

Synthesizing Tabular Data Using Selectivity Enhanced Generative Adversarial Networks

[25pt Research Project]

by

Youran Zhou

Supervised by

Dr. Jianzhong Qi

THE UNIVERSITY OF MELBOURNE

Faculty of Science School of Mathematics and Statistics

This thesis submitted to the University of Melbourne for partial fulfillment of the degree of

Master of Data Science

March 2025

THE UNIVERSITY OF MELBOURNE

Abstract

While the fast pace of economic development, E-commerce platforms face significant challenges in handling excessive customer transactions during major online shopping events like Black Friday. To be prepared for large volumes of transactions, those platforms need to utilize synthesized data to run stress tests and derive the computational resources needed to cope with such transactions. The synthesized data for such patterns are usually in the form of tables.

Generating Adversarial Networks (GAN) are used in most recent tabular data synthesizing studies and have shown impressive performance in generating tabular data while fulfilling privacy constraints and downstream machine learning model training needs. However, existing studies do not apply to the E-commerce stress testing scenarios directly because the computational resources required to process the data generated by GAN have not been considered. A core concept in computational resource estimation for database transaction processing is query selectivity. To the best of our knowledge, no study has been conducted on supporting selectivity constraints in the tabular data synthesizing field.

This thesis considers query selectivity constraints in tabular data generation and offers solutions by designing a novel method for tabular generation GAN models. We add a pre-trained deep neural network component for an additional supervision signal to model the query selectivity constraint that maintains the selectivity consistency between ground truth data and synthetic data. We implement our method on top of two GAN models and evaluate them with extensive experiments against the three state-of-the-art GAN models and a VAE model on five real-world datasets. The results show that the synthetic data generated by our model resembles the real data, increasing the selectivity estimation accuracy by up to 20% and machine learning utilities by up to 6%.

Keywords: GAN, data synthesis, tabular data, selectivity estimation

Declaration of Authorship

I certify that this report does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text. The report is 11460 words in length (excluding text in images, tables, bibliographies and appendices).

Signed: Youran Zhou	
Date: 05/06/2022	

Acknowledgements

I would like to thank our supervisors, Dr Jianzhong Qi as well as Dr Wei Wang from the Hong Kong University of Science and Technology, for your guidance and constant patience and encouragement throughout the year. As I look back, this precious experience with you has been the highlight of my Master's program. When I struggled, you always pointed me in the right direction and supported me during our weekly meetings.

Thank you to Dr Qi for leading me to this research topic and showing me a different world I've never seen before. Preparing papers for me, scheduling our meetings, and helping me understand our project. Thank you for guiding my codes and experiments. Thank you for bearing my writing skills, providing detailed feedback and helping me with my thesis. It is my pleasure to have a great supervisor like you for my research project. I did grow and learned a lot from the past year. I am truly grateful for your kind words and encouragement.

Thank you to all kind staff from Spartan and IT support from the University of Melbourne for fixing my slurms and teaching me how to use the Spartan properly. Without you, I could not finish all of my experiments.

Lastly, I would like to thank my parents, my twin sister and our family mascot Jinzhi for sending me videos and voice calls to cheer me up, supporting and believing in me unconditionally.

Contents

A	bstra	\mathbf{ct}		j				
D	eclar	ation o	of Authorship	ii				
A	cknov	wledge	ments	iii				
Li	st of	Figure	es	vi				
Li	st of	Tables	5	vii				
1	Intr	Introduction						
	1.1	Proble	em	3				
	1.2	Contri	butions and Thesis Outline	4				
2	Rela	Related Work						
	2.1	Tabula	ar Data Generative models	6				
		2.1.1	Bayesian network	7				
		2.1.2	Autoencoder	9				
		2.1.3	Generative Adversarial Network	11				
		2.1.4	GAN Variants for Tabular Data Generation	15				
	2.2	Selecti	ivity Estimation	20				
		2.2.1	Regression-based Models	20				
3	Met	thodol	\mathbf{ogy}	22				
	3.1	Propo	sed Method	22				
		3.1.1	Data Transforming	22				
		3.1.2	Pre-trained Selectivity Model	25				
		3.1.3	Base Model Training	27				
		3.1.4	Base Model	29				
4	Exp	erime	\mathbf{nts}	31				
	4.1	Datase	et	31				
	4.2	Baseli	ne Models	32				
	4.3	Evalua	ation Metrics	33				
		4.3.1	Mode Collapse	33				
		122	Vigualization	22				

Contents v

		4.3.3	Selectivity Estimation	 . 33
		4.3.4	Machine Learning Utility	 . 34
	4.4	Param	neter Setting	 . 34
	4.5	Result	ts analysis	 . 35
		4.5.1	Mode Collapse	 . 35
		4.5.2	Visualization	 . 35
		4.5.3	Selectivity Estimation	 . 38
		4.5.4	Machine Learning Utility	 . 38
	4.6	Ablati	ion Study	 . 40
5	Con	clusio	ons and Future Work	42
	5.1	Conclu	lusions	 . 42
	5.2		re Direction	
A	Not	ations	S	46
В	Figu	ires		47
Bi	bliog	raphy	7	$\bf 52$

List of Figures

1.1	Example of using GAN to solve the tabular data shortage problem 3
2.1	Bayesian network over five attributes from PrivBayes [1]
2.2	Autoencoder scheme from [2]
2.3	Training evolution of DCGAN [3]
2.4	GAN Training Process
2.5	Architecture of MedGAN
2.6	Architecture of table-GAN
2.7	Architecture of CTGAN
3.1	Methodology Overview
3.2	Example of multi-modal distribution
3.3	Example of mode-specific normalization
3.4	Architecture of Pre-trained Selectivity Model
3.5	Base Model Training Process
4.1	age in Adult
4.2	aspect in Covertype
4.3	global subjectivity in News
4.4	Correlation Heap Map for Adult
4.5	Correlation Heap Map for News
4.6	ACC and F1 Score for Classification Task over five datasets
B.1	education-num in Adult
B.2	fnlwgt in Adult
B.3	capital-loss in Adult
B.4	elevation in Covertype
B.5	slope in Covertype
B.6	hillshade noon in Covertype
B.7	Amount in Credit
B.8	MktDistance in Ticket
	Passengers in Ticket
	title subjectivity in News
	shares in News
	average token length in News
	LDA00 in News
	Correlation Heap Map for Covertype
	Correlation Heap Map for Ticket
B.16	Correlation Heap Map for Credit

List of Tables

2.1	Summary for commonly used Tabular data generation GAN models	15
4.1	Summary of datasets	32
4.2	Model Compatible table	32
4.3	Rate of Repeated Data (%)	35
4.4	Difference in pair-wise correlation	37
4.5	Selectivity Estimation MSE in 10^2	38
4.6	Classification Accuracy (F1)	40
4.7	Regression Accuracy (MSE)	40
4.8	Ablation Selectivity Estimation MSE in 10 ²	41
4.9	Ablation Study: Regression Accuracy (MSE)	41
4.10	Ablation Study: Classification Accuracy (F1)	41
A.1	Notations	46

Chapter 1

Introduction

Back in 2017, The Economist published a story titled, 'The world's most valuable resource is no longer oil, but data.' Companies from a variety of industry fields gain valuable insights from using internal and external data sources. The desire of data raises the issue of data shortage. For instance, in the medical field, new technologies are utilizing patient health histories to create predictive models that can be used to improve diagnosis and understanding of illness. Rare diseases are challenging to study since we can only find a limited number of real-life datasets. In the E-commerce field, the online shopping platforms face the challenge from gigantic amount of transaction data during the major shopping events such as Black Friday, where a lack of computation resources may result in a blockage of user transactions, thus negatively impacting the revenue. They need a sufficient amount of data to do the stress testing to avoid such loss. As another example, scientists who work in the data science area are often faced with the problem of insufficient data when they are trying to train new and robust machine models.

On the other hand, big data often compromises privacy and results in unjustified analyses because of its immense knowledge. Some European governments implemented the European General Data Protection Regulation in order to prevent misuse of data and violations of privacy rights and to implement strict rules with respect to data protection in order to prevent privacy leaks. This poses a new challenge for the industries that are driven by big data to find solutions that will allow them to make big discoveries while respecting the privacy rights of individuals as well as mandatory government regulations.

One emerging solution is to rely on synthetic data rather than real data, which is statistically very close to real data and can satisfy privacy requirements due to its synthetic nature. However, there are some challenges for tabular data synthesizing tasks.

The major issues can be summarized as follows:

- 1. Data Shortage Issue: Generative models aim to produce sufficient outputs with a wide variety. Due to the limited number of input data, it is hard to consider both quantity and variety for synthetic data. Some models may suffer from mode collapse problems, which means the model can not generate various data. Thus it keeps generating the same output.
- 2. Data Privacy Issue: Tabular data usually contains users' sensitive information, which could be used to identify individuals and harm their privacy. The more similarity between synthetic data and real data indicates the better quality of synthetic data. However, synthetic data with high similarity could reveal users' information. Therefore, the tabular data synthesizer should try to protect users' information and maintain the high generating quality.
- 3. Data Quality Issue (Machine Learning Utility): Synthetic data are used for downstream machine learning model training needs in specific applications. That requires high-quality synthetic data. These high-quality data should satisfy the machine learning utility to complete the downstream tasks. That means if we train the machine learning model using the synthetic data, that should have a similar performance to the machine learning model trained by real data.
- 4. Data Quality Issue (Database Constraint): Databases are commonly used to store and manage tabular data. Some database constraints define specific properties that data in a database must comply with. The origin data define those properties. The synthetic data with high quality should fulfil these constraints as well.

Generative Adversarial Network (GAN) [4] is one of the most promising approaches to synthesize data. Initially, GAN was designed in the past to generate images. Now, it has been migrated to produce tabular datasets as well [5] [6] [7] [8]. Normally, a GAN model is trained on a real dataset. Once it is constructed, it can be used to efficiently generate tabular data.

Figure 1.1 shows the example of using GAN model to solves the **issue 1** successfully. Most of the recent works pay considerable attention to the **issue 2** and **issue 3**. TGAN [5] uses a reversible data transformer and Gaussian Mixture Model (GMM) to pre-process the categorical data, and numerical data further improves the ability to generate categorical data and the distribution of numerical data. The state-of-the-art CTGAN [6] augments the training procedure with mode-specific normalization and treats categorical variables as condition vectors and addresses data imbalance by employing a conditional generator. TableGAN-MCA [9] proposes a novel Membership Collision Attack against GANs, which

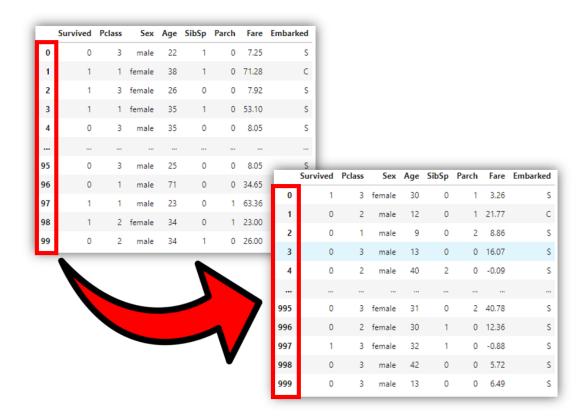


FIGURE 1.1: Example of using GAN to solve the tabular data shortage problem

allows an adversary given only synthetic entries randomly sampled from a black-box generator to recover partial GAN training data to immune Membership Inference attack. ITS-GAN [10] studies an incomplete table synthesis problem (using a very small proportion of real data to train GAN models) for tabular data augmentation. It used pre-trained functional dependencies models to enhance the GAN model in order to ensure the generated data satisfied the machine learning utility.

Although current works yielding success on the first three issues, $issue\ 4$ has not been well developed and solved.

1.1 Problem

Recall the E-commerce problem: the online platforms need to conduct the stress test and use sufficient data to estimate the computational resources required during major shopping events. Required computational resources can be formulated as the query execution cost. The query cost occurs when we take any actions on a table from our database. The query execution cost is computed by combining the cost of each of the operators appearing in the query plan. The standard operators include selection, projection, joint and so on. However, calculating the actual cost of query plans is usually

impossible without actually executing the plan. The only method we could calculate the execution cost is to estimate the cost of each operator separately and combine them. Equation 1.1 shows the idea of how to estimate the query execution cost.

$$\hat{\mathcal{C}}_{\text{Query execution}} = \hat{\mathcal{C}}_{\text{Selection}} + \hat{\mathcal{C}}_{\text{Projection}} + \hat{\mathcal{C}}_{\text{Joint}} + \cdots$$
 (1.1)

where \hat{C} indicates the estimated cost.

