SYNTHETIC DATA GENERATION IN FINANCIAL INSTITUTIONS

M.Tech Final Semester Thesis Report

Under The Supervision of

Primary Supervisors

Debarshi Banerjee, Lead Vice-President, HSBC

Rajeev Pareek, Senior Asst. Vice-President, HSBC

Secondary Supervisor

Prof. Kuntal Ghosh, ISI Kolkata

Host company

HSBC

Submitted by

Arnab Das (CrS2109)

M.Tech, CrS, ISI Kolkata, Risk Analytics Intern, HSBC

CRYPTOLOGY AND SECURITY

INDIAN STATISTICAL INSTITUTE

203 B.T Road, Kolkata 700108

JULY, 2023

Abstract

Sharing financial data, particularly for research purposes outside institutions, is subject to stringent restrictions due to the sensitive and personally identifiable information it contains. Accessing data within financial institutions can also be time-consuming due to complex data sharing policies. This thesis aims to address these challenges by leveraging machine learning algorithms to generate synthetic data, enabling companies to adopt more flexible data sharing policies.

The thesis consists of three main experiments. Firstly, various open-source libraries that utilize different algorithms, such as Gaussian Copula and various types of Generative Adversarial Networks (GANs), are explored. The goal is to identify the most effective algorithms for generating synthetic datasets. A proposed method is developed to evaluate and select the best version of the synthetic datasets generated by these algorithms.

In the second experiment, a simple GAN structure is built from scratch, without relying on pre-existing libraries. This approach provides insights into the challenges and intricacies involved in implementing GANs for synthetic data generation.

The third experiment involves the use of a small dataset to explore and evaluate the applications of synthetic data. This step aims to demonstrate the feasibility and potential benefits of using synthetic data in various scenarios.

By generating synthetic data, this project aims to provide a viable alternative to original financial datasets, facilitating easier and more permissive data sharing within the company. The utilization of machine learning algorithms and the exploration of different approaches contribute to advancing the field of synthetic data generation and its practical applications.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and appreciation to my primary supervisors, Debarshi Banerjee and Rajeev Pareek, for their invaluable guidance, support, and expertise throughout the course of my M.Tech thesis. I am truly grateful for their mentorship and the opportunity to work under their supervision.I would also like to extend my sincere thanks to my secondary supervisor, Prof. Kuntal Ghosh, for his valuable inputs, suggestions, and expertise. I would like to acknowledge HSBC for providing me with the opportunity to pursue my M.Tech thesis in collaboration with their organization. The exposure and support I have received from the professionals at HSBC have been immensely valuable, and I am grateful for their guidance and cooperation throughout my research.

I would also like to express my gratitude to my family and friends for their unwavering support.Lastly, I would like to thank all the individuals who have directly or indirectly contributed to the successful completion of my thesis. Your assistance, advice, and encouragement have been greatly appreciated.

Place: Kolkata Arnab Das

Date: 04.07.2023

Contents

Chapter 1

INTRODUCTION

1.1 Motivation:

1.2 Motivation:

The financial services sector is experiencing a technological revolution in which data related business models are changing the way the industry functions. Because the data maintained by these institutions is highly sensitive, secure, trustworthy, and efficient handling of information will form the foundation of a modern financial institution. Aside from being sensitive, data is also complex and is frequently held in multiple silos inside organizations. Part of the reason for this is due to regulatory regulations, which limit data sharing across departments and outside of the corporation. It is either prohibited for privacy reasons or excessively time-consuming due to the clearance procedure [1]. This reduces the data's usability, which eventually harms the companies' value proposition. As a result, it is essential.For this reason, approaches that might streamline, enhance, and reduce the dangers of data sharing are of major importance to financial institutions and other regulated industries.

Generating synthetic data can be a solution to this problem as it does not contain any real data. In addition to for a financial institutions like HSBC can also use synthetic data for following business and technical purposes:

Lack of historical data: Historical data is often essential for training machine learning models, but in certain situations, obtaining a sufficient amount of real-world data may be challenging or impossible.As a large number of realistic Synthetic data can be generated from a small number of dataset, we can handle the situation of lack of historical data.

Scenarios generation: Synthetic data can be used to create simulated economic scenarios,such as recessions, market fluctuations, or policy changes and also can be utilized to perform stress testing, which involves simulating extreme or adverse scenarios to evaluate the resilience of financial models, portfolios, and risk management systems. It helps identify vulnerabilities and assess the institution's ability to withstand adverse market conditions.

Realistic relational database design: Sometimes HSBC may require to have a relational database for software testing or they need to share data for analytical purposes with a vendor or internal hackathon and here synthetic data may lead to the solution of generating relational database.

Conditional Generation: Conditional generation of synthetic data can help to improve training of machine learning algorithm used for testing and refining their fraud detection algorithms, developing personalized marketing strategies, or assessing the impact of new products or services on different customer segments

Missing value imputation: By providing the value of known columns we can generate by conditional sampling the unknown columns to tackle with missing value imputation problem.

We have a difficulty with stress testing at the financial institute due to a lack of data. Secure data governance is critical in the public eye, as illustrated by the reaction towards the Cambridge Analytica controversy cite2 and the ensuing financial market response cite3. Organizations may afford a more freed data sharing strategy using synthetic data, enabling better collaboration both internally and outside.

1.3 Problem Statements :

The goal of this project is to investigate how synthetic data can be generated and what open source libraries are available, as well as to evaluate the quality of synthetic data generated by different algorithms and to try to modify the results obtained by applying different algorithms.
approaches are classified into two types: process-driven approaches and data-driven methods cite6. The former is derived from a mathematical or computer model of the underlying physical process. Consider agent-based modeling or Monte-Carlo simulations for example.Methods based on data,on the other hand,create artificial data using generative models that have been trained on observed data. We will discuss about these two methods and try to generate the best quality of synthetic data out of these two worlds.Statistical methods has some restriction regarding complexity of data and data size [8] . The topic of whether GANs would outperform their statistical counterparts arose, and various articles have attempted to address it [7, 8]. Xu and Veeramachaneni [7]evaluate GANs to numerous statistical techniques on various data sets and show that GANs beat those conventional approaches in most tasks. GAN's non-convergence and instability hamper the training process6.The non-convergence dilemma forces the user to stop the training blindly because the loss values from the architecture's components don't give any helpful information about the model's intended use-case. Due to this characteristic and the volatile behavior of GAN, there is a significant risk that the model will stop working properly and perform poorly.

Also we'll discuss about the acceptance of synthetic tabular data generated by these methods in the real world scenario and how we can use these synthetic data to solve some practical examples.All in all this leads to following questions:
(Q1): Can we use Gaussian copula to improve the result achieved by GAN and generate the a more closer distribution to the original dataset and can it be possible to use that dataset as a substitute of original one ?
(Q2): If we want to create our own library inside HSBC what should be our steps?
(Q3): How synthetic data can help HSBC in practical scenario?

1.4 Contribution

1.4.1 Hybrid Algorithm: Improving the Quality of Synthetic Data

Statistical methods work well for datasets with a small number of rows and distribution of variables has less number of modes and also number of distinct classes corresponding to a particular categorical variable is less. The neural network methods like CTGAN is able to predict multimodal distribution well and it is highly effective to prevent modal collapses.[7] .The first contribution is to take the best column between two generated datasets and make a hybrid dataset so that the quality of synthetic data has been improved.

1.4.2 Exploring Different Approaches for Synthetic Data Generation

Five different open source libraries has been used to generate synthetic data based on two different type of datsets and also compared their quality. Also tried to get the best result using hybrid algorithm approach.

1.4.3 Creating a Simple GAN for Tabular Data Generation: Challenges and Insights

A simple GAN has been created for generating tabular data to determine what the challenges are to be faced if we want to generate synthetic data using GAN. Also this provides a view what the necessary steps are to create a library for generating synthetic data.[1]

[1]this work is available on `https://github.com/iamarnabdas/mthesis-synth-hybrid/tree/master`

Chapter 2

Preliminaries

To generate synthetic data we have two types of approaches and one of them is statistical methods. In statistical method we'll try to guess the joint distribution of the original data and sample data from the parameters of the original distribution of the real dataset.There is another approach, that is machine learning. In machine learning we'll use neural network techniques specially deep learning algorithms like GAN(generative adversarial network) and VAE(variational autoencoder).In general synthetic data has been used for the purpose of anonymization and software testing. In some cases, these two methods also has been overlapped. In the recent days, with the rise of big data, privacy of data is a big factor and good quality of realistic synthetic data is a holy grail .

2.1 Statistical Methods:

2.1.1 Gaussian Copula:

Gaussian copula is a special type of copula function.We will first understand what actually copula does and then discuss about Gaussian copula and how it'll be used to genrate synthetic data.

Copula:

Copula means coupling together.Copulas are functions that can take n univariate distributions and can map to a corresponding joint multivariate distribution. In particular, by defining the copula and marginal distributions independently, any multivariate distribution can be created.

Definition:
A d dimensional copula $C : [0,1]^d \to [0,1]$ is cumulative distribution function(CDF) with uniform marginals **u**$= (u_1, u_2, ..., u_n)$ and it has the following properties:

1. $C(u_1, ..., u_n)$ for each component u_i is a non decreasing function
2. The i^{th} marginal distribution is obtained from providing the value of $u_j = 1$ and $i \neq j$

$$C(1, .., u_i, .., 1) = u_i$$

3. For $a_i \leq b_i$, $P(U_1 \in [a_1, b_1], \ldots, U_d \in [a_d, b_d])$ must be non-negative. This implies the rectangle inequality

$$\sum_{i_1=1}^{2} \cdots \sum_{i_d=1}^{2} (-1)^{i_1+\ldots+i_d} C(u_{1,i_1}, \ldots, u_{d,i_d}) \geq 0$$

where $u_{j,1} = a_j$ and $u_{j,2} = b_j$.

In this we need to know the most important theorem related to Copula is **Sklar's Theorem**.

Theorem(Sklar's Theorem):

Let us consider any joint CDF H with marginal distribution $H_1, H_2, ..., H_d$ then there exists a copula $C : [0,1]^d \to [0,1]$ such that

$$H(x_1, \cdots, x_d) = C(H_1(x_1), \cdots, H_d(x_d))$$

Here the domain of C is $Ran(H_1) \times Ran(H_2) \times \cdots \times Ran(H_n)$ where $Ran(H_i)$ is the range of H_i

Gaussian Copula:

The Gaussian copula is a type of copula that is based on the Gaussian distribution, also known as the normal distribution. It is commonly used to model the dependence structure between random variables, independent of their marginal distributions. The Gaussian copula function, denoted as C_ρ, is defined as follows:

$$C_\rho(u_1, \ldots, u_d) = \Phi_\rho(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d))$$

where $u_1, \ldots, u_d$ are the probabilities associated with the corresponding random variables, and Φ_ρ is the cumulative distribution function of the multivariate Gaussian distribution with correlation matrix ρ. The function Φ^{-1} represents the inverse of the cumulative distribution function of the standard univariate Gaussian distribution.

The correlation matrix ρ determines the strength and nature of the dependence between the random variables. A positive correlation indicates a positive dependence, while a negative correlation represents a negative dependence. The Gaussian copula allows for capturing various types of dependence structures, including linear and nonlinear relationships.

2.1.2 Bayesian Network:

A Bayesian network is a graphical representation of the probabilistic relationships among a set of variables. It is sometimes referred to as a belief network or a probabilistic graphical model. It is made out of directed edges and nodes that indicate the interdependence between variables. Bayesian networks are frequently employed for inference in uncertain situations and complicated system modeling.

When it comes to generating synthetic data, Bayesian networks offer an effective approach. Given a set of observed data, a Bayesian network can learn the underlying probabilistic dependencies between variables. This is typically done through a learning algorithm, such as the maximum likelihood estimation or Bayesian parameter estimation. Once the network structure and parameters are learned, synthetic data can be generated by sampling from the learned distribution.

To generate synthetic data using a Bayesian network, one can start by sampling values for the root nodes, which have no parent nodes. Then, values for the child nodes are sampled based on the conditional probability distributions defined by the network. This process is repeated for each variable in the network until all variables have been assigned values, resulting in a complete synthetic dataset.

By using a Bayesian network, the generated synthetic data can preserve the statistical properties and dependencies observed in the original data. This is because the network structure captures the probabilistic relationships between variables, allowing for the generation of data that closely resembles the original dataset. Furthermore, the flexibility of Bayesian networks enables the modeling of complex dependencies and the incorporation of prior knowledge into the data generation process.

It is worth noting that the accuracy and reliability of the synthetic data generated from a Bayesian network depend on the quality of the learned network structure and parameters. Proper model selection and validation techniques, along with careful consideration of the data characteristics, are essential to ensure the generated data accurately represents the underlying distribution.

In summary, Bayesian networks provide a powerful tool for generating synthetic data by capturing and utilizing the probabilistic dependencies between variables. They enable the generation of data that preserves statistical properties and dependencies observed in the original dataset, making them valuable in various applications where privacy and data integrity need to be maintained.

2.2 Generative Adversarial Network:

In 2014, Ian Goodfellow et al. introduced Generative Adversarial Networks (GANs) [29]. Two hostile "players" battle in a minimax game using the game theory-based GAN neural network design. The main concept underlying GANs is to pit a generative model against a discriminative model.

The generative model aims to generate synthetic data samples that closely resemble real data. On the other hand, the discriminative model is trained to distinguish between the synthetic data generated by the generative model and the real data.

The competition between the generative and discriminative models drives them to improve their respective methods iteratively. The generative model learns to generate synthetic samples that become increasingly difficult for the discriminative model to differentiate from real data. Simultaneously, the discriminative model learns to improve its ability to correctly classify samples as real or synthetic.

GAN Formally

GANs consist of a generator model G and a discriminator model D. The generator takes random noise z from a Gaussian distribution $p_z(z)$ as input and maps it to synthetic data samples $\tilde{x}$. The discriminator aims to distinguish between real data samples x and the synthetic samples generated by the generator.

Generator Structure:

The generator transforms the input noise z into synthetic samples $\tilde{x}$ using a function $G(z; \theta_g)$, where θ_g represents the parameters of the generator model.

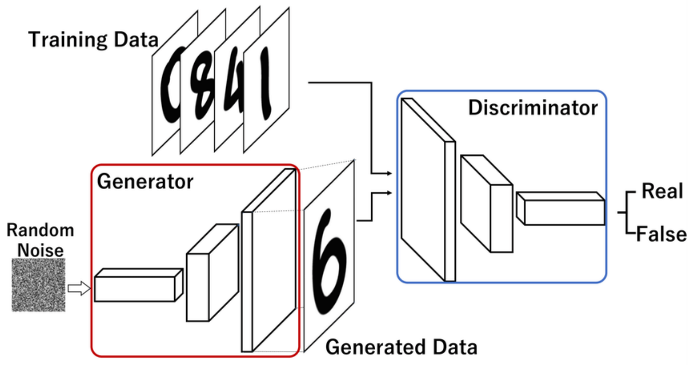


Figure 2.1: This is an example that how GAN works for MNIST dataset.The generator generates data from random noise and sends to discriminator along with real data sets and discriminator tries to separate them.From [30]

Discriminator Structure:

The discriminator model $D(x; \theta_d)$, parameterized by θ_d, predicts the probability that a given input sample x is real.

Loss Function for GAN:

The generator and discriminator play a minimax game to determine the GAN aim. The generator seeks to reduce the discriminator's capacity to discriminate between actual and synthetic samples, while the discriminator seeks to maximize its likelihood of accurately classifying genuine and synthetic samples.

The discriminator loss function L_D can be defined as:

$$L_D = -\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(D(x))] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))].$$

The generator loss function L_G is formulated as:

$$L_G = -\mathbb{E}_{z \sim p_z(z)}[\log(D(G(z)))].$$

KL Divergence and JS Divergence:

The Kullback-Leibler (KL) divergence and Jensen-Shannon (JS) divergence are commonly used to measure the dissimilarity between probability distributions.

The KL divergence between the true data distribution $p_{\text{data}}(x)$ and the generator distribution $p_g(x)$ is given by:

$$\text{KL}(p_{\text{data}}(x) || p_g(x)) = \int p_{\text{data}}(x) \log \left(\frac{p_{\text{data}}(x)}{p_g(x)} \right) dx.$$

The JS divergence, which is symmetric, can be calculated as the average of the KL divergence between $p_{\text{data}}(x)$ and $\frac{p_{\text{data}}(x)+p_g(x)}{2}$ and the KL divergence between $p_g(x)$ and $\frac{p_{\text{data}}(x)+p_g(x)}{2}$.

Vanishing Gradient:

One challenge in GAN training is the issue of vanishing gradients, where the gradients become extremely small during backpropagation, hampering the learning process. This issue can arise when the generator and discriminator models have poor initialization or when they diverge too far from each other.

To mitigate the vanishing gradient problem, techniques such as careful weight initialization, alternative loss functions, and architectural modifications have been proposed, such as using batch normalization or avoiding certain activation functions that are prone to vanishing gradients.

