Synthetic Data Generation: A Comparative Study

Asha Mannarapotta Venugopal
University of Passau
Passau, Germany
asha.mannarapottavenugopal@uni-passau.de

Tung Son Tran
University of Passau
Passau, Germany
tungson.tran@uni-passau.de

Markus Endres
University of Passau
Passau, Germany
markus.endres@uni-passau.de

ABSTRACT

Generating synthetic data similar to realistic data is a crucial task in data augmentation and data production. Due to the preservation of authentic data distribution, synthetic data provide concealment of sensitive information and therefore enable Big Data acquisition for model training without facing privacy challenges. Nevertheless, the obstacles arise starting with acquiring real-world open-source data to effectively synthesizing new samples as genuine as possible. In this paper, a comparative study is conducted by considering the efficacy of different generative models like *Generative Adversarial Network* (GAN), *Variational Autoencoder* (VAE), *Synthetic Minority Oversampling Technique* (SMOTE), *Data Synthesizer* (DS), *Synthetic Data Vault with Gaussian Copula* (SDV-G), *Conditional Generative Adversarial Networks* (SDV-GAN), and *SynthPop Non-Parametric* (SP-NP) approach to synthesize data with regard to various datasets. We used the pairwise correlation and Synthetic Data (SD) metrics as utility measures respectively between real data and generated data for evaluation. Accordingly, this paper investigates the effects of various data generation models, and the processing time of every model is included as one of the evaluation metrics.

CCS CONCEPTS

• **General and reference** → *Surveys and overviews.*

KEYWORDS

Synthetic Data, Neural Networks, Generative Models

ACM Reference Format:
Asha Mannarapotta Venugopal, Tung Son Tran, and Markus Endres. 2022. Synthetic Data Generation: A Comparative Study. In *International Database Engineered Applications Symposium (IDEAS'22), August 22–24, 2022, Budapest, Hungary*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3548785.3548793

1 INTRODUCTION

Synthetic data refers to artificial information rather than those that are recorded from real-world events via direct measurement [7]. Artificial data is used when real data is not available, cannot be used due to privacy concerns and avoids business data vulnerable to data breaches.

IDEAS'22, August 22–24, 2022, Budapest, Hungary

ACM ISBN 978-1-4503-9709-4/22/08…$15.00
https://doi.org/10.1145/3548785.3548793

Using real data for different purposes like algorithm testing, Machine Learning training or various Data Science applications in business and industries suffer from different problems: original data is often highly secure, takes a lot of time to be accessible, and it cannot be used for testing hypothetical scenarios. Therefore, generating synthetic data is quite important for researchers and business developers to overcome real data usage restrictions. It allows to simulate not yet encountered conditions and it can be generated to meet specific needs or conditions that are not available in existing (real) data. Another familiar constraint is the lack of specific characteristics in a dataset required for certain applications or domains, which typically cannot be obtained effortlessly and economically.

In practice, often scanty authentic data serve as template from which synthetic data can be produced algorithmically, e.g. when privacy requirements limit data usage and variability. Typical applications where synthetic data is a "must-have" are autonomous driving, financial services, healthcare, model training, and consumer behaviour evaluation in marketing and social media analysis.

Research communities and organizations involved in Machine Learning development need adequate datasets consisting of various characteristics for experimental purposes. Large, accessible and privacy compromising-free datasets therefore are much desired. Nevertheless in many cases, real data is often not sufficient to comprise such a dataset that meets the demand required to carry out experiments and approaches due to numerous concerns as aforementioned. Missing values or corrupted records due to errors in measurement or data encryption and storage, data is too expensive to acquire due to technological constraints or consent requirements, are among the many popular contributing reasons. Synthetic data henceforth becomes a promising alternative to alleviate these limitations and opens up opportunities in a wide range of domains like privacy protection, image generation, healthcare, data mining, pattern recognition, etc.

In this paper, we present a comparative study in order to identify the most efficient data generation method according to certain use cases. Seven models were inspected, namely *Generative Adversarial Network* (GAN), *Variational Autoencoder* (VAE), *Synthetic Minority Oversampling Technique* (SMOTE), *Data Synthesizer* (DS), *Synthetic Data Vault with Gaussian Copula* (SDV-G), *Conditional Generative Adversarial Networks* (SDV-GAN), and *SynthPop Non-Parametric* (SP-NP). For each use case, identical configurations such as working environment, hardware, training dataset (Adult Census Data, Airbnb Data, and Airlines Data) from open sources and programs are applied to every model to ensure fairness in performance evaluation, which considers processing time, generation speed and generated data quality. To the best of our knowledge, many of these models have been widely applied in contemplate

works, but there have not been a research that examines their efficacy in comparison and thus, this becomes the objective of our contribution.

This paper is structured as follows: Section 2 describes our used data generation models, the architecture pipeline, and the used datasets. Details on the implementation strategies and challenges can be found in Section 3. All experiments and results are presented in Section 4. In Section 5 we discuss related work, and Section 6 contains a summarization.

2 BACKGROUND

In this section we provide an outline of our synthetic data generation pipeline. We give a brief overview on our architecture, the data preprocessing step, the used models and datasets. We also present details of our evaluation techniques.

2.1 Pipeline Architecture

Our workflow is defined by four steps (Figure 1), starting with *data collection and preprocessing* where the preprocessed data becomes the input of the training phase, in which various features and characteristics such as data types, value ranges, pattern and distribution, etc. are extracted and captured. Data quality requirements include

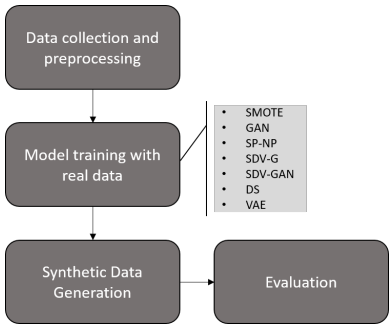


Figure 1: Pipeline architecture

the consistency, accuracy, integrity, timeliness, interpretability, and believability of the dataset. The raw data, which is the original form of the real data supposed to be used as the training data, most often comprises incomplete, inconsistent data and lack of values or attributes depending on the different types of datasets. Therefore, we transformed the raw dataset into a clean and understandable dataset that can be used for training with the generator models [11]. In order to achieve this, we performed *Data Collection*, *Data Cleaning*, and *Data Analysis and Extraction*. *Data Collection* involves gathering and measuring the featured variables of the datasets enabling the target outcome for the next phases such implementation and evaluation. *Data Cleaning* is applied to check and fill in the missing values, remove noise data, detect or remove outliers, and correct inconsistent data. *Data Analysis and Extraction* takes place by reviewing the data and checking them for missing values, and removing noise data if any.