Despite the current GAN models having made great successes, the existing methods could not ensure their synthetic data fit the requirement for E-commerce platforms as there is a research gap between the **Issue 4** and current methods. Motivated by this problem, we start from the selection cost and take a further step to selectivity. To maximize the accuracy of selectivity cost, we should let the generated data satisfy the selectivity constraints from the original data.

The problem can be stated as:

'How to develop a tabular data generation GAN to model selectivity constraints in tabular data synthesizing'

Motivated by this question, this thesis conducts a series of studies on query selectivity constraint modeling for GAN-based tabular data generation.

1.2 Contributions and Thesis Outline

In this thesis, we design a GAN based tabular data synthesizer that fulfills query selectivity constraints. We propose a novel method to combine with state-of-the-art GAN models by introducing a pre-trained selectivity estimation deep neural network to provide additional control of the selectivity of generated data. By modifying the loss term of the GAN model to ensure the generated data could fit the selectivity constraint. We combined the method with two GAN models and tested on five widely used machine learning datasets against three GAN-based tabular data generation methods and one VAE-based method.

The main contributions of our work can be summarized as following:

- Improves the current data reversible transforming method to ensure the GAN model is suitable with any mixed-type data.
- Pre-trains a selectivity estimation model. Incorporates the selectivity score in training the generator, thus the synthesizing data can fulfill the selectivity constraints.

• The proposed augmentation method is flexible that could compatible any GAN based tabular data synthesizer.

The rest of the chapters are organized below:

In Chapter 2, we will discuss common approaches of tabular data synthesizing methods and the selectivity estimation methods.

In Chapter 3, we will introduce pre-trained selectivity model, GAN model architecture and how to combine them together.

In Chapter 4, we will talk about the implementation, parameter setting and present sufficient experimental results.

In Chapter 5, we will gave a summary of the experiments, proposes the limitation of our method, and provides the future improvement direction.

Chapter 2

Related Work

In this chapter, we will introduce some background information regarding traditional generative models for tabular data generation, Bayesian networks in statistics, variational autoencoders (VAEs) and generative adversarial networks (GANs) in computer science. Furthermore, we will discuss the different approaches to selectivity estimation, which include traditional estimation models and novel regression-based estimation models.

2.1 Tabular Data Generative models

Generative models are unsupervised machine learning models that attempt to discover the regularities, patterns and distributions from the input origin data, then use the learned knowledge to generate plausible data. The purpose of synthetic data generation is to resolve four issues, which are discussed in Chapter 1: data shortage issues, data privacy issues, data quality issues, including Machine Learning and Data Base issues. The models learn from the existing real data and generate their distributions from the acquired data, fulfilling requirements from the different industries and addressing the four issues.

Statistical generative models such as Bayesian networks and Gaussian mixture models are suitable for fitting certain probability distributions. However, in the real world, the datasets are often more complex and come in different formats. As a result, the statistical models are not usually compatible with image or text datasets.

2.1.1 Bayesian network

The Bayesian network [11] model is widely studied in the field of statistical and machine learning.

Assume A is the set of attributes on the dataset D. D has a joint probability distribution over the cross-product of A's attribute domains. A Bayesian network can be used to describe the distribution through the particular conditional independence between the attributes of A. To be specific, a Bayesian network is a directed acyclic graph (DAG) that represents each attribute in A as a node and conditional independence between attributes using directed edges. A simple Bayesian network schematic is shown in Figure 2.1. Bayesian networks are simple but powerful graphical models. They can approximate the complete-dimensional data distribution by combining low-dimensional data distributions.

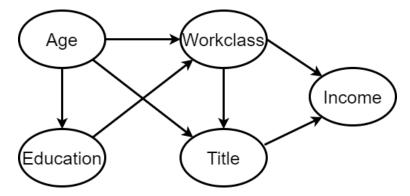


FIGURE 2.1: Bayesian network over five attributes from PrivBayes [1]

The standard Bayesian network generating synthetic data can be summarized as following steps:

- 1. Train a standard Bayesian network
- 2. Compute the differential privacy distributions of the data and then inject the Laplace noise to each parameter of the learned Bayesian network
- 3. Generate synthetic data from the noisy Bayesian network

However, the inappropriate amount or content of noise would lead the Bayesian network to a very poor generation performance. The other limitation of the traditional Bayesian networks is that they cannot handle continuous data, but they can represent a joint distribution of discrete variables.

PrivBayes

PrivBayes [1] was developed by ZHANG et al. in 2017. A novel Bayesian Network-based model provides a solution to protect differential privacy. The formal definition of ε -differential privacy is as follows:

$$\Pr[G(D_1) = O] \le e^{\varepsilon} \cdot \Pr[G(D_2) = O]$$

where $Pr[\cdot]$ is the probability of an event.

PrivBayes uses traditional Bayesian network architecture but a differential privacy learning algorithm to reduce the amount of noise that needs to be inserted. They compute a differential private Bayesian network that approximates the full-dimensional distribution using the Laplace mechanism and the exponential mechanism. Before the model, it discretized all continuous variables into 16 equal-sized bins to make the model more flexible for mixed-type datasets.

The PrivBayes can be constructed in to three stages:

- 1. Using ε_1 -differential privacy method to build a k-degree Bayesian network N through the attributes in dataset D.
- 2. Using ε_2 -differential privacy method to generate d, a set of conditional distributions for D. For example, each pair of conditional distribution $\Pr[X_i|\Pi_i]$ got a noisy distribution $\Pr^*[X_i|\Pi_i]$.
- 3. Use the Bayesian network N and the d noisy conditional distributions to derive an estimated distribution of the tuples in D, then sample tuples from the estimated distribution to generate a synthetic dataset D^* .

In phase 1, the choice of k is non-trivial. It involves a trade-off between the Bayesian network's original quality. A Bayesian network with a larger k keeps more information from the full-dimensional distribution $\Pr[A]$. For instance, a (d-1) - degree Bayesian network can fit the distribution. In contrast, if a 1 - degree Bayesian network is present, there is much information loss when fitting the distribution. To resolve this problem, they use a measure called θ - usefulness to provide a more choose k automatically to balance the accuracy of the Bayesian network. Through the constraint of θ - usefulness, \PrivBayes uses a greedy algorithm to maximize the mutual information and to optimally structure the tree-based Bayesian network.

2.1.2 Autoencoder

Back in 1987, Autoencoder [12] was first developed by Ballard. An autoencoder is a type of deep neural network used to solve an unsupervised learning task — representation learning. That means the autoencoder could efficiently learn the coding of datasets. Thus, it can usually be used to remove the redundancy and extract the important data features. More specifically, a deep neural network contains a bottleneck inside the network and then forces a compressed knowledge representation from the input data. Figure 2.2 shows the sample scheme for autoencoder.

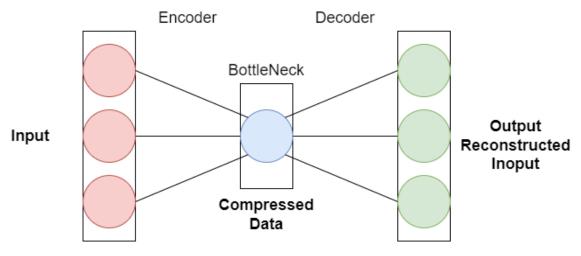


Figure 2.2: Autoencoder scheme from [2]

A standard autoencoder contains two components: an encoder $\mathcal{E}(\cdot)$ and a decoder $\mathcal{D}(\cdot)$. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code.

To be more specific, provide a dataset \mathbf{X} , the encoder $\mathcal{E}(\cdot)$ compresses the input data x from \mathbf{X} into a hidden distributed representation z (Compressed data from Figure 2.2). The encoding step can be shown as:

$$z = \mathcal{E}(x)$$
, where $x \sim \mathbf{X}$.

Then the decoder $\mathcal{D}(\cdot)$ takes the hidden representation z and reconstructs it. Lastly, it will produce \hat{x} :

$$\hat{x} = \mathcal{D}(z)$$
, where $\hat{x} \approx x$.

The generated data \hat{x} is approximate to the original data x because a successful autoencoder aims to extract all the essential features of x. The x and \hat{x} should share the same properties. Therefore, the autoencoder network is trained by minimizing the reconstruction error:

$$\mathcal{L} = (x, \hat{x}) = \mathbb{E}_{x \sim \mathbf{X}}[\|\mathcal{D}(\mathcal{E}(x)) - x\|_2^2]$$

Typically, the reconstruction error is the mean squared error, which measures the differences between the original input data and the later reconstruction \hat{x} .

Variational Autoencoder

The Variational Autoencoder (VAE) [13] is a variant of standard autoencoder. The 'variational' means that the encodings distribution is regularised during the training process to keep the approximate features and generate new data.

The architecture of a VAE is the same as a standard autoencoder. It contains an encoder $\mathcal{E}[\cdot]$ and a decoder $\mathcal{D}[\cdot]$. The VAE is trained using the similar reconstruction error $\mathcal{L} = (x, \hat{x})$, which aims to minimize the difference between origin data and the generated data as well.

As mentioned before, some regularisation term is introduced to VAE. A small modification is applied to the encoding-decoding step to achieve the regularisation. In a traditional autoencoder, the input data is a single point x from the original data \mathbf{X} , and the output is the hidden representation z. In VAE, the hidden representation is seen as a Gaussian distribution $\mathcal{N}[\cdot]$. Therefore the hidden representation from a VAE encoder forms a normal distribution $\mathcal{N}(\mu, \sigma^2)$.

$$\mu, \sigma = \mathcal{E}(x)$$
, where $x \sim \mathbf{X}$.

The decoder takes one sample z from the $\mathcal{N}(\mu, \sigma^2)$ and reconstructs the data.

$$\hat{x} = \mathbb{E}_{z \sim \mathcal{N}(\mu, \sigma)}[\mathcal{D}(z)], \text{ where } \hat{x} \approx x.$$

Additionally, VAE made a constraints to force all the aggregated distribution of z over all the data \mathbf{X} to be $\mathcal{N}(0, \mathbf{I})$. Under this constraint, we can input any vector sampled from $\mathcal{N}(0, \mathbf{I})$ into the trained decoder to generate new data.

The encoder-decoder pair is multi-input and multi-output deep neural networks and trained by stochastic gradient descent (SGD). The Equation 2.1 shows the reconstruction error is further modified as a evidence lower-bound (ELBO) loss.

$$\mathcal{L} = [\mathbb{E}_{x \sim \mathcal{N}(\mu, \sigma \mathbf{I})} [\|\mathcal{D}(\mathcal{E}(x)) - x\|_2^2] + \mathbb{KL}(\mathcal{N}(\mu, \sigma \mathbf{I}) \|\mathcal{N}(0, \mathbf{I}))].$$
 (2.1)

The first term is precisely the same as the traditional autoencoder. The second term $\mathbb{KL}(p||q)$ is the Kullback–Leibler (KL) divergence. The KL divergence is used to measure the distance between two distributions p and q using the following formula:

$$\mathbb{KL}(p||q) = -\int_{x} p(x) \log \frac{q(x)}{p(x)}$$

In the VAE case, we have the p as the standard normal distribution $\mathcal{N}(0, \mathbf{I})$, the q as the distribution of $z \sim \mathcal{N}(\mu, \sigma \mathbf{I})$. Thus, this term makes a constraint to ensure the z to be the standard normal distribution. The outer expectation is computed by taking the average over minibatch. The training process will end when the model converges. The learned \mathcal{D} is an approximated mapping from a multivariate Gaussian distribution to the data distribution.

2.1.3 Generative Adversarial Network

Generative Adversarial Networks (GANs) were firstly proposed by Goodfellow [4] in 2014. GAN is a generative model using deep learning techniques to generate different types of data to fit the requirements of variant industry needs.

Generative modelling is an unsupervised learning task. GAN models convert the problem from an unsupervised learning task to a supervised learning task masterly using two submodels: A generator \mathcal{G} and a discriminator \mathcal{D} .

