Synthetic data generative models for DC government transaction data

May Chang, Yaoqi Liao

1 Introduction

The General Data Protection Regulation (GDPR) was officially implemented on 25th May 2018 to enforce a standardized data security law on all European Union members. This law determines how organizations and businesses should handle the information and aims to protect consumers' personal information. With privacy protection, many companies face a significant challenge in using the data at hand, especially for companies operating in the banking and financial industries since they have tons of sensitive customers' transactional data. To solve this challenge, one of the solutions could be anonymizing Personally Identifiable Information. However, there is still a risk of reidentifying customers' personal information by cross-referencing from other datasets. A safer approach could be replacing the real data with synthetic data, which reflects the real data mathematically and statistically. If some algorithms artificially generate the data, there is no way to cross-reference and reidentify consumers' information. Hence, in this project, we are going to produce a fully synthetic transactional data set by trying some existing data-generative algorithms that could help those companies generate synthetic data.

Some important contributions of the project include:

1. Experience with the most advanced synthetic data generation package thoroughly (i.e., Synthetic data vault or SDV for short) in Python.
2. Try out another synthetic data generation algorithm developed in the R community(i.e., R synthpop). Originally, the synthpop library was released as an R package, but for code consistency, we used the reimplementation version of this package in Python called "py-synthpop."
3. Compare the pros and cons of different algorithms and discuss the frameworks and implementation considerations encountered during the project.
4. Discuss the models' performances by doing machine learning detection tests and checking statistical properties.

Section 2 details work that has been carried out in the area of synthetic data generation and is relevant to this project's scope. Section 3 describes the data set we used and the methods we tried by providing the necessary background information on each algorithm and explaining why we chose them. Section 4 provides the metrics we used and visualizes the performance comparisons among the different algorithms. Section 5 discusses the overall outcome, a summary of what we learned by doing this project, and some potential future work.

2 Related Work

2.1 Copula-based generative model

Copulas are functions that enable us to separate the marginal distributions from the dependency structure of a given multivariate distribution. A d-dimensional copula, $C : [0, 1]^d : \rightarrow [0, 1]$ is a cumulative distribution function (CDF) with uniform marginals. To apply copula to synthetic data generation, we can specify a probability distribution of a set of marginal distributions and a copula, both with learnable parameters. These parameters can then be learnt from the real data by maximum likelihood estimation. Then we can get the distribution with these parameters and draw samples from it to create a synthetic dataset. A commonly used copula is the Gaussian copula but other families of copula models could also be used to better model the underlying dependency structures [1].

2.2 GAN-based generative model

A generative adversarial network (GAN) is a class of machine learning frameworks designed by Ian Goodfellow and his colleagues in June 2014. The basic GAN architecture is good at generating fake images by learning from real images. A few MIT scholars and fellows proposed the Conditional Tabular Generative Adversarial Network (CTGAN)[2], making the model more suitable for generating synthetic tabular data. CTGAN is a modified version of the basic GAN architecture. The SDV project team implemented the CTGAN model in its library, and we will evaluate it in section 3.

2.3 VAE-based generative model

Besides the GAN architecture in deep learning, the Variational Autoencoders (VAEs) are also very popular in generating fake images. Unlike GAN, the VAE consists of an encoder and a decoder that is trained to minimize the reconstruction error between the encoded-decoded data and the initial data. But instead of encoding an input data point as a single point in the latent space, the VAE encodes it as a distribution. Hence we can use the trained decoder to generate synthetic data.[3]. The SDV project team also implemented the VAE model in its library, and we will evaluate it in section 3.

3 Methods

3.1 Data description

Although we mentioned that we are interested in studying consumers' transactional data, we could not find any real transactional data online, given it is one of the most sensitive pieces of consumer information that needs to be protected. Therefore, we used the open data source from the DC government[1]. The government is providing Purchase Card transaction data to let taxpayers know how their tax dollars are being spent. The data includes transaction records for different government agencies from 6th January 2009 to 16th December 2016. Each row represents one transaction record the government agency spent on a given date. Since we want to use the government transaction records to mimic consumer credit card transactions, we only picked three columns from the data set: agency name(this column is to mimic the transaction description in a real consumer transaction), transaction date, and transaction amount. Given the limited computing resources we have, we only select the data from January 2009 to December 2009 (28,235 rows) as our real data for training.