Trained models subsequently can be readily used for inference tasks, where synthetic data can be generated based on user configurations queried to the models. Such configurations can specify desired sample quantity to be produced, value range as well as distribution. The data distribution present in the input schema (original data) is achieved to be similar as that of the output schema (generated data). Finally, the evaluation is conducted on the generated data to assess quality as well as to compare models performance according to certain evaluation metrics which will be further described subsequently.

2.2 Synthetic Data Generation Models

Seven frameworks considered in this study, namely *Synthetic Minority Oversampling Technique* (SMOTE), *Generative Adversarial Network* (GAN), *SynthPop Non-Parametric* (SP-NP), *Synthetic Data Vault with Gaussian Copula* (SDV-G), *Variational Autoencoder* (VAE), *Data Synthesizer* (DS) and *Conditional Generative Adversarial Networks* (SDV-GAN) are outlined in Table 1. A brief summary of each model is disclosed in Section 3, while their detailed descriptions are referred to the bibliography.

Table 1: List of used models

Year	Name	Full name	Ref.
2002	SMOTE	Synthetic Minority Oversampling Technique	[2]
2014	GAN	Generative Adversarial Network	[6]
2015	SP-NP	SynthPop Non-Parametric	[13]
2016	SDV-G	Synthetic Data Vault with Gaussian Copula	[14]
2017	DS	Data Synthesizer	[16]
2017	VAE	Variational Autoencoder	[8]
2019	SDV-GAN	Conditional Generative Adversarial Networks	[17]

2.3 Datasets

For our experiments we used datasets consisting of varying records. First, the *Adult Census Data*[1] (30,162 records with 11 attributes after data preprocessing) is the census data from 1994 based on income (originally possess 48,842 records with 14 attributes) with multivariate dataset characteristics consisting of both categorical and numerical data. Second, *Airbnb Data*[2] (213,451 records with 11 attributes after data preprocessing) has details of new users from Airbnb with demographics, web records and other statistics (initially with same number of records and 16 attributes) holding different type of attributes that consists of categorical, numerical and timeseries data. And third, *Airlines Dataset*[3] (1,046,595 records after data preprocessing) has flight travel details of adult passengers (1,048,575 records with 30 attributes before data preprocessing) with several attributes consisting of categorical and numerical data. We have choosen these datasets because of different type of attributes holding varying data distribution and enabling the betterment of implementation process in finding the advantages and disadvantages of each model while handling the datasets.

2.4 Evaluation Techniques

This section describes the two evaluation techniques used to evaluate the datasets on the compared models on the basis of it efficacy and utility. The used techniques are *Pairwise Correlation* and *SD Metrics*.

2.4.1 Pairwise Correlation and Proximity Value. Pairwise correlation is the correlation distance observed between two variables. Correlation distance is a famous method of estimating the distance

[1] UCI Machine Learning Repository (2019): https://archive.ics.uci.edu/ml/datasets/census+income

[2] Airbnb: https://www.kaggle.com/competitions/airbnb-recruiting-new-user-bookings/overview

[3] AirlinesCodrnaAdult: https://www.openml.org/d/1240

between two random variables with limited variances. When the distance of two items is zero, at that point they are the equivalent, and vice versa [1]. The method evaluates the correlation of real data and synthetic data. The concept behind this technique is to determine if the relationship between the variables in the real data are preserved in the generated synthetic data, which is done as follows:

(1) Firstly by observing the n variables of real data and storing it in a matrix `df_real.corr`.
(2) Then the n variables of synthetic data are observed and stored in a matrix `df_synth.corr`.

By exploring the output values from data generation, we check whether the pairwise correlation structure of the real data is firmly reflected by the correlation structure of the synthetic data and then determine the pairwise correlation distance for both real data and synthetic data and calculate the difference between them. To accomplish this, the correlation difference `diff` is calculated as follows:

```
diff = df_real.corr - df_synth.corr        (1)
```

For a generated dataset of good quality, diff should be close to '0', which is the **proximity value**. The *proximity values* of bad quality are far from '0'. Proximity is the measure of similarity and dissimilarity between the data points. The proximity value referred in this case mean the value obtained after the observation of correlation distance between real data and synthetic data [5].

2.4.2 SD Metrics. SD metrics are developed under the SDV project [4] [14] both to evaluate the relationship of synthetic data with real data and to test the quality of the former one. It is a dataset-agnostic tool that supports various data modalities such as single column, column pairs, single table, multi table and timeseries. It also includes a variety of metrics such as statistical, detection, efficacy, privacy, Bayesian Network and Gaussian Mixture. These metrics are a combination of several metrics such as CSTest (Chi-Squared), KSTest (Inverted Kolmogorov-Smirnov D statistic), KSTestExtended, LogisticDetection (Logistic Regression Detection), SVCDetection (Support Vector Classification Detection), BNLikelihood (Bayesian Network Likelihood), BNLogLikelihood (Bayesian Network Log Likelihood), LogisticParentChildDetection (Logistic Regression Detection), and SVCParentChildDetection (Support Vector Classification Detection). Here, values close to '0' mean that the data is not of good quality, and values close to '1' mean the data is of good quality. SD Metrics is available as a library in **Python**.

