetic data. Numeric variables are being discretized by binning these into 10 equally spaced buckets. Categorical variables are considering the top 10 values, and lump together all remaining values into a single bucket.

First, for each subject a single event is randomly selected. Then, for each column, respectively each combination of 2, 3 or 4 columns the distances between the empirical distributions of actuals vs. synthetic data are being calculated, and then averaged across a random subset of max. 100 of such combinations. Note, that the depth of combinations is capped at 4 as for higher orders the resulting contingency tables would be too sparsely populated.

- Max (MAX) = maximum deviation in relative frequency
- L1-Distance (L1D) of empirical distributions = sum over all deviations in relative frequency

In addition, we developed two coherence measures for each attribute over the sequence, and calculate
- L1-Distance for share of subjects per category
- L1-Distance for share of distinct categories per subject

See https://mostly.ai/2020/06/05/how-to-unlock-your-behavioral-data-assets-part-iii/ for further details on these.



The output from `metrics.compare` is

Metric Name	**Definition**
MAX univariate	The maximum relative frequency deviations with respect to 1 column.
L1D univariate	The sum of relative frequency deviations with respect to 1 column.
L1D bivariate	The sum of relative frequency deviations with respect to 2 columns.
L1D 3-way	The sum of relative frequency deviations with respect to 3 columns.
L1D 4-way	The sum of relative frequency deviations with respect to 4 columns.
L1D Users per Category	The sum of relative frequency deviations between how many users per category. For each category column we count the unique number of users and normalize that number, simply by dividing with total number of unique users. After

	that we calculate an absolute difference and sum up those for each category. As a last step we calculate the mean value over all categories to get the final metric.
L1D Categories per User	The sum of relative frequency deviations between how many categories per user. Here we group by users and count the unique number of values of the binned categories. In this case we also normalize, but the category counts by users, then calculate absolute difference and sum that up. The last step is to calculate mean value over all users.

4 Privacy Metrics

The privacy measures are based on individual-level distances. These quantify the distance between individual synthetic data records to their closest target records. These distances are then related to the same distance measures applied to actual holdout data.

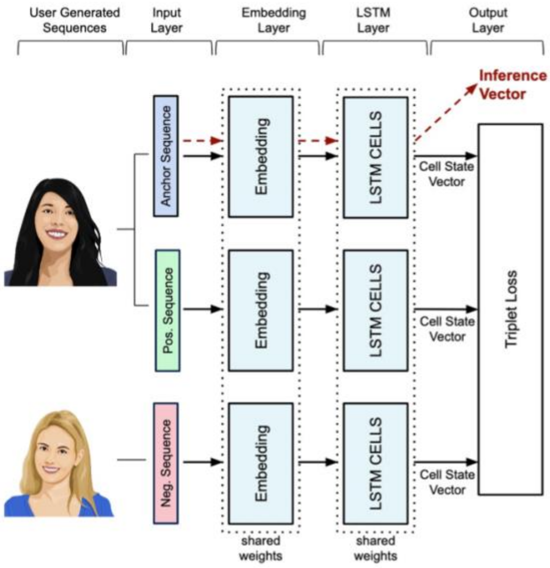
- Distance to Closest Records (DCR): Calculates the distance of each synthetic record to a record in the target distribution. These distances must not be systematically smaller than the distances between holdout and target.

- Nearest Neighbour Distance Ratio (NNDR): Ratio of the distance of each synthetic record to its closest to the second closest record in the target distribution. Again, these ratios should not be systematically smaller than the ratios derived for the holdout set.

An in-depth presentation of these privacy metrics has been published at https://mostly.ai/2020/11/04/truly-anonymous-synthetic-data-legal-definitions-part-ii/.

4.1 TL-RNN based privacy assessment

A key challenge for sequential data is the definition of a meaningful, relevant distance metric, that can then be used for DCR/NNDR calculations. For that purpose, we developed a novel embedding space for mixed-type sequential data that leverages a Triplet-Loss RNN model. This embedding space can then be used to calculate L1- and L2-distances between samples.