Chapter 3

BACKGROUND

3.1 Anonymization Methods

Anonymizing data methods can be categorized into two main types: syntactic and synthetic methods. Syntactic methods involve anonymizing data at the attribute level by employing techniques such as data removal, generalization, masking, or combining, aiming to prevent individual records from being identified. Well-known techniques like K-anonymity and L-diversity have been developed to meet these privacy requirements. K-anonymity ensures that a minimum number of K individuals cannot be distinguished from each other, while L-diversity focuses on enhancing privacy by adding diversity to sensitive attribute values. However, it's important to note that as the privacy constraint increases, the anonymized data further deviates from the original data, resulting in a reduction in data utility. This trade-off between privacy and utility was exemplified in the case of Netflix in 2008. The company released movie ratings for a data analysis competition, but researchers successfully de-anonymized the dataset using background knowledge from IMDB, leading Netflix to cancel the competition.
To address the challenges of achieving both privacy and data utility, recent research has turned to the generation of synthetic data as a more practical but complex method. Synthetic data generation involves approximating data points from the original distribution, either explicitly or implicitly, without removing any information from the dataset. This approach offers the advantage of preserving data utility while ensuring privacy. However, modeling synthetic data accurately is a more challenging task. The privacy level of synthetic data remains an active area of research, with some proponents claiming that it provides no room for information leakage, while others maintain a more cautious stance. Concerns primarily revolve around the risk of neural networks being overtrained, potentially leading to the inadvertent disclosure of training data. Although previous research on Generative Adversarial Networks (GANs) has not exhibited significant signs of this risk, the absence of a formal privacy guarantee in their design means that the possibility cannot be entirely ruled out.
To enhance the credibility of synthetic data generation, researchers have introduced privacy algorithms into the models. One promising approach is differential privacy, where noise is injected during the calibration process to mitigate the risk of data leakage. However, similar to K-anonymity and L-diversity, incorporating differential privacy measures can impact data utility. Given that synthetic data generation with deep learning is still in its early stages, there is optimism for promising results in the near future as researchers continue to explore and refine these techniques.

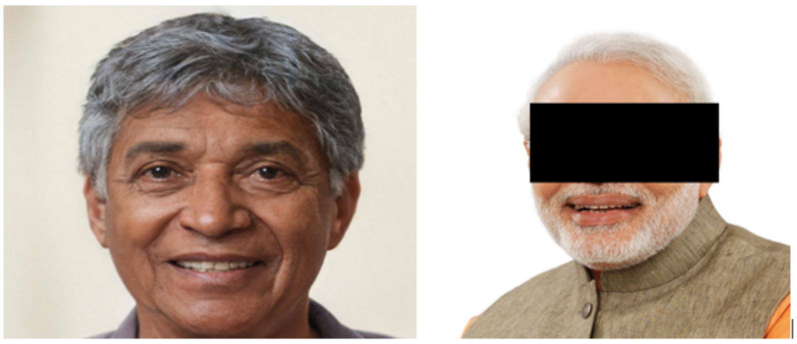


Figure 3.1: Illustrative example of the difference in data utility between syntatic and synthetic anonymization. The left image is artifical, generated with styleGAN

3.2 Open Source Libraries

3.2.1 Synthetic Data Vault (SDV)

The Synthetic Data Vault (SDV) is an ecosystem of libraries designed for synthetic data generation using PyTorch. Developed by students from MIT and now under the stewardship of their startup DataCebo [21], SDV provides a user-friendly framework to learn from and generate synthetic data that matches the format and statistical properties of the original dataset.

SDV offers a wide range of modeling techniques for different types of datasets. For single-table datasets, it includes Gaussian Copula [7], CTGAN [9], and TVAE [22] models. For sequential or time series datasets, SDV incorporates the PARSynthesizer [23], which can preserve the temporal relationships and patterns. Additionally, for multi-table datasets, SDV provides the HMASynthesizer, allowing the generation of synthetic data that maintains the inter-table relationships.

One notable feature of SDV is its ability to perform conditional sampling, where users can generate synthetic data based on predefined relationships between columns. This feature enables users to create synthetic datasets that adhere to specific conditions or constraints.

To address privacy concerns, SDV offers anonymization capabilities. Users can specify sensitive columns in the dataset, and SDV will anonymize those columns while preserving the overall patterns and characteristics of the original data. The library also supports the integration of Faker , a popular Python library for generating fake data, to further anonymize columns.

To evaluate the quality and privacy of the synthetic data generated by SDV, the library includes SDmetrics. SDmetrics provides various evaluation metrics to assess the fidelity of the synthetic data to the original dataset and measure the privacy guarantees offered by the synthetic data generation process.

Moreover, SDV provides the option to save a parameter file instead of the entire synthetic dataset. This parameter file contains the learned model and can be used to generate as many rows as desired in a different environment, reducing storage requirements and facilitating the reproducibility of the synthetic data generation process.

In summary, the Synthetic Data Vault (SDV) offers a comprehensive ecosystem of libraries for synthetic data generation. With its diverse modeling techniques, anonymization capabilities, evaluation metrics, and flexibility in generating synthetic data, SDV empowers researchers and practitioners to create privacy-preserving synthetic datasets that retain the statistical properties of the original data.

3.2.2 DataSynthesizer

DataSynthesizer is a tool that generates synthetic data while preserving differential privacy [18]and it needs data to be in 1NF [18] Its standout feature is its usability, as it eliminates the need for data owners to specify parameters to start generating and sharing data securely.

DataSynthesizer is made up of three main parts: DataDescriber, DataGenerator, and ModelInspector. DataDescriber examines the private dataset's data kinds, correlations, and attribute distributions and generates a data summary by adding noise to the distributions to maintain privacy. The resulting summary is then sampled by DataGenerator to provide synthetic data. ModelInspector gives an understandable explanation of the data summary, allowing data owners to evaluate the accuracy of the summarizing process and, if necessary, make parameter revisions.

DataSynthesizer offers three operational modes. In the correlated attribute mode, it learns a differentially private Bayesian network to capture attribute correlations and samples from this model to construct the synthetic dataset. When computational complexity or insufficient data hinders the correlated attribute mode, the independent attribute mode can be used. This mode derives histograms for each attribute, adds noise to achieve differential privacy, and then draws samples for each attribute. Alternatively, the random mode can be employed for highly sensitive data, which simply generates random values consistent with the attribute types.

With its emphasis on usability and differential privacy preservation, DataSynthesizer provides a practical solution for generating synthetic data with varying levels of attribute correlations and privacy requirements.

3.2.3 Synthpop

Synthpop is an open-source library that specializes in generating synthetic microdata for survey datasets. It utilizes statistical modeling techniques to capture the complex relationships and distributions within the original data. Synthpop is the Python implementation of the R package *synthpop* [24].

Synthpop can synthesize univariate synthetic data using decision tree classifiers and classification and regression trees (CART). The procedure for synthesis by a CART model is as follows:

1. Fit a classification or regression tree through binary recursive partitioning.
2. For each matrix of synthesized covariates (k x p), find the terminal node.
3. Randomly draw a donor from the members of the node and take the observed value of the target variable (y) from that draw as the synthetic value.

3.2.4 nbsynthetic

nbsynthetic is a project developed by *Nextbrain.ml* [25] to generate high-quality synthetic data from small or medium datasets with poor data quality, such as a larger number of missing values. The project utilizes Vanilla GAN or WGAN for generating synthetic data without any conditioning for GAN models.

However, nbsynthetic has a limitation when the input data is highly dimensional and contains only numerical features.

3.3 Related Work:

In the past ten years, there has been a significant advancement in the generation of synthetic data by modeling joint multivariate probability distributions and sampling from them. This approach involves treating each column in a table as a random variable and using various modeling techniques to simulate different types of variables.

One commonly used technique is the use of decision trees and Bayesian networks to simulate discrete variables in the synthetic data generation process [10]. These models provide a structured approach to capture the dependencies and interactions among the variables. Additionally, spatial decomposition trees have been employed for modeling spatial data, which has found applications in fields such as geospatial analysis and environmental sciences [11].

When dealing with continuously associated variables that are not necessarily linearly correlated, copulas have proven to be a valuable tool in the generation of synthetic data [9]. Copulas allow for modeling the joint distribution of variables while capturing their individual marginal distributions and dependence structure.

However, the quality of synthetic data generated using these models is influenced by the limitations inherent to the chosen distributions and computational constraints. These factors can impact the accuracy and representativeness of the synthetic data.

In recent years, generative models such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and their extensions have emerged as powerful techniques for synthetic data generation [7, 8, 13]. These models offer performance and flexibility in data representation, enabling the generation of high-quality synthetic data that captures complex patterns and distributions.

GANs, in particular, have gained popularity in generating tabular data, including healthcare records [14]. For instance, GANs have been used to generate continuous time-series medical records, allowing researchers to generate realistic data for analysis and testing purposes. Another notable example is medGAN, which combines an auto-encoder and a GAN to generate heterogeneous non-time-series continuous and/or binary healthcare data [6].

To address the challenge of producing synthetic data that preserves the quality of the label column, tableGAN employs a convolutional neural network to optimize the generation process [12]. This approach ensures that the generated synthetic data maintains the desired properties in the label column, enabling the use of the data for training classifiers and other machine learning tasks.

In summary, the field of synthetic data generation has witnessed significant progress in the past decade, with advancements in modeling techniques such as decision trees, Bayesian networks, copulas, VAEs, and GANs. These techniques have enabled the generation of high-quality synthetic data for various applications, including healthcare, spatial analysis, and more.

Chapter 4

METHODOLOGY AND WORK DONE

This chapter will provide an idea of basic structures of the algorithms that we have used.And also in the last part it'll provide about the experiments that we have done.

4.1 Generation Algorithms

4.1.1 CTGAN

Conditional tabular GAN(CTGAN)[7] is a follow up paper of TGAN [26].But TGAN and CTGAN are not similar , they are different models .In the first part, the paper discusses three main challenges encountered when using GANs for tabular data generation tasks. These challenges are as follows:

1. **Mixed data types and non-Gaussian distribution**: Tabular data often consists of both discrete and continuous columns. Generating a mix of these column types requires GANs to apply softmax for discrete columns and tanh for continuous columns. This presents a challenge because continuous values in tabular data typically have non-Gaussian distributions, making min-max transformations problematic due to the vanishing gradient problem.

2. **Multimodal distributions**: Many continuous columns in real-world datasets exhibit multimodal distributions, meaning they have multiple modes. Vanilla GAN models struggle to capture and model all modes accurately, as demonstrated by previous research. Therefore, generating synthetic tabular data that accurately represents the multimodal distributions of continuous columns is a challenge.

3. **Learning from sparse one-hot-encoded vectors**: In tabular data, categorical variables are often represented as one-hot-encoded vectors, where each category is assigned a binary value. However, when training a generative model to generate synthetic samples, softmax is typically used to generate a probability distribution over all categories. This creates a problem because a discriminator can easily distinguish between real and fake data by checking the sparseness of the distribution instead of considering the overall realism of the entire row.

4. **Highly imbalance data**: In some datasets, certain categorical columns exhibit severe imbalances, where the majority category appears in over 90% of the rows. This imbalance can lead to mode collapse, as minor categories have insufficient representation and can be easily missed by the discriminator, resulting in a biased and unrealistic synthetic data distribution.

Data Pre-processing

CTGAN addresses the challenges of categorical and continuous data representation in the following ways:

CTGAN employs *mode-specific normalization* to handle non-Gaussian and multimodal distributions in continuous attributes. The process involves the following steps:

1. A Variational Gaussian Mixture Model (VGMM) estimates the number of modes m_i for each continuous column C_i by fitting a Gaussian mixture. This results in a probabilistic model $P_{C_i}(c_{i,j}) = \sum_{k=1}^{m_i} \mu_k \mathcal{N}(c_{i,j}; \eta_k, \phi_k)$, where μ represents the weights, η_k are the modes, and ϕ the standard deviations of the modes.

2. For each value $c_{i,j}$ in C_i, the probability of $c_{i,j}$ coming from each mode is computed and $\rho_k = \mu_k \mathcal{N}(c_{i,j}; \eta_k, \phi_k)$ are the probability densities. mode is drawn from the calculated probability density., and $c_{i,j}$ is represented as a one-hot vector denoting the mode and a scalar $\alpha_{i,j}$indicating the value within the mode.

The finally we get by concatenating the continuous and discrete columns as follows:

$$r_j = \alpha_{1,j} \oplus \beta_{1,j} \oplus \ldots \oplus \alpha_{N_c,j} \oplus \beta_{N_c,j} \oplus d_{1,j} \ldots \ d_{N_d,j}$$

Here, $d_{i,j}$ represents the one-hot encoding of the categorical values.

Conditional Generator and Training by Sampling

To address the challenge of even sample distribution in discrete columns, CTGAN introduces a conditional generator. The conditional generator transforms the input to the original distribution given a condition on a specific discrete value. The probability of a row is computed as:

$$P(\text{row}) = \sum_{k \in D_{i^*}} P(\text{row}|D_{i^*} = k^*) P(D_{i^*} = k)$$

Here, k represents values from the discrete attribute D, and D_{i^*} denotes the set of all discrete attributes.

To enable conditional sampling, a conditional vector is introduced in the generator and discriminator. This vector concatenates all categorical columns together. For example, if we have two discrete columns $d_1 = \{1, 2\}$ and $d_2 = \{1, 4\}$, and the condition is $d_1 = 2$, the resulting conditional vector is $[0, 1, 0, 0]$. A penalty is added to the loss function to ensure that the generator produces data points that maintain the conditioned value.

To enforce the representation of all categorical attributes during training, a training-by-sampling technique is employed. This involves randomly selecting one categorical column with equal probability, computing the corresponding probability mass function (PMF) using the logarithm of the frequency as the mass, and sampling from the PMF. The value is set to one, and the conditional vector is reconstructed by concatenating all categorical columns.

Network Architecture

The architecture of CTGAN consists of two fully-connected layers with 256 neurons each to capture the correlations between columns. The components of CTGAN, namely the generator and discriminator, have the following specific characteristics:

Generator
The generator utilizes batch normalization and ReLU activation function between the hidden layers. The output layers have different activation functions: - The scalar value α is generated using the tanh activation function. - The mode indicator β and the categorical values d_i are generated using Gumbel softmax.

Discriminator
The discriminator uses Leaky ReLU activation functions with dropout applied to each hidden layer.

The loss function in CTGAN is adopted from WGAN-GP (Wasserstein GAN with Gradient Penalty). The network is trained using the Adam optimizer with the following parameters:

Parameter	Value
Learning rate	2×10^{-4}
β_1	0.5
β_2	0.9
L2-penalty	10^{-6}

Table 4.1: Default optimization parameters for CTGAN using Adam optimizer.

To train the generator and discriminator, CTGAN utilizes the PacGAN framework. PacGAN allows the discriminator to make decisions on multiple samples from the same class, which penalizes generators that suffer from mode collapse. In CTGAN, the discriminator is fed with a pac-size of ten.

The training parameters for CTGAN are as follows:

Parameter	Value
Epochs	300
Batch size	500

Table 4.2: Default training parameters for CTGAN.

4.1.2 Gaussian Copula

Using Gaussian copula to generate synthetic data has been described in a paper by Neha Patki et all [21].In this paper they have made the following contributions -**Recursive modelling technique**(for recursively modelling tables into datasets),**Creation of synthetic data**(taking input the dataset and after defining the schema,user will be able to generate data as much as they want)**Enable privacy protection**(perturbing model parameters and adding noise to data)**Demostration of its utility**(using synthetic data for predictive modelling instead of real data and compared the results)

Overview of SDV system

: The SDV workflow, as depicted in Figure 1, enables users to collect and format data, specify the structure and data types, run the modeling system, and finally generate new synthetic data based on the learned model.
The SDV workflow involves four main steps: organizing the data, specifying the table structure, learning the generative model, and synthesizing new data.

Organize

The user prepares the data by formatting it into separate files for each table in the database.

Specify Structure

The user provides metadata about the structure of each table, including relationships between tables if any.

Learn Model

SDV's modeling system iterates through the tables, discovering dependencies and computing aggregate statistics for related tables. An extended table is formed, capturing generating information and dependencies.

Synthesize Data

After learning the generative model, the user can use the provided API(`database.get_table`) to synthesize new data. The API includes functions to retrieve a model for a specific table, synthesize rows(`table.synth_row`) with inferred missing data, and synthesize(`table.synth_children`) complete tables that reference the current table. The synthesized data matches the original data in terms of structure and values.

This process allows the user to generate fully synthetic data that can be used in place of the original data.

Standalone Table Model

Ithas two parts, i) Distribution of each column and ii) Covariances between the columns
Distribution: Distribution of each column is guessed to be some known distribution like Gaussian distribution , truncated Gaussian distribution,uniform,beta and exponential distribution and their parameters to be determined.

Covariances: In addition to estimating the distributions, a generative model needs to calculate the covariances between columns. However, the shape of the distributions can introduce bias in covariance estimates. To address this issue, the Gaussian Copula is used in the multivariate case.

The steps to model a Gaussian Copula are as follows: 1) Given the columns of the table $0, 1, ..., n$ and their respective cumulative distribution functions $F_0, ..., F_n$. 2) Iterate through the table row-by-row and consider each row as a vector $X = (x_0, x_1, ..., x_n)$. 3) Convert the row using the Gaussian Copula: $Y = (\phi^{-1}(F_0(x_0)), \phi^{-1}(F_1(x_1)), ..., \phi^{-1}(F_n(x_n)))$, where $\phi^{-1}(F_i(x_i))$ is the inverse cumulative distribution function of the Gaussian distribution applied to the original distribution's cumulative distribution function. 4) After converting all the rows, compute the covariance matrix σ of the transformed values in the table.

The generative model for the table consists of the parameters for each column distribution and the covariance matrix σ. This model contains all the necessary information from the original table in a compact form and can be used to synthesize new data for the table.