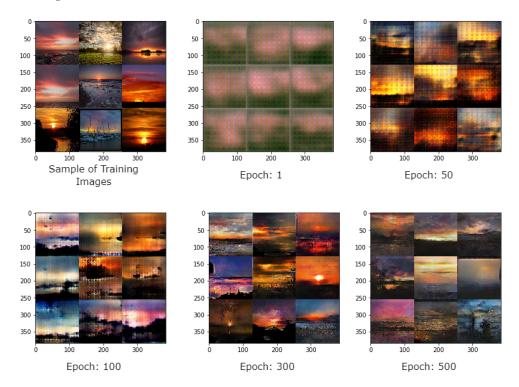


FIGURE 2.3: Training evolution of DCGAN [3]

The generator \mathcal{G} is trained to generate new samples, and the discriminator \mathcal{D} is required to recognize if the generated samples are real or fake. The training process is like an adversarial zero-sum competition as the \mathcal{G} have to foolish the \mathcal{D} and the \mathcal{D} tries its best to classify the provided data. During the training, the \mathcal{G} and \mathcal{D} grow together,

the quality of generated data is higher and higher, as well as the ability to recognize \mathcal{D} . Figure 2.3 shows the training evolution of a deep convolutional generative adversarial network [3] which is commonly used to generate images. It is an example of generating sunset images from the first epoch to the 500th epoch. The growth of \mathcal{G} and \mathcal{D} shows the adversarial process during training.

Vanilla GAN

As mentioned in the last section, the GAN model contains two deep neural networks that make the training process quite complete; it has to solve those complications:

- Handle two different training tasks (Generating and Classification)
- Identify training convergence

Figure 2.4 shows the flow chart for Vanilla GAN training. Vanilla GAN is the origin GAN training method proposed by Goodfellow [4]. The main training process can be broken down into two alternating steps:

- 1. Update Discriminator \mathcal{D} .
- 2. Update Generator \mathcal{G} .

The two steps run repeatedly to continuous training the \mathcal{G} and \mathcal{D} .

Step 1:

The Discriminator \mathcal{D} is a classifier which needs to distinguish if the data is generated by the Generator \mathcal{G} . To train the classifier more accurately, we should use the data from both data sources. The real data from the original data as the positive samples. The fake data generated by \mathcal{G} as the negative samples. During the training, the \mathcal{D} classifies both real data and fake data and produces a D loss. The discriminator loss will penalize the \mathcal{D} for all misclassified samples. The \mathcal{D} updates its weights through backpropagation from the D loss function Equation 2.2:

$$\mathcal{L}_{\mathcal{D}} = -\mathbb{E}_{x \sim \mathbf{X}}[\log(\mathcal{D}(x))] + \mathbb{E}_{z \sim \mathcal{N}(0, \mathbf{I})}[\log(1 - \mathcal{D}(\mathcal{G}(z)))]$$
(2.2)

 $\mathcal{L}_{\mathcal{D}}$ is the cross-entropy loss for binary classification.

Step 2:

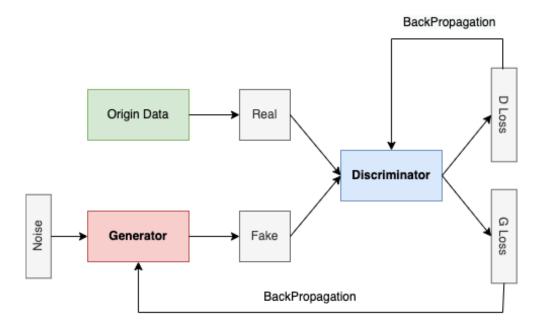


FIGURE 2.4: GAN Training Process

The Generator \mathcal{G} is used to create fake data incorporating feedback from the Discriminator. It aims to fool the \mathcal{D} and let \mathcal{D} can not complete the classification task well. The training process for \mathcal{G} is more complicated. The whole training process involves three components: Random Input which makes sure the GAN produce a wide variety of data; Generator \mathcal{G} which generates the data using the random feed input and the D loss, which enhances the generating ability by penalizes the \mathcal{G} for correctly classified samples. The Generator \mathcal{G} is optimized using the equation:

$$\mathcal{L}_{\mathcal{G}} = \mathbb{E}_{z \sim \mathcal{N}(0, \mathbf{I})}[\log(\mathcal{D}(\mathcal{G}(z)))]$$
(2.3)

Minimax Loss Function:

The MiniMax loss function (Equation 2.4) is commonly uses in standard GAN which is combined from the $\mathcal{L}_{\mathcal{G}}$ (Equation 2.3) and $\mathcal{L}_{\mathcal{D}}$ (Equation 2.2). In this function the generator tries to minimize the following function while the discriminator tries to maximize it.

$$\mathbb{E}_{x \sim \mathbf{X}}[\log(\mathcal{D}(x))] + \mathbb{E}_{z \sim \mathcal{N}(0,\mathbf{I})}[\log(1 - \mathcal{D}(\mathcal{G}(z)))]$$
(2.4)

In the first term, $\mathcal{D}(x)$ is the probability estimation to indicate if the real instance x is real. The E_x is the expected value among all real data instances. In the second term, $\mathcal{G}(z)$ is the generated data using the input noise z. $\mathcal{D}(\mathcal{G}(z))$ is the probability estimation to indicate if fake instance $\mathcal{G}(z)$ is real. The E_z is the expected value among all fake

data instances. The formula derives from the cross-entropy between the real and fake distributions. However, the Vanilla GAN always suffers from the mode collapse problem. Mode collapse means the Generator keeps producing the same output data.

Wassersttein GAN

Wassersttein GAN [14] is a improved training method to solve the mode collapse problem. WGAN uses a critic network $\mathcal{C}[\cdot]$. The output of the critic network $\mathcal{C}[\cdot]$ is dynamic. When the input is more realistic, the output value is larger or vice versa.

To achieve this, the critic network is trained using:

$$\mathcal{L}_{\mathcal{C}} = -\mathbb{E}_{x \sim \mathbf{X}}[\log(\mathcal{C}(x))] + \mathbb{E}_{z \sim \mathcal{N}(0,\mathbf{I})}[\log(1 - \mathcal{C}(\mathcal{G}(z)))]$$
(2.5)

That means, the critic network is trying to maximize the output on real data and minimized the output of the generated data. Conversely, the generator is trying to maximize the output of the critic network using the following function:

$$\mathcal{L}_{\mathcal{G}} = -\mathbb{E}_{z \sim \mathcal{N}(0, \mathbf{I})}[\log(1 - \mathcal{C}(\mathcal{G}(z)))]$$

WGAN aims to minimize the Wasserstein distance between the generated and real data distribution. The Wasserstein distance will provide the between probability distributions on a given metric space. The Wasserstein distance $\mathbb{W}(\cdot)$ can be shown as:

$$\mathbb{W}(\mathbf{x}, \mathbb{P}_{\mathcal{G}}) = \sup_{\|f\|_{L} \le 1} \mathbb{E}_{x \sim \mathbf{X}}[f(x)] - \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\mathcal{G}}}[f(\hat{x})]$$

where $||f||_L \leq 1$ indicates f is a 1-Lipschitz function.

Equation 2.5 can be seen as approaching the Wasserstein distance equation. Therefore, the training process for the Generator can be seen as minimizing the Wasserstein distance. Note that parameters in the \mathcal{C} are controlled to fit the 1-Lipschitz constraint.

The WGAN is also trained by stochastic gradient descent (SGD) with the similar steps to Vanilla GAN: In the first step, WGAN updates Critic \mathcal{C} , the next step is to update Generator \mathcal{G} . The two steps repeatedly run to continuous training the \mathcal{C} and \mathcal{D} .

Mostly, the training processes are the same as the Vanilla GAN training processes. Besides, we use Critic \mathcal{C} instead of Discriminator \mathcal{G} . Additionally, the parameter in \mathcal{C} has to be modified to satisfy the 1-Lipschitz condition in step 1.

2.1.4 GAN Variants for Tabular Data Generation

The initial GAN model was used to synthesize the image [4]. The development of GAN has led to more and more varieties being proposed in different fields. Now, GAN models have migrated from image data to tabular data. This session summarizes the most commonly used GAN-based tabular data generation models.

Model	Discrete	Numerical	Enhanced
MedGAN (2017)		√	High-Dimensional
$\mathtt{tablgeGAN}\ (2017)$		✓	Data Semantic
${\tt CTGAN}\ (2019)$	✓	✓	Imbalanced Data
$\mathtt{OCTGAN}\ (2021)$	✓	✓	Imbalanced Data

Table 2.1: Summary for commonly used Tabular data generation GAN models

MedGAN

MedGAN [15] was proposed by Choi et al. in 2017 to produce high-dimensional electronic health record (EHR) data in discrete variables. High-dimensional data suffers from the curse of dimensionality. To overcome this, MedGAN develop an autoencoder to learn the data representation. The original data can be represented as a low-dimensional data representation without any information loss by using the autoencoder.

MedGAN is limited applied to binary responses and continuous features. The binary features are represented as 1 or 0. The continuous features should apply a min-max normalization method to normalized to the range [0,1] using the MinMax normalization formula:

$$\frac{c_{i,j} - \min(C_i)}{\max(C_i) - \min(C_i)} \tag{2.6}$$

where C_i means the ith continuous column and c_{ij} means the jth value in the ith continuous column.

Figure 2.5 shows the Architecture of MedGAN. The discrete x comes from the source EHR data, z is the random prior for the generator \mathcal{G} ; \mathcal{G} is a feedforward network with shortcut connections (right-hand side figure); An autoencoder (i.e, the encoder **Enc** and decoder **Dec**) is learned from x; The same decoder **Dec** is used after the generator \mathcal{G} to construct the discrete output. The discriminator \mathcal{D} tries to differentiate real input x and discrete synthetic output $\mathbf{Dec}(\mathcal{G}(z))$. The pre-trained autoencoder is used to learn discrete features, and then it can be applied to decode the continuous output of \mathbf{G} . The autoencoder uses to mean a squared loss for the loss function to check if only continuous features exist in the generated data and cross-entropy loss to check if the

discrete columns are in binary. The loss function for the GAN model is the standard Minimax loss function (see Equation 2.6) from Vanilla GAN [4]. The loss function for the autoencoder is the mean squared error if the table contains only continuous columns and cross-entropy loss if the columns are all binary. The generator and discriminator are trained using the same loss function as a vanilla GAN.

However, the MedGAN does not support tabular data with mixed data types. Only continuous and binary discrete data is acceptable. The real-world data is usually complicated with the mixed data type, and this is not very suitable in most real-world scenarios.

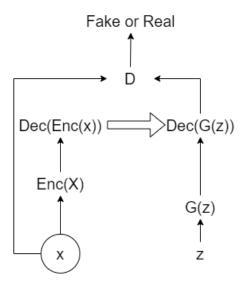


FIGURE 2.5: Architecture of MedGAN

Table-GAN

Table-GAN [8] is a variation on the GAN Architecture published by Park et al. in 2017. It applies the idea of DCGAN [3] which is a typically used image generation GAN model to generate tabular data to protect people's privacy. In this paper, Table-GAN is developed against three attacks: re-identification attack, attribute disclosure and membership attack.

The Table-GAN is developed from Deep Convolutional Generative Adversarial Network. Unlike the GAN model mentioned in the previous section, the Generator \mathcal{G} and the Discriminator \mathcal{D} are a pair of Convolutional neural networks and De-convolutional neural work. Additionally, Table-GAN add another neural network Classifier \mathcal{C} to supervise the semantic. Since DCGAN [3] was used for image generation. Thus the input data is not a vector but a matrix with a number. Thus, all the data should be prepossessed before training. For continuous variables, the MinMax normalization Equation 2.4 is applied; the discrete variables are converted to floating-point numbers or one-hot vectors. After

that, the preprocessed data should be re-arranged into a squared matrix. If the number of columns can not fill in a squared matrix, zeros are padded behind to increase the vector length to form a squared matrix. For example, if the vector length for preprocessed features is 12, then four zeros are padding behind, then the vector after padding can form a 4×4 matrix.