3 IMPLEMENTATION DETAILS AND CHALLENGES

In this section we present more details on the used models and also discuss implementation issues and challenges we had so solve in order to compare all models.

3.1 SMOTE

SMOTE [2] is an over-sampling technique that generates data for minority classes by inputting a sample from each of such classes. Then it creates synthetic samples based on its corresponding k Nearest Neighbors (kNN) in the feature space. The new samples are generated by multiplying the difference between the input feature vector and that of a selected nearest neighbor with a random value between 0 and 1, then add the result to the input feature vector. This results in the decision boundary of the minority classes becoming more general.

Implementation. We adopted the SMOTE technique with *Label Encoding* and *Logistic Regression*. Label Encoding is used in converting the labels (data in each row or column or cell) into numeric format making it easier for the Machine Learning models to process the dataset whereas Logistic Regression helps in predicting the variables which are categorically dependent. In order to generate duplicate/fake values, SMOTE requires to have one or two columns. This allows the data to be manipulated and to produce new random values. SMOTE is initialized for oversampling so that the data can be transformed. This is done with a random state of *2* and also by setting a *sampling_strategy* ratio to assign the number of samples to be generated because the data was earlier split into two forms, one with one or two columns and reference and the other set that includes all the features of all the columns.

Challenges. SMOTE faces certain challenges while generating synthetic data. The main challenges include overlapping of classes, creating additional noise and it is also hard to implement it for very high dimensional data. We overcome this by analyzing the samples and their respective labels from the results of Label Encoding. The hyperparameters were standard to all the 3 datasets where we have the random_state as 2 and k_neighbours as 1. By tuning the hyperparameters in random_state and k_neighbours, the similarity or differences in resulting values were very minute. SMOTE in general requires sufficient amount of samples to compare during the oversampling phase, we must ensure there are sufficient samples (specifically where we have more than 2 labels) in the columns (particular column attributes) of the dataset. This was done during the data preprocessing where we analyze the data and check if all the attributes hold sufficient number of samples possessing more than 2 labels.

3.2 GAN

Generative Adversarial Network (GAN) [6] is a Neural Network (NN)-based approach consisting of two components: a generative model G that learns the distribution from training data, then produces samples from it, and a discriminative model D that tries to distinguish the generated samples as real or fake. The objective of the training process is to maximize the error of the discriminator D on classifying samples produced by the generator G, thus inherently maximizing the synthesized samples similarity to authentic samples. Both models are trained independently, in which the generator gain more efficiency at producing similar (better) synthetic data, while the discriminator becomes increasingly skilled at identifying these data.

Implementation. The implementation of GAN mainly uses Label Encoding (similar like Synthetic Minority Oversampling Technique (SMOTE)) and Clustering. Clustering helps in dividing the data points into groups where the data points of the same group are similar to those in the other group. The batch size is set to 100 so that the whole set of data is handled in batches of 100 on each iteration. Next, two variables are initialized, one for the discriminator and the other for the generator with the value 1. It is mainly the number of discriminator network updates per adversarial training step and

[4] SD Metrics: https://github.com/sdv-dev/SDMetrics

number of generator network updates per adversarial training step, respectively. These act as the key hyperparameter arguments which are standard for all the datasets. Then the column labels and column data are initialized with the columns of the trained data for the class and if not for the column labels. The training data with no label is initialized with the training data of the data columns.

Challenges. In general, it is difficult to train GANs as they have several problems such as *non-convergence* and *mode collapse*. *Non-convergence* is when the parameters are unstable and do not converge. *Mode collapse* is when the generator collapses and generates limited variety of samples. There occurs an unbalance between generator and discriminator that causes overfitting and when it comes to hyperparameter selection, they become highly sensitive. To overcome this, it is important to balance between generator and discriminator to avoid overfitting in the training.

3.3 SP-NP

SynthPop Non-Parametric (SP-NP) [13] is a Python package of the original **R** package *SYNTHPOP* which uses the functions and structure of the mice multiple imputation package and extends it for the purpose of synthetic data generation. The basic functionality of *SYNTHPOP* is to generate synthetic forms of microdata containing sensitive information. It has two modes, namely parametric and non-parametric. Parametric is a mixture of logistic regression with linear regression, which uses binary, numeric, ordered and unordered factor of data types designated in a vector (default parametric method) and can be customized if needed. Non-parametric uses Classification and Regression Trees (CART) (default non-parametric method) that is based on classification and regression trees, which handles all types of variables having predictors and can be applied for all data types. In this paper, we use only the non-parametric version.

Implementation. The implementation of SP-NP involves Label Encoding, similar to SMOTE where the input data is encoded and given as input to the generator function. In general, SYNTHPOP requires dtypes (data types) as one of its main hyperparameters apart from the feature contents. So it is necessary to specify the data types of every column attribute in this dtypes respectively for each dataset. The results obtained from the generator is decoded to the original input data structure which serves to be our synthetic data.

Challenges. SYNTHPOP had to be customized for every single dataset based on their attribute type. It also does not work well with float and categorical data formats. This is the reason why we use Label Encoding to overcome this challenge.

3.4 SDV-G

Synthetic Data Vault with Gaussian Copula (SDV-G) [14] is a framework that generates synthetic data by utilizing a multivariate model derived from the intersection of various tables in a relational database. The modeling of the whole database is performed by taking input consisting of the data tables themselves and their corresponding metadata. The relationship between tables is computed recursively using Conditional Parameter Aggregation (CPA) to handle foreign key relations, while the relationship between columns of a table is computed using multivariate Gaussian Copula to calculate covariance. The framework is capable of performing model-based and knowledge based synthesis, where the former allows user to synthesize a complete database using solely the computed model, while the latter allows the completion of data based on some given information.

Implementation. The framework of SDV is advantageous for direct implementation for all types of dataset. Input and generated synthetic data are in the form of dataframes. It is also to be noted that Label Encoding can be used for all the SDV models. SDV-G is implemented directly by only assigning the Gaussian Copula model for the generator.