Data Preprocessing

Missing Values

When SDV encounters any column with missing values ,it creates two columns . One column is the original column of data with randomly filled by a number in the missing value position and another column tracks if the corresponding value of real data is missing or not. Hence, the generated dataset contain missing value in a similar proportion of the real dataset.

Categorical column

Categorical columns in a table cannot be modeled using the Gaussian Copula . When the SDV encounters a categorical column, it replaces it with a numerical column containing values in the range [0, 1]. The conversion process involves the following steps:

- 1) Sort the categories in descending order based on their frequency of occurrence.
- 2) Divide the interval [0, 1] into sections proportional to the cumulative probability of each category.
- 3) To convert a category, find the interval [a, b] within [0, 1] that corresponds to the category.
- 4) Choose a value between a and b by sampling from a truncated Gaussian distribution with a mean (μ) at the center of the interval and a standard deviation (σ) equal to $\frac{b-a}{6}$.

Datetime

Date and time column is also replaced by numerical columns based on number of seconds passed Epoch (January 1,1970)

Data Synthesis

Data synthesis part is divide into two portion, one is **Model based**(sampling the whole dataset without any condition) and **KnowLedge based** (sampling the dataset based on previous knowledge i.e conditional sampling)

Model Based

All numerical values in the dataset can be sampled from the specified distributions and covariances of the columns. The process involves the use of cumulative distribution functions F and the covariance matrix Σ. The algorithm for sampling numerical values is outlined in Algorithm 3, assuming there are n columns.

The resulting vector x provides values for all columns that were converted to numerical data. Further post-processing steps may be applied to convert the numerical values back to their original formats, such as datetime or categorical values. Additionally, values for columns that were not originally present in the table, including derived columns , should be removed. Missing values can be identified by considering the binary ”Yes” or ”No” value that is sampled.

1	**function SAMPLE(F, Σ)**
2	$v \leftarrow$ random n-dimensional Gaussian vector
3	Find Cholesky decomposition, $LL^T = \Sigma$
4	$u \leftarrow Lv$
5	$x \leftarrow [F_0^{-1}(\mathrm{u}_0), F_1^{-1}(\mathrm{u}_1), ..., F_n^{-1}(\mathrm{u}_n)]$
6	**return** x

Table 4.3: Sampling numerical values from distribution and covariances of the columns.

Knowledge Based

We need to update the covariance matrix Σ and the mean vector μ for observed data. Initially, $\mu = 0$ due to the Gaussian Copula process.

Let k represent all the observed (known) variables, and u represent the unobserved (unknown) variables that the user wishes to synthesize.

$$\Sigma = \begin{pmatrix} \Sigma_{uu} & \Sigma_{uk} \\ \Sigma_{ku} & \Sigma_{kk} \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_u \\ \mu_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

According to SDV new matrices the updated parameters $\Sigma\prime$ and $\mu\prime$ for the unknown variables:

$$\Sigma^{'} = \Sigma_{uu} - \Sigma_{uk}\Sigma_{kk}^{-1}\Sigma_{ku}, \quad \mu\prime = \mu_u + \Sigma_{uk}\Sigma_{kk}^{-1}(obs - \mu_k) = \Sigma_{uk}\Sigma_{kk}^{-1}\mathrm{obs}$$

Here, *obs* is the user-inputted vector containing the known values. Note that Σ_0 has dimensions $|u| \times |u|$ and μ_0 has $|u|$ elements, as they only describe the relations for the columns with unobserved values.

Now, the SDV has the updated Σ_0 and $\boldsymbol{\mu}_0$, along with the corresponding cumulative distribution functions F_i for the unknown variables $i \in \mathbf{u}$. These values can be used in the sampling algorithm (Algorithm 3) with a slight modification: in step 4, the mean vector $\boldsymbol{\mu}_0$ is added to the vector $\mathbf{u}$. This process returns numerical values that can be further post-processed into categorical or datetime values. However, inference is required to determine the parent for foreign key information.

4.2 Evaluation Framework:

It can be challenging to judge how useful synthetic data is.Utility measures, sometimes referred to as broad and narrow measures, are typically divided into general and concentrated utility measures [20]. The broad evaluation measures the statistical disparity between artificial and actual data but ignores the specific future research that will be conducted using the data. Instead, this is where particular utility enters the picture, which frequently contrasts the similarity of inference across models trained on fictitious data and real data. The advantage of general metrics is that they are mostly automated, whereas the specialized measure needs to be manually defined based on its purpose. If numerous criteria produce noticeably disparate findings, [19].Broad and narrow utility measurements, also referred to as broad and narrow measures [20], make up the majority of the evaluation.Broad and narrow utility measurements, also referred to as broad and narrow measures, make up the majority of the evaluation. The overall evaluation examines the statistical difference between the real and synthetic data, not the specific analysis that will ultimately be performed on the data. Instead, this is where particular utility comes into play, which frequently looks at how similarly models can infer things when trained on synthetic and actual data. The benefit of general metrics is that they are generally automated, as opposed to specialized metrics, which must be manually defined according to their intended use.

4.2.1 General utility:

Distribution:

This metric will make sure that each column of synthetic data has a distribution that resembles the original distribution. Graphing the distribution of synthetic and actual columns can also help us determine whether the mode-collapse problem affects the created data.The probability mass, probability density, and cumulative distribution functions will all be visually inspected as part of the investigation. To compare the similarity of two distributions, we employ the Kolmogorov-Smirnov statistic. We transform a numerical distribution into its cumulative distribution function (CDF) in order to calculate this statistic. The largest discrepancy between the two CDFs is represented by the KS statistic. **Overall quality** is the overall score of how our synthetic data is mathematically close to real dataset.It is the average of column shape score and column pair trends.

For numerical and date time variable we use **KSComplement**.

KSComplement :This metric, also known as the marginal distribution or 1D histogram of the column, calculates the resemblance of a real column to a synthetic column in terms of the column forms. This metric disregards blank values. Utilizing the Kolmogorov-Smirnov statistic is the KSComplement. We transform a numerical distribution into its cumulative distribution function (CDF) in order to calculate this statistic. The largest distinction between the two CDFs is shown by the [KS statistic].The distance has a value that ranges from 0 to 1. It is reversed in SDMetrics: A higher score indicates higher quality because the KSComplement returns 1 (the KS statistic).

For categorical variable, we have used TVComplment.

TVComplment: This metric, also known as the marginal distribution or 1D histogram of the column, calculates the resemblance of a real column to a synthetic column in terms of the column forms. The Total Variation Distance (TVD) between the real and synthetic columns is calculated using this test. The frequency of each category value is first calculated and expressed as a probability in order to accomplish this. The TVD statistic compares the differences in probabilities, as shown in the formula below:

$$\beta(R, S) = 1/2 * \sum(|R_\omega - S_w|)$$

Here, *omega* describes every potential category in a single column.The terms R and S denote the respective real and artificial frequencies for those categories. A higher score indicates higher quality because the TVComplement returns $1 - TVD$ as the score.

$$score = 1 - \beta(R, S)$$

Correlation:

CorrelationSimilarity: This metric compares the trends of 2D distributions by computing the similarity between real and synthetic data as well as the correlation between two numerical columns. Both the Pearson and Spearman's rank coefficients are supported by this metric when used to calculate correlation. Here, the Pearson correlation coefficient has been employed.

This test calculates a correlation coefficient for a pair of columns, A and B, using both actual and fake data, R and S. Two different correlation values result from this. Using the algorithm below, the test normalizes and provides a similarity score.

$$score = 1 - (1/2) * (|R_{A,B} - S_{A,B}|)$$

ContingencySimilarity:

This metric compares 2D distributions by computing the similarity of two categorical columns in the real and synthetic datasets. The test creates a normalized contingency table for the genuine and fake data for a pair of columns, A and B. The percentage of rows with each A and B combination of categories is shown in the table below. The Total Variation Distance is then used to calculate the difference between the contingency tables. The distance is then subtracted from 1 to ensure that a high score indicates a high degree of similarity. The formula below provides a summary of the procedure.

$$score = 1 - (1/2) * \sum_{\alpha} \sum_{\beta} (|R_{\alpha,\beta} - S_{\alpha,\beta}|)$$

In the formula, α describes all the possible categories in column A and β describes all the possible categories in column B. Meanwhile, R and S refer to the real and synthetic frequencies for those categories.

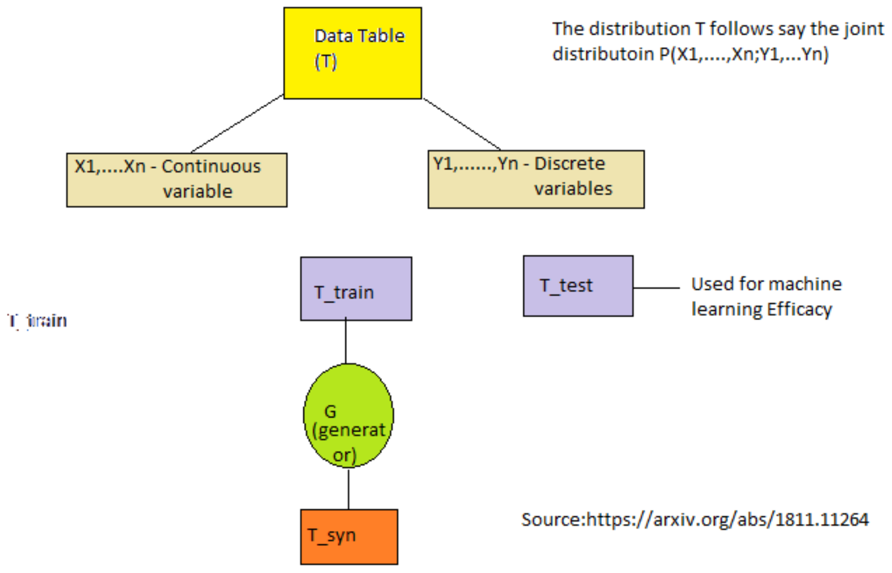

Data Table (T)
The distribution T follows say the joint distributoin P(X1,....,Xn;Y1,...Yn)
X1,....Xn - Continuous variable
Y1,......,Yn - Discrete variables
T_train
T_test
Used for machine learning Efficacy
G (generat or)
T_syn
Source:https://arxiv.org/abs/1811.11264

Figure 4.1: An illustrative image of evaluation of synthetic data

4.2.2 Machine learning efficacy:

By training a machine learning model on the data and comparing its inference capability to a model trained on real data, one can assess the quality of synthetic data using the machine learning efficacy metric.The question we addressed is if a classifier or regressor learnt from T_{syn} can achieve a similar performance on T_{test} as a model learned on T_{train}, as illustrated in figure 2.1, when trained to predict one column using other columns as features.

4.2.3 Privacy:

Using differential privacy as a metric, we can establish if it is possible to retrieve data from our initial dataset.Differential privacy is defined as follows:

Diffferntial Privacy:
Let ϵ be a positive real number and A be a randomized algorithm that takes a dataset as input (representing the actions of the trusted party holding the data). The algorithm A is said to be ϵ-differential privacy for all datasets D_1, D_2 such that:

$$Pr[A(D_1) \in S] \leq exp(\epsilon).Pr[A(D_2) \in S] \tag{4.1}$$

4.3 Baselines:

Due to varying datasets and measurements, one of the main issues with such a new field of study is that evaluations are not always comparable. Additionally, due to the usage of various definitions of what constitutes a ”good” sample, particularly with tabular data, evaluating synthesized data is difficult in and of itself.

In this thesis, we have used different baselines. Our baseline includes CTGAN, TVAE, Gaussian Copula and CART and to evaluate we have used metrices of SDV library.

4.4 Improved Result:

First we have generated $T_{1_{syn}}$ using Gaussian copula method and then generated $T_{2_{syn}}$ using CTGAN method. Next we have evaluated each column of these two synthesized datasets using **KSComplement** [4.2.1] and TVComplement [4.2.1] and compared their results. Now we will check which columns of $T_{2_{syn}}$ has a better result than the columns of $T_{1_{syn}}$ and replace those columns in $T_{1_{syn}}$. During replacing columns we'll sort respective columns and the replace for numerical variables. And for categorical variables , we will label them according to the the frequency of different classes of those variables.

Consider that T_1 and T_2 are two generated synthetic dataset from same real dataset and we want to make a synthetic dataset that will contain the best generated column (w.r.t KSComplement and TVComplement) among these two different algorithms. Here column shape score is basically the quality score for each column w.r.t KSComplement and TVComplement.

Algorithm 1 Hybrid Algorithm

```
1. Make a list of column shape score for all columns for $T_1$ and $T_2$
2. Transform all categorical columns using label encoder
3. For all columns $col$:
      if column shape score of $col$ of $T_1$ data $<$ column shape score of $col$ of $T_2$ data:
         Sort $T_1$ and $T_2$ with respect to $col$ in decreasing order and then swap $col$ of $T_1$
   with $col$ of $T_2$


   Return $T_1$
```

4.5 Experiment 1:

In this experiment, we selected two distinct types of datasets. The first dataset comprises a large number of rows, specifically 30,000 rows, while the second dataset is relatively smaller, with only 1,000 rows. Moreover, the first dataset predominantly consists of numerical columns, whereas the second dataset primarily comprises categorical columns.

To generate synthetic datasets, we employed four different libraries and assessed the generated data based on various statistical properties. This evaluation allowed us to compare and analyze the effectiveness of the different algorithms in generating realistic synthetic data.

Subsequently, a hybrid algorithm combining Gaussian copula and CTGAN was utilized to generate a hybrid dataset. This hybrid dataset was designed to incorporate the strengths of both algorithms, aiming to enhance the quality and representativeness of the generated synthetic data.

Furthermore, the hybrid algorithm was employed once again, this time to generate synthetic data using both the newly generated hybrid dataset and the dataset generated by the CART algorithm. This additional step enabled us to explore the potential synergies between the two approaches and investigate any improvements achieved by combining them.

By conducting these experiments, we aimed to gain insights into the performance and capabilities of various synthetic data generation techniques. Additionally, the comparison and evaluation of the generated datasets facilitated the identification of the most suitable algorithms for generating synthetic data with desired statistical properties and characteristics.

4.6 Experiment 2

In this part of the study, a simple vanilla GAN network implemented using the Keras library was employed. The objective was to explore the challenges that might arise when attempting to develop a proprietary library specifically tailored to the needs of HSBC.

The synthetic dataset generated by the GAN network provided valuable insights into the process of data generation using GANs. It helped to identify potential issues and challenges that may arise when building a custom library. By analyzing the results and understanding the limitations of the vanilla GAN implementation, we gained valuable knowledge and insights on how to proceed with the development of a proprietary library.

This experiment served as a baby step towards building a customized solution that aligns with the specific requirements and objectives of HSBC. The challenges encountered provided valuable guidance on the areas that need to be addressed and improved upon to ensure the successful development and implementation of an in-house library for synthetic data generation.

The findings from this experiment will serve as a foundation for future research and development efforts, allowing HSBC to refine and enhance the synthetic data generation process and create a library that is tailored to their unique needs and requirements

4.7 Experiment 3

n this particular part of the study, we focused on a small dataset related to climate risk and its impact on economic losses. The reason for selecting this dataset was the scarcity of available data in this field. We aimed to explore whether the challenge of limited historical data could be overcome through synthetic data generation techniques.

Additionally, we aimed to verify the correlation among five columns in the dataset. As existing methods typically calculate the correlation between pairs of columns, we encountered a limitation when trying to determine the correlation among multiple columns simultaneously. To overcome this, we generated a specific column in the dataset while having knowledge of its exact correlation with the other columns. This approach allowed us to evaluate the accuracy of the generated column and assess its correlation with the remaining columns in the dataset.

By conducting this experiment, we sought to address the issue of data scarcity in the climate risk domain and explore the feasibility of using synthetic data to augment the available datasets. Furthermore, we aimed to verify the accuracy of the generated synthetic column by comparing it to the known correlation values, providing insights into the effectiveness of the synthetic data generation process.

The outcomes of this experiment contribute to advancing the understanding of generating synthetic data in situations where historical data is limited, specifically in the context of climate risk and its economic implications. These findings serve as a basis for future research and provide valuable insights for decision-making processes in the field.

Chapter 5

EXPERIMENT 1

This chapter provides an overview of experimenting with four open source libraries - SDV, DataSynthesis,nbsynthetic and synthpop and using different algorithms Gaussian Copula, CTGAN, Bayesian Network, TVAE, Vanilla GAN and WGAN.In the last part , we'll compare the quality of synthetic data on these baselines with the hybrid version of all CART, CTGAN and Gaussian copula algorithms based on 3.2.

5.1 Data:

We have used two datasets downloaded from external resources . One is UCI credit dataset [15] based on default payments of customers of Taiwan and another one is german credit dataset also downloaded from UCI [16]. The difference is that first one contains mostly numerical variables and the second one categorical and boolean variables.These datasets were chosen because they contain a variety of various data types with varying and complex sorts of distributions, and they also reflect data that is frequently found in the financial domain.In order to assess the effectiveness of the model's multi-classification, an additional attribute was chosen because there aren't any publicly available datasets in the financial area that include a target variable including multiple classes.

We have used an open source library named SDV library [17] and DataSynthesis [18] and also used three diffent algorithms : Gaussian copula [9], CTGAN [8], Bayesian network [10].