Figure 2.6 shows the architecture of the Generator \mathcal{G} and the Discriminator \mathcal{D} from Table-GAN. The Generator \mathcal{G} performs a series of deconvolution operations to generate data, while the Discriminator \mathcal{D} has the corresponding convolution layers to classify the real and fake data. The final loss after the sigmoid activation can be back-propagated to the Generator. The dimensions of the latent vector input z and intermediate tensors should be configured considering the number of attributes (e.g., $16 \times 16 = 196$ attributes in this figure). The Classifier \mathcal{C} increases the semantic integrity of synthetic records that have the same structure of the Discriminator \mathcal{D} . For example, (cholesterol=50, diabetes=1) is not a correct record because cholesterol=50 is too low to be diagnosed as diabetes. The \mathcal{C} is trained by the ground-truth label from the original table, therefore it can recognize if the generated data is semantic correct.

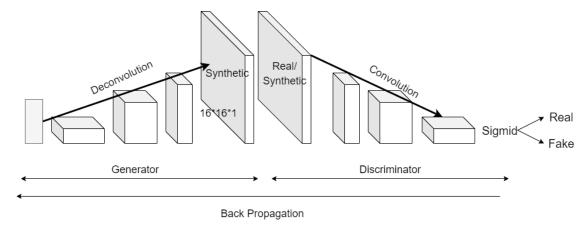


FIGURE 2.6: Architecture of table-GAN

The Table-GAN is trained using standard GAN loss function (Equation 2.4) with two additional term.

Information Loss

The information loss is defined as the discrepancy between two statistics of synthetic and real records. The information loss compares the first-order statistics(mean) and second-order statistics(sd) using the Equation 2.7:

$$\mathcal{L}_{mean} = \|\mathbb{E}[\mathbf{f}_x]_{x \sim p_{data}(x)} - \mathbb{E}[\mathbf{f}_{\mathcal{G}_{(z)}}]_{z \sim p(z)}\|_2$$

$$\mathcal{L}_{sd} = \|\mathbb{SD}[\mathbf{f}_x]_{x \sim p_{data}(x)} - \mathbb{SD}[\mathbf{f}_{\mathcal{G}_{(z)}}]_{z \sim p(z)}\|_2$$
(2.7)

The less value of \mathcal{L}_{mean} and \mathcal{L}_{sd} indicates that real and synthetic records have the statistically same features from the perspective of the discriminator. To make the privacy degree more controllable, the two loss terms are combined with two thresholds using Equation 2.8.

$$\mathcal{L}_{info}^{G} = \max(0, \mathcal{L}_{mean} - \delta_{mean}) + \max(0, \mathcal{L}_{sd} - \delta_{sd})$$
(2.8)

Classification Loss

Classification loss maintains the semantic integrity using the Equation 2.9. It measures the between the label of a generated record and the label predicted by the classifier for that record.

$$\mathcal{L}_{class}^{\mathcal{C}} = \mathbb{E}[|l(x) - \mathcal{C}(remove(x))|]_{x \sim p_{data}(x)},$$

$$\mathcal{L}_{class}^{\mathcal{G}} = \mathbb{E}[|l(\mathcal{G}(x)) - \mathcal{C}(remove(\mathcal{C}(x)))|]_{z \sim p(x)}$$
(2.9)

where $l(\cdot)$ is a function that returns the ground truth label, $remove(\cdot)$ is to remove the label attribute and $C(\cdot)$ is a label predicted by the classifier neural network.

Table-GAN is a novel approach for tabular data generation with convolutional neural network, but the CNN and classifier architecture restrict it only compatible with limited data type.

CT-GAN

CTGAN [6] was proposed by Xu et al. from MIT in 2019. This paper uses Mode-specific normalization to solve mixed data type and Non-Gaussian distribution assumption problems and uses Conditional vectors to solve imbalanced categorical column problems.

Data Preprocessing

The discrete columns are represented in a one-hot vector, the ith discrete column is donated as d_i . A special method called Mode-specific normalization is used to process continuous values. This method is used since the continuous columns in the real-world are usually not following a normal distribution but a Multi-modal distribution. In this paper, for each continuous column C_i , variational Gaussian mixture model is used to find the number of Modes. Then fit a Gaussian mixture and find the parameters Mean, Weight and Standard Deviation of a mode respectively. For each value $c_{i,j}$ in C_i compute the probability of $c_{i,j}$ coming from each mode. Using the most possible mode to normalize $c_{i,j}$, the normalized value donates as $\alpha_{i,j}$ and the chosen mode is represent in one-hot vector $\beta_{i,j}$.

After that, the representation of a row becomes the concatenation of continuous and

discrete columns can be written as follows:

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

Conditional Generator

Usually, when a GAN model is trained using an unbalanced dataset, the data in the minor category will not be sufficiently represented. CTGAN uses a conditional generator to enforce that the Generator matches a given category.

The **cond vector** is introduced to indicate the condition. Recall that after the data preprocessing, all the discrete columns $D_1...D_N$ end up as one-hot vector $d_1...d_{Nd}$, such that the ith one hot vector is d_i . The one-hot representation vector is the mask vector m_i for each discrete column D_i . All mask vectors are concated to form a **cond vector**. For instance, for two discrete columns, $D_1 = \{1, 2, 3\}$ and $D_2 = \{1, 2\}$, the condition for D_2 assigned to 1. Thus, the D_1 mask vectors $m_1 = [0, 0, 0]$ since no condition is assign to D_1 , and the D_2 mask vectors $m_2 = [1, 0]$; The final **cond vector** = [0, 0, 0, 1, 0].

CTGAN use fully connected hidden layers in both generator and critics (discriminator). For the generator, the input is a random variable z and output is our synthetic data. For the Critics, the PACGAN [16] framework is used to prevent mode collapse. The model is trained using WGAN loss(Equation 2.5) with gradient penalty and Adam optimizer. Figure 2.7 shows the architecture of CTGAN, the **cond** vector feed into the conditional generator, that generates synthetic conditioned rows. With training-by-sampling, the **cond** and training data are sampled according to the log-frequency of each category, thus CTGAN can evenly explore all possible discrete values.

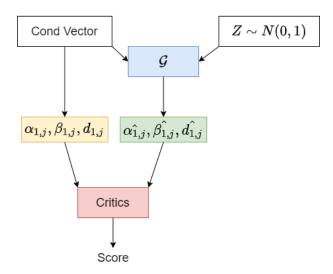


FIGURE 2.7: Architecture of CTGAN

Now, CTGAN has become a commonly used GAN-based tabular data generation method,

and it can generate data effectively with high-quality synthetic. Thus, CTGAN is one of the models we used as the base model to further develop our method.

2.2 Selectivity Estimation

Query execution cost estimates the number of computational resources (e.g. CPU and I/O resources) for running queries in the current query plan. The total query plan cost is computed by combining the cost of each of the operators appearing in the plan. Selection, Projection and Joint are three commonly used standard operators. In this thesis, we focus on the selection operation. Therefore, this session will introduce some regular approaches to the selectivity estimation method. In database systems, selectivity estimation has been extensively studied. Common approaches are sampling [17] and histograms [18]. However, most of them suffer from the curse of dimensionality, which means they could not be compatible with the high-dimensional data. Mattig et al. [19] use the Kernel-based cardinality to highlight the distribution of metric space. However, the kernel function usually needs to be supported by strong assumptions, and a single kernel function sometimes is insufficient to solve the complicated inner correlation of high-dimensional data.

2.2.1 Regression-based Models

Another method is to see the Selectivity estimation as a regression problem with query object and threshold as input features. Wang et al. [20] developed a Regression based model called **Selnet** to estimate the selectivity as well as maintain the consistency for high-dimensional data. Definition 2.1 shows how to form the selectivity to a regression problem.

Definition 2.1 (Selectivity). Given a database with d dimensional vector $\mathbb{D} = \{\mathbf{o}_i\}_{i=1}^n$, $\mathbf{o}_i \in \mathbb{R}^d$. Provide a distance function $dist(\cdot)$, a scaler threshold t and a query object $x \in \mathbb{R}^d$. The estimate the selectivity in the database can be written as:

$$|\{\mathbf{o} \mid dist(\mathbf{x}, o) \leq t, \mathbf{o} \in \mathbb{D}\}|$$

The selectivity (i.e., the ground truth label) y of a query object \mathbf{x} and a threshold t as generated by a value function:

$$y = f(\mathbf{x}, t, \mathbb{D})$$

Thus, the selective estimation can be seen as estimate $f(\mathbf{x}, t, \mathbb{D})$ using $\hat{f}(\mathbf{x}, t, \mathbb{D})$. However, f is complex, thus f is broke into small sub-functions to rather than using one

function to estimate f directly. For example, let $y = y_1 + y_2$ and $t = t_1 + t_2$. Then we can use two linear model to estimate corresponding y_1 and y_2 with t_1 and t_2 in the range of $[0, t_1]$ and $(t_1, t_2]$. Following this idea, the Threshold Partitioning method (Definition 2.2) is adopt.

Definition 2.2 (Threshold Partitioning). Assume the maximum threshold is t_{max} , the t_{max} is divided it with an increasing sequence of (L+2) value: $[\tau_0, \tau_1, \dots, \tau_{L+1}]$. We have $\tau_0 = 0$ and $\tau_{L+1} = t_{max} + \epsilon$ where ϵ is a small positive quantity to cover corner cases. Let $g_i(\mathbf{x}, t)$ be an interpolant function for interval $[\tau_{i-1}, \tau_i)$ and we have:

$$\hat{f}(\mathbf{x}, t, \mathcal{D}) = \sum_{i=1}^{L+1} \mathbf{1}[t \in [\tau_{i-1}, \tau_i)] \cdot g_i(\mathbf{x}, t)$$
(2.10)

Where $\mathbf{1}[\]$ is the indicator function.

Selnet uses L + 1 continuous piece-wise linear functions to implement $g_i(x, t)$ for the range of $[\tau_{i-1}, \tau_i)$ from Equation 2.10. The τ_i values are called **control points**. Let p_i be the estimated **Selectivities at Control Points** τ_i . The g_i function can written using τ_i and p_i through:

$$g_i(\mathbf{x},t) = p_{i-1} + \frac{t - \tau_{i-1}}{\tau_i - \tau_{i-1}} \cdot (p_i - p_{i-1})$$
(2.11)

Hence, τ_i and p_i are the two parameters for each sub-regression model. The final estimation function can be re-parameterized as $\hat{f}(\mathbf{x}, t, \mathbb{D}; \Theta)$ where Θ represents $\{(\tau_i, p_i)\}_{i=0}^{L+1}$. Regression-based estimation methods need to find the best control points and estimate the corresponding p_i , then combine the τ_i and p_i to estimate the total selectivity y.

Chapter 3

Methodology

This chapter will discuss the metrics we used and the methods used in our experiments. In section 3.1, we will discuss the proposed methodology we use to enhance the GAN model to fulfil query selectivity constraints. Section 4.3 will introduce the metrics used to measure the quality of the synthesizing data.

3.1 Proposed Method

We use the proposed methodology to enhance the Base Model to fulfil query selectivity constraints through the Pre-Trained Selectivity Model. The Base Model could be any existing GAN-based tabular data generation model. Figure 3.1 shows the overview of the method process. The whole process can break into three steps:

- 1. Data Transforming: Make the GAN model compatible with the mixed data type.
- 2. Pre-trained Selectivity Model: Provide supervision for generated data.
- 3. Base model Training: Incorporates the selective score during the Base Model training.

3.1.1 Data Transforming

Tabular data usually contains multiple columns with continuous mixed type and categorical type. From Chapter 2 we know that generating tabular data with mixed data types could be challenging for some of the existing GAN-based methods. CTGAN uses a Reversible Data Transforms(RDT) for data pre-processing to handle mixed data types.

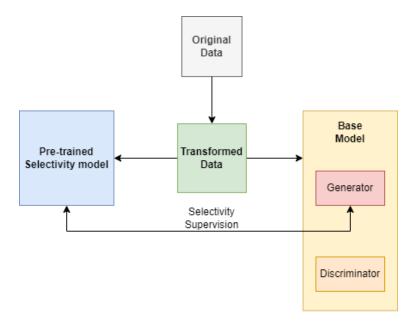


FIGURE 3.1: Methodology Overview

The RDT converts the categorical features to one-hot vectors and uses the "Mode-Specific Normalization" method to convert a continuous feature to the corresponding Mode and normalized value.

