Synthetic Data Generation Application User Manual

Table of Contents

This document is intended to guide users through the usage of the synthetic data generation web application version 1.0.0.

F. Hoffmann-La Roche Ltd

Grenzacherstrasse 124
4070 Basel
Switzerland

E-mail: XXX@roche.com

The Synthetic Data Generation Application (“the app”) is a python web browser application. The app enables users to generate synthetic data from source data. Additionally, the app also enables users to evaluate the quality of synthetic data.

App Installation

Prerequisites

The app runs on machines with the following minimum requirements:

1. Microsoft Windows version 10 or above
2. Conda, accessible from the Command Prompt (https://conda.io/projects/conda/en/latest/index.html)
3. Internet connection.

Installation

To install the app, perform the following steps on the Command Prompt:

1. Clone app repository

   Clone the app code repository and change directory:

```python
git clone
https://code.roche.com/sdg-pipeline/sdg-development/sdg-interface.git

cd sdg-interface
```

2. Create a new conda environment

   Create a new conda environment named `sdg-interface-env`:

```Python
conda create -n sdg-interface-env python=3.8 -y
```

Activate the environment:

```Python
conda activate sdg-interface-env
```

3. Install R and `synthpop`

There are two ways for this step.

You may need to try both and see which one works for your machine.

Method 1: Installing R using conda

Install `R` using conda:

```Python
conda install r -y
```

Then, enter `R`:

```Unset
R
```

In `R`, install package `synthpop`:

```Unset
install.packages("synthpop")
```

When prompted in a pop-up window, choose any CRAN mirror site.

You may need to press Enter in the Command Prompt to proceed.

Exit `R`:

```
Unset
quit()
```

When prompted whether to save your `R` workspace image, enter `n` and press Enter.

Set R path environment variable:
in the `sdg-interface` folder, open the `.env` file and modify the path:

```
Unset
R_path= 'C:\Users\YOURUSERNAME\Anaconda3\envs\sdg-interface-env\Lib\R'
```

Substitute `YOURUSERNAME` with your Windows username.

Method 2: Installing R from source

Alternatively, if installing `R` using conda causes errors, you can do the following:

- download and install the latest version of `R` (version 4.2.1 or above) from the official R website.
    - Note that the default installation path is `C:\Program Files\R\R-4.2.1`
    - You could also specify other directory for installation
- Run the `R` command line tool from the *bin* folder
- install package `synthpop`:

```
Unset
install.packages("synthpop")
```

When prompted in a pop-up window, choose any CRAN mirror site.

- Set R path environment variable
  In the `sdg-interface` folder, open the `.env` file and modify the path, e.g:

```
Unset
R_path = 'C:\Program Files\R\R-4.2.1'
```

If you specified another directory for installation, please modify accordingly.

4. Install environment dependencies

   Finally, install the environment dependencies using `pip`:

```
Python
pip install -r requirements.txt
```

NOTE: Please see Frequently Asked Questions (FAQs) if dependencies installation fails.

Source Data Prerequisites

Before running the application, make sure that your source data conforms to the following requirements:

1. In uncompressed CSV (comma-separated values) format
2. In a single table format
    - If the data is composed of several tables, please merge them into a single table
3. Entities (e.g patients) are represented in rows, and features (e.g gender, year of birth, height etc.) are represented in columns.
4. Tabular and non-longitudinal in nature; i.e, one patient is represented by only one row
5. The data should be large enough for the model to work successfully
    - we recommend at least 200 rows
6. Contains only categorical and/or numerical columns
    - Note that the app currently does not support datetime columns
        - if you wish to include datetime columns, you may:
            1. first transform the columns into numerical columns
            2. run the generation pipeline

3. and re-transform the synthetic numerical columns back into datetime column

7. If a categorical column has too many (more than 60) categories, bin the categories to reduce complexity
    - failure to do so could result in running time being too high.
8. Related to (7), the data should not contain free text that can vary widely for a column, for example doctor's notes that are highly variable across patients.
    - failure to do so will result in the app identifying such columns as categorical columns, and will result in the app encountering the problem with too many features
9. Column names should contain no special symbols
    - to remove such symbols, run the following command on your source data:

```python
column=column.str.replace('[^\w\s]','')
```

10. As a general rule, remove columns that have less than 70% of values filled from the source data
    - Even though the app will work fine, we have observed some quality issues when there are too many NaNs (missing values) in a column.
11. Related to (10), missing data may be imputed.
    Some of the imputation methods that users may use:
    - Using mean value of the column
    - Using methods such as k-Nearest Neighbor (kNN)
12. If you wish to use a column for the ML performance test in the evaluation pipeline, the column should be categorical with only two features.