5.1.1 Credit card default Taiwan dataset:

This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. The dataset contains 25 variables including categorical and numerical variables and 30000 rows in total.The target variable of this dataset is *"default.payment.next.month"* and it's a binary classification problem to determine whether a customer will be default in the next month or not.

Data Pre-processing:

In this dataset there is no missing values , we have noticed. *'Education','Marriage','Sex','default. payment.next.month'* are the columns that have datatype as *'float64','int64'*, but clearly they are not. So we have updated the datatype of these variables.We have created a meta-data of the dataset as a .json file . Also there is another column *'ID'* and it's meaningless to generate this column using our algorithms. Unique value of each row will be generated but not be in a perfect order. So we'll remove this column and next will index our rows according to the format of this column.

5.1.2 German credit risk data:

The dataset contains 1000 entries with 20 independent variables (7 numerical, 13 categorical) and in the dataset each entry represents a person who takes a credit by a bank. Each person is classified as good or bad credit risks according to the set of attributes.

Data Preprocessing:

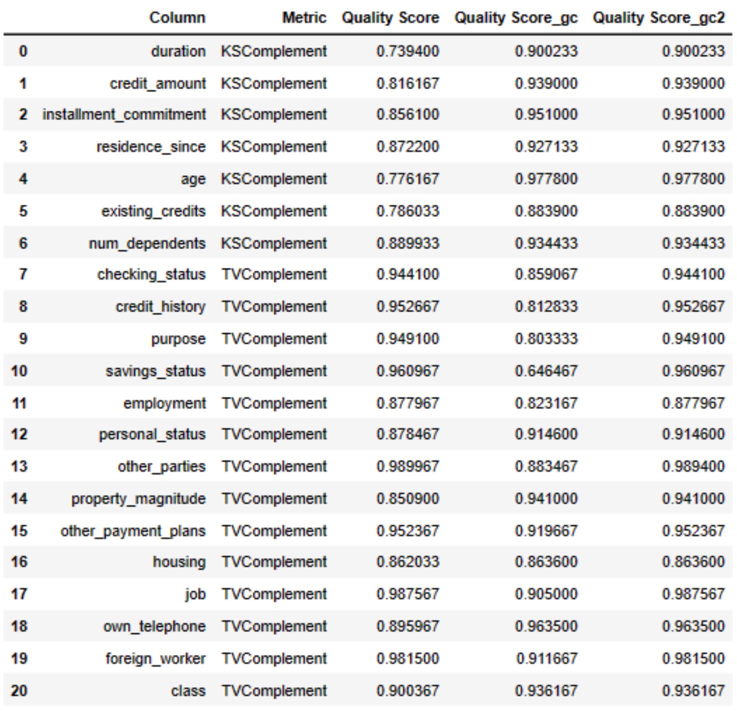
Some variables have to be renamed for the purpose of easy handling this data.There were no missing values and also no primary key of this dataset. So we created a column with the index value of each row and also created a .json file as a metadata of the dataset as described earlier.

5.2 Results and analysis

:

5.2.1 General Utility:

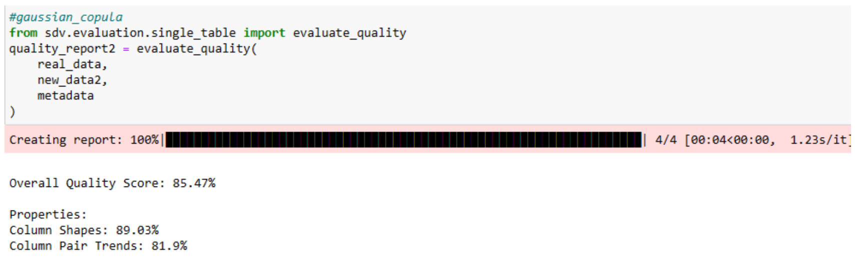
Distribution score:



	Column	Metric	Quality Score	Quality Score_gc	Quality Score_gc2
0	duration	KSComplement	0.739400	0.900233	0.900233
1	credit_amount	KSComplement	0.816167	0.939000	0.939000
2	installment_commitment	KSComplement	0.856100	0.951000	0.951000
3	residence_since	KSComplement	0.872200	0.927133	0.927133
4	age	KSComplement	0.776167	0.977800	0.977800
5	existing_credits	KSComplement	0.786033	0.883900	0.883900
6	num_dependents	KSComplement	0.889933	0.934433	0.934433
7	checking_status	TVComplement	0.944100	0.859067	0.944100
8	credit_history	TVComplement	0.952667	0.812833	0.952667
9	purpose	TVComplement	0.949100	0.803333	0.949100
10	savings_status	TVComplement	0.960967	0.646467	0.960967
11	employment	TVComplement	0.877967	0.823167	0.877967
12	personal_status	TVComplement	0.878467	0.914600	0.914600
13	other_parties	TVComplement	0.989967	0.883467	0.989400
14	property_magnitude	TVComplement	0.850900	0.941000	0.941000
15	other_payment_plans	TVComplement	0.952367	0.919667	0.952367
16	housing	TVComplement	0.862033	0.863600	0.863600
17	job	TVComplement	0.987567	0.905000	0.987567
18	own_telephone	TVComplement	0.895967	0.963500	0.963500
19	foreign_worker	TVComplement	0.981500	0.911667	0.981500
20	class	TVComplement	0.900367	0.936167	0.936167

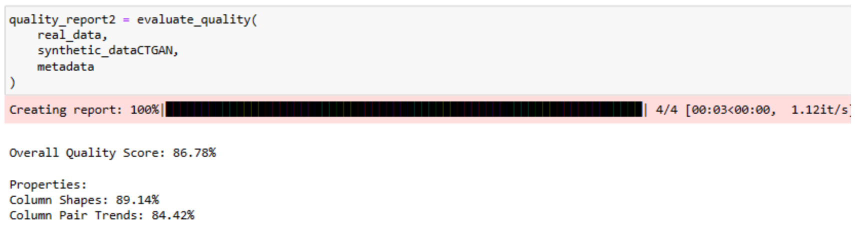
Figure 5.1: quality score of synthetic data generated by three different methods

Figure 5.1 shows that after performing on german credit data, the generated distribution is how close to real distribution of each column based on KS-test. 1 is the maximum



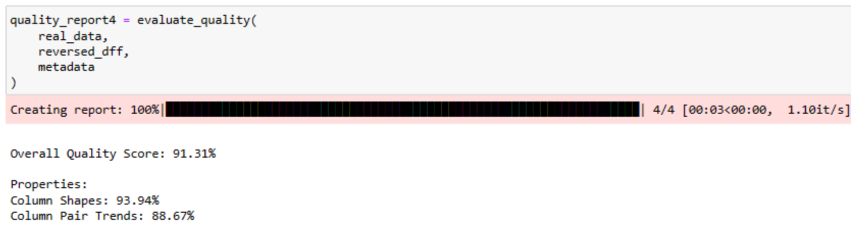
```
#gaussian_copula
from sdv.evaluation.single_table import evaluate_quality
quality_report2 = evaluate_quality(
    real_data,
    new_data2,
    metadata
)
Creating report: 100%|██████████████████████████████| 4/4 [00:04<00:00,  1.23s/it]

Overall Quality Score: 85.47%

Properties:
Column Shapes: 89.03%
Column Pair Trends: 81.9%
```

(a) Gaussian copula



```
quality_report2 = evaluate_quality(
    real_data,
    synthetic_dataCTGAN,
    metadata
)
Creating report: 100%|██████████████████████████████| 4/4 [00:03<00:00,  1.12it/s]

Overall Quality Score: 86.78%

Properties:
Column Shapes: 89.14%
Column Pair Trends: 84.42%
```

(b) CTGAN



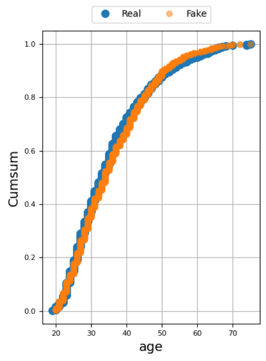
```
quality_report4 = evaluate_quality(
    real_data,
    reversed_dff,
    metadata
)
Creating report: 100%|██████████████████████████████| 4/4 [00:03<00:00,  1.10it/s]

Overall Quality Score: 91.31%

Properties:
Column Shapes: 93.94%
Column Pair Trends: 88.67%
```

(c) hybrid algorithm1 of copula and CTGAN

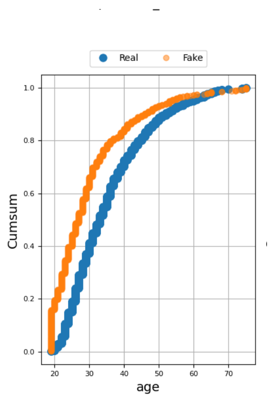
Figure 5.2: overall quality score by three different algorithms on german credit data

score and 0 is the lowest score that a column can have. In figure 5.1 *Quality Score* denotes the score achieved by CTGAN,*Quality Score_gc* is the score for Gaussian copula and *Quality Score_gc2* is for our hybrid algorithm1 of copula and gan. Clearly, we have achieved the best possible score for each column. From figure 5.2, the overall column shape score is 94% for our algorithm and we improved the shape of distribution .Also we can visualize from fig 5.3 that the CDFs of each column for our algorithm takes the closest shape to the original distribution among the generated columns by CTGAN and Gaussian Copula. If we see the column for *age* variable Gaussian has predicted better than CTGAN. CTGAN is unable to predict the numerical columns well and Gaussian copula is unable to predict some categorical columns well. So replacing with the best column we can see that our algorithms has performed well. For discrete variables like *purpose* Gaussian copula is unable to predict well comparing to CTGAN.(Figure 5.4)
We have also seen that the data generated by synthpop(CART) is better(Quality score for individual column) than Gaussian Copula and CTGAN for some columns .So we used our hybrid algorithm for our hybrid1(Gaussian Copula and CTGAN) and CART generated data. We named it hybrid2. Table 5.1 shows that hybrid2 is clearly far better than other methods. This may also solve the problem of volatility for any methods.

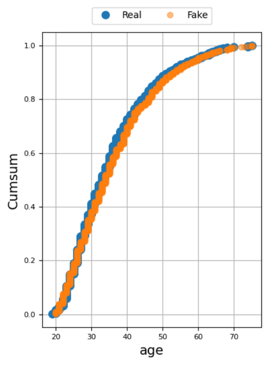
In credit card Taiwan dataset interestingly CART has performed well , even it has a better result than the hybrid result of CTGAN and Gaussian Copula. Also there is high improvement in column shape score . The table 5.2 shows the result of overall quality of all algorithms.



(a) Gaussian copula

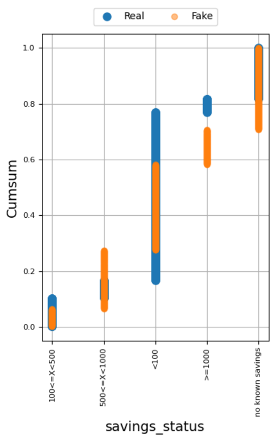


(b) CTGAN

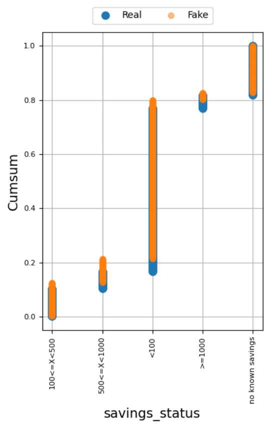


(c) hybrid algorithm1 of copula and CTGAN

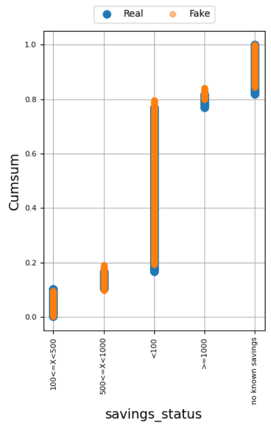
Figure 5.3: CDF of age columns generated by different algorithms on german credit data



(a) Gaussian copula



(b) CTGAN



(c) hybrid algorithm of copula and CTGAN

Figure 5.4: CDF of saving status columns generated by different algorithms

Table 5.1: Data quality score on german credit data

Library	Methods	Overall_Qs	Column Shape	Correlation Score
SDV	GC	85.47	89.03	81.9
	CTGAN	86.78	89.14	84.42
nbsynthetic	WGAN	50.62	55.86	45.39
	VGAN	55.35	62.87	47.83
Synthpop	CART	78.72	84.06	73.38
DataSynthesis	Bayesian Network	58.08	63.18	52.97
	Hybrid1 (GC and CTGAN)	91.31	93.34	88.67
	Hybrid2 (Hybrid1 and CART)	92.86	95.18	90.54

Table 5.2: Comparison of Methods for Column Synthesis Credit Card Taiwan Dataset

Library	Method	Overall Qs	Column Shape
SDV	GC	88.49	84.93
	CTGAN	93.56	92.03
	TVAE	89.43	88.55
Synthpop	CART	95.93	95.09
	Hybrid1	94.28	92.81

Correlation:

For German credit risk dataset ,CTGAN has captured the correlation between the columns well comparing to Gaussian copula. Column-pair trends has a better score for CTGAN 84.42% but our algorithm has improved this result to 88.67%. So we may say that our model may not be over-fitting.Improving the distributions of all variables also lead to a better correlation between the variables.(figure 5.5) The problem with CTGAN is that we may stop at some point that may produce bad quality of synthetic data due to its volatile nature. But if we mix with Gaussian copula we can get a better result. We have also seen that for some variables that CTGAN is unable to capture the right distribution but Gaussian copula does. By replacing the columns we can get close to very accurate result. Similarly , we get better result for taking the best column of this hybrid data and CART generated data (Table 5.1)

Machine Learning Efficacy:

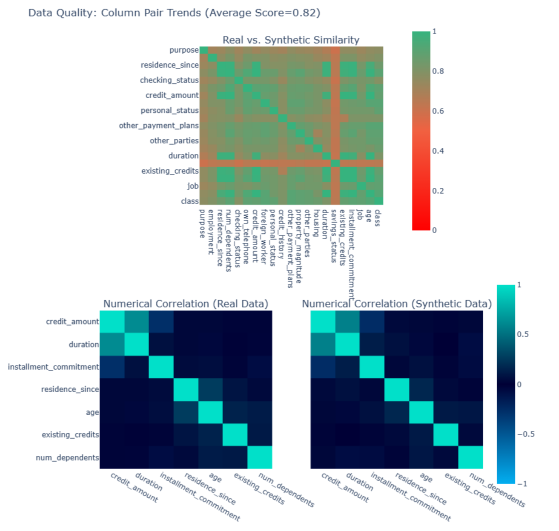
We have compared the F1 score of real and synthetic data using different type of machine learning algorithms like decision tree classifier , random forest ,logistic regression etc based on [3.2.2].For decision Tree Classifier we take 'binary' F1-score[28] and rest of them we have used F1-score 'micro' [28]. CTGAN has performed best for three out of four classifiers (Table 5.3). Our hybrid1 and hybrid2 has performed best for Random Forest Classifier and for the rest of the classifiers it's close to CTGAN. But, some interesting result we can see for WGAN and VGAN (nbsynthetic).For Logistic regression VGAN has outperformed all dataset even real dataset . For this though we have taken only 2000 rows instead of 30000 rows.