Categorical Data

There are two types of categorical data in tabular data, ordinal and nominal. Nominal data is classified without a natural order or rank (for example: Male and Female), whereas ordinal data has a predetermined or natural order (for example: Small, Medium and Large). CTGAN uses One-hot Encoding to convert both nominal and ordinal data. One-hot Encoding is a simple encoding method to creates additional features based on the number of unique values in the categorical feature. This encoding method usually used to convert nominal type categorical data, but it may not suitable for ordinal data. One-hot Encoding may not catch the natural ordered relationship between each response for ordinal data. For example, if we have a nominal categorical variable D_{gender} and an ordinal categorical variable D_{size} . One-hot Encoding represents $D_{gender} = \{F, M\}$ are [1,0] and [0,1], represents $D_{size} = \{small, medium, large\}$ are [1,0,0], [0,1,0] and [0,0,1]. Ont-hot Encoding provides an equal distance between outcomes. However, for ordinal variable D_{size} the distance between small and large should be larger than the distance between small and medium since the order matters in ordinal variables. The One-hot Encoding loses natural order information when converts ordinal variables.

As a result, we decide to use **Ordinal Encoding** for ordinal variables to maintain the inner ordered relations to make sure the GAN model can understand and harness this relationship during the generation step. By using the Ordinal Encoding, the representation of $D_{size} = \{small, medium, large\}$ is [1, 2, 3]. We use $d_{i,j}$ to represent the converted value from the ith categorical column and jth row.

Continuous Data

We continuous data we used the Mode-specific normalization method, which has been introduced in CTGAN from session 2.1.4. The Mode-specific normalization method normalizes a continuous column with multi-modal distributions. Figure 3.2 shows the histogram of daily temperature records through one year. It is a typical two-modal distribution example where the two distribution indicates the temperature in winter and summer.

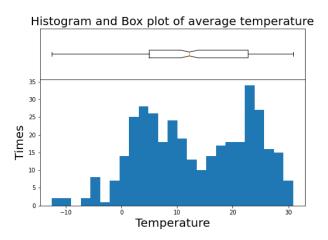


FIGURE 3.2: Example of multi-modal distribution

To be specific, the Mode-specific normalization uses variational Gaussian mixture (VGM) model to find the number of Modes for each continuous column C_i and the parameters mean η_k and standard deviation ϕ_k of each mode. Then, for each value $c_{i,j}$ in C_i compute the probability ρ_k of $c_{i,j}$ coming from each mode. Using the ρ_k to choose which mode the value belong to and using the parameters from the selected mode to normalized the $c_{i,j}$. Finally, record the normalized value $\alpha_{i,j}$ and the chosen mode represent in one-hot vector $\beta_{i,j}$.

For example, in Figure 3.3 there are three mode are found by VGM with the mean value of η_1, η_2 and η_3 . Each mode is a Gaussian distribution with the parameter mean η_k and standard deviation ϕ_k . Then a continuous value $c_{i,j}$ from column C_i appears. $c_{i,j}$ has the ρ_1 , ρ_2 and ρ_3 of coming from each mode. It is more likely to come from the third mode η_3 . Thus, $c_{i,j}$ is normalized by the parameters from the third mode, using the formulae:

$$\frac{c_{i,j}-\eta_k}{\phi_k}$$

We use $\alpha_{i,j}$ to represent the normalized value, $\beta_{i,j}$ (Mode indicator) is a one-hot vector which represents it allocated into the third mode [0,0,1]. Finally, a continuous value $c_{i,j}$ can be converted into the concatenate of $\alpha_{i,j}$ and $\beta_{i,j}$

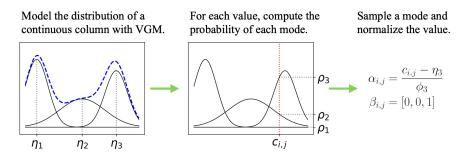


Figure 3.3: Example of mode-specific normalization

After the above transformation, the representation of the jth row becomes the concatenation of continuous and discrete columns can be written as follows:

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

A modified Reversible Data Transformer(mRDT) is built to transform the origin data into the processed data and convert the processed back to the original form.

3.1.2 Pre-trained Selectivity Model

In the second step, we need to train a model to estimate the selectivity for the original tabular data. After step one, the dimension of transformed data increases a lot. Therefore, we need a selectivity estimation model that can handle high-dimensional data. Finally, We decided to employ the simple version of Selnet [20] as our Pre-trained Selective Model. Selnet is designed to estimate the selectivity for high-dimensional data, which is quite suitable for our case. It is a regression-based deep learning model that learns a query-dependent piecewise linear function as a selectivity estimator. The original implementation of Selnet contains a Data Partitioning part to improve the accuracy of estimation on large-scale datasets. We drop this part for the current implementation.

Recall the **Definition 2.1 Selectivity** from Section 2.2.1:

Given a database with d dimensional vector $\mathbb{D} = \{\mathbf{o}_i\}_{i=1}^n$, $\mathbf{o}_i \in \mathbb{R}^d$. Provide a distance function $dist(\cdot)$, a scaler threshold t and a query object $x \in \mathbb{R}^d$. The estimate the selectivity in the database can be written as:

$$|\{\,\mathbf{o}\,|\,dist(\mathbf{x},o)\leq t,\mathbf{o}\in\mathbb{D}\}|$$

The selectivity (i.e., the ground truth label) y of a query object \mathbf{x} and a threshold t as generated by a value function:

$$y = f(\mathbf{x}, t, \mathbb{D})$$

Through the **Definition 2.2 Threshold Partitioning**, the estimation of selectivity can be written as Equation (2.10). We use this Threshold Partitioning method, which means L+1 piece-wise linear function is used to implement the interpolant function $g_i(x,t)$. Each g(x,t) contains corresponding τ_i and p_i . The final estimation function can be re-parameterized as $\hat{f}(\mathbf{x},t,\mathbb{D};\Theta)$ where Θ represents $\{(\tau_i,p_i)\}_{i=0}^{L+1}$. Therefore, we need to find the best control points τ and estimate the selectivity p, then combine all the τ and p to calculate the total selectivity \hat{y} .

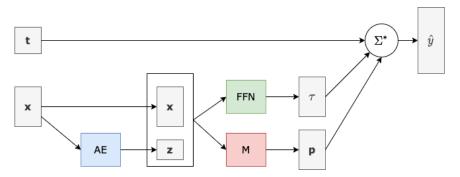


FIGURE 3.4: Architecture of Pre-trained Selectivity Model

Figure 3.4 shows the architecture of the Pre-trained Selectivity Model. It is a complicated model which combines three deep neural network components. We firstly input the query objects \mathbf{x} to an Autoencoder(\mathbf{AE}) to learn a latent representation. The \mathbf{AE} encourages the model to use latent representation and query distributions for the piecewise linear function. The \mathbf{AE} makes the model more generalized and better for handling query objects beyond the training data. Then the query objects \mathbf{x} is feed in to the \mathbf{AE} to learn the representation \mathbf{z} . The origin \mathbf{x} and representation \mathbf{z} is concatenated together to form [x;z]. After that the [x;z] is fed into two independent deep neural networks: a feed-forward network (\mathbf{FFN}) and \mathbf{M} . The \mathbf{M} is a encoder-decoder model. In the encoder, an \mathbf{FFN} is used to generate ($\mathbf{L} + 2$) embeddings:

$$[h_0; h_1; ...; h_{L+1}] = FFN([x; z]),$$

where h_i s are high-dimensional representations to represent the latent information of **p**. Then, we adopt adopt (L + 2) linear transformations with the ReLU activation function:

$$k_i = \text{ReLu}(w_i^T h_i + b_i)$$

Then, we have $\mathbf{p} = [k_0, k_0 + k_1, ... \sum_{i=0}^{L+1} k_i]$. The output of **FFN** and **M** can be converted to the τ and \mathbf{p} vector. Finally they can combine with the threshold t and fed into the operator Σ^* to compute the estimated selectivity \hat{y} . The Estimation Loss is used to estimate the loss between the true selectivity y and the estimated value \hat{y} of a query (x,t).

$$J_{\mathrm{est}}(\hat{j}) = \sum_{((x,t),y) \in \mathcal{T}_{\mathrm{train}}} l(f(x,t,\mathbb{D}), \hat{f}(x,t,\mathbb{D})) = l(y,\hat{y})$$

Due to the use of \mathbf{AE} , the final loss function (3.1) is a linear combination of the Estimation loss and the loss of the $\mathbf{AE}(J_{AE})$ during the training.

$$J(\hat{f}) = J_{\text{est}}(\hat{j}) + \lambda \cdot J_{AE} \tag{3.1}$$

In our setting, the Selnet is trained to make sure the GAN model satisfies the selectivity constraint. We denote the transformed T_{origin} through mRDT as \mathbb{D}_{origin} . Using the entire data set as the training query objects $\mathbf{Q}_{\text{train}}$ to ensure the Selnet can be trained properly. Then, the labels $\mathbf{y}_{\text{train}}$ and thresholds $\mathbf{t}_{\text{train}}$ are generated based on \mathbb{D}_{origin} . Then, we use the training query objects $\mathbf{Q}_{\text{train}}$ and $\mathbf{y}_{\text{train}}$ to train the selectivity model. After the training is done, the model is ready to evaluate the performance of any arbitrary synthesizing data through Mean Squared Error (MSE). The evaluation result can be written as follows:

$$\mathcal{L}_{Sel} = \text{MSE}(y, \hat{y}) \tag{3.2}$$

3.1.3 Base Model Training

After the Selectivity Model is trained, we use the trained selectivity model to enhance the Base Model. The Base Model could be modified from any existing GAN-based tabular data generation model. Figure 3.5 shows the flowchart of the training process of Base Model.

In a standard GAN model, the Generator \mathcal{G} will generate Fake data each iteration after receiving a random noise. Then the Real data from the original input and Fake data will both send into the Discriminator \mathcal{D} . The \mathcal{D} recognizes the Fake and Real, and then the feedback is produced to Generator Loss and Discriminator Loss, respectively. After that, the loss will be back-propagated to the \mathcal{G} and \mathcal{D} and start the next iteration. In our method, we can pick any arbitrary standard GAN model as the Base Model and then make two changes during the Base Model training so that the produced data from Base Model can satisfy our requirements. One is the Selectivity Evaluation for the Fake data, and the other is the Generator Loss Function Modification before back-propagation.

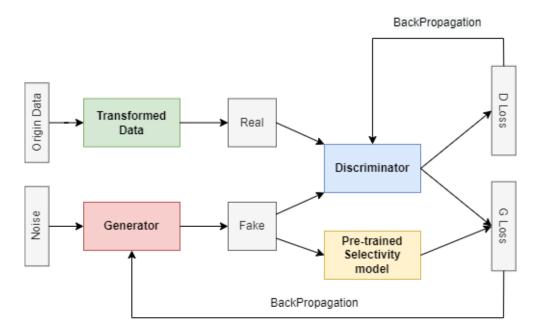


FIGURE 3.5: Base Model Training Process

Selectivity Evaluation

During the GAN training process, the generator synthesis a fake tabular data in each iteration. The synthesised data has the same format and dimension with the transformed data, therefore there is no need to further per-processed the synthesised data. Then for each Fake data, we generate the test query \mathbf{Q}_{test} , labels \mathbf{y}_{test} and thresholds \mathbf{t}_{test} . Note that the labels \mathbf{y}_{test} are computed on $\mathbb{D}_{\text{origin}}$, not \mathbf{Q}_{test} , thus the estimated selectivity performance will not be overestimated. Then the \mathbf{Q}_{test} and labels \mathbf{y}_{test} will send to the Pre-trained Selectivity Model to predict the $\hat{\mathbf{y}}_{\text{test}}$. The evaluation metric is MSE which is calculated through equation (3.2). The evaluation result is donated as \mathcal{L}_{Sel} . The \mathcal{L}_{Sel} indicates that if the \mathbf{Q}_{test} fulfill the selectivity constraints, the less MSE score means the better performance.