Metadata

Metadata is required to run the app. The metadata should be in JSON (JavaScript Object Notation) format. The metadata defines the source data and synthetic data, as well as a parameter for the evaluation pipeline.

We recommend using the metadata file in the `sample-data` directory as a starting point. You can copy paste this file and modify its parameters for your use case.

Metadata Prerequisites

The metadata contains the following:

1. `entity_columns` (list): specifies the entity column(s).
2. `fields` (dictionary): specifies the name of all columns and their datatype.
    - Possible values for datatype are:
        i. `categorical`
        ii. `integer`
        iii. `numerical`
3. `evaluation_parameters` (array): contains parameters for the evaluation pipeline.
    - Currently is used to define the target feature in a categorical column for machine learning (ML) performance test as such: `{"non_longitudinal_ml_performance": {"cat_col_feature_to_predict": "CATCOL_FEATURE"}}`, where:
        i. `CATCOL` is target categorical column with only two features, and
        ii. `FEATURE` is one of the features of the column

Example Metadata

The following is an example of a metadata:

```JavaScript
{
    "entity_columns": ["patientid"],
    "fields": {
        "patientid": {
            "type": "categorical"
        },
        "birthyear": {
            "type": "integer"
```

```
        },
       "gender": {
            "type": "categorical"
        },
...
   "evaluation_parameters":{
        "non_longitudinal_ml_performance": {
            "cat_col_feature_to_predict": "gender_F"
        }
    }
}
```

In the above example:

1. `patientid` is the entity column
2. Among the columns,
    a. `patientid` and `gender` are categorical columns
    b. `birthyear` is numerical column
3. For the ML performance test in the evaluation pipeline,
    a. `gender` is the target categorical column with only 2 features
    b. `F` is the target feature

Running the app

To run the app, follow these steps:

1. Start Anaconda prompt or access conda through the Command Prompt
2. Change directory to the app working directory:

```
Python
cd sdg-interface
```

3. Activate the `sdg-interface-env` environment:

```
Python
conda activate sdg-interface-env
```

4. Run the app on your default web browser:

```
Python
streamlit run app.py
```

The Generation Pipeline

Models in the Generation Pipeline

The app has capabilities to generate synthetic data using two models:

- CTGAN, conditional tabular generative adversarial network
- Synthpop, that runs on classification and regression trees (CART)

Running the Generation Pipeline

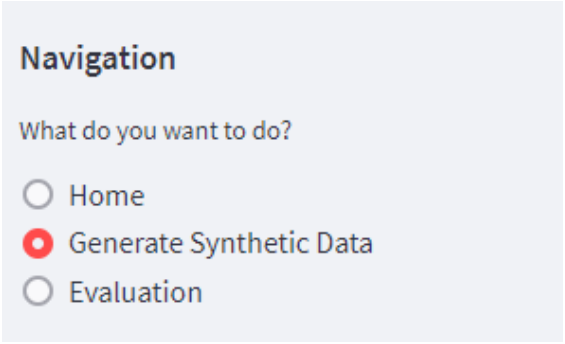
To generate synthetic data, run the generation pipeline by following these steps:

1. On the Command Prompt, go to the app directory (run cd sdg-interface) and start the app:

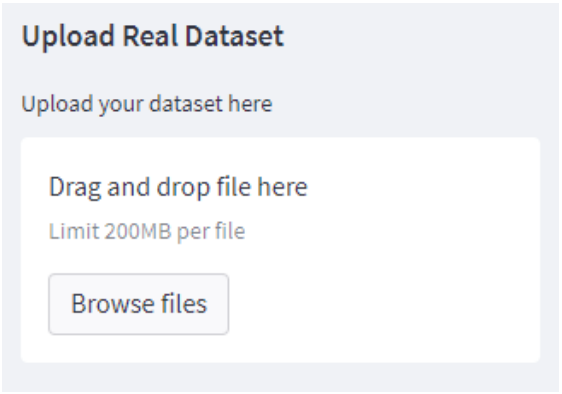
```
Python
streamlit run app.py
```

The app will start in your default web browser.