Table 5.3: Classifier Performance for German data

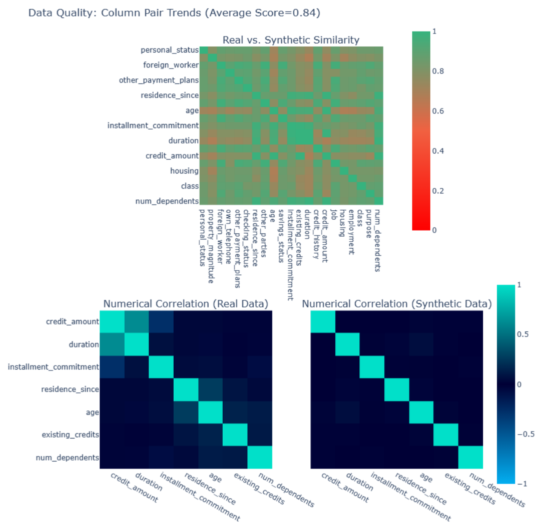
Classifier	Real	GC	CART	VGAN	WGAN	CTGAN	Hybrid1	Hybrid2
DecisionTree	0.74	0.73	0.728	0.78	0.74	0.72	0.69	0.71
LogisticRegressor	0.70	0.67	0.65	0.84	0.41	0.73	0.67	0.70
MLPClassifier	0.72	0.48	0.59	0.52	0.5	0.84	0.69	0.72
RandomForest	0.73	0.63	0.62	0.63	0.48	0.67	0.71	0.68

5.2.2 Privacy:

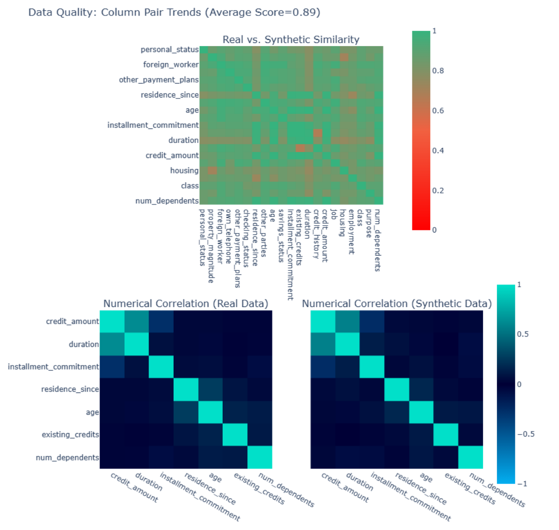
For privacy evaluation we have checked the common columns between real and synthetic data. We noticed that none of the three methods(GC, CART, CTGAN) has common columns and that's a good result for us(figure 5.6) and Table **??**. Only VGAN and



(a) Gaussian copula



(b) CTGAN



(c) Hybrid algorithm of copula and CT-GAN

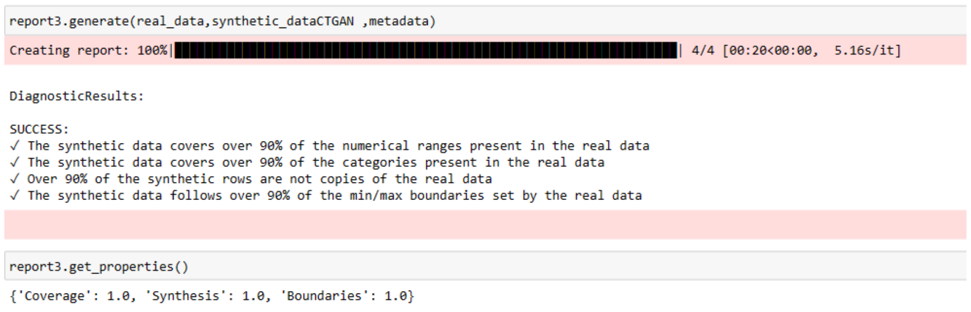
Figure 5.5: Correlation matrix comparison of columns generated by different algorithms for german credit data

WGAN have repeated columns 9 and 2 respectively. Secondly also, instead of sharing whole synthetic dataset, we can share .pkl file that contains only the parameters of our distribution and from that .pkl file we can sample as our desired number of rows .Another part, we measure the risk of disclosing sensitive information through an inference attack. We assume that *'purpose', 'class','personal_status'*- these columns in the real data are public knowledge. An attacker is combining this with synthetic data to make guesses about other sensitive column (for this, we considered *'property_magnitude'*) of real dataset. The score is also close to 0.7 for Gaussian copula and our hybrid algorithm(figure 5.7).

In credit card dataset , the synthetic data generated by CART algorithm has 12 common columns.

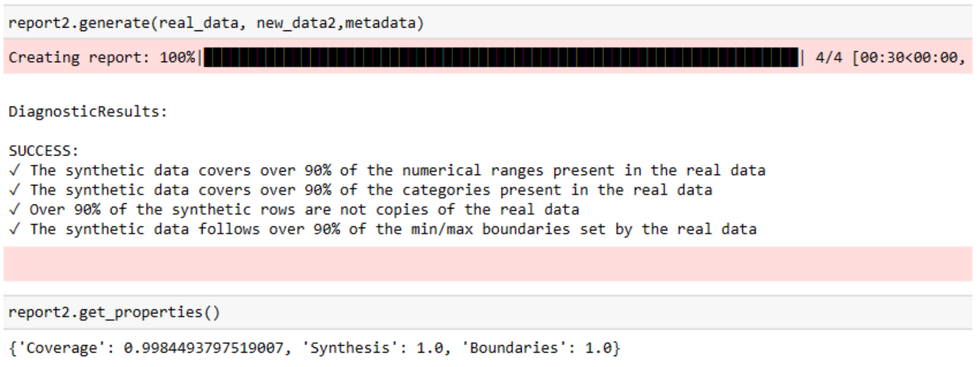
GC	0
CART	0
VGAN	9
CTGAN	0
WGAN	2
Hybrid1	0
Hybrid2	0

Table 5.4: Number of Repeated Columns



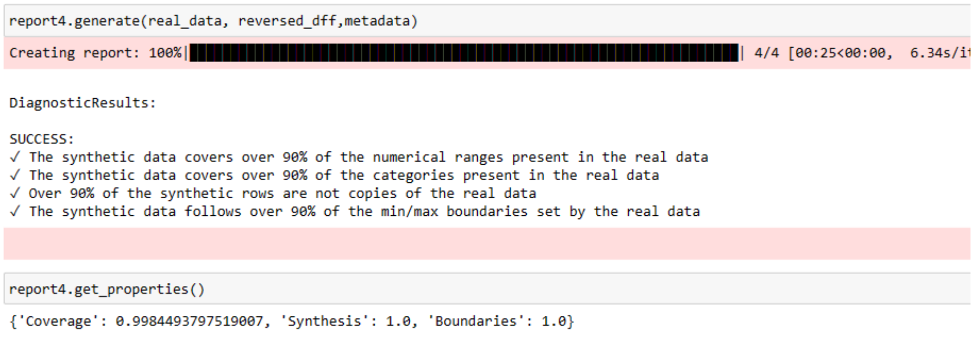
```
report3.generate(real_data,synthetic_dataCTGAN ,metadata)
Creating report: 100%|██████████████████████████████████████████████████| 4/4 [00:20<00:00,  5.16s/it]

DiagnosticResults:

SUCCESS:
✓ The synthetic data covers over 90% of the numerical ranges present in the real data
✓ The synthetic data covers over 90% of the categories present in the real data
✓ Over 90% of the synthetic rows are not copies of the real data
✓ The synthetic data follows over 90% of the min/max boundaries set by the real data

report3.get_properties()
{'Coverage': 1.0, 'Synthesis': 1.0, 'Boundaries': 1.0}
```

(a) Gaussian copula



```
report2.generate(real_data, new_data2,metadata)
Creating report: 100%|██████████████████████████████████████████████████| 4/4 [00:30<00:00,

DiagnosticResults:

SUCCESS:
✓ The synthetic data covers over 90% of the numerical ranges present in the real data
✓ The synthetic data covers over 90% of the categories present in the real data
✓ Over 90% of the synthetic rows are not copies of the real data
✓ The synthetic data follows over 90% of the min/max boundaries set by the real data

report2.get_properties()
{'Coverage': 0.9984493797519007, 'Synthesis': 1.0, 'Boundaries': 1.0}
```

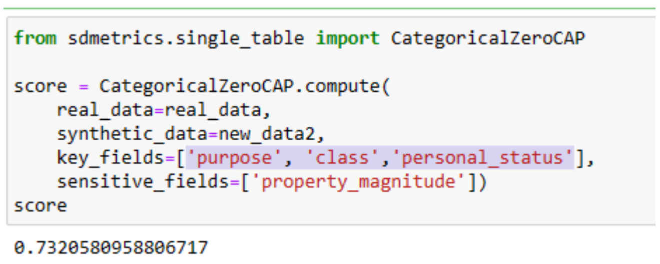
(b) CTGAN



```
report4.generate(real_data, reversed_dff,metadata)
Creating report: 100%|██████████████████████████████████████████████████| 4/4 [00:25<00:00,  6.34s/i

DiagnosticResults:

SUCCESS:
✓ The synthetic data covers over 90% of the numerical ranges present in the real data
✓ The synthetic data covers over 90% of the categories present in the real data
✓ Over 90% of the synthetic rows are not copies of the real data
✓ The synthetic data follows over 90% of the min/max boundaries set by the real data

report4.get_properties()
{'Coverage': 0.9984493797519007, 'Synthesis': 1.0, 'Boundaries': 1.0}
```

(c) hybrid algorithm(copula and CTGAN)

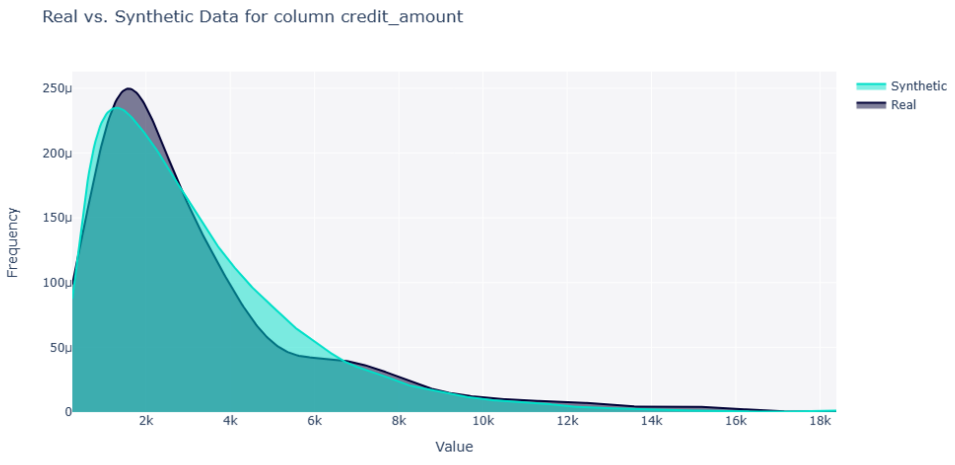
Figure 5.6: Checking if there are common columns and range difference of synthetic data and real data (german ceredit data)



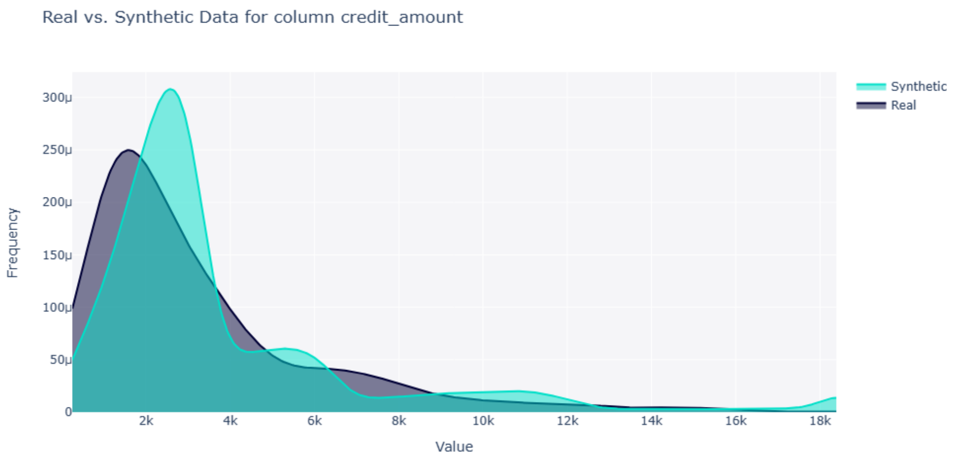
```
from sdmetrics.single_table import CategoricalZeroCAP

score = CategoricalZeroCAP.compute(
    real_data=real_data,
    synthetic_data=new_data2,
    key_fields=['purpose', 'class','personal_status'],
    sensitive_fields=['property_magnitude'])
score
```

```
0.7320580958806717
```

Figure 5.7: Privacy score for synthetic data generated by Gaussian copula

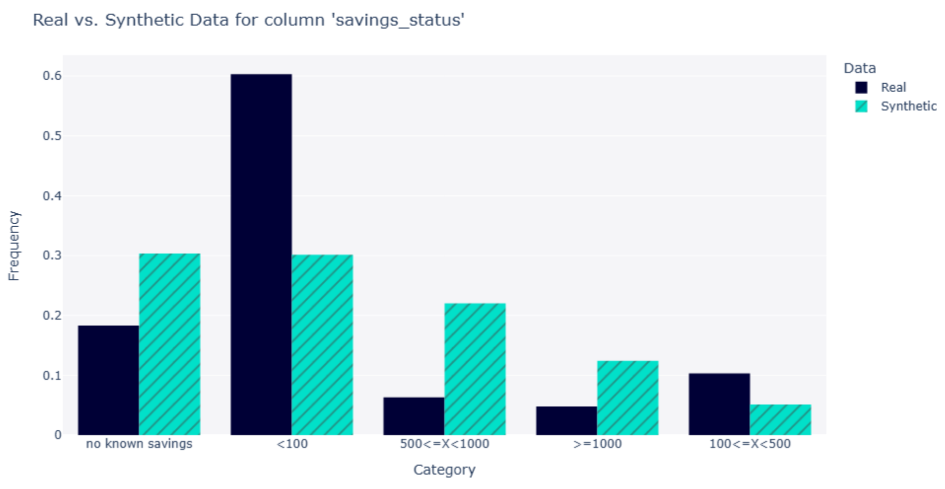


(a) Hybrid algorithm of copula and CTGAN

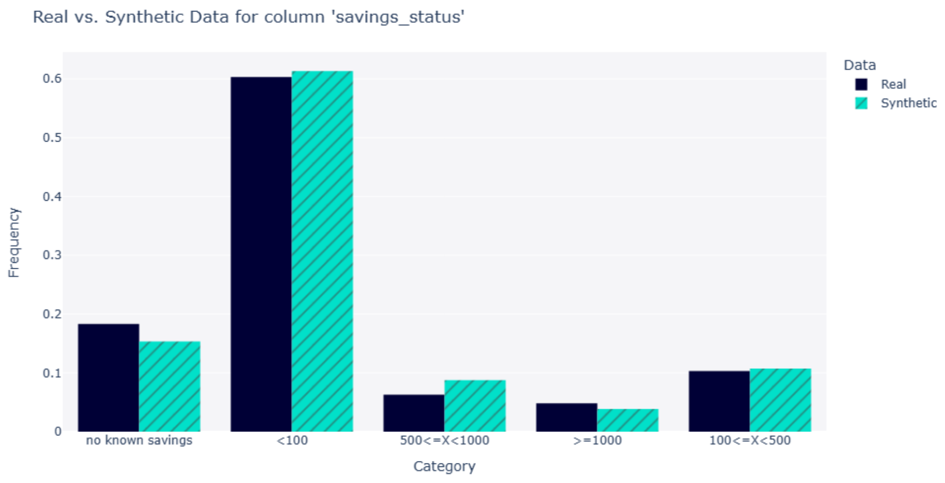


(b) CTGAN

Figure 5.8: Checking probability distribution of Credit amount



(a) Gaussian copula



(b) Hybrid algorithm of copula and CTGAN

Figure 5.9: Checking probability mass function of saving status

Chapter 6

EXPERIMENT 2

Chapter 2 provides a brief overview of the insights and challenges encountered while creating a custom library for generating synthetic data. The approach involved developing a simple GAN using *Keras*, without relying on existing open-source libraries for synthetic data generation. .

6.1 Data

4.1.2 (German credit risk data) Data has been used for this evaluation.

6.1.1 Data Preprocessing

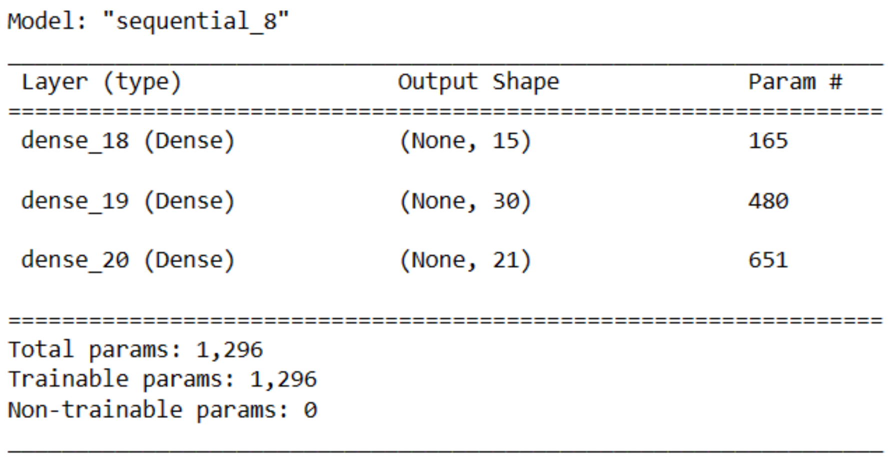
Regarding data preprocessing, it was necessary to address the issue that Keras does not directly support categorical variables. To overcome this limitation, ordinal encoding was applied to the categorical variables in the dataset. This encoding technique assigns numerical values to the different categories while preserving the order information. Fortunately, there were no missing values in the dataset.
The target column in the dataset, referred to as *'class'*, comprised two distinct classes: 'good' and 'bad'. To effectively use this column in the GAN model, a binary mapping was implemented, where 'good' was assigned the value of 1 and 'bad' was assigned the value of 0.

6.2 GAN Network

Generator

Input Layer: The generator takes a noise vector as input. The dimensionality of this vector, latent_dim, is a parameter that determines the complexity and variety of the generated samples.
Hidden Layers:
Dense Layer1: The first hidden layer is a dense layer with 15 units, which applies a linear transformation to the input data followed by a rectified linear activation function (ReLU). This layer helps in learning nonlinear relationships and extracting meaningful features from the noise vector.
Dense Layer2: The second hidden layer is another dense layer with 30 units and ReLU activation. It further refines the learned features and contributes to the complexity of the generated samples.