Challenges. In general, Gaussian Copula works only for linear correlation problems and not when there is tail dependence. Tail dependence is a concept of clustering of large events which are extremely important for risk management problems. So, for this task SDV-G servers to beneficial, but for clustering problems it is in question.

3.5 SDV-GAN

Conditional Generative Adversarial Networks (SDV-GAN) [17] is an adapted framework of SDV-G using Conditional GAN (CTGAN), a data generator to generate synthetic tabular (categorical) data based on GAN, which can handle varied feature types for both discrete and continuous data that claim to outperform the existing GAN models, Variational Autoencoder (VAE) and Bayesian Networks when applied on benchmarking datasets. This also includes some new techniques such as *mode-specific normalization* to augment the training process, change in architecture and to resolve the data imbalance by implementing *conditional generator* and *training by sampling*. We call this model as SDV-GAN since CTGAN is also a part of the SDV-G framework.

Implementation. Since the framework of SDV can be directly implemented for all types of datasets where the input data is in the form of a dataframe and the generated synthetic data is also in the form of a dataframe, the setup is similar to the implementation of SDV-G. So, SDV-GAN is also implemented directly by only assigning the CTGAN model for the generator.

Challenges. SDVs also have problems with producing complete results when the samples are too less for an attribute type. Larger datasets with sufficient amount of categorical data can overcome this issues and also smaller datasets, when trained multiple times, overcome this in certain cases.

3.6 VAE

VAE [8] is a NN-based approach capable of capturing complicated data distribution and produce synthetic data that are similar to the original data, which consists of two components: an encoder and a decoder. The former captures dimensional dependencies by mapping the input data into the latent space, while the latter takes the latent variable as input to generate samples. Mathematically, VAE is classified as a variational Bayesian method and thus is different in formulation to autoencoders despite the architecture similarity.

Implementation. The implementation of VAE is as difficult as GAN, escpecially for the purpose of generating categorical data. Here, we use Label Encoding just like SMOTE, GAN and SP-NP. Since VAEs already possess an encoder and decoder, the process of generation takes more time compared to any other model. But it is not possible to avoid the label encoding because the encoder and decoder functions result in some null values when implemented directly.

Challenges. The main challenge of VAE is the consumption of processing time. In future, we might be able to overcome this when implementing and running on a GPU environment.

3.7 DS

Data Synthesizer (DS) [16] is a framework consisting of three modules: a *Data Describer*, a *Data Generator* and a *Model Inspector*. The Data Describer inspects the input data types, correlations and attributes distribution to create a summary, from which synthetic samples are generated by the Data Generator. The Model Inspector is responsible for visualizing the summary, allowing accuracy evaluation and parameters adjustment. It has three modes: the default *correlated attribute mode*, the *independent attribute mode* for the cases of expensive correlation computation cost or inadequate samples, and the *random mode* for cases of exceedingly sensitive data.

Implementation. The framework of DS is implemented directly using the random mode as we need our data to be handled sensitively, but it has problems in generating timeseries data. We included the timeseries generation by using the datetime Python library. The main implementation is carried out by inputting the dataset file as a whole and drawing its features into a JSON file which acts a medium in generating the synthetic data having the same features as the real data. This is one of the reasons why it is time efficient. The threshold value is the hyperparameter for Data Describer which is standard for all the datasets and it is noted that this threshold is always less than the domain size when the attribute is categorical.

Challenges. Inability to generate timeseries is a major drawback to the model. But if this can be fixed using some additional timeseries generation methods, then this serves to be one very beneficial for synthetic data generation as the time consumption is very less.

4 EXPERIMENTS AND RESULTS

In this section, we present our comparative study and discuss the experiments conducted elaborately by describing and presenting the results obtained throughout the experimental phase [5].

4.1 Experimental Setup

The experiments are fourfold to the goals of this paper after successful completion of the implementation and evaluation. The experiments are mainly carried out on different datasets stored in CSV files, namely Adult Census Data (30,162 records), Airbnb Data (213,451 records) and Airlines Data (1,046,595 records). For more details we refer to Section 2.3.

The environmental setup for the experiment consists of Python 3.9, Jupyter Notebook with Anaconda3, PyCharm IDE, and required libraries installed on a Ubuntu platform. The Ubuntu platform is a server with Ubuntu 20.04, 754GB RAM, Intel(R) Xeon(R) Gold processor and CPU at 2.60GHz. The Data Preprocessing is done using Jupyter Notebook, Synthetic Data Generation and Evaluations are carried out on PyCharm.

4.2 Accumulated Results

The results obtained from the experiments are depicted in Tables 2 – 5 as well as in the form of heatmaps (Figures 2 – 4 in the Appendix). Finding the proximity and SD Metrics (cp. Section 2.4) is evaluated for each column of synthetic data against each column of real data. The models have good proximity, where '0' means we have a closer proximity and any value far from '0' means the proximity is low. Also we have a good SD Metric score, where '1' means the generated synthetic data is of good quality whereas '0' means the quality of data is low. The heatmaps represent the pairwise correlation distance between each column attribute of the synthetic data and each column attribute of the real data.

We performed different experiments on different datasets as described below:

(1) The first experiment begins with generating and evaluating the *Adult Census Data* with 10K records for the 7 models SMOTE, GAN, SP-NP, SDV-G, SDV-GAN, VAE and DS.
(2) The second experiment is with *full Adult Census Data* where we have 30,162 records for which we perform experiments for the models SMOTE, GAN, SP-NP, SDV-G, SDV-GAN and DS. We excluded VAE here because it took longer runtime of about 4 days and resulting in a value error occurred from the float point matrix of the generated dataset.
(3) The fourth experiment is for *Airbnb* Data (213,451 records) which includes all necessary types of attributes such as categorical, numerical and timeseries. Here, we performed experiments for the models SMOTE, SP-NP, SDV-G and DS and not for GAN, SDV-GAN and VAE, because the models produce memory error due to huge number of records.
(4) Finally, the last experiment is conducted on *Airlines Data* (1,046,595), which possess more than a million records on models SMOTE, SP-NP, SDV-G and DS. Similar to the Airbnb Data, we do not perform experiments on GAN, SDV-GAN and VAE.

4.2.1 Results on Adult Data. The first experiment with 10K records of Adult Census data is conducted and shows that the models SMOTE, GAN, SP-NP, SDV-G, SDV-GAN, VAE and DS are successfully implemented. From the represented results in Table 2, it is feasible to understand the performance nature and the comparison of the proximity levels and SD Metrics.

Table 2: Adult Census Data (10K records)

Adult Census Data (10K records)			
	Proximity Level	SD Metrics	Processing Time
GAN	0.0127	0.8564	26.2 minutes
VAE	0.1903	0.3704	21.2 hours
SMOTE	0.0055	0.7833	2.6 minutes
DS	0.0190	0.0967	0.4 seconds
SDV-G	0.0002	0.6434	7.0 seconds
SDV-GAN	0.0065	0.6830	3.8 minutes
SP-NP	0.0007	0.8595	2.6 minutes

Table 3 shows the results conducted on the Adult Census Data with 30,162 records, whereas the heatmaps (Appendix, Figure 2) represent the pairwise correlation distance of synthetic data and real data. The models SDV-G and SP-NP have better scores with respect to the proximity level, while SMOTE, GAN, SP-NP and SDV-GAN have better scores with respect to SD Metrics and in terms of processing time, DS and SDV-G outperform the other models, but SMOTE and SP-NP have resulted in a reasonable processing time.

[5] Source Code: https://github.com/ashamvenu/SyntheticData.git

Table 3: Adult Census Data Results

Adult Census Data			
	Proximity Level	SD Metrics	Processing Time
GAN	0.0402	0.8621	1.9 hours
SMOTE	0.0372	0.7296	8.9 minutes
DS	0.0219	0.0977	1.3 seconds
SDV-G	0.0029	0.6426	17.9 seconds
SDV-GAN	0.0100	0.7134	25.7 minutes
SP-NP	0.0054	0.8644	9.0 minutes

4.2.2 Results on Airbnb Data. We carried out the experiment on the Airbnb Data. The models of GAN, VAE and SDV-GAN failed to run, producing a termination of the program due to huge amount of data samples ultimately causing a memory error. Hence, this experiment was performed only on SMOTE, SP-NP, SDV-G and DS. The results (Table 4) obtained from Airbnb data show that SMOTE and SP-NP result in better proximity compared to DS and SDV-G. In terms of processing time, DS scores far better than the other three models. The heatmaps found in the Appendix, Figure 3, show the correlation distance between the real data and synthetic data for each column and feature in the Airbnb Data. Table 4 depicts the results in terms of proximity level and processing time of Airbnb Data. SD Metrics is not evaluated for Airbnb Data as the SD Metrics is not capable for large datasets and result in memory error as SD Metrics framework is designed to evaluate the dataframes of synthetic data and real data as a whole.

Table 4: Airbnb Data Results

Airbnb Data		
	Proximity Level	Processing Time
SMOTE	0.0036	1.5 hours
DS	0.0093	33.6 seconds
SDV-G	0.0102	2.6 minutes
SP-NP	0.0008	1.6 hours

4.2.3 Results on Airline Data. Similarly to Airbnb Data, the experiments are carried out only on SMOTE, SP-NP, SDV-G and DS models. The results (cp. Table 5) show that SMOTE and SP-NP result in better proximity compared to DS and SDV-G. Also here, the processing time of DS is far better than the other three models. Again, the heatmaps are shown in Figure 4 and present the correlation distance between the real data and synthetic data for each column in the Airlines dataset. Table 5 depicts the results in terms of proximity level and processing time of Airlines Data. Similar to Airbnb Data, we did not evaluate SD Metrics due to memory errors.

Table 5: Airlines Data Results

Airlines Data		
	Proximity Level	Processing Time
SMOTE	0.0148	7.7 hours
DS	0.0287	90.3 seconds
SDV-G	0.0282	11.9 minutes
SP-NP	0.0008	6.9 hours

5 RELATED WORK

Synthetic data generation has been a significant research topic for the past two decades. However, most of the research work is based on artificial image generation and the focus on text data has been rising only in the recent years. There are many papers dealing with synthetic data generation and we are not able to mention all of them. In this section, beside the models proposed in Section 2 and 3, we mention the most relevant work related to our study.

In a comparative work conducted by Dandekar et al. [4] on Linear Regression, Decision Tree, Random Forest and Neural Networks, the results show that NNs are more effective w.r.t. utility and privacy on the basis of running time. However, the evaluation also shows that the normalized Kullback–Leibler divergence scores are more or less the same for all the four models. Considering the importance of implementing flexible models for synthetic data generation by calculating nuances of multivariate structures, Manrique-Vallier and Hu [12] proposed a Bayesian non-parametric method to preserve complex multivariate data relationships between different variables subject to structured zeros by a tool called Truncated non-Parametric Latent Class Model (TNPLCM) using Full Conditional Specification (FCS) approach, CART method and Random Forests. This results in producing high quality of analytical data which exposes low risk. For this reason, we have used SP-NP in our research, which uses CART method for its non-parametric version.

Peng and Telle [15] published a complete tool for synthetic data generation, employing three algorithms namely Random Data Generation, Decision Tree and Multilinear Regression for different use cases depending on their respective data mining pattern. While being satisfactory on the aspect of software functionality, adequate evaluation of output quality and processing time was not provided.

Beside textual data, synthetic time series generation is also in demand, particularly in domains where sensor data analysis is involved such as healthcare applications. Dahmen and Cook [3] introduced a Machine Learning approach based on hidden Markov models and models. The generated data have been evaluated using time series distance measures against the authentic, manually annotated smart-home data and exhibited to be highly realistic as well as capable of improving the accuracy of the model it was used to train on. However, the results show only the accuracy of which we find the scores to be not satisfactory considering the quality of the generated data in comparison with the real data.

Lecanzo and Arias [10] introduced three different generation methods with traditional datasets based on item-based generative models, where two of the models Itemset Generating Model (IGM) and Interesting Itemset Miner (IIM) are over itemsets and the third one Latent Dirichlet Allocation (LDA) is using textual corpora. Evaluation is carried out based on characteristics (pattern similarity), preservation of itemsets, privacy and runtime. This determines the strength and weakness of each model in which IGM has the lowest learning phase runtime, while IIM scores best in data generation. Though the results depicts the runtime in seconds, the amount of data used is unclear to consider the evaluation.

Leduc and Grislain [9] proposed an architecture called Composable Generative Model (CGM) which is an auto-regressive model that inputs column embeddings through a transformer, evaluated using Synthetic Data Gym (SDGym) benchmark and claiming CGM

to be the best state-of-the-art approach. The concept of using an encoder, decoder and loss function seems reliable but the evaluation scores do not justify the claims with respect to synthetic data.

6 CONCLUSION AND DISCUSSION

In this paper we considered the challenge of generating synthetic data based on real data. The objective of our paper was to compare several state-of-the-art approaches to synthesize data in order to find out the efficiency of the proposed models. We used different real world datasets as a basis for our data-driven tasks, namely Adult Census data, Airbnb data, and Airlines data to analyze and depict the difference in performance as well as the behaviour of all seven models. We conducted several experiments of which the initial experiments show that SMOTE, SP-NP, SDV-G and DS are better among the 7 models chosen in terms of proximity, SD Metrics and processing time. GAN, SDV-GAN and VAE are not capable for large datasets, hence we run further experiments on SMOTE, SP-NP, SDV-G and DS to analyze which among these 4 models give us more promising results. By the end of all the experiments, we come to a conclusion that SMOTE and SP-NP are the most effective methods in terms of proximity and SDV-G and DS are effective in terms of processing time.

Our next step will be to consider models for text data generation, which is out of scope in this work due to being a more challenging task. Unlike the categorical data types, natural language generation involves language modelling which is a complicated and demanding process that requires more sophisticated model architectures, enormous amount of data, robust hardware to accommodate and longer training time. In addition, evaluating generated text quality is also an obstacle to be addressed, as identifying suitable metrics for automatic validation can be much less straightforward and counter intuitive, while human evaluation is certainly way too expensive to be included.

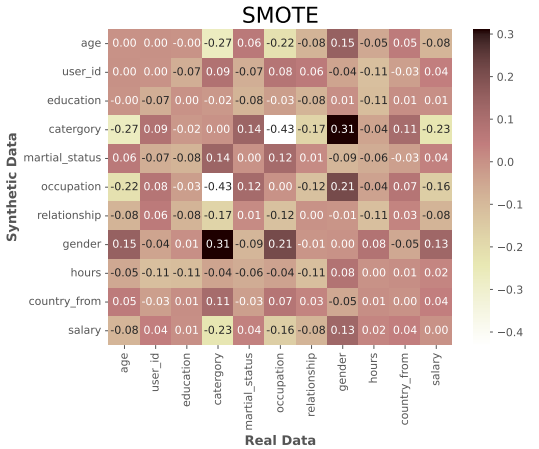
ACKNOWLEDGMENTS

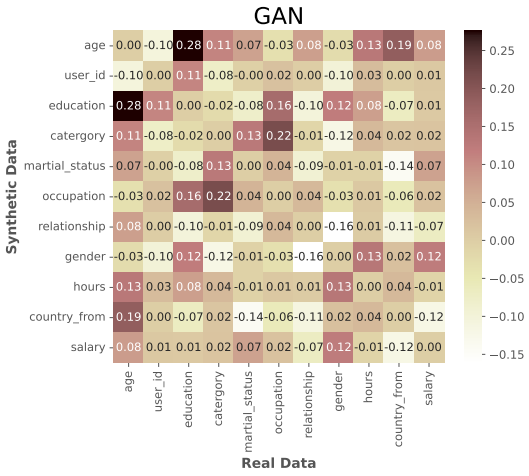
This research is part of the project "BigScience", which is funded by the Bavarian State Ministry for Economic Affairs, Regional Development and Energy under the grant number DIK0259/01.

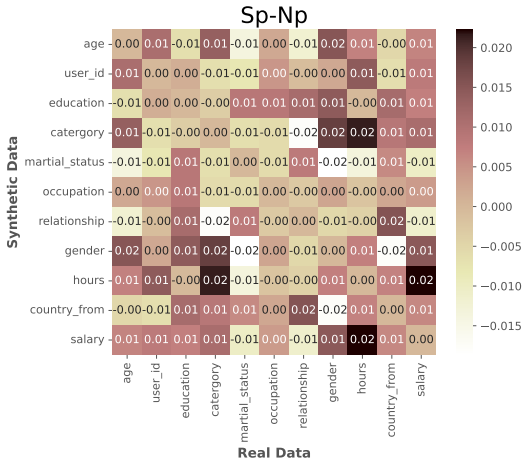
REFERENCES

[1] Atanu Bhattacharjee. 2014. Distance Correlation Coefficient: An Application with Bayesian Approach in Clinical Data Analysis. *Journal of Modern Applied Statistical Methods* 13, 1 (2014), 23.
[2] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. 2011. SMOTE: Synthetic Minority Over-sampling Technique. *CoRR* abs/1106.1813 (2011).
[3] Jessamyn Dahmen and Diane Cook. 2019. SynSys: A Synthetic Data Generation System for Healthcare Applications. *Sensors* 19, 5 (2019). https://doi.org/10.3390/s19051181
[4] Ashish Dandekar, Remmy A. M. Zen, and Stéphane Bressan. 2018. A Comparative Study of Synthetic Dataset Generation Techniques. In *Database and Expert Systems Applications*. Springer International Publishing, 387–395.
[5] B.S Everitt and David C. Howell. 2005. *Encyclopedia of Statistics in Behavioral Science*. Vol. 3. Wiley, 1621–1628.
[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. *Advances in neural information processing systems* 27 (2014).
[7] McGraw Hill and S.P. Parker. 2003. *McGraw-Hill Dictionary of Scientific and Technical Terms*. McGraw-Hill Education. https://books.google.de/books?id=xOPzO5HVFfEC
[8] Diederik P Kingma and Max Welling. [n.d.]. Auto-Encoding Variational Bayes. https://doi.org/10.48550/ARXIV.1312.6114
[9] Johan Leduc and Nicolas Grislain. 2021. Composable Generative Models. *CoRR* abs/2102.09249 (2021). arXiv:2102.09249 https://arxiv.org/abs/2102.09249
[10] Christian Lezcano and Marta Arias. 2020. Synthetic Dataset Generation with Itemset-Based Generative Models. *CoRR* abs/2007.06300 (2020). arXiv:2007.06300 https://arxiv.org/abs/2007.06300
[11] Jasdeep Singh Malik, Prachi Goyal, and Mr. Akhilesh K Sharma. 2010. A Comprehensive Approach Towards Data Preprocessing Techniques & Association Rules.
[12] Daniel Manrique-Vallier and Jingchen (Monika) Hu. 2018. Bayesian Non-parametric Generation of Fully Synthetic Multivariate Categorical Data in the Presence of Structural Zeros. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 181 (02 2018). https://doi.org/10.1111/rssa.12352
[13] Beata Nowok. 2015. synthpop : An R package for generating synthetic versions of sensitive microdata for statistical disclosure control.
[14] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. 2016. The Synthetic Data Vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 399–410. https://doi.org/10.1109/DSAA.2016.49
[15] Taoxin Peng and Alexander Telle. 2018. A Tool for Generating Synthetic Data. In *Proceedings of the First International Conference on Data Science, E-Learning and Information Systems*. Association for Computing Machinery, New York, NY, USA, Article 22, 6 pages. https://doi.org/10.1145/3279996.3280018
[16] Haoyue Ping, Julia Stoyanovich, and Bill Howe. 2017. DataSynthesizer: Privacy-Preserving Synthetic Datasets. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*. Article 42, 5 pages. https://doi.org/10.1145/3085504.3091117
[17] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular data using Conditional GAN. *CoRR* abs/1907.00503 (2019). arXiv:1907.00503 http://arxiv.org/abs/1907.00503

APPENDIX







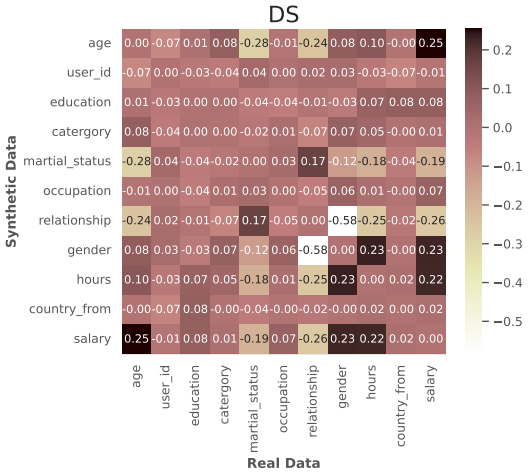
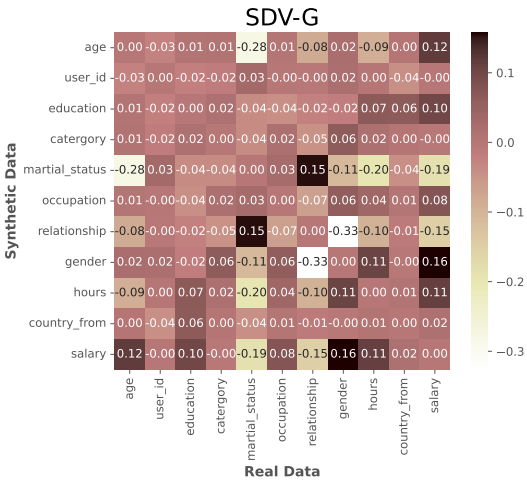
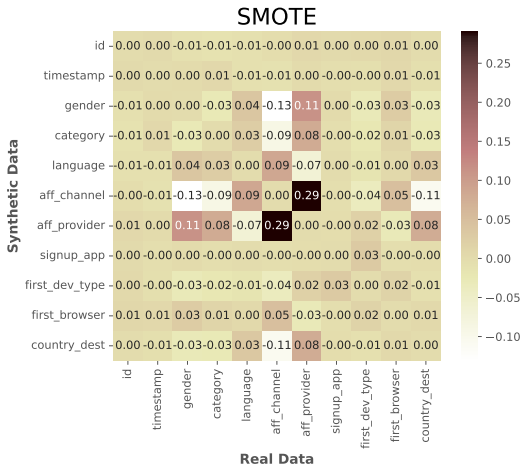
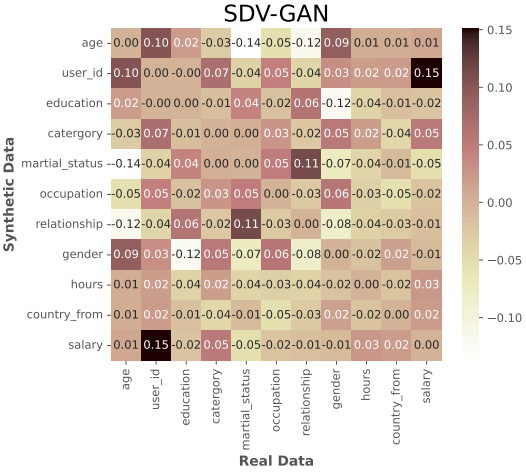
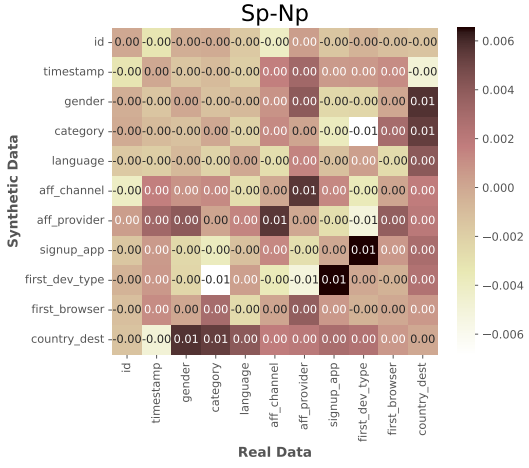


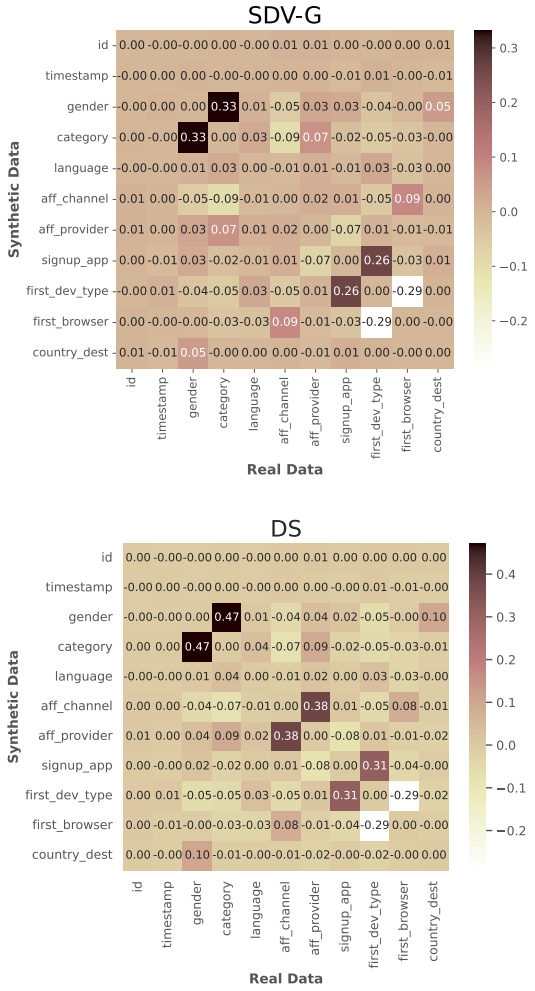
Figure 2: Heatmaps of Adult Data











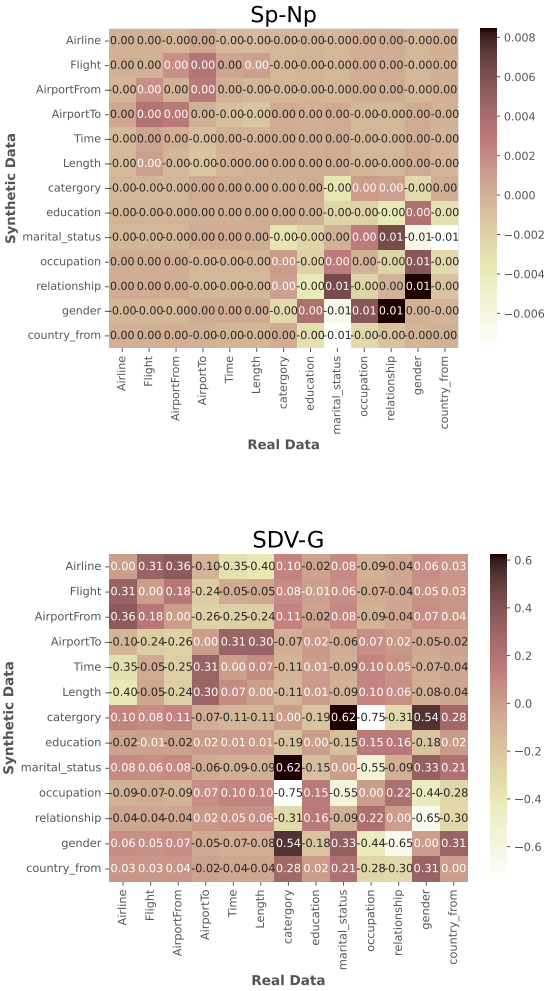
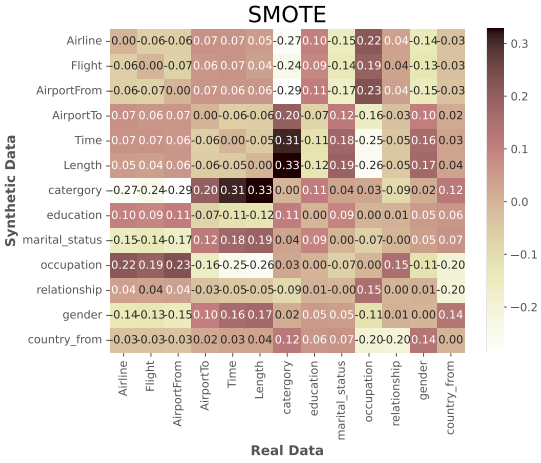


Figure 3: Heatmaps of Airbnb Data



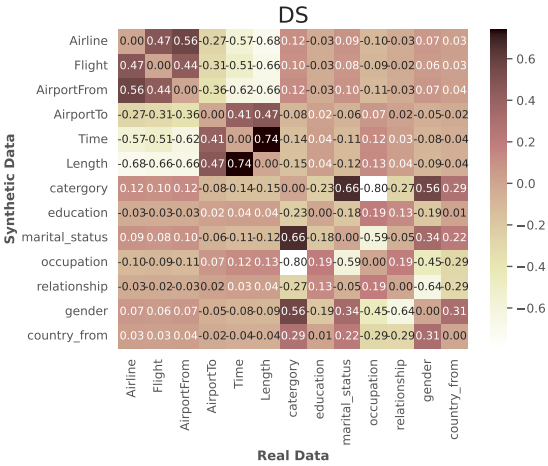


Figure 4: Heatmaps of Airlines Data