3.2 Models and Algorithms

3.2.1 Baseline model

To better understand and compare with the advanced models we used, we developed a baseline model. Given we have tabular data, the most straightforward approach to creating synthetic data would be to assume independence across variables. Hence we model the transaction date and agency name via categorical distributions and estimate the relevant statistics empirically. Then we used the Gaussian Kernel density estimation (KDE) for the transaction amount column to get the continuous empirical distribution. At last, we generated synthetic columns by sampling independently from these learned distributions. However, this baseline model we designed does

[1] The data source can be found at: https://data.world/codefordc/purchase-card-transactions

not consider dependencies between variables, which is not ideal. That's why we tried other more advanced methods to compare with it.

3.2.2 Synthpop CART model

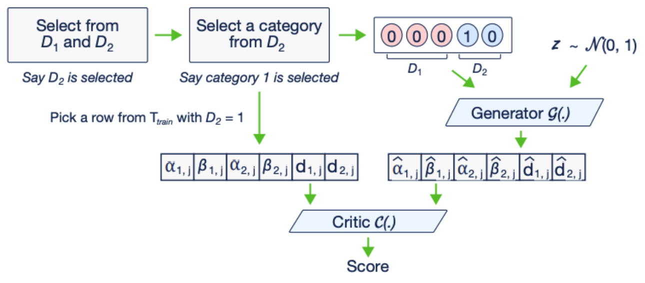
The CART model in the synthpop package is a little more advanced than our baseline model. It can capture the correlation to some extent. The algorithm behind the model follows these steps:

1. Assume that we have a data set with p variables
2. Pick a column and take bootstrap samples of that column $X_{1,obs}$ and set as $X_{1,syn}$
3. Fit a decision tree model $f(X_{2,obs}|X_{1,obs})$ and draw $X_{2,syn}$ from $f(X_{2,syn}|X_{1,syn})$
4. Fit another decision tree model $f(X_{3,obs}|X_{1,obs},\ X_{2,obs})$ and draw $X_{3,syn}$ from $f(X_{3,syn}|X_{1,syn},\ X_{2,syn})$
5. And so on, until $f(X_{p,syn}|X_{1,syn},\ X_{2,syn},...,\ X_{p-1,\ syn})$

Based on the algorithm, we picked the transaction date as the first column to get bootstrapped samples. Then we used the samples to train the CART model to get the transaction amount column and generated synthetic data for that column. Lastly, we trained the model to the agency name column and generated the synthetic agency names. The reason behind the chosen column order is that the algorithm cannot use the agency name as the first column to predict the transaction amount or transaction date. This is because the agency name is pure text, and the model cannot do label encoding or one hot encoding to transform it into numbers. Even if we do the one-hot encoding on the agency name column before fitting the model, the model will perform poorly because we have 89 different agency names, and the encoded columns will be highly unbalanced. From our implementation, one can notice that this model is unable to capture the correlations among all variables thoroughly but it is slightly better than the baseline model.

3.2.3 CTGAN model

The CTGAN model [2] is a GAN-based method to model tabular data distribution and sample rows from the distribution. The GAN architectures are often used for image generation. That's because there are many challenges associated with tabular data generation using GAN. One of the biggest challenges is the mixed data types. In real-world tabular data, we usually have both discrete and continuous columns. Unlike images, the continuous column in tabular data usually follows some non-gaussian distributions where min-max transformation will lead to a vanishing gradient problem. The discrete categorical columns are usually highly imbalanced, which will create severe mode collapse. Therefore, the CTGAN model employed model-specific normalization, a conditional generator, and a training-by-sampling technique to deal with the challenges.