Output Layer: The final layer of the generator produces the synthetic data samples. In this case, the output layer is a dense layer with n_outputs units and linear activation. The number of output units (n_outputs) corresponds to the dimensionality of the generated samples. The activation function used here is linear, meaning the output values are not constrained within a specific range.



```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_18 (Dense)            (None, 15)                165

 dense_19 (Dense)            (None, 30)                480

 dense_20 (Dense)            (None, 21)                651

=================================================================
Total params: 1,296
Trainable params: 1,296
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.1: Generator-network

Discriminator

Input Layer: The discriminator takes an input sample, typically a data sample, as input. In this case, the input dimensionality is specified as n_inputs, which determines the number of features in the input data.
Hidden Layers: The discriminator consists of one or more dense (fully connected) layers. These layers extract relevant features from the input sample to aid in the classification task.

Dense Layer1: The first hidden layer is a dense layer with 25 units. It performs a linear transformation on the input data followed by a rectified linear activation function (ReLU). This layer helps in capturing complex relationships and extracting meaningful features from the input.
Dense Layer2: The second hidden layer is another dense layer with 50 units and ReLU activation. It further refines the learned features and contributes to the discriminative capability of the network.
Output Layer: The final layer of the discriminator produces a single output that represents the probability of the input sample being real or fake. It is a dense layer with 1 unit and a sigmoid activation function. The sigmoid activation function ensures that the output value falls between 0 and 1, representing the probability of the input being real.
Compilation: After defining the network architecture, the model is compiled using the binary cross-entropy loss function ('binary_crossentropy'). This loss function is commonly used in binary classification problems. The optimizer used is Adam, a popular optimization algorithm, and the accuracy metric is specified for monitoring the performance of the discriminator during training.

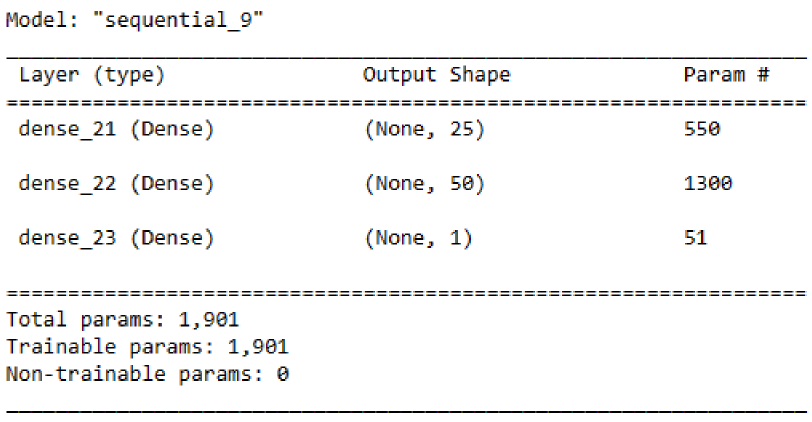

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_21 (Dense)            (None, 25)                550

 dense_22 (Dense)            (None, 50)                1300

 dense_23 (Dense)            (None, 1)                 51

=================================================================
Total params: 1,901
Trainable params: 1,901
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.2: Discriminator summary

6.3 Training of GAN

Batch Size: The function takes the batch size (n_batch) as a parameter. It determines the number of real and fake samples used in each training iteration. In this case, half of the batch size is used for real samples, and the other half is used for fake samples.

Initialization: The function initializes empty lists (d_history and g_history) to store the discriminator and generator losses, respectively, for each epoch.

Epoch Iteration: The function iterates over the specified number of epochs (n_epochs).

a. Real and Fake Samples: In each epoch iteration, the function generates real samples (x_real, y_real) using the generate_real_samples function. These samples are taken from the real data distribution.

b. Fake Samples: Fake samples (x_fake, y_fake) are generated using the generator model and the latent dimension (latent_dim) as inputs. The generator synthesizes fake samples based on random noise vectors.

c. Discriminator Update: The discriminator model is trained using the real and fake samples. The train_on_batch function is called twice: once with the real samples (x_real, y_real) and once with the fake samples (x_fake, y_fake). The resulting discriminator loss and accuracy are calculated for each batch.

d. Discriminator Loss: The discriminator loss is computed as the average of the losses from the real and fake samples. It is stored as d_loss.

e. Latent Space and Inverted Labels: Points in the latent space (x_gan) are generated as input for the generator model. In addition, inverted labels (y_gan) are created for the fake samples. These inverted labels represent the generator's objective to fool the discriminator.

f. Generator Update: The generator model is trained using the generator model (gan_model) and the inverted labels (y_gan) with the generated latent points (x_gan) as input. The generator aims to generate samples that the discriminator will classify as real.

g. Loss Monitoring: The discriminator loss for real and fake samples (d_loss_real, d_loss_fake) and the generator loss (g_loss_fake) are printed for each epoch. The discriminator and generator losses are appended to their respective history lists (d_history, g_history).

Plotting History: After training, the function plots the discriminator and generator losses over the epochs using the plot_history function.

Model Saving: Finally, the trained generator model (g_model) is saved for future use.

Parameters that have been used for this network is described in Table 5.1

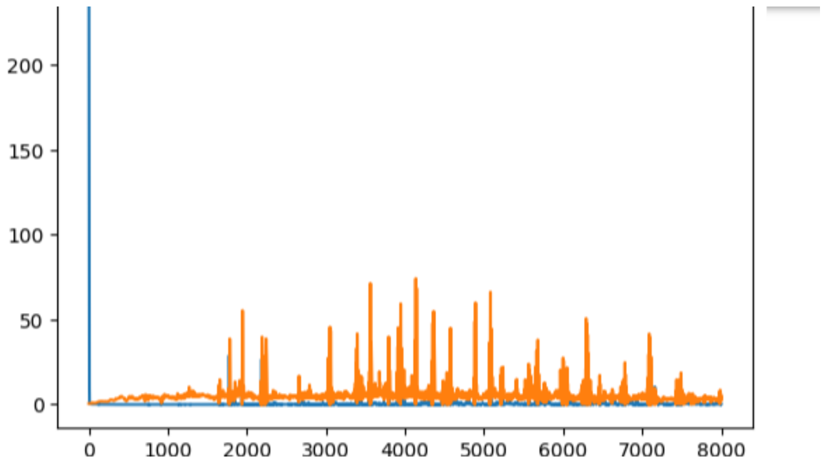


Figure 6.3: training losses for discriminator and generator

6.4 Evaluation:

As expected data quality is not so good. Specially, categorical columns are badly generated. But, it's a good hand on experience to generate synthetic data.Fig 6.4 and Fig 6.5 shows *'age'*,*'credit_amount'* ,*'saving_status'* columns.
Machine learning efficacy table 6.2 shows a huge gap between the performance of estimators trained on synthetic dataset and tested on real dataset.
For privacy perspective it's also not so good as it has generated 12 common columns between synthetic dataset and real datset.

Table 6.1: GAN Training Parameters

Parameter	**Value**
Latent Dim	10
Learning Rate	0.001
Number of Epochs	8000
Batch Size	128
Optimizer Parameters	$\beta_1 = 0.9$ $\beta_2 = 0.999$ $\epsilon = 1e - 07$

Table 6.2: Classifier Performance

Classifier	**Real Dataset**	**Synthetic Dataset**
DecisionTreeClassifier	0.74	0.57
Logistic Regression	0.70	0.49
MLPClassifier	0.72	0.52
RandomForest	0.73	0.52

6.5 Challenges:

1. Data Pre-processing is a challenging part. We need a proper transformer that can transform categorical variable to numerical one and memorize the mapping. So that it can revert back to original form when synthesizer create data in numerical form.

2. Also when synthetic data is generated , it'll consider the whole dataset as a vector and it can't generate data according to distinct classes of original dataset. Rather it'll treat the whole dataset as a continuous variable . We need some cluster method to land back to discrete classes. I have tried to map numerical values to its nearest class but it does not perform well.

3. We also need a method that can predict the type of variables like continuous or categorical. It'll help to preprocess the data.

4.Also we need a function to let the model know some predefined relations of columns or constraint.

5. Mode collapse can happen and to secure our synthetic data we must add some noise to it and we've to decide how much noise it should tolerate. My synthetic data has 12 columns same as original dataset.

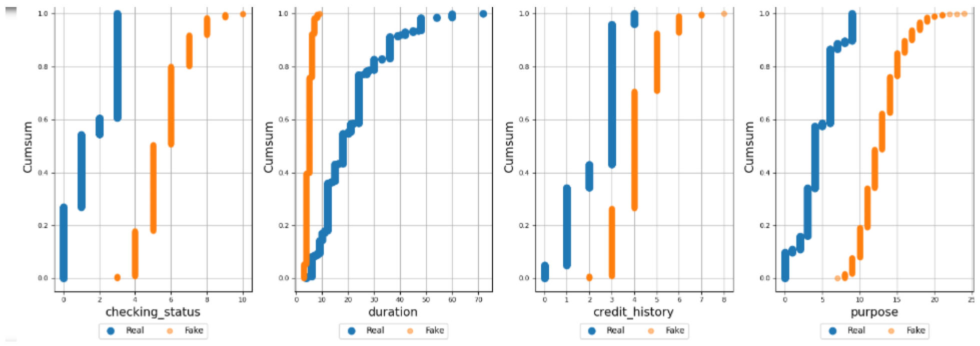


Figure 6.4: CDF of some generated columns

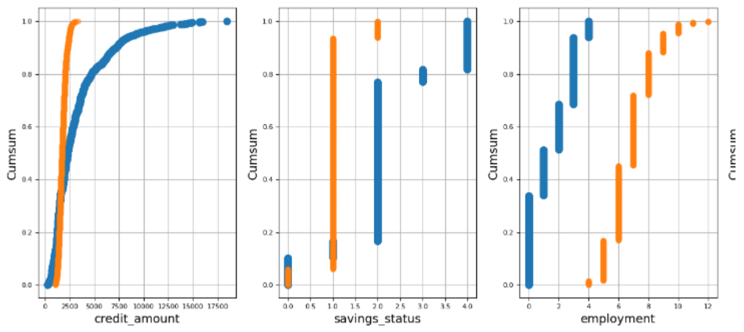


Figure 6.5: CDF of some generated columns

Chapter 7

EXPERIMENT 3

This chapter provides the overview of application of synthetic data on real world dataset. We'll focus on three part of the application of synthetic dataset- 1. We have generated 1000 rows from 181 rows and so, we'll examine how good is our data when we have a small amount of data. 2.The data contains missing values and we'll try to generate those missing values. 3.We have generated data based on some conditions and scenarios.

7.1 Data and Data-Preprocessing

Data was downloaded from this source[27].Data contains information related to economical losses due to climate change. Some of the are :
cri_rank: Rank of the country on the Climate Risk Index (Integer)
cri_score: Score of the country on the Climate Risk Index (Integer)
fatalities_per_100k_rank: Rank of the country in terms of fatalities per 100,000 people (Integer)
fatalities_per_100k_total: Total number fatalities per 100,000 people (Integer)
fatalities_rank: Rank of the country in terms of total fatalities (Integer)
fatalities_total: Total number of fatalities (Integer)

Data has only 181 rows and 14 columns .Out of 14 columns, ten columns are numerical and rest of them are categorical. Five columns consist the ranking of columns based on some other columns. As for example *fatalities_rank* column is based on *fatalities_total* in a decreasing order.So, all rank related columns are deleted and after generating rest of columns, rank related columns are also generated based on their respected columns.
cri_score has a particular formula , but we have generated this column without the formula. So we can evaluate the quality of generated column.

7.2 Data Generation and Quality of Synthetic data

Methods , used for this are CTGAN , Gaussian Copula and CART. Quality of CTGAN is 76, Gaussian Copula is 74 and CART is 93.08. The correlation between the column is also good.