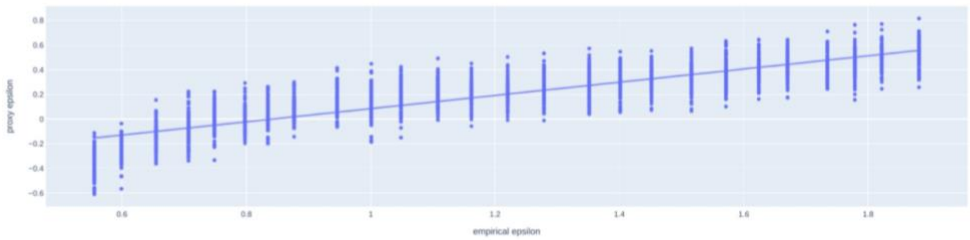
Further details will be shared as part of WP7, with a dedicated paper on TL-RNN (Working paper by Vamosi, Reutterer, Platzer) and its usage for re-identification of sequential data (upcoming paper, presented at EMAC 2021 titled "AI-based re-identification exposes privacy risk of behavioral data. A case for synthetic data").

4.2 Empirical differential privacy proxy

Another developed approach to assess privacy, was to track a proxy measure for empirical epsilon during the training phase. As a true empricial epsilon would be computationally expensive, the validation loss might serve as a good enough proxy.

Key Finding was, that the proxy epsilon gives us an indication of whether the training starts to focus on the specifics of rare users in the training dataset, but it does NOT give us the value of the empirical epsilon in the specific run.

Figure 1.3 Relationship of empirical epsilon and proxy epsilon: quantile 0.9



5 Synthesizers

The Virutal Data Lab ships with three reference implementations of synthesizers. These primarily serve the purpose of demonstration of the interface. In particular, these do not represent state-of-the-art methods in data synthetization, as these are not being publicly released as part of the the Virtual Data Lab.

5.1 IdentitySynthesizer

This synthesizer simply returns samples of the provided data. It serves as a demonstration for the implementation interface, plus validates that the accuracy and privacy measures do work as expected.

5.2 ShuffleSynthesizer

This synthesizer simply returns column-wise shuffled samples of the provided data. It serves as a demonstration for the implementation interface, plus validates that the accuracy and privacy measures do work as expected.

5.3 FlatAutoEncoder

This synthesizer is based on a fully connected encoder-decoder neural network, implemented in PyTorch. It serves as a demonstration for the implementation interface to integrate AI-based generators.

5.4 Interface for Custom Synthesizes

New synthesizers can be easily provided by extending the `BaseSynthesizer` call. Additionally, their `train` and `generate` must invoke parent method via `super()`. Parent functions ensure that common data format is respected and that models cannot be expected to generate if they have not been trained yet.

All synthesizer classes MUST accept the common data format. As a result, synthesizers are responsible for transformation of input data.

Example:

```
class MyGenerator(BaseSynthesizer):

    def train(self,data):
        super().train(data)
        data_model = some_transformation(data)
        self.train_data_ = data
        #  model is now trained
        self.data_model_ = data_model

    def generate(self,number_of_subjects):
        super().generate(self)
        generated_data = some_generation(number_of_subjects)

        return generated_data
```

6 Datasets

Four datasets are included as part of the Virtual Data Lab. These are all datasets, that have already been previously made public by a data owner, and serve as a test ground for benchmarking then the generators.

These datasets are all sequential, contain mixed-type variables, and have been limited to a fixed sequence length. The latter ensures, that we do not have to require that synthetic data generators need to handle varying sequence lengths across customers, which might pose an additional challenge. Along the same line, the datasets do not include any missing values, to keep the focus on the coherence along the sequence in the evaluation.

6.1 CDNOW

These are 3,925 users with a sequence length of 5 purchases, with 3 attributes each:

- number of CDs purchased: numeric
- dollar amount of the transaction: numeric
- day of the week of the transaction: categorical

```python
from virtualdatalab.datasets.loader import load_cdnow
load_cdnow()
```

id	sequence_pos	cds	amt	wday
3	0	4	20.96	Tue
	1	5	57.45	Sat
	2	1	16.99	Thu
	3	2	20.76	Sun
	4	2	20.76	Thu
...	...	...	...	...
23556	0	1	11.77	Tue
	1	2	26.73	Tue
	2	2	29.33	Sat
	3	3	45.74	Sat
	4	3	31.47	Sat

19625 rows × 3 columns

6.2 BERKA

These are 4,400 users with a sequence length of 10 transactions, with 5 attributes each:

- Transcation Amount: numeric
- Transfering Bank: categorical
- Transaction Operation: categorical
- Transaction Type: categorical
- Transaction Symbol: categorical

```python
from virtualdatalab.datasets.loader import load_berka
load_berka()
```

id	sequence_pos	amount	bank	operation	type	k_symbol
1	0	1000.0		VKLAD	PRIJEM	
	1	3679.0	AB	PREVOD Z UCTU	PRIJEM	
	2	12600.0		VKLAD	PRIJEM	
	3	19.2			PRIJEM	UROK
	4	3679.0	AB	PREVOD Z UCTU	PRIJEM	
...	...	...	...	...	...	...
4000	6	74.8			PRIJEM	UROK
	7	4987.0		VKLAD	PRIJEM	
	8	2300.0		VYBER	VYDAJ	
	9	95.2			PRIJEM	UROK
	10	7480.0		VKLAD	PRIJEM	