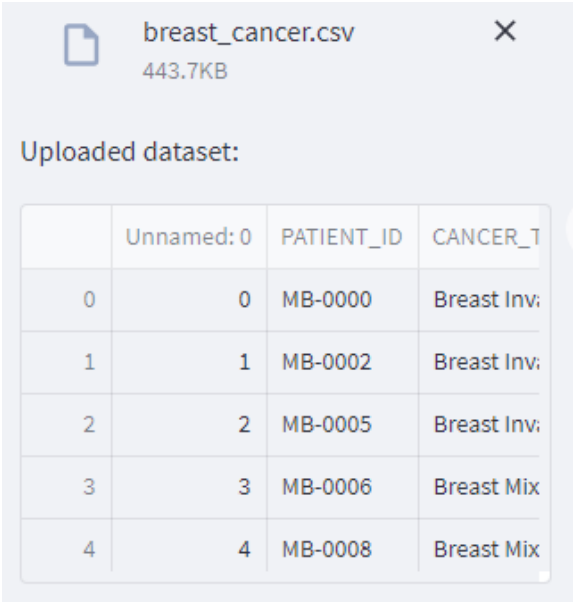
2. On the left panel under **Navigation**, select the radio button **Generate Synthetic Data**:



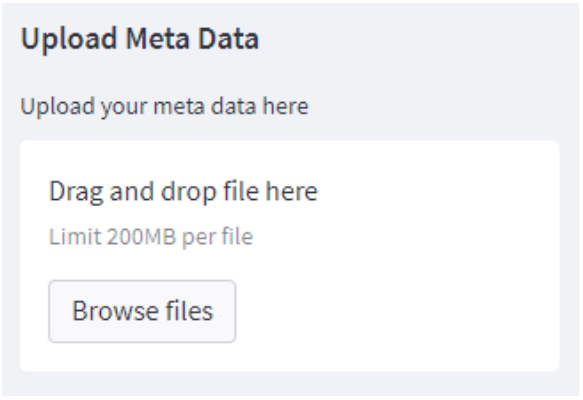
3. Upload source data under the heading **Upload Real Dataset**. Users may drag the file from the Desktop and drop it into the white area, or click **Browse files** to browse for and select the source data file:



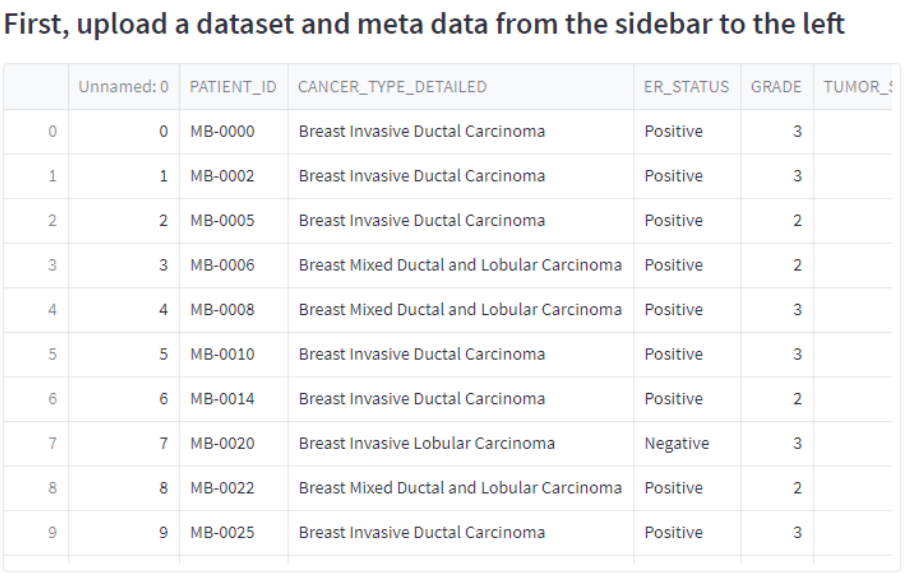
If the upload is successful, users will see a snippet of the source data on the left panel:



4. Upload metadata under the heading **Upload Meta Data**: Similar to uploading the source data, users can either drag and drop the file, or browse the file:

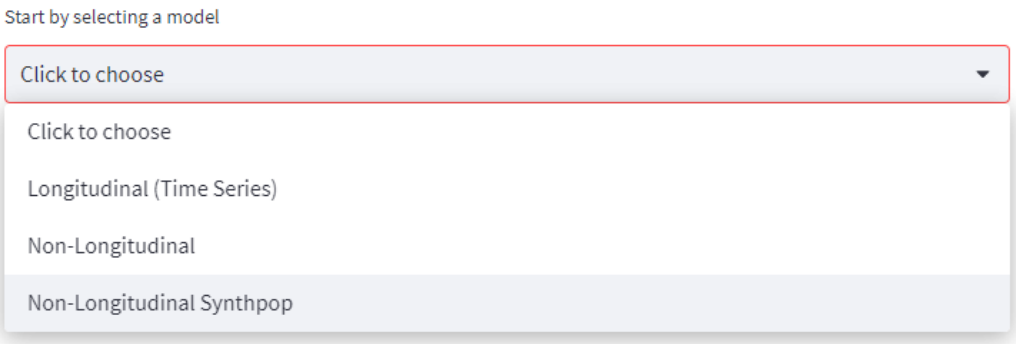


If the upload of the metadata is successful, users will see a snippet of source data at the center of your screen (the main panel):

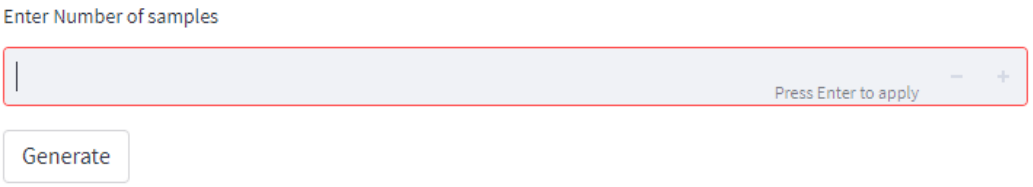


	Unnamed: 0	PATIENT_ID	CANCER_TYPE_DETAILED	ER_STATUS	GRADE	TUMOR_S
0	0	MB-0000	Breast Invasive Ductal Carcinoma	Positive	3	
1	1	MB-0002	Breast Invasive Ductal Carcinoma	Positive	3	
2	2	MB-0005	Breast Invasive Ductal Carcinoma	Positive	2	
3	3	MB-0006	Breast Mixed Ductal and Lobular Carcinoma	Positive	2	
4	4	MB-0008	Breast Mixed Ductal and Lobular Carcinoma	Positive	3	
5	5	MB-0010	Breast Invasive Ductal Carcinoma	Positive	3	
6	6	MB-0014	Breast Invasive Ductal Carcinoma	Positive	2	
7	7	MB-0020	Breast Invasive Lobular Carcinoma	Negative	3	
8	8	MB-0022	Breast Mixed Ductal and Lobular Carcinoma	Positive	2	
9	9	MB-0025	Breast Invasive Ductal Carcinoma	Positive	3	

5. Select a model from the dropdown menu at the bottom center of your screen:



6. Enter the number of synthetic rows to be generated:



Click **Generate** to start generating your synthetic data.

Note that at the top right corner, the animation indicates that the generation pipeline is in progress:

When the generation pipeline is finished, the animation will stop. Users can scroll down the main panel and see a snippet of the synthetic data:

Synthetic Data Generated

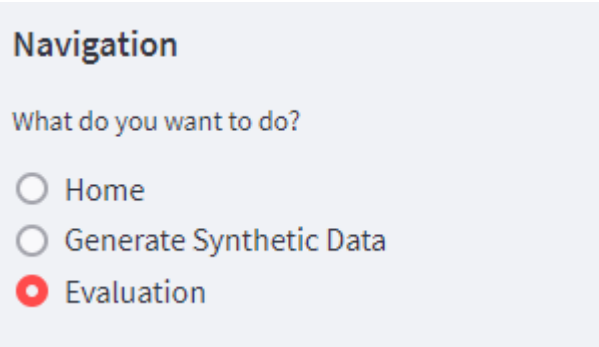
	PATIENT_ID	CANCER_TYPE_DETAILED	ER_STATUS	GRADE	TUMOR_SIZE	TMB_NONSYNONY
0	P_SYNTH-0	Breast Invasive Ductal Carcinoma	Positive	3	45	1
1	P_SYNTH-1	Breast Invasive Ductal Carcinoma	Negative	3	45	1
2	P_SYNTH-2	Breast Invasive Ductal Carcinoma	Positive	2	28	
3	P_SYNTH-3	Breast Invasive Lobular Carcinoma	Positive	2	28	
4	P_SYNTH-4	Breast Invasive Ductal Carcinoma	Positive	2	16	
5	P_SYNTH-5	Breast Invasive Ductal Carcinoma	Positive	3	23	
6	P_SYNTH-6	Breast Invasive Ductal Carcinoma	Negative	NA	NA	
7	P_SYNTH-7	Breast Invasive Ductal Carcinoma	Positive	3	19	
8	P_SYNTH-8	Breast Invasive Ductal Carcinoma	Positive	2	25	
9	P_SYNTH-9	Breast Invasive Ductal Carcinoma	Positive	1	9	

The synthetic data is saved in the working directory of the app as `synthpop_output.csv`.

The Evaluation Pipeline

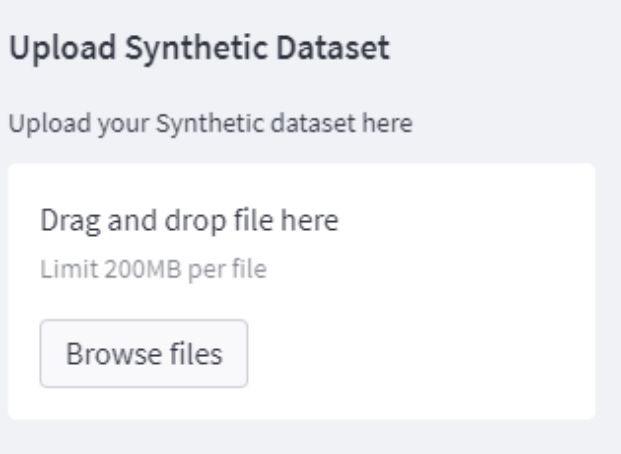
To compare the source data with the synthetic data, users may run the evaluation pipeline via the following steps:

1. On the left panel under **Navigation**, select the radio button **Evaluation**:

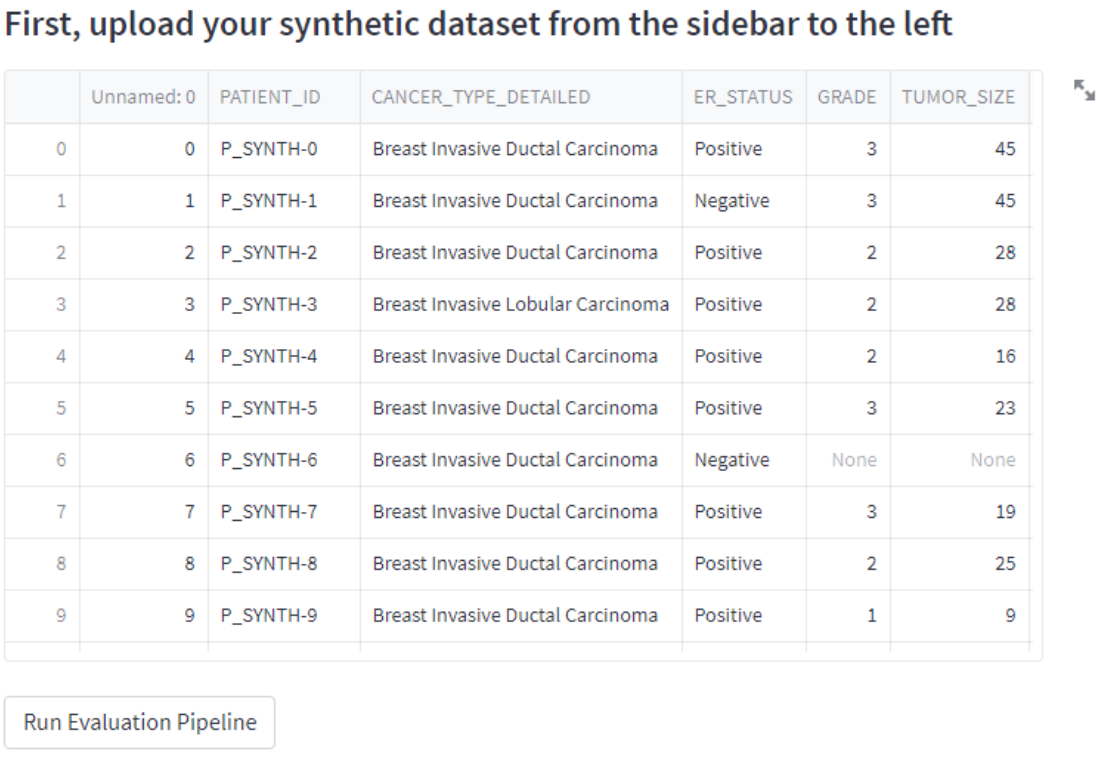


2. If users haven't done so already, upload the source data and metadata as previously described in Running the Generation Pipeline

3. Upload the synthetic data under the heading **Upload Synthetic Dataset**. Similar to uploading the source data and metadata, users can either drag and drop the file, or browse the file:



If the upload of the synthetic data is successful, users will see a snippet of synthetic data at the center of the screen:



	Unnamed: 0	PATIENT_ID	CANCER_TYPE_DETAILED	ER_STATUS	GRADE	TUMOR_SIZE
0	0	P_SYNTH-0	Breast Invasive Ductal Carcinoma	Positive	3	45
1	1	P_SYNTH-1	Breast Invasive Ductal Carcinoma	Negative	3	45
2	2	P_SYNTH-2	Breast Invasive Ductal Carcinoma	Positive	2	28
3	3	P_SYNTH-3	Breast Invasive Lobular Carcinoma	Positive	2	28
4	4	P_SYNTH-4	Breast Invasive Ductal Carcinoma	Positive	2	16
5	5	P_SYNTH-5	Breast Invasive Ductal Carcinoma	Positive	3	23
6	6	P_SYNTH-6	Breast Invasive Ductal Carcinoma	Negative	None	None
7	7	P_SYNTH-7	Breast Invasive Ductal Carcinoma	Positive	3	19
8	8	P_SYNTH-8	Breast Invasive Ductal Carcinoma	Positive	2	25
9	9	P_SYNTH-9	Breast Invasive Ductal Carcinoma	Positive	1	9

4. Click **Run Evaluation Pipeline**

The web application will now run the evaluation pipeline. The results of the evaluation pipeline will be displayed on the screen as the evaluation pipeline progresses.

The results of the evaluation pipeline are also saved in the directory `report`:

1. `my_images.pdf` contains the plots and graphs of the evaluation results,
2. `tables.pdf` contains tables of the evaluation results, and
3. `final_report.pdf` contains a combination of (1) and (2).

Interpreting the Evaluation Report

The results of the evaluation report are made up of several sections. The following describes the contents of `final_report.pdf`:

1. Univariate comparisons
    a. Barplots
        - The height of the barplots denotes the median value
        - the lines in the middle of the bars indicate standard deviations
    b. Violin plots
        - The shapes of the violin indicate the distribution of the values
        - The white dot in the middle of the violin indicate the mean value
        - The black thick line in the middle of the violin indicate the 25th and 50th percentile
        - The lines extending out from the black thick line indicate the 5th and 95th percentile

2. Bivariate comparisons
    The heatmaps indicate the strength of the relationship between columns:
    a. Left heatmap
        The strength of relationship between columns in source data. The values are calculated as follows:
        - categorical versus categorical column relationships were calculated using Theil's U values
        - Numerical versus numerical column relationships were calculated using Pearson correlation of coefficients

- Categorical versus numerical column relationships were calculated using correlation ratios.