Generator Loss Function Modification

The selectivity estimation score \mathcal{L}_{Sel} could not only indicate the performance of the Fake data but also shows the ability of the Generator \mathcal{G} in the current status. To improve the capability of \mathcal{G} , we add the \mathcal{L}_{Sel} to the Generator Loss $\mathcal{L}_{\mathcal{G}}$. Thus the \mathcal{G} will modify itself to minimize the loss function then the selectivity constraints will be satisfied. Thus, the loss function for the Generator \mathcal{G} is adapted as:

$$\mathcal{L}_{G}^{*} = \mathcal{L}_{G} + \alpha \cdot \mathcal{L}_{Sel} \tag{3.3}$$

where the \mathcal{L}_{Sel} is the Selectivity Loss (3.2) and α is the hyper-parameter indicates the weight of the Selectivity Loss term to avoid the large selectivity Loss dominating the whole loss value. For the Discriminator \mathcal{D} we remain the same loss function. In general, the loss functions should be globally continuous and differentiable. Fortunately, the added term to Generator loss is MSE, one of the simplest and most typical loss functions. Thus the modified loss function should work well theoretically.

Overall, Algorithm 1 shows the Generator training algorithm.

Algorithm 1 Generator Training Algorithm

```
Input: T_{origin}
Output: Trained Generator \mathcal{G}
  1: Selnet \leftarrow Pre-Trained Selectivity model
  2: \mathcal{G} \leftarrow \text{Generator}
  3: \mathcal{D} \leftarrow \text{Discriminator}
  4: \mathbb{D}_{origin} \leftarrow \text{mRDT}(T_{origin})
     for number of epoch do
            for k steps do
  6:
                 Create a mini-batch of random noise Z = \{z_1, ..., z_n\}
  7:
                 Sample a mini-batch of real data X = \{X_1, ..., X_n\} from \mathbb{D}_{origin}
  8:
                 Train \mathcal{D} by maximizing equation (2.2)
  9:
            end for
10:
            Create a mini-batch of random noise Z = \{z_1, ..., z_n\}
11:
            Generate \mathbf{Q}_{\text{test}} and Labels \mathbf{y}_{\text{test}} using \mathcal{G}(Z) \triangleright Labels are computed on \mathbb{D}_{\text{origin}}
12:
            \mathcal{L}_{Sel} \leftarrow \mathtt{Selnet}([\mathbf{Q}_{	ext{test}}: \mathbf{y}_{	ext{test}}])
13:
            \mathcal{L}_{\mathcal{G}}^* = \mathcal{L}_{\mathcal{G}} + \alpha \cdot \mathcal{L}_{Sel}
14:
```

3.1.4 Base Model

Train \mathcal{G} by minimizing $\mathcal{L}_{\mathcal{G}}^*$

This thesis employs two existing models as our Base Model to test our proposed method's compatibility.

CTGAN

15:

16: end for 17: return \mathcal{G}

CTGAN is a commonly used baseline tabular data synthesizing GAN model. It could synthesize data in high quality and solve imbalanced data problems by introducing additional conditions. It uses RDT to prepossess data and then send the prepossessed data to the CTGAN model. CTGAN uses PACGAN framework to prevent mode collapse and WGAN loss (2.5) with gradient penalty and Adam optimizer. Figure 2.7 shows

the architecture of CTGAN. We use CTGAN as our Base Model and combine it with our method. The completed model we name it SelGAN.

Daisy

Daisy [21] is a survey paper which compares the different GAN-based frameworks in the tabular data synthesizing field. We want to use a standard GAN-based tabular data generation method to conduct further ablation studies to test whether our proposed method works well. In Daisy work, we find that utilizing a sequence generation mechanism (such as recurrent neural networks(RNN) and long short-term memory (LSTM)) as Generator could generate attributes separately in sequential timesteps and provide robust results. As a result, we decide to employ an LSTM as the Generator and a standard MLP as the Discriminator as the Base Model. Again, we use RDT to prepossess data and then send the prepossessed data to the Base Model and add our proposed method. The final resulted model we call it Daisy-sel.

Chapter 4

Experiments

In this chapter, we will discuss the implementation of experiments. We will briefly introduce the five real-world datasets in section 4.1. Section 4.2 and Section 4.4 will talk about the four baseline models and parameter setting for each models and dataset. Section 4.5 shows the evaluation results using the metrics mentioned in section 4.3. An ablation study is also conducted to demonstrate the efficacy of our method.

4.1 Dataset

In our experiments, we choose five commonly used real-world from UCI Machine Learning Repository and Kaggle.

- Adult¹: Contains the work-hour attribute has the information of work hours per week for each individual.
- Covertype²: Contains cartographic variables for four wilderness areas located in the Roosevelt National Forest of northern Colorado.
- Ticket³: Contains the data for the plane tickets.
- News⁴: Contains a heterogeneous set of features about articles published.
- CreditCard⁵: Contains PCA data with 28 dimensions of fraudulent credit card transactions information.

Those data sets contains both high and low dimension data with mixed type which means all dataset contain both numerical and categorical data. Table 4.1 shows the summary of chosen datasets. As a result, all data can be sent to both regression and

classification task in the later machine learning utility tests. For regression task we use educutation_num from Adult, soil_type from Covertype, amount from CreditCard, shares from News, and passengers from Ticket as our response. For classification task we use sex' from Adult, cover_type from Covertype, class from CreditCard, is_lifestyle from News, and mktcoupons from Ticket as our response.

Name	#Instance	#Columns	#Continuous	#Ordinal	#Nominal
Adult	30148	15	6	1	8
Covertype	77469	13	10	0	3
Ticket	5000	38	4	0	34
News	39644	60	46	0	14
CreditCard	14241	30	28	0	1

Table 4.1: Summary of datasets

4.2 Baseline Models

For baseline models, we mentioned existing works VAE, MedGAN, tablgeGAN, CTGAN, and OCTGAN in Chapter 2. MedGAN is limited to generated binary response which is not suitable for our data set. Therefore, we abandon the MedGAN. OCTGAN failed in Ticket due to mode collapse. tablgeGAN has a classifier component inside to maintain the semantic, which needs a binary response column as the label input for the classifier. Only Adult and CreditCard datasets contain a binary response. Therefore tablgeGAN only successfully works for those two datasets. Also, generated data could not produce the label input, and we do not adopt tablgeGAN to the Selectivity Estimation Test. SelGAN is one of our resulting model mentioned in 3.1.4.

Table 4.2 shows the compatibility for each model.

Model	Adult	Covertype	Ticket	News	CreditCard
SelGan	✓	✓	✓	✓	✓
CT-GAN	\checkmark	✓	✓	✓	✓
OCT-GAN	\checkmark	✓		✓	✓
tablGAN	\checkmark				✓
VAE	\checkmark	✓	✓	✓	✓

Table 4.2: Model Compatible table

¹Adult: http://archive.ics.uci.edu/ml/datasets/adult

 $^{^2} Covertype: \verb|http://archive.ics.uci.edu/ml/datasets/covertype|$

³Ticket: https://www.transtats.bts.gov/DataIndex.asp.

⁴News: https://archive.ics.uci.edu/ml/datasets/online+news+popularity

⁵CreditCard: https://www.kaggle.com/mlg-ulb/creditcardfraud

4.3 Evaluation Metrics

Quality assessment of generated data is not a simple task. Through different propose and setting, different evaluation metric is designed. We will introduce the designed metric and how to evaluate the synthesizing data in the three aspects.

4.3.1 Mode Collapse

Mode Collapse is a common problem for the GAN-based model. The synthesizing data is supposed to be diverse. Since the generator is always trying to find the one output that seems most plausible to the discriminator, thus if a generator produces a plausible output, it might learn only to produce that output. Therefore generators keep producing a small set of output over and over again. We will input the same amount of origin data for each model by testing this problem and letting them generate the same amount of data. Check if there are duplicated records exist in the generated data.

4.3.2 Visualization

We expect the synthesized data to be close to the original data. For numerical columns, one simplest way is to visualize the data distribution. Visualizations could provide us with a clear view of the comparison results. We could recognize if the model generates the correct number of modes or if the model can handle outliers. We compared the cumulative distribution functions (CDFs) of each column's origin data and synthesized data to evaluate whether a generated synthesized data is close to the origin data statistically. Also, we plot Pearson correlation heat maps. A correlation heat map is a graphical representation of a correlation matrix representing the correlation between different variables. These Pearson Correlation Heat Maps can test if the synthetic data contains the inner linear correlation between features.

4.3.3 Selectivity Estimation

As one of the major contributions of our works, we will evaluate the selectivity estimation score for T_{origin} and T_{Synth} . This test is conducted to compare the ability to handle selectivity for models. Similar to before, we used a pre-trained selectivity estimation model metric. Each T_{Synth} will generate the 1000 queries \mathbf{Q}_{Synth} based on \mathbb{D}_{origin} . The queries are sent into the pre-trained model and result in MSE score to indicate if the T_{Synth} satisfy the selectivity constraints. Each experiment is repeated ten times. The average lower MSE score shows better performance on fulfilling selectivity constraints.

4.3.4 Machine Learning Utility

Data utility is highly dependent on the specific needs of the synthetic data for down-streaming tasks. We use the Machine learning score to evaluate the effectiveness of using synthetic data as training data for machine learning tasks. We conduct both supervised learning tasks and unsupervised learning to evaluate the data quality thoroughly. For the supervised learning task, we first need to choose our task and separate the features X and labels y from the original data table T_{origin} . Then split the origin data feature X into the training set X_{train} and test set X_{test} . Then, we extract the features from synthesizing data table T_{Synth} denote as X_{Synth} . We use X_{train} and X_{Synth} to train the machine learning models. Then, we could use X_{test} to test the performance of each model. XGBoost, RandomForest and SVM are employed as regressors and classifiers to make the results more accurate. For the classification task, the accuracy or F1 score will be used. In the regression task, R^2 and MSE will be used.

4.4 Parameter Setting

All experiments are conducted on the 'Spartan' [22] server, which is the general purpose High-Performance Computing (HPC) system operated by Research Computing Services at The University of Melbourne. The partition **deeplearn** we used contains 13 nodes, each with four NVIDIA V100 graphics cards and six nodes each with four NVIDIA A100 graphics cards. The implementation for the Python version is 3.7.4 (compatible with TensorFlow 1.15, which is required for tableGAN) and 3.8.6.

We use source code for baseline models CT-GAN⁶, OCT-GAN⁷, tableGAN⁸, VAE and Daisy⁹ with their default parameters.

As our method is orthogonal to any existing GAN model, the performance of our method is largely dependent on the capability of the Base Model, and we do not need to do the parameter truing for the Base Model. SelGAN and Daisy-Sel uses the same parameters with CTGAN and Daisy. To ensure the fairness of experiments, we use batch size 500 for VAE and all GAN-based models with 300 training epochs using an Adam optimizer.

Selnet¹⁰ is used as the pre-trained selectivity component. We use complete origin data T_{origin} to train Selnet models with batch size 512. The training epoch for **AE** is 100

 $^{^6\}mathrm{CT} ext{-}\mathrm{GAN}$: https://github.com/sdv-dev/CTGAN

OCT-GAN: https://github.com/bigdyl-yonsei/OCTGAN

 $^{^8} able ext{-GAN https://github.com/mahmoodm2/tableGAN}$

⁹VAE and Daisy https://github.com/ruclty/Daisy

 $^{^{10}\}mathrm{Selnet:https://github.com/yyssl88/SelNet-Estimation}$

and 120 for other models. The parameter α from equation (3.3) is set as 0.01 to avoid large selectivity loss dominate the whole loss value.

4.5 Results analysis

In this session, we conduct a complete evaluation of our method to understand the quality of synthetic data in the metrics mentioned before.

4.5.1 Mode Collapse

Table 4.3 show the rate of repeated data. We use this method to check if the model suffers from mode collapse. The higher value indicates that the mode collapse phenomenon is serious. From the table, we can see that the VAE model suffers from the mode collapse problem the most. SelGAN is not suffering from this problem as well as its Base Model CTGAN. The standard capability of SelGAN depends on the Base Model. We can only comment that our method will not cause any mode collapse, but we could not make any comment on if our method could ease mode collapse.

Model	Adult	Covertype	Ticket	News	CreditCard
CT-GAN	0	0	0	0	0
OCT-GAN	0	0	1	0	0
table GAN	3.2	_	_	_	0
VAE	0.779	0	9.78	0	84.83
SelGAN	0	0	0	0	0

Table 4.3: Rate of Repeated Data (%)

4.5.2 Visualization

Due to the space limit, we do not reveal the complete visualization plots in this chapter, and more plots are shown in Appendix B.