The number of missing values in original data were 51 and the synthetic data were 264(all the missing values were for a single column. So the ratio of total rows and missing values is approximately same .
To evaluate synthetic data one of the toughest question is the data is acceptable in the

practical sense or not. So human expert's knowledge on that particular field is required. There is a formula for the column *cri_score* and actually it can be formulated based on the other columns. But we have generated this column and compared the genrated value with the actual value(calculated by formula).The histogram is below(Fig 7.1).In this case , we have generated same number of rows i.e 182 rows and 90% of rows has cri_score that differ by atmost 12. So they are unable to generate exact *cri_score* but they have genarated a cri_score close to it.(Highest value of cri_score is near about 125.

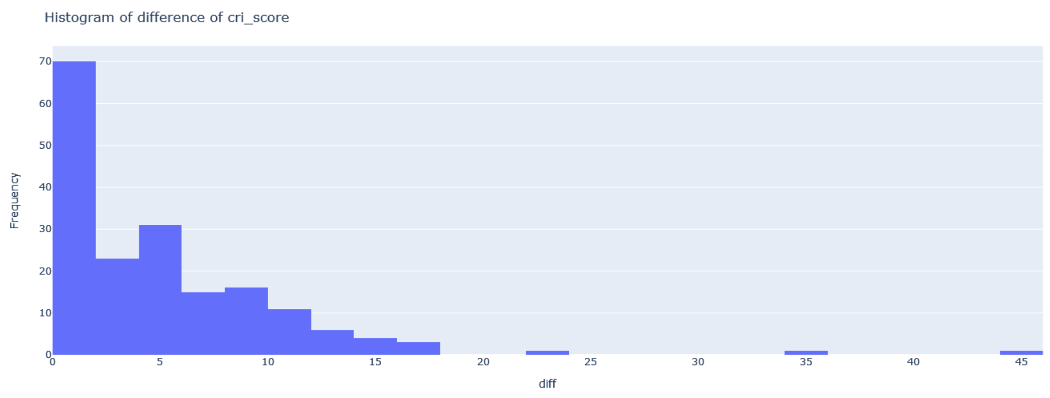
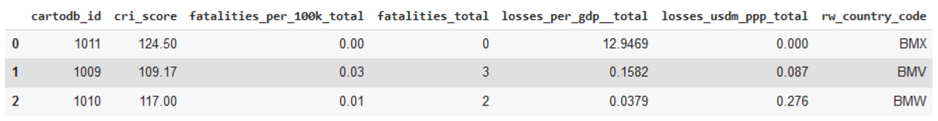


Figure 7.1: Difference of cri_score

7.3 Missing Values

Though there are actually 51 missing values in the original data but no of distinct rows is only three.Also only *losses_per_gdp_total* has missing values. We have used CTGAN and Gaussian Copula to generate those rows by providing the values of other columns and leaving the value of missing value. CTGAN used reject sampling process, so it's unable to generate all the rows. Only 1 row has been generated.From fig 7.2 we can say that the missing values are close to zero. According to our calculation , the values should be more closer to zero. Hence it's not perfect but it's closer to the perfect value.



	cartodb_id	cri_score	fatalities_per_100k_total	fatalities_total	losses_per_gdp__total	losses_usdm_ppp_total	rw_country_code
0	1011	124.50	0.00	0	12.9469	0.000	BMX
1	1009	109.17	0.03	3	0.1582	0.087	BMV
2	1010	117.00	0.01	2	0.0379	0.276	BMW

Figure 7.2: missing values

7.4 Scenario Generation

For scenario generation , we have considered to generate a dataset that will contain 90% of rows with value of *fatalities_total* is zero and rest of them is non zero. We have generated this with the help of Gaussian Copula and CTGAN.

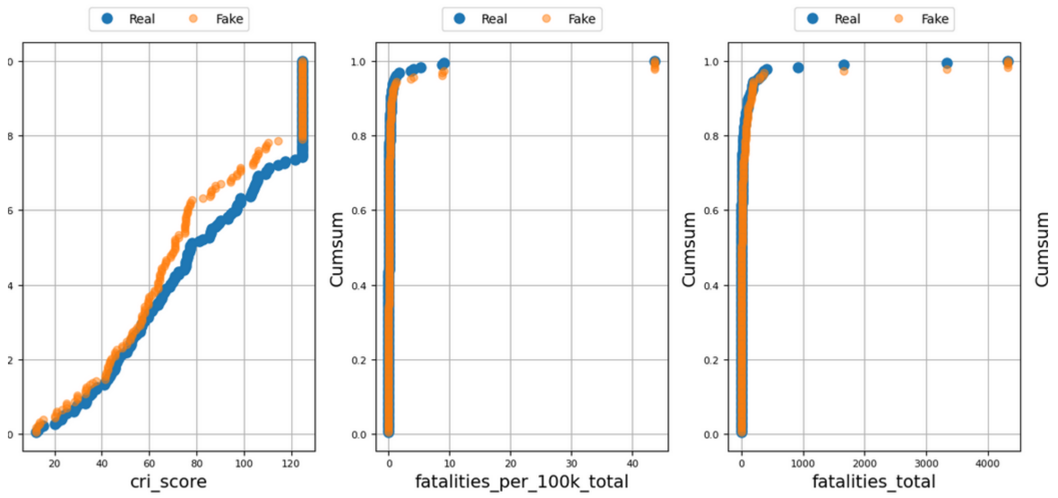


Figure 7.3: CDF of some generated columns

Chapter 8

FUTURE PLAN:

- Privacy Evaluation: We will closely examine our data to evaluate the privacy of our synthetic data. We will determine which privacy metrics are most meaningful for financial data. To enhance data security, we will introduce noise into our data, and we need to determine the acceptable level of noise.
- Larger Data Implementation: We plan to implement our algorithms on a larger dataset, preferably using internal data from HSBC. This will allow us to validate the effectiveness and scalability of our synthetic data generation methods.
- Outlier and Skewness Detection: We will focus on capturing outliers and skewness in variables within our synthetic data as accurately as possible. We aim to develop modules to evaluate the skewness of our data distributions, enabling more realistic representations of financial scenarios.
- Development of a Synthetic Data Library: Our long-term goal is to create a comprehensive library for generating synthetic financial data. While we have already developed a simple GAN structure for synthetic data generation, our objective is to further refine and expand the library to provide a complete and robust solution. .

Bibliography

[1] Samuel Assefa. “Generating Synthetic Data in Finance: Opportunities, Challenges and Pitfalls”. In: SSRN Electronic Journal (2020).

[2] Brian Tarran. “What can we learn from the Facebook—Cambridge Analytica scandal?” In: Significance (2018).

[3] Niaz Kammoun et al. “Financial market reaction to cyberattacks”. In: Cogent Economics and Finance (2019).

[4] Arvind Narayanan and Vitaly Shmatikov. “Robust de-anonymization of large sparse datasets”. In: *Proceedings - IEEE Symposium on Security and Privacy. 2008.*

[5] Steve Lohr. *Netflix Cancels Context After Concerns Are Raised About Privacy.* `https://www.nytimes.com/2010/03/13/technology/13netflix.html` New York, Mar. 2010.

[6] Edward Choi et al. *Generating multi-label discrete patient records using generative adversarial networks.* 2017.arXiv: 1703.06490.

[7] Lei Xu and Kalyan Veeramachaneni. *Synthesizing tabular data using generative adversarial networks.* In: arXiv (2018). arXiv: 1811 . 11264.

[8] L. Xu, M. Skoularidou, and A. Cuesta-infante. Modeling Tabular data using Conditional GAN.

[9] Neha Patki, RoyWedge, and Kalyan Veeramachaneni. The synthetic data vault. In *International Conference on Data Science and Advanced Analytics. IEEE,* 2016.

[10] Zhang et al .25 PrivBayes: Private Data Release via Bayesian Networks. In*ACM Transactions on Database Systems, Vol. 42, No. 4, Article 25.* ,2017

[11] Jerome P Reiter. Using cart to generate partially synthetic public use microdata. *Journal of Official Statistics,* 21(3):441, 2005.

[12] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. In *International Conference on Very Large Data Bases,* 2018.

[13] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *international conference on computer vision,* pages 2223–2232. IEEE, 2017.

[14] Alexandre Yahi, Rami Vanguri, Noémie Elhadad, and Nicholas P Tatonetti. Generative adversarial networks for electronic health records: A framework for exploring and evaluating methods for predicting drug-induced laboratory test trajectories. In NIPS *workshop on machine learning for health care,* 2017.

[15] default of credit card clients Data Set,URL: `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`,

[16] Statlog (German Credit Data) Data Set, URL: `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`.

[17] The Synthetic Data Vault Project, URL: `https://github.com/sdv-dev`.

[18] DataResponsibly/DataSynthesizer,URL: `https://github.com/DataResponsibly/DataSynthesizer`.

[19] Lucas Theis, AäronVan Den Oord, and Matthias Bethge. "A note on the evaluation of generative models". In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings.* 2016. arXiv: 1511.01844

[20] Joshua Snoke et al. "General and specific utility measures for synthetic data". In: *Journal of the Royal Statistical Society. Series A: Statistics in Society (2018).*arXiv: 1604.06651

[21] Patki, Neha and Wedge, Roy and Veeramachaneni, Kalyan. *Synthetic Data Vault.* 2016, Available at: `https://github.com/sdv-dev/SDV`.

[22] Xu, Lei and Skoularidou, Maria and Cuesta-Infante, Alfredo and Veeramachaneni, Kalyan, *Modeling Tabular data using Conditional GAN*, *Advances in Neural Information Processing Systems*, 2019

[23] Zhang, Kevin and Veeramachaneni, Kalyan and Patki, Neha, "Sequential Models in the Synthetic Data Vault", *unpublished* 2022

[24] Hazy. *synthpop.* 2020, Available at: `https://github.com/hazy/synthpop`

[25] nextbrain.ml. *nbsynthetic.* 2020, Available at: `https://github.com/NextBrain-ai/nbsynthetic`

[26] L. Xu and K. Veeramachaneni. "Synthesizing Tabular Data using Generative Adversarial Networks" 2018. URL `http://arxiv.org/abs/1811.11264.`

[27] Data.World URL `https://data.world/dataworldadmin`

[28] sklearn -F1 score URL `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`

[29] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. URL `http://www.github.com/goodfeli/adversarial`.

[30] O. Carey. Generative adversarial networks (GANs) — a beginner's guide, 2018. URL `https://towardsdatascience.com/generative-adversarial-networks-gans-a-beginners-guide-5b38eceece24`.

Chapter 9

APPENDIX

9.1 Credit card default Taiwan dataset:

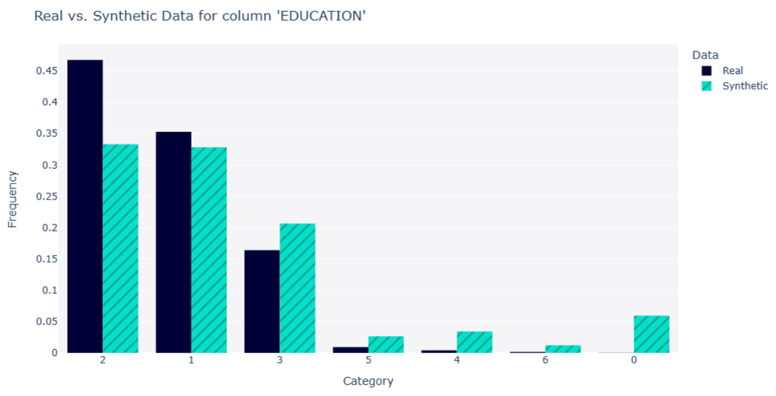


Figure 9.1: PDF of synthectic column *EDUCATION* of Taiwan dataset , generated by CTGAN algorithm

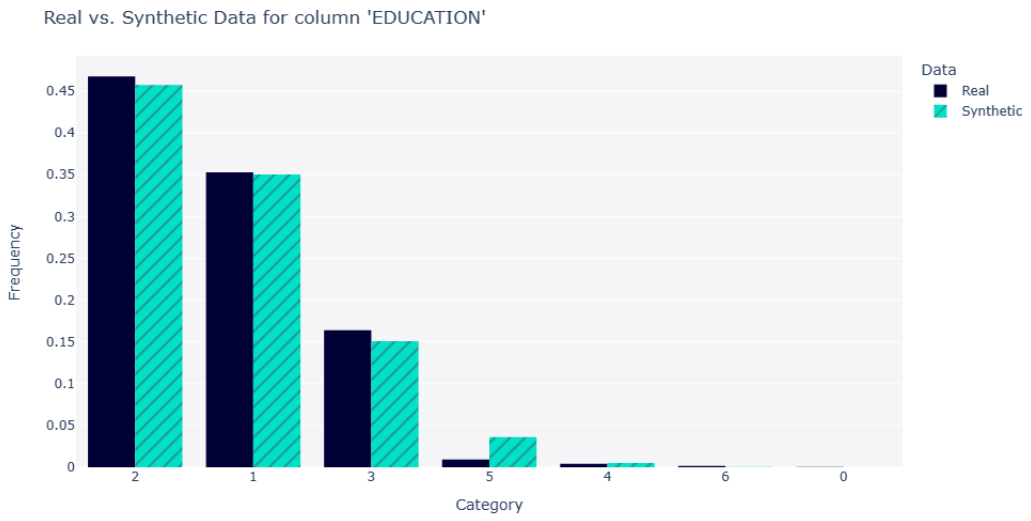


Figure 9.2: PDF of synthectic columns of Taiwan dataset , generated by improved algorithm

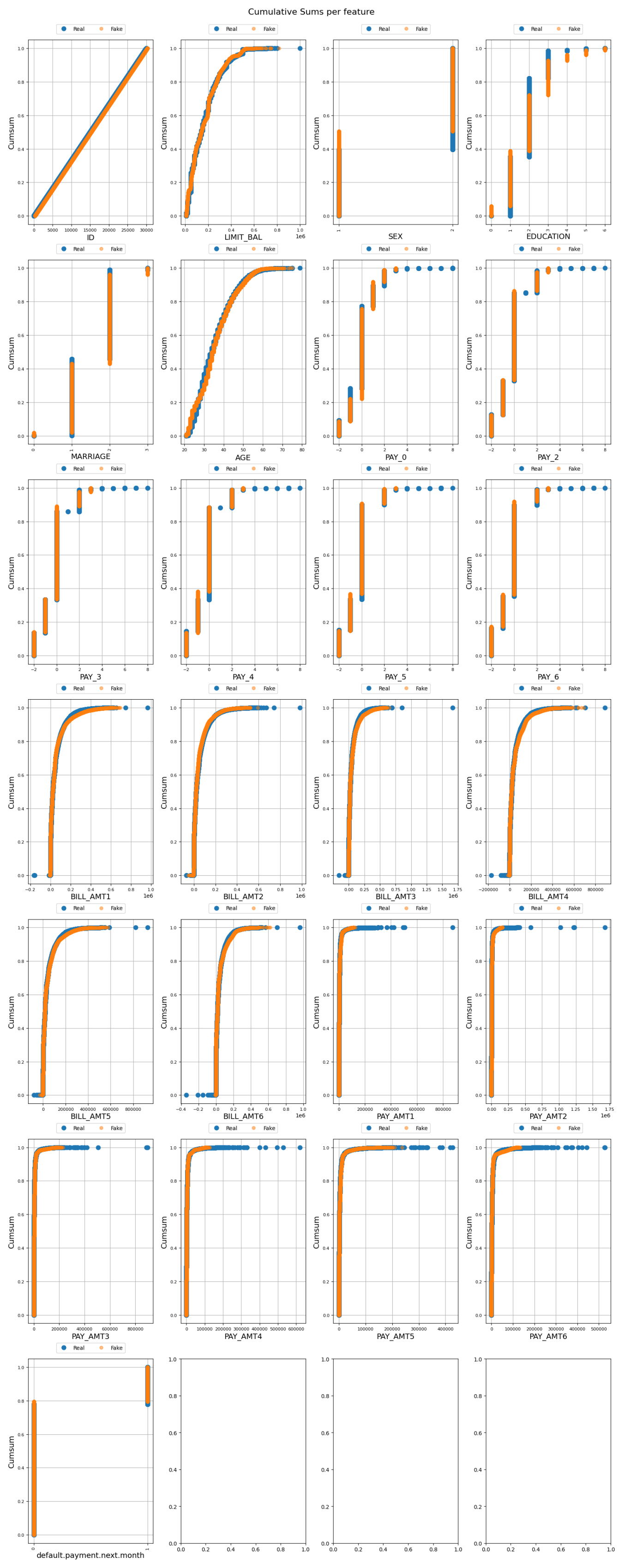


Figure 9.3: CDFs of synthectic columns of Taiwan dataset , generated by CTGAN algorithm

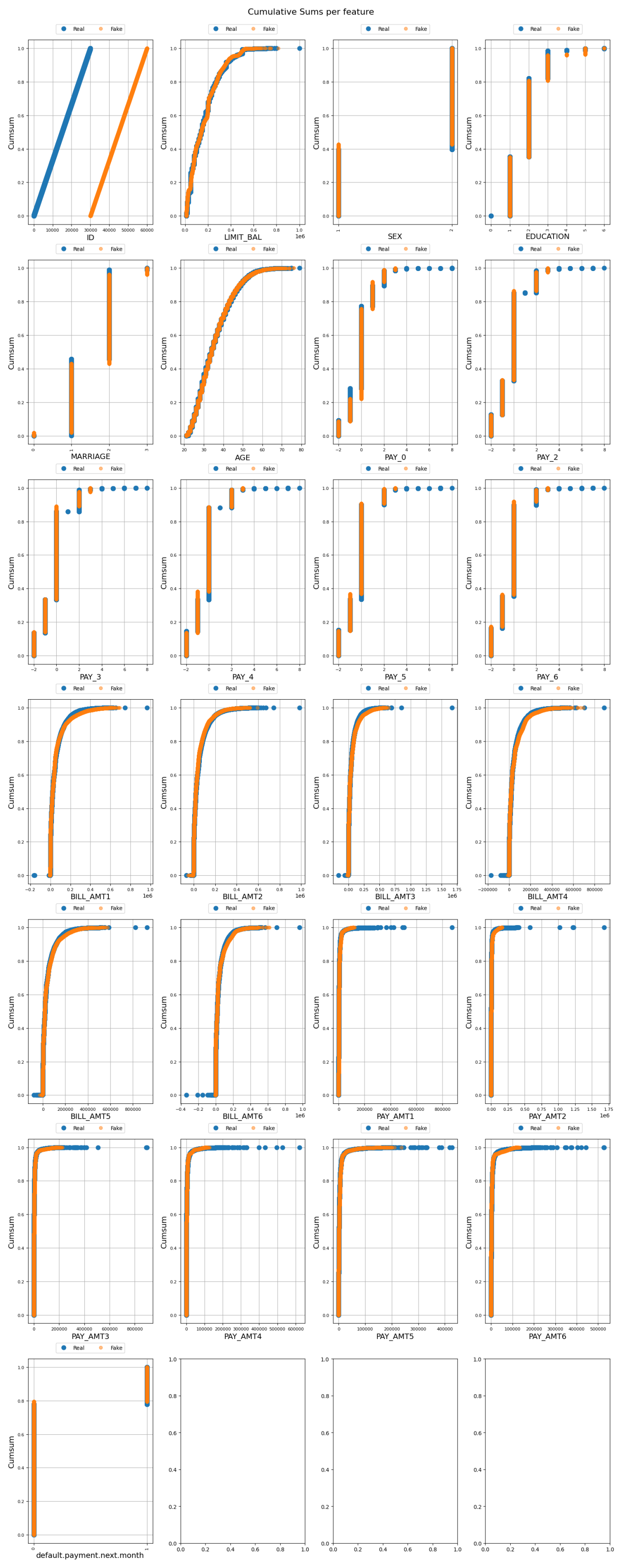
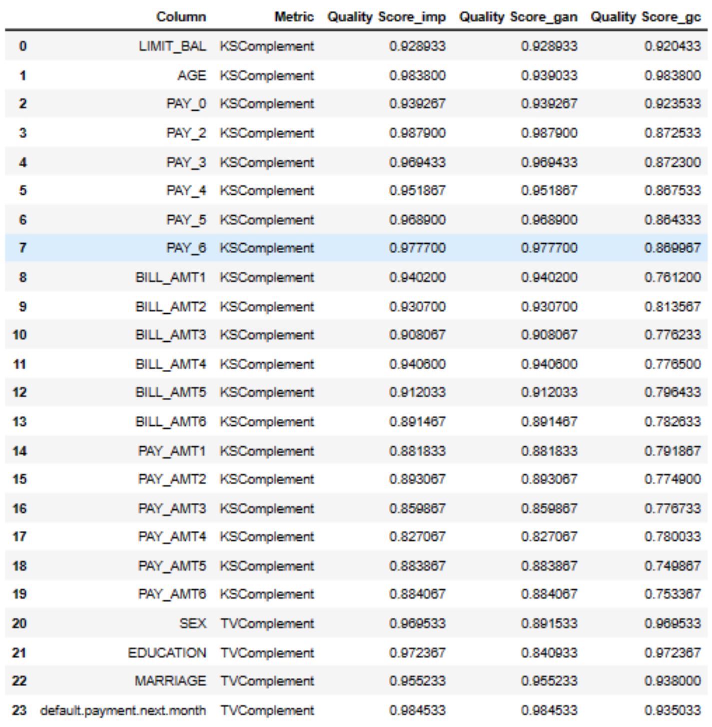
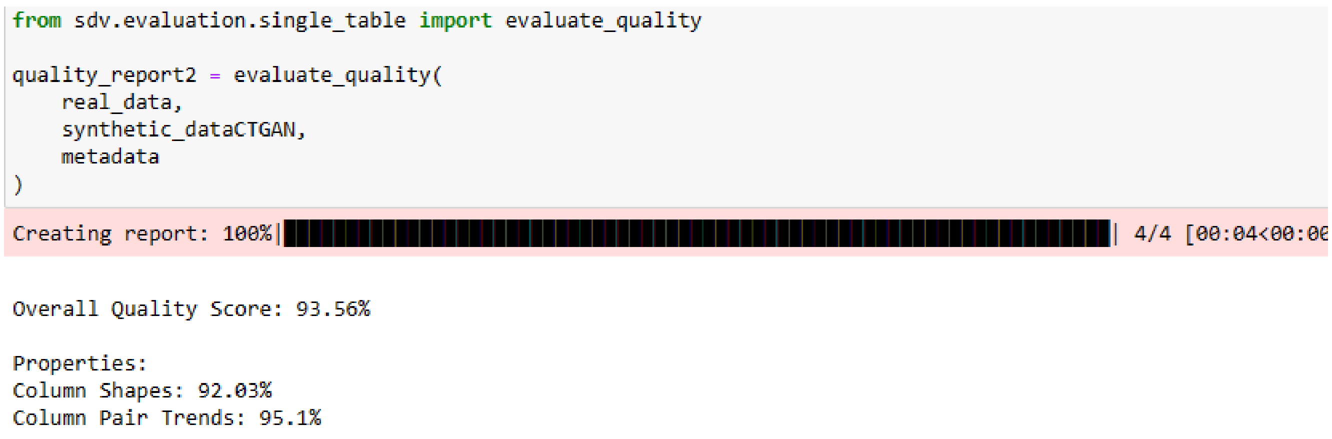


Figure 9.4: CDFs of synthectic columns of Taiwan dataset , generated by improved algorithm



	Column	Metric	Quality Score_imp	Quality Score_gan	Quality Score_gc
0	LIMIT_BAL	KSComplement	0.928933	0.928933	0.920433
1	AGE	KSComplement	0.983800	0.939033	0.983800
2	PAY_0	KSComplement	0.939267	0.939267	0.923533
3	PAY_2	KSComplement	0.987900	0.987900	0.872533
4	PAY_3	KSComplement	0.969433	0.969433	0.872300
5	PAY_4	KSComplement	0.951867	0.951867	0.867533
6	PAY_5	KSComplement	0.968900	0.968900	0.864333
7	PAY_6	KSComplement	0.977700	0.977700	0.869967
8	BILL_AMT1	KSComplement	0.940200	0.940200	0.761200
9	BILL_AMT2	KSComplement	0.930700	0.930700	0.813567
10	BILL_AMT3	KSComplement	0.908067	0.908067	0.776233
11	BILL_AMT4	KSComplement	0.940600	0.940600	0.776500
12	BILL_AMT5	KSComplement	0.912033	0.912033	0.796433
13	BILL_AMT6	KSComplement	0.891467	0.891467	0.782633
14	PAY_AMT1	KSComplement	0.881833	0.881833	0.791867
15	PAY_AMT2	KSComplement	0.893067	0.893067	0.774900
16	PAY_AMT3	KSComplement	0.859867	0.859867	0.776733
17	PAY_AMT4	KSComplement	0.827067	0.827067	0.780033
18	PAY_AMT5	KSComplement	0.883867	0.883867	0.749867
19	PAY_AMT6	KSComplement	0.884067	0.884067	0.753367
20	SEX	TVComplement	0.969533	0.891533	0.969533
21	EDUCATION	TVComplement	0.972367	0.840933	0.972367
22	MARRIAGE	TVComplement	0.955233	0.955233	0.938000
23	default.payment.next.month	TVComplement	0.984533	0.984533	0.935033

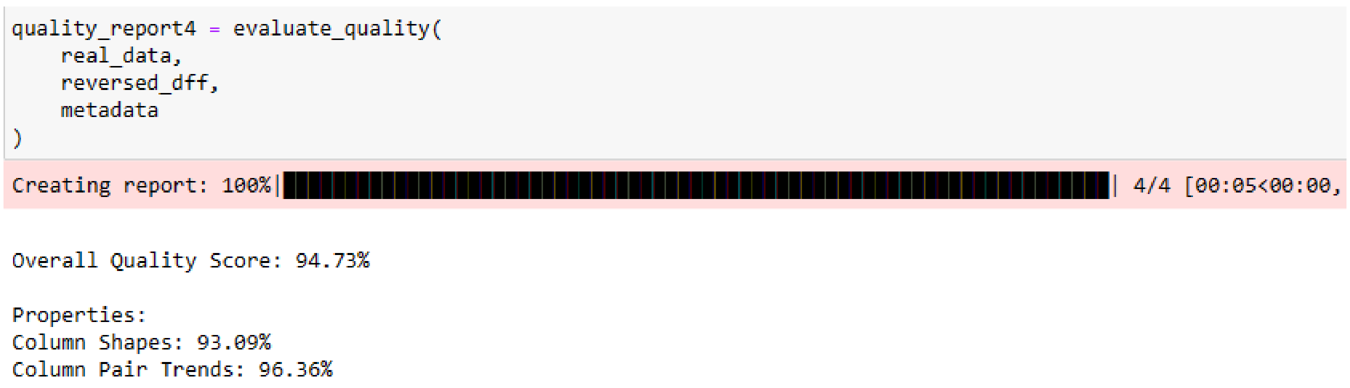
Figure 9.5: comparision of quality score of synthetic data generated by three algorithms.'Quality score_imp' by improved one