44000 rows × 5 columns

6.3 MLB

These are 4,000 baseball players with a sequence length of 8 seasons, with 5 attributes each:

- Baseball team: categorical
- Baseball league: categorical
- Number of Games played: numeric

- Number of At Bats: numeric
- Number of Hits: numeric

```
from virtualdatalab.datasets.loader import load_mlb
df = load_mlb()
df
```

id	sequence_pos	team	league	G	AB	H
1	0	MIN	AL	6	0	0
	1	MIN	AL	28	5	1
	2	MIN	AL	14	0	0
	3	MIN	AL	29	5	0
	4	MIN	AL	31	2	0
...	...	...	...	...	...	...
4000	3	PIT	NL	39	33	2
	4	PIT	NL	34	51	5
	5	PIT	NL	37	69	7
	6	PIT	NL	33	24	2
	7	SDN	NL	53	25	3

32000 rows × 5 columns

6.4 RETAIL

These are 10,000 users with a sequence length of 100 ordered product, with 2 attributes each:

- An indicator whether this record represents a new order: boolean
- The ordered product: categorical

Due to its size, its sequence length, and its high cardinality for attribute product, this represents the most challenging dataset among those included within the Virtual Data Lab.

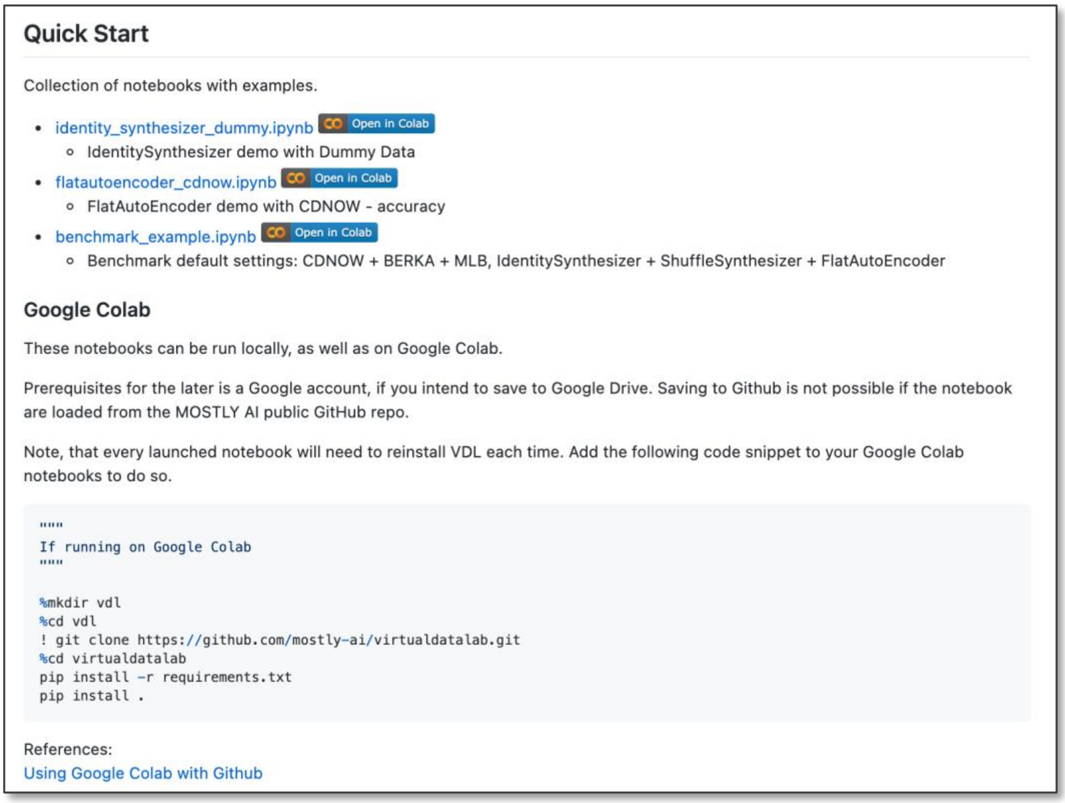
```
from virtualdatalab.datasets.loader import load_retail
df = load_retail()
df
```

id	sequence_pos	new_order	prod
1	0	Yes	(other)
	1	No	bags & foil
	2	No	by hand dish washings
	3	No	canned vegetables
	4	No	carbonated water
...	...	...	...
10000	95	No	instant soup
	96	No	meat
	97	No	melted cheese
	98	No	milk dessert nature
	99	No	ready meals