CDF Comparison

Three interpret-able continuous columns are chosen for the CDF comparison. Figure 4.1, Figure 4.2 and Figure 4.3 shows age in Adult, aspect in Covertype and global_subjectivity in news. The left-hand side shows an overall CDFs comparison among all baseline model and our resulting model SelGAN. The comparison of CDFs

for SelGAN and SelGAN-w/o Sel are showns in the right hand side. The SelGAN-w/o Sel model removes the selectivity estimation component for ablation studies. From these plots, CDFs of SelGAN in orange are close to CDFs of T_{origin} in blue, suggesting that SelGAN performs quite well. However, SelGAN could not fit well at the beginning and end in Figure 4.2, that means our SelGAN does not always successfully capture the statistics of the T_{origin} . But, SelGAN still outperform among other base line models visually.

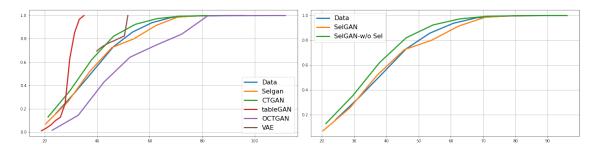


FIGURE 4.1: age in Adult

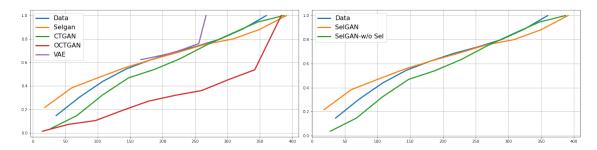


FIGURE 4.2: aspect in Covertype

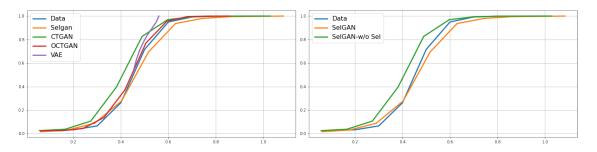


Figure 4.3: global subjectivity in News

Correlation Heat Map

Figure 4.4 and Figure 4.5 shows the Correlation Heap Map for Adult and CreditCard. From Figure 4.4, we can see SelGAN generates a similar pattern and color with the T_{origin} . CTGAN and VAE also generate similar patterns and colours, but there are some massive patterns inside. OCTGAN and tableGAN fail to produce the inner correlations.

From Figure 4.5, we find OCTGAN and VAE simulate the correlation pretty good, but CTGAN failed. SelGAN produce a similar colour but fails to generate the pattern. We realise that it is hard to evaluate the performance just by visuals. Thus, we conduct a difference in pair-wise correlation comparison test to assess the correlation performance at the quantity level. Table 4.4 summarize the difference in pair-wise correlation between the correlation matrix of origin data T_{origin} and correlation matrix of synthetic data T_{Synth} . The smaller value indicates that the synthetic data can mimic the correlation well. From this table, we find that SelGAN outperforms the other GAN-based models in two datasets. We also investigate an interesting phenomenon, VAE models have the lowest distance in Adult and Covertype, but it does not show ideal results in the other three datasets. However, SelGAN has the second-best results in Adult and Covertype. Thus we can conclude that overall, SelGAN can handle the inner correlation between dataset features well.

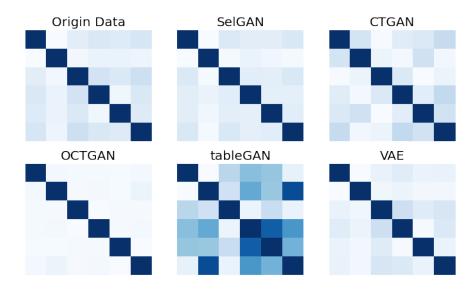


FIGURE 4.4: Correlation Heap Map for Adult

Model	Adult	Covertype	Ticket	News	CreditCard
CT-GAN	2.79	21.94	59.59	79.00	100.48
OCT-GAN	2.04	26.96	_	46.24	85.82
table GAN	10.09	_	_	_	467.84
VAE	0.76	15.99	91.41	54.9	554.05
SelGAN	0.93	19.79	45.54	75.07	66.13

Table 4.4: Difference in pair-wise correlation

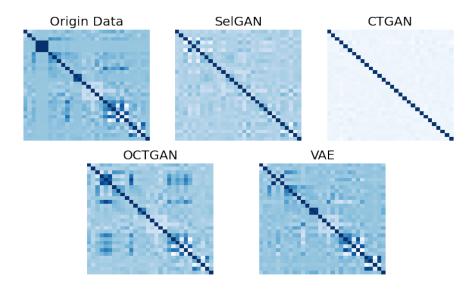


Figure 4.5: Correlation Heap Map for News

4.5.3 Selectivity Estimation

Table 4.5 shows the average summary results for the Selectivity estimation accuracy. From the table, we could see that there are clear reductions for SelGAN among all baseline models. The average decrease rate in MSE score is around 20%. Since the ability of our model is also dependent on the ability of the Base Model, we find that the scores are various between baseline models. Thus, further ablation studies are still required to understand better if our method can successfully enhance the existing model to fulfil the selectivity constraints.

Model	Adult	Covertype	Ticket	News	CreditCard
CT-GAN	40.93	98.82	66.12	111.38	230.75
OCT-GAN	55.53	128.04	_	163.73	247.22
VAE	63.49	97.27	68.07	132.29	238.40
SelGAN	23.86	$\bf 82.50$	54.83	104.36	200.01

Table 4.5: Selectivity Estimation MSE in 10^2

4.5.4 Machine Learning Utility

We conducted both classification and regression for all datasets. The predicted labels were mentioned in section 4.1. For classification tasks, We plot the Accuracy vs F1-score for all three machine learning models for five datasets (Figure 4.6). This plot shows that for the dataset Ticket and CreditCard, SelGAN largely outperforms all others

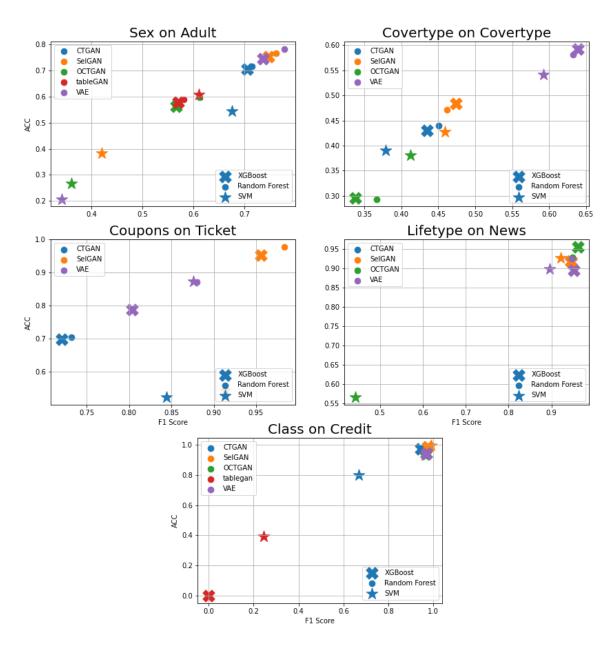


FIGURE 4.6: ACC and F1 Score for Classification Task over five datasets

across all used machine learning models. For dataset Adult and Covertype, SelGAN outperforms all GAN-based models. For datasets News, the best result and worst both come from OCTGAN, which means the performance of OCTGAN is quite volatile. SelGAN shows relatively stable results with high compatibility.

We also use Table 4.6 and Table 4.7 table to summarize the averaged machine learning utility score between T_{origin} and synthetic data in terms of accuracy, F1 score and MSE. A better synthetic data is expected to have a similar performance to the T_{origin} . The best evaluation scores have been labelled in boldface. From the two tables, we can see, that mostly SelGAN outperforms all other GAN-based state-of-the-art methods in

terms of F1-score and MSE. The averaged F1 across the three machine learning models increases up to 6%, and the averaged MSE decreases up to 20%.

Model	Adult	Covertype	Ticket	News	CreditCard
Origin	0.66	0.69	0.98	0.96	0.99
CT-GAN	0.65	0.42	0.80	0.91	0.91
OCT-GAN	0.47	0.32	_	0.81	0.93
table GAN	0.59	_	_	_	0.75
VAE	0.57	0.51	0.84	0.90	0.93
SelGAN	0.63	0.42	0.85	0.92	0.95

Table 4.6: Classification Accuracy (F1)

	Model	Adult	Covertype	Ticket	News	CreditCard
	Origin	4.17	49.03	51.67	12.03	4562
	CT-GAN	7.66	247.05	53.62	12.62	49226
]	OCT-GAN	9.38	116.50	_	74.23	68724
	table GAN	23.98	_	_	_	58099
	VAE	16.43	159.35	53.69	12.77	52410
	SelGAN	5.18	103.27	53.26	12.28	39484

Table 4.7: Regression Accuracy (MSE)

4.6 Ablation Study

To illustrate the efficiency of our method, we implement an ablation study. Due to the limitation of data resources, from Table 4.1 we realized there are not many ordinal columns in the experimental datasets. In our proposed method, we use an m-RDT preprocessing method for the proposed data to send to the downstream model more easily. Through m-RDT, we should use Ordinal encoding for ordinal attributes rather than one-hot vectors. That means we will not have enough experimental data to support the impact of using Ordinal encoding or One-Hot encoding. Therefore, we abandon the ablation study of the m-RDT component.

As mentioned in Section 3.1.4. We combined our method with two existing GAN base models. These two resulting models are used to test our method's flexibility and compatibility and check if our method can successfully enhance the synthetic data quality. We remove the pre-trained selectivity component from the model and then redo the test to see the performance differences.

The following are the two pairs of models we used for the ablation study:

- SelGAN and SelGAN-w/o Sel
- Daisy-Sel and Daisy

On the right-handed side of each figure from Figure 4.1, Figure 4.2 and Figure 4.3, the CDS distributions of between T_{origin} and T_{Synth} are revealed. Table 4.8, Table 4.9 and Table 4.10 shows the quantity comparison of selectivity estimation and machine learning performance. In Table 4.8, we find that the average decrease rate for adding the selectivity component is 27%, which is a significant drop in the selectivity score. That is strong evidence to say that our method does help the current GAN model meet the selectivity constraints. As well as the machine learning utility, we can observe trivial differences among them in most cases. In Table 4.10, Daisy-sel pair in CreditCard dataset and SelGAN pair in Covertype dataset share the same results. Therefore, both Daisy-sel and SelGAN show better results than the Daisy and SelGAN-w/o Sel respectively. Considering the high data utility in several datasets, it is crucial to use the selectivity component.

Model	Adult	Covertype	Ticket	News	CreditCard
Daisy	39.77	73.86	88.83	139.99	225.22
Daisy-sel	11.21	67.47	48.77	$\bf 82.35$	184.46
SelGAN-w/o Sel	40.93	98.82	66.12	111.38	230.75
SelGAN	23.86	82.50	54.83	104.36	200.01

Table 4.8: Ablation Selectivity Estimation MSE in 10²

Model	Adult	Covertype	Ticket	News	CreditCard
Daisy	8.89	261.19	152.66	15.20	55135
Daisy-sel	7.83	143.36	104.87	12.07	50798
SelGAN-w/o Sel	7.66	247.05	53.62	12.62	49226
SelGAN	5.18	103.27	53.26	12.28	39484

Table 4.9: Ablation Study: Regression Accuracy (MSE)

Model	Adult	Covertype	Ticket	News	CreditCard
Daisy	0.54	0.47	0.77	0.72	0.97
Daisy-sel	0.63	0.52	0.83	0.75	0.97
SelGAN-w/o Sel	0.65	0.42	0.80	0.91	0.91
SelGAN	0.63	0.42	0.85	0.92	0.95

Table 4.10: Ablation Study: Classification Accuracy (F1)

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis proposed a flexible method that could enhance any existing GAN-based tabular data generation model to fulfil the selectivity constraints. Through this method, we can synthesize data with a similar selectivity to the origin data T_{origin} . Then the synthetic data can be used to estimate query execution cost more accurately and further estimate the computational resources required if we create queries to the T_{origin} .