```
from sdv.evaluation.single_table import evaluate_quality

quality_report2 = evaluate_quality(
    real_data,
    synthetic_dataCTGAN,
    metadata
)
```

```
Creating report: 100%|██████████████████████████████████████████████████| 4/4 [00:04<00:0

Overall Quality Score: 93.56%

Properties:
Column Shapes: 92.03%
Column Pair Trends: 95.1%
```

Figure 9.6: Quality score of Taiwan synthetic dataset , generated by CTGAN



```
quality_report4 = evaluate_quality(
    real_data,
    reversed_dff,
    metadata
)
```

```
Creating report: 100%|██████████████████████████████████████████████████| 4/4 [00:05<00:00,

Overall Quality Score: 94.73%

Properties:
Column Shapes: 93.09%
Column Pair Trends: 96.36%
```

Figure 9.7: Quality score of Taiwan synthetic dataset , generated by improved one

9.2 German Credit Risk Data

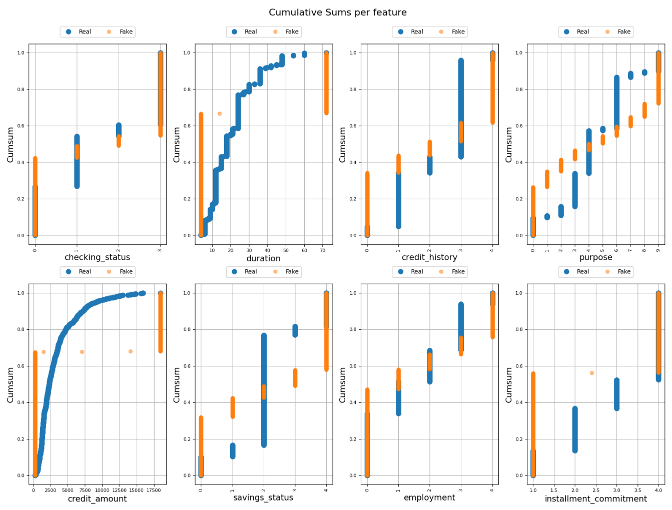


Figure 9.8: CDFs of some variables of synthetic data generated by VGAN on German data

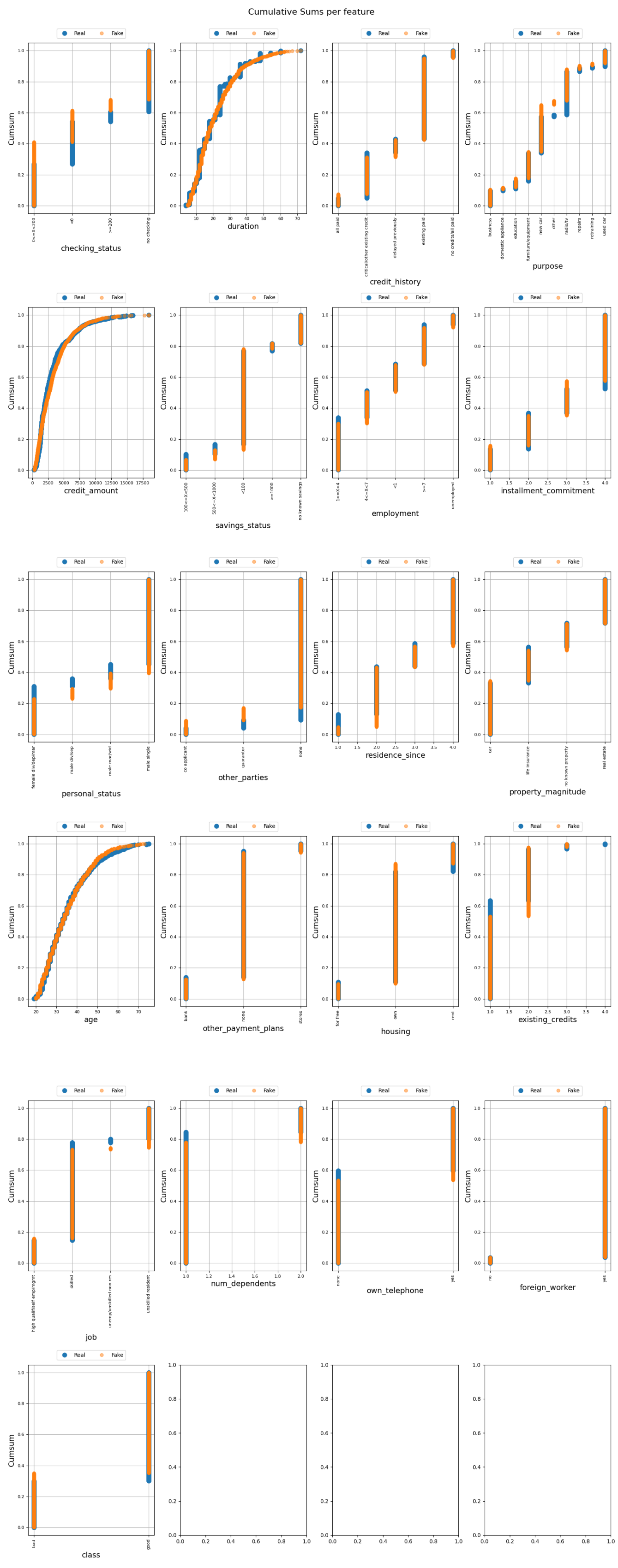


Figure 9.9: cdf of hybrid1 dataset on german credit data

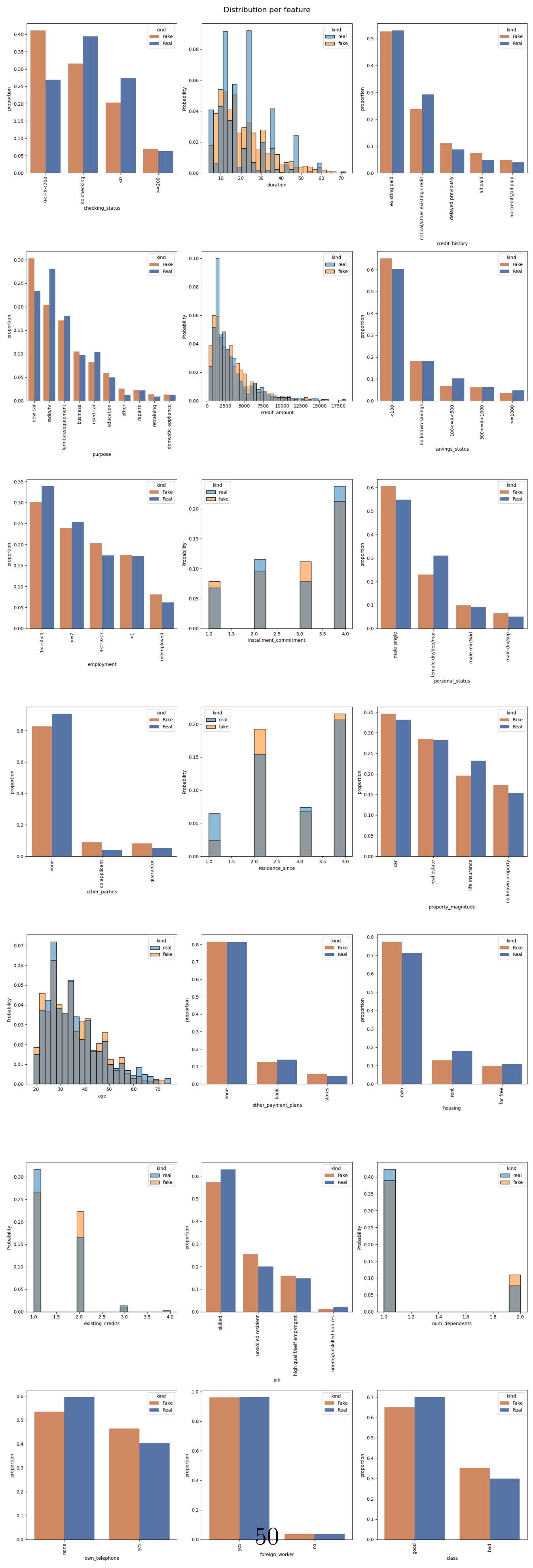


Figure 9.10: pdf of hybrid1 dataset on german credit data

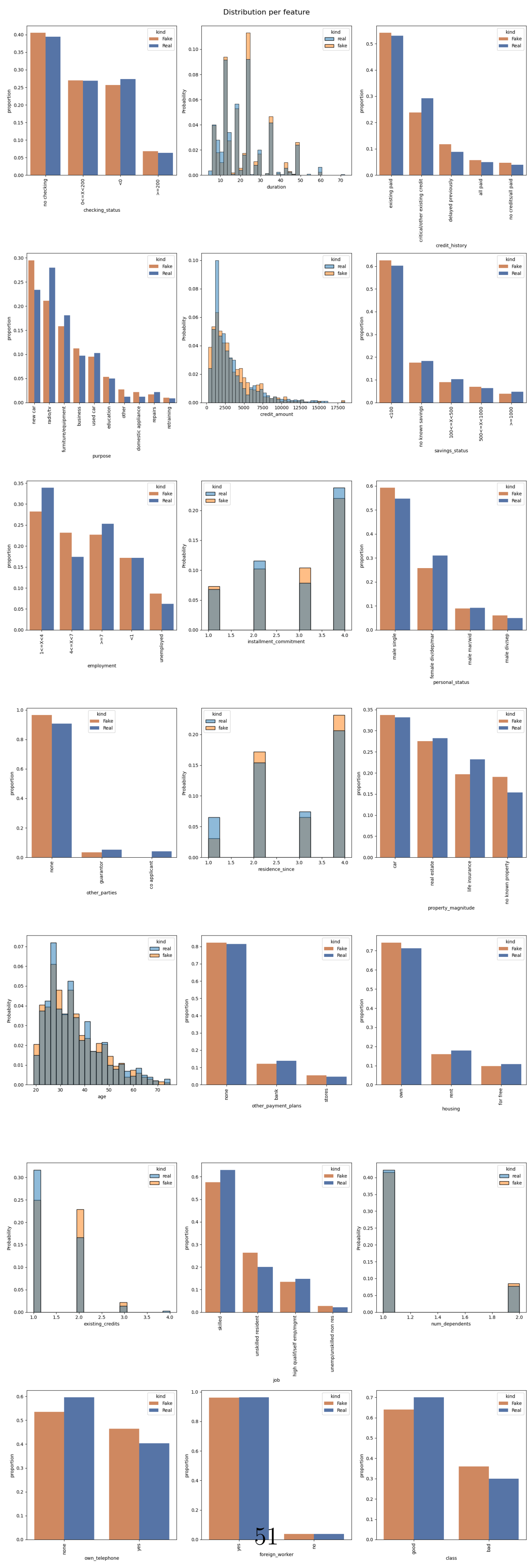


Figure 9.11: pdf of hybrid2 dataset on german credit data

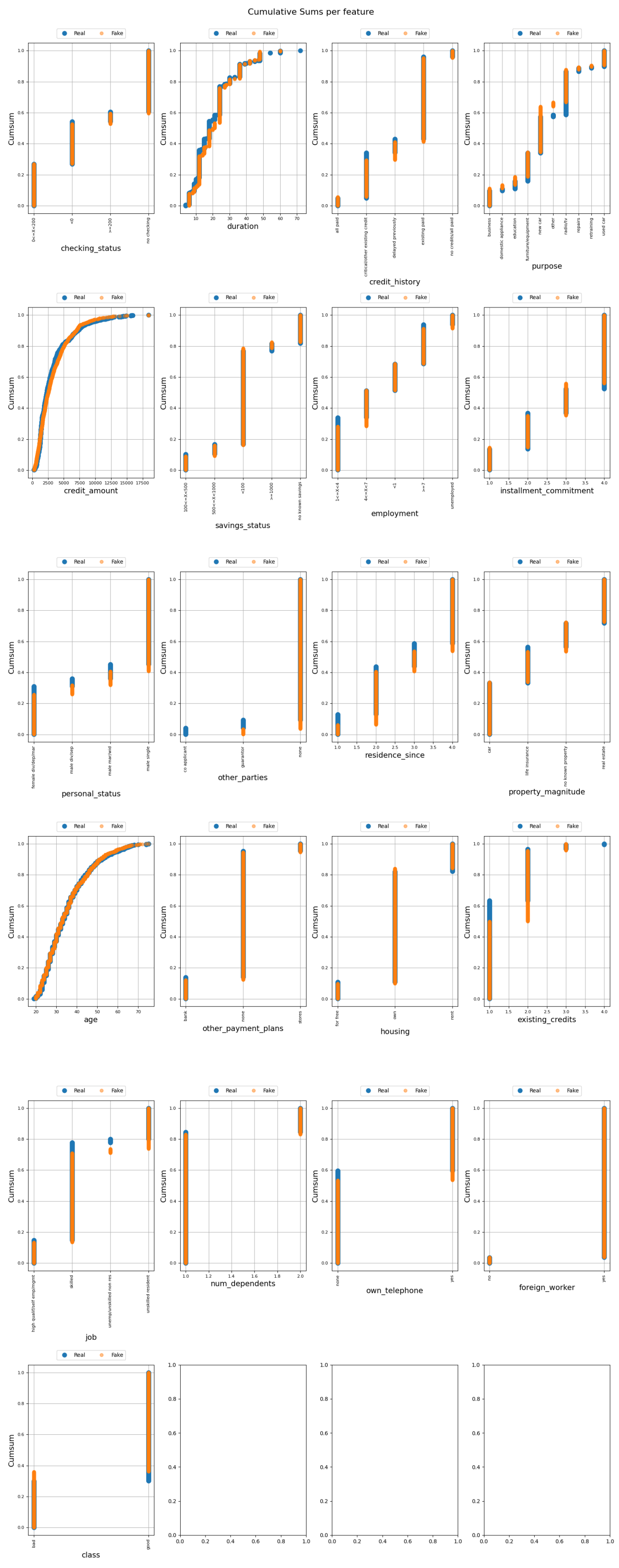


Figure 9.12: cdf of hybrid2 dataset on german credit data