Select from D_1 and D_2
Say D_2 is selected
Select a category from D_2
Say category 1 is selected
0 0 0 1 0
D_1
D_2
$z \sim \mathcal{N}(0,1)$
Generator $\mathcal{G}(.)$
Pick a row from T_{train} with $D_2 = 1$
$\alpha_{1,j}$ $\beta_{1,j}$ $\alpha_{2,j}$ $\beta_{2,j}$ $d_{1,j}$ $d_{2,j}$
$\hat{\alpha}_{1,j}$ $\hat{\beta}_{1,j}$ $\hat{\alpha}_{2,j}$ $\hat{\beta}_{2,j}$ $\hat{d}_{1,j}$ $\hat{d}_{2,j}$
Critic $\mathcal{C}(.)$
Score

In our project, the model performs the one-hot encoding transformation on the agency name column and uses those encoded columns as conditional vectors to generate the synthetic rows for the other columns (i.e., transaction date and transaction amount). But since we have more than 80 categories in the agency name columns, the overall training time took roughly 2 hours for less than 30,000 rows. In the end, we conclude that the model is not ideal for the data set with many categorical variables.

3.2.4 TVAE model

Variational autoencoder (VAE) is another generative model that is popular in the deep learning area. Like GAN, it is more common to see its application in image generation instead of tabular data generation. It is still possible to apply the encoder and decoder architecture and the evidence of lower-bound (ELBO) loss to train tabular data. The details of the modeling process have been studied by Xu and other fellows in their paper, and they named the model Tabular VAE (TVAE) [2].

In our project, we leveraged the TVAE model written in the SDV package and trained our data set using Adam with a learning rate of 1e-3. The data time column has been transformed into a continuous variable using its timestamp. The TVAE model also leverages the mode-specific normalization technique and uses a scalar to represent the continuous variable within the mode. Hence one of the underlying assumptions of our data set is that the continuous variables within their modes follow Gaussian distributions with different means and variance. However, we found that the TVAE model cannot perform as well as the CTGAN model, and we will discuss the metrics and performances in more detail in section 4.

3.2.5 HMA model

The Hierarchical Modeling Algorithm (HMA) [4] is very different from the deep learning models we discussed above. The idea behind the HMA is to iterate through tables sequentially using a modeling algorithm designed to account for the relations between the tables.
The relations between tables must follow the rules in a relational database. For example, the parent table has to have a primary key and the child table has to have a foreign key that refers to the object stored in the parent table.

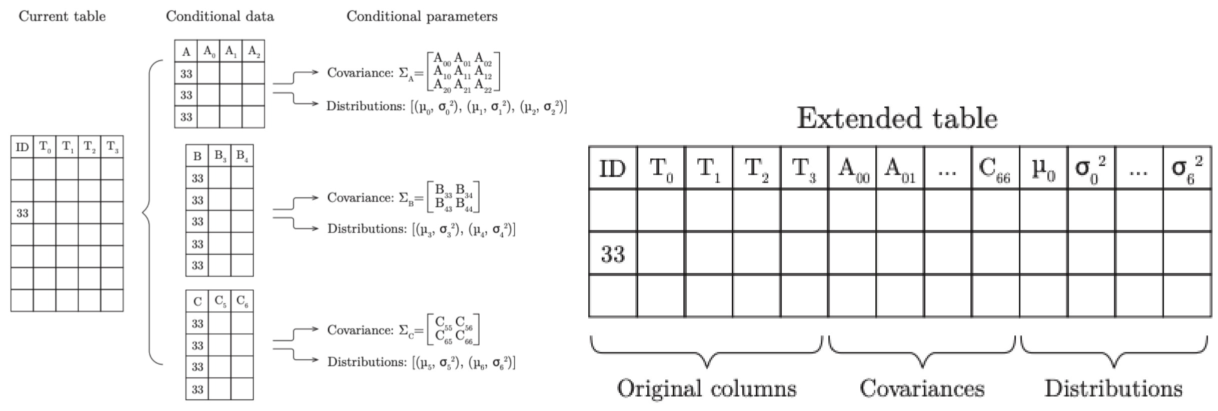
The algorithm[2] can be described as following steps:

1. Iterate through each row in the table.
2. Perform a primary key lookup in the entire database using the ID of that row. The rows that have the same foreign key in the child table form a conditional data set.
3. For each set of conditional data, perform the Gaussian Copula process and yield distributions and covariance matrices. We call these parameters as conditional parameters.
4. Add the conditional parameters as additional values for the row in the parent table.

Based on the algorithm, we split the original data set into two tables. One is called the agency table, which contains the agency name and agency ID (primary key); the other is called the transaction table, which includes the transaction ID, agency ID (foreign key), transaction date, and transaction amount. As we know, the Gaussian Copula cannot handle the categorical variable; hence when the HMA model encounters a categorical column, it will replace the column with numerical values in the range [0, 1] based on the proportions of that class. For

[2] More details and graphs of this algorithm can be found on page 4 of the paper The Synthetic data vault: https://dai.lids.mit.edu/wp-content/uploads/2018/03/SDV.pdf

example, if we have 50% class A, 40% class B, and 10% class C in the categorical column, then "A" is assigned to the interval [0, 0.5]; "B" is assigned to the interval [0.5, 0.9] and "C" is assigned to the interval [0.9, 1], and each occupies its allocated interval with a Gaussian distribution.



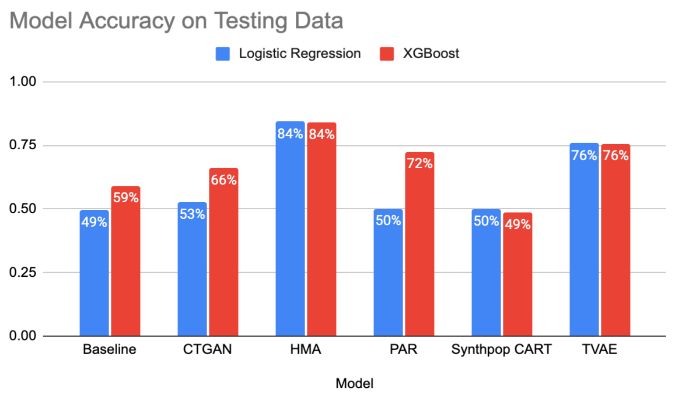
Current table
Conditional data
Conditional parameters
Covariance: $\Sigma_A = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix}$
Distributions: $[(\mu_0, \sigma_0^2), (\mu_1, \sigma_1^2), (\mu_2, \sigma_2^2)]$
Covariance: $\Sigma_B = \begin{bmatrix} B_{33} & B_{34} \\ B_{43} & B_{44} \end{bmatrix}$
Distributions: $[(\mu_3, \sigma_3^2), (\mu_4, \sigma_4^2)]$
Covariance: $\Sigma_C = \begin{bmatrix} C_{55} & C_{56} \\ C_{65} & C_{66} \end{bmatrix}$
Distributions: $[(\mu_5, \sigma_5^2), (\mu_6, \sigma_6^2)]$
Extended table
Original columns
Covariances
Distributions

3.2.6 PAR model

Unlike the models mentioned above, the Probabilistic AutoRegressive (PAR) model allows learning multi-type, multivariate time series data rather than single table data. Timeseries data exhibits inter-row dependencies as the order of the rows matter, and the sequence index parameter defines the order of the data. PAR model generates new synthetic data based on the external and abstract entities and their property associated with the time series data. Contextual variables that provide information about the entities related to the entity would provide additional information about each time series. However, our data doesn't have any contextual attributes that are expected to remain constant alongside each combination of entity variables. Hence we did not pass any context attributes to the model.

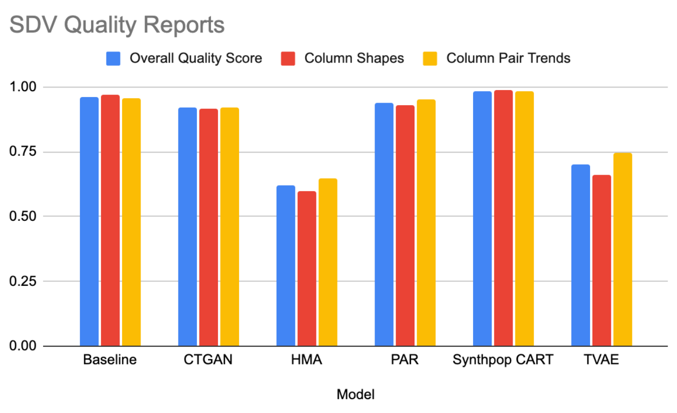
4 Evaluation and Analysis

Synthetic data have played a vital role in recent years as models have become more versatile and complex yet have reached the limitation of data generation and usage. Synthetic data is modeled data without sensitive information but still remains the crucial essence. If successful, a machine learning detection model cannot know if the data is synthetic. Therefore, we decided to use supervised models to evaluate the synthetic data quality and see how hard it is to distinguish the synthetic from the real data. Note that we aim to achieve a high accuracy score in most supervised learning tasks. However, in this detection task, we would like to see a low accuracy score indicating high-quality generated data.



To implement the detection task, we first labeled the real and synthetic data as 1 and 0. Then we split data into training and testing sets with a 70:30 ratio. We trained both logistic regression and XGboost classifiers with five-fold cross-validations and evaluated the model's accuracy on testing data.

Now, let's take a closer look at the models' performances: The synthetic data generated by the Synthpop CART model performs best, with low accuracy scores on both logistic regression and the XGBoost classifiers. Synthetic data generated by CTGAN and PAR models have similar low accuracy from the logistic regression classifier, but the XGBoost model has a higher accuracy score on data generated by the PAR model, which might be because we did not have the context attributes to help the PAR model to learn. Both classifiers can easily distinguish data generated by HMA and TAVE methods, which indicates that produced data from these two models does not closely resemble the actual data.



Using the evaluation metrics offered by the SDV library, we evaluated the shapes of the columns (marginal distributions) and the pairwise trends between the columns (correlation). A score is between 0 and 1, where 0 indicates the lowest quality and 1 indicates the highest. Different statistical distribution tests are used based on the metadata. The Kolmogorov-Smirnov Goodness-of-Fit Test (KSComplement) is used to decide if the synthetic transaction amount and date come from the known cumulative frequency distributions. Total Variation Distance (TVComplement) evaluates the difference in probabilities of frequency of each agency seen in the data.

The Synthpop CART model again produces high-quality synthetic data that closely resembles the actual data statistically and mathematically. It is important to note that, in the baseline model, as we calculated the empirical distributions for the real data and then generated new data from those distributions, the baseline model inevitably has high scores on statistical tests. Among the deep learning data synthesizers we tried in this project, the performance of the CTGAN model stood out - it captured the sample dataset's characteristics and performed well among other statistical-based models.

5 Discussion and Conclusion

As mentioned in the project proposal, we defined our success as exploring at least four different models and summarizing their pros and cons. Within a limited time frame, we successfully designed a baseline model and explored five different modern methods used for artificial data generation. We provided explanations of each of them and evaluated them using statistical tests and machine-learning detection techniques.

Having said that, we also acknowledged that there are a few limitations in this project as well:

1. Since we cannot find real customers' transactional data, the data set we used has its limitations on applications.
2. In addition, if we have a real consumers' transactional data set, we can do a more comprehensive evaluation of different models and try other metrics, such as privacy metrics and machine learning efficacy tests (e.g., train a machine learning model on real

data and synthetic data separately and see the difference between the model performances.)

Working on this project, we learned to apply machine learning concepts and techniques appropriately to solve supervised (detection tests) and unsupervised (data synthesizer) machine learning problems on a real-world dataset. We read multiple research papers and documentation to learn how to properly process our data to fit the models and correctly evaluate and interpret results. Our focus on synthetic data generation also makes us aware of data privacy issues, algorithmic bias, transparency, fairness, and social context in machine learning applications. If we have more time to continue working on the project, we will further explore the opportunity to utilize other attributes in the dataset. And we might be able to convert the transaction table to image format and generate synthetic data using a deep Convolutional Neural Network (CNN) model.

References

[1] Zheng Li, Yue Zhao, Jialin Fu. 2020. SYNC: A Copula based Framework for Generating Synthetic Data from Aggregated Sources

[2] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, Kalyan Veeramachaneni. 2019. Modeling Tabular data using Conditional GAN

[3] Zhiqiang Wan, Yazhou Zhang, Haibo He. 2017 Variational autoencoder based synthetic data generation for imbalance learning

[4] Neha Patki, Roy Wedge, Kalyan Veeramachaneni 2016 The Synthetic data vault