In Chapter 1, we introduce the background and aim of our project. We listed four challenges for current tabular data synthesizers, including Data Shortage Issue, Data Privacy Issue, and Data Quality Issues in both the Machine Learning utility and Data Constraints aspects. Then we talked about how the promising tabular data generation method GAN handles these four challenges. We found that current state-of-the-art GAN models have made successes in the first three challenges, but there is a research gap between the current method and the fourth challenge. Motivated by the E-commerce platforms problem, we decided to develop a tabular data generation GAN to model selectivity constraints in tabular data synthesizing and contribute to solving the last Issue.

In Chapter 2, we reviewed some common approaches of tabular data synthesizing methods from statistical and machine learning aspects. Bayesian network [11] is a statistical data generation method. It can describe a dataset as a directed acyclic graph. The dependency or relations can be represented through edges and nodes. In the machine learning aspect, we introduce the Variational Auto-encoder. It is a variant of a standard auto-encoder whose encoding distribution is regularised during the training process to keep the approximate features and generate new data. Then, we introduced some GAN

variants for tabular data generation including MedGAN, tableGAN, CTGAN and OCTGAN. They developed different methods to make the GAN model solve the first three challenges mentioned before. MedGAN is designed for high-dimensional medical record data, but it can only generate numerical and binary data. tableGAN uses a convolutional neural network as the generator \mathcal{G} and adds a classifier to maintain the semantics of synthetic data. CTGAN and OCTGAN are all aim to solve the in-balanced data distribution problem through a conditional vector and neural ordinary differential equations (NODEs). We also studied different approaches for selectivity estimation, such as the histogram-based and regression-based methods.

In Chapter 3, we talked about the proposed method after consulting many relevant materials and learning from GAN-based tabular data generation methods. Finally, we decided to develop a flexible method to combine any existing GAN-based tabular data generation method to model the selectivity constraints. We first need to send the origin data T_{origin} to train the selectivity estimation model. The pre-trained selectivity estimation model will supervise the Generator's performance during the GAN model training process. The supervision feedback term is added to the Generator loss function, and the Generator can use this feedback to adjust itself in each iteration. We also modified the current data pre-processing step to make the method more suitable for the various data types.

In Chapter 4, we introduce the detail of the experiments. Overall, our proposed method is an enhanced method to the current GAN based model; thus, the major capability in large depends on the Base Model we used to combine. To satisfy the selectivity constraints as well as the quality of the synthetic data, we implement our method to the state-of-art tabular data generation model CTGAN and resulting SelGAN. The selectivity estimation results show that SelGAN can model tabular data with selectivity constraints successfully, and it is also robust over different datasets to compare with three GAN-based models and one VAE model. Using the results from machine learning utility tests and statistical tests in visual, we can conclude SelGAN can also effectively model tabular data and generate high-quality synthetic data. We also combine a GAN model without outstanding performance, resulting Daisy-Sel to test the compatibility of our method. Finally, we conduct an ablation study for SelGAN and Daisy-Sel to analyze the importance of the pre-trained selectivity estimation term. We remove the pre-trained term and rerun the tests. The results show that our method is efficient.

5.2 Future Direction

There are still some limitations of our method. Therefore, in the future work, we plan to explore the following aspects:

1. More query operators could be considered

There are many query operators in a query execution plan. Now, we only focus on the selection. More commonly used query operators like projection and joint could be considered in the future.

The selection and projection involve only one single data table. Therefore, the modelling cost for selection and projection should be similar. We use projection queries to train the current model or to replace the current model with a projection estimation model.

However, modelling joint constraints should be more complicated since the joint operation involves two or more data tables. To model the joint constraints, we consider that we should use multiple parallel GAN models for each data table. Each GAN model will produce a single data table in each iteration, and then we should calculate the joint cost for the current generated table and provide them feedback. Then the feedback is sent to each GAN model to update the \mathcal{G} . The multiple GAN training method should be parallel to ensure all sub-GAN models can grow and interact together. One limitation for that the multiple GAN model could be too complicated. Since the number of GAN components depends on the joint constraints, we have to run the same number of sub-GAN components simultaneously if a joint constraint involves a large number of the data table. That could be a huge cost of computational resources.

2. Improvement on Selectivity component

Currently, we use Selnet as our pre-trained selectivity estimation model. We picked this Selnet because it can handle the high-dimensional data. We failed to use the sampling and histograms based or Gradient boosting trees regression-based estimation method due to the curse of dimensionality. In the future, more selectivity estimation models are worth trying to improve estimation accuracy further. To solve the high-dimensional data, we could try to train an Auto-Encoder to learn the representation of synthetic data. We sent the synthetic data into the Encoder to get the representation. Then we sent the representation into the estimation model to receive feedback and use the feedback to update \mathcal{G} . Even though the model could not handle the high-dimensional data, it can still handle the representation with lower dimensions. Nevertheless, we need to consider that

could the representation have any statistically meaning to fit the logic of estimation models.

In addition, we employed an existing selectivity estimation model previously. Nevertheless, we should have a novel, own-designed selectivity model for our scenario.

3. Satisfy industrial needs

This project is motivated by the E-commence platform cases. However, our current method is not yet satisfied industrial needs. We have to solve the previous two future directions and then consider them for industry application. There is one big challenge for the E-commence platform in a real-world scenario: we have to dynamically update the users' data. In the E-commence platform cases, users' data or transactions should be updated frequently. Our method can only handle the static data table. If the data table changes, we must redo the GAN model training. However, that should be a common issue for GAN-based models or even all static machine learning models. To solve this problem, we should consider a Dynamic Training method.

4. More experiments in other GAN variants

In this thesis, we only test two existing models. These two existing models have different architectures but still can not be representative enough. There are lots of varieties in GAN-based tabular generation methods. We can implement our method to more GAN models and further analyze whether the selectivity improvement would be changed due to the other GAN architectures.

5. Hyper-parameter setting

During the GAN training process, we add the \mathcal{L}_{Sel} into the $\mathcal{L}_{\mathcal{G}}$. We define the parameter $\alpha = 0.01$ to control the weight of \mathcal{L}_{Sel} . This weight is to ensure the \mathcal{L}_{Sel} will not dominate the whole loss function. However, the scale of \mathcal{L}_{Sel} for the different datasets is various, and a single value of α may not work well. Thus, more precision experiments are required. We may need to test new α for different datasets or use some normalization function to normalize the \mathcal{L}_{Sel} in a certain range.

Appendix A

Notations

Notation	Description
General:	
T_{origin}	Original Data
T_{Synth}	Synthetic data produced by model
Pre-processing:	
C_i	ith continuous column
D_i	ith discrete column
$c_{i,j}$	Value in the i-th continuous column of j-th row
$d_{i,j}$	Value in the i-th discrete column of j-th row
m-RDT	Modified Reversible Data Transforms
$lpha_{i,j}$	normalized value for $c_{i,j}$
$eta_{i,j}$	Mode representation for $c_{i,j}$
m-RDT	Modified Reversible Data Transforms
GAN model:	
\mathcal{D}	Discriminator
${\cal G}$	Generator
Query Generation:	
\mathbb{D}_{origin}	Transformed Original Data
\mathbf{Q}_{train}	testing query objects
\mathbf{Q}_{test}	training query objects
GAN training:	
$\mathcal{L}_{\mathcal{G}}$	Generator Loss
\mathcal{L}_{Sel}	Selectivity Loss
$\mathcal{L}_{\mathcal{D}}$	Discriminator Loss
$X \sim \{X_1, X_2, X_2\}$	Real data sampled from \mathbb{D}_{origin}
$Z \sim \{Z_1, Z_2, Z_2\}$	Noisy data sampled from $N \sim (0,1)$

Table A.1: Notations

Appendix B

Figures

Complete CDFs and correlation heat maps visualization plots from Chapter 4

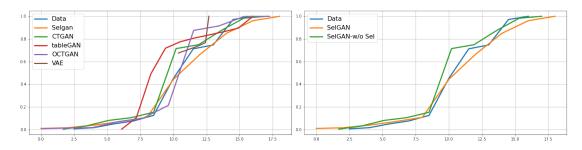


FIGURE B.1: education-num in Adult

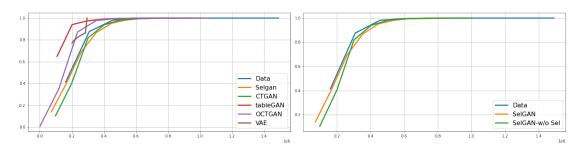


FIGURE B.2: fnlwgt in Adult

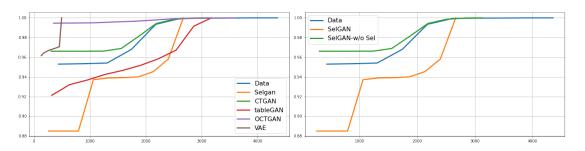


Figure B.3: capital-loss in \mathbf{Adult}

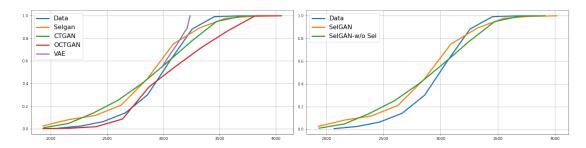


FIGURE B.4: elevation in Covertype

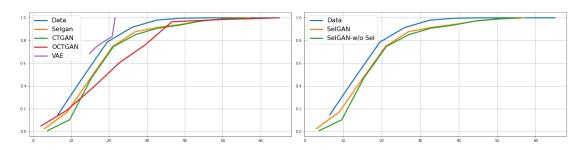


FIGURE B.5: slope in Covertype

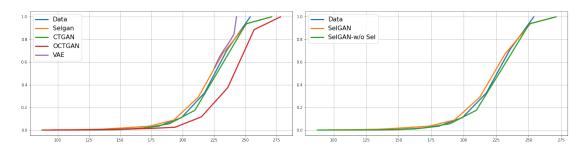


FIGURE B.6: hillshade noon in Covertype

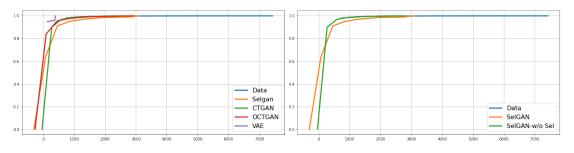


FIGURE B.7: Amount in Credit

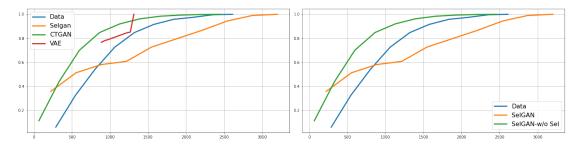


FIGURE B.8: MktDistance in Ticket

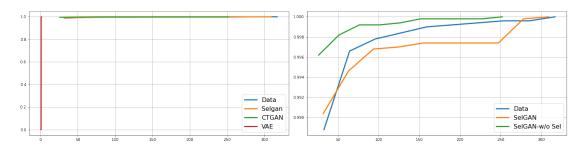


Figure B.9: Passengers in Ticket

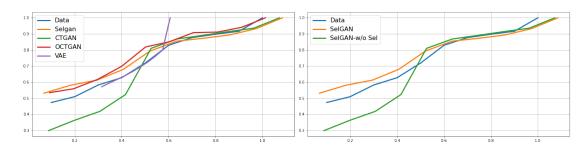


FIGURE B.10: title subjectivity in News

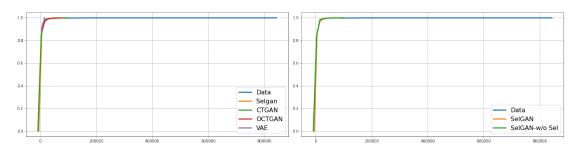


Figure B.11: shares in News

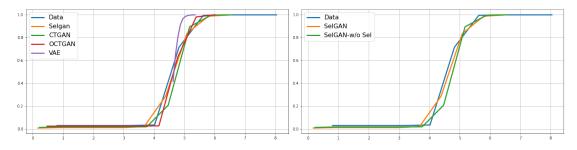


FIGURE B.12: average token length in News

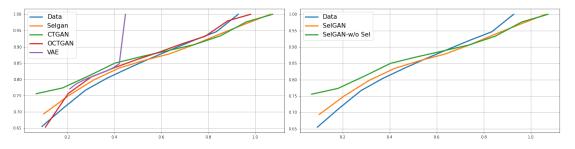