1000000 rows × 2 columns

7 Demo Notebooks

The Virtual Data Lab includes 3 Python Jupyter Notebooks, that demonstrate the basic workflow when using the library. In particular, it's being demonstrated how these benchmarks can be easily and instantly launched on GPU-powered resources hosted on the Google Cloud, via Colab.



Quick Start

Collection of notebooks with examples.

- identity_synthesizer_dummy.ipynb Open in Colab
  - IdentitySynthesizer demo with Dummy Data
- flatautoencoder_cdnow.ipynb Open in Colab
  - FlatAutoEncoder demo with CDNOW - accuracy
- benchmark_example.ipynb Open in Colab
  - Benchmark default settings: CDNOW + BERKA + MLB, IdentitySynthesizer + ShuffleSynthesizer + FlatAutoEncoder

Google Colab

These notebooks can be run locally, as well as on Google Colab.

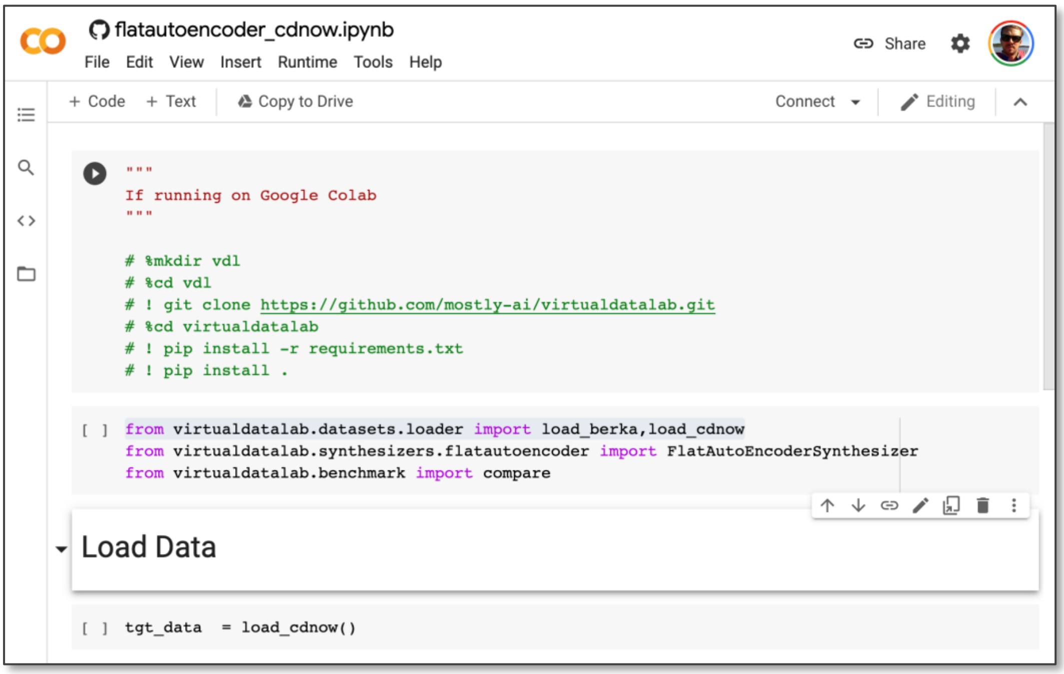
Prerequisites for the later is a Google account, if you intend to save to Google Drive. Saving to Github is not possible if the notebook are loaded from the MOSTLY AI public GitHub repo.

Note, that every launched notebook will need to reinstall VDL each time. Add the following code snippet to your Google Colab notebooks to do so.

```
"""
If running on Google Colab
"""

%mkdir vdl
%cd vdl
! git clone https://github.com/mostly-ai/virtualdatalab.git
%cd virtualdatalab
pip install -r requirements.txt
pip install .
```

References:
Using Google Colab with Github



8 License

The Virtual Data Lab is released under the GNU General Public License v3.0, which can be found here.

Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights.

Permissions

- Commercial use
- Modification
- Distribution
- Patent use
- Private use

Limitations

- Liability
- Warranty

Conditions

- License and copyright notice
- State changes
- Disclose source
- Same license