The color intensities represent the absolute values of the bivariate relationships.

b. Middle heatmap
   Similar to left heatmap (2a), but calculated for synthetic data
c. Right heatmap
   The difference of relationship strengths between source data and synthetic data. The raw bivariate relationship values of synthetic data (2b) was subtracted from that of source data (2a). The color intensities represent the absolute values of the differences.

If the synthetic data closely mimics the source data, left and middle heatmaps should look similar, and right heatmap should have only cells with low intensity colors.

3. Distinguishability confusion matrix
   The matrix depicts the results of a distinguishability test. Here, the app aggregates the rows from source data and synthetic data into a single table, and then trains an XGBoost (eXtreme Gradient Boosting) model. The app then scores the model for its ability to identify whether each row comes from source data or synthetic data.

   The number in each cell denotes the number of rows identified, e.g, the number in top left denotes the number of rows predicted to be true to also be true.

   If the synthetic data closely mimics the source data, we should see a high number of correctly identified rows (high numbers in top left and bottom right cells).

4. ML performance confusion matrices for source data and synthetic data.
   The matrices depict the results of machine learning performance tests using either source data or synthetic data. Here, app trains XGBoost models to predict the target feature in a categorical column (defined by the parameter `evaluation_parameters:non_longitudinal_ml_performance:cat_col_feat`

`ure_to_predict` in metadata). The app then scores the models for its ability to correctly identify the target feature in either source data or synthetic data.

If the synthetic data closely mimics the source data, we should see a similar performance between the models trained on source data and synthetic data, i.e the two confusion matrices show a similar pattern.

5. Basic statistics of numerical columns for source data and synthetic data.
   The tables depict the following statistics for source data and synthetic data:
   1. Number of values
   2. Mean value
   3. Standard deviation
   4. Minimum value
   5. 25th percentile
   6. 50th percentile
   7. 75th percentile
   8. Maximum value

   If the synthetic data closely mimics the source data, values in (5-1) to (5-8) for source data should be close to those for synthetic data.

6. Results of testing for significance of data distribution between source data versus synthetic data.
   This table shows the number of categorical and numerical columns, as well as the number of those columns that are deemed to be statistically different in source data versus synthetic data. The statistical tests used are Student's two-tailed t-test or Chi-squared test at a P-value cutoff of less than 0.05.

   If the synthetic data closely mimics the source data, the number of columns deemed statistically different should be as low as possible.

7. Mean absolute difference across bivariate relationships.

This value depicts the difference of inter-column relationships between the source data and synthetic data. The value is the mean of values depicted in the right heatmap in (2).

If the synthetic data closely mimics the source data, this value should be close to zero.

8. Distinguishability test's F1 score, accuracy, sensitivity and specificity.
   Depicts the metrics from the distinguishability test, (3).

   If the synthetic data closely mimics the source data, these values should be close to 0.5.

9. ML performance test's F1 score, accuracy, sensitivity and specificity.
   Depicts the metrics from the ML performance tests, (4).

   If the synthetic data closely mimics the source data, these values should be close to 1.

Frequently Asked Questions (FAQs)

Q: What can I use the synthetic data for?
A:

Q: How much is enough data to obtain high quality outputs?
A:

Q: Why did Roche build this application?
A:

Q: Will there be capabilities to generate synthetic longitudinal synthetic data?
A:

Q: Will there be privacy evaluation capabilities included in the app?

A:

Q: I tried to install the conda environment but failed when running the command `pip install -r requirements.txt`. How do I fix this?
A: Several packages that are in `requirements.txt` require valid access tokens to be downloaded from Roche. Please contact the Synthetic Data Squad for access.

Q: I tried to install synthpop but failed when running the command `install.packages("synthpop")` in `R`. How do I fix this?
A: This error may surface if some of the synthpop dependencies cannot be installed automatically. You can remedy this by

1. tracing back which packages that have failed installation, and
2. installing these packages manually.

Note that sometimes several synthpop packages could fail installation, and they each have to be installed manually.
Once you have installed these packages, you can run the command `install.packages("synthpop")` and experience a smooth installation.
Otherwise, you could also follow another installation method. Please see the section App **Installation: Installation: Install R from Source**.

Q: Who can I contact if I have further questions about the application?
A:

Submitting Feedback

We always welcome your feedback. Please submit your feedback to the Synthetic Data Generation squad via email: XXX@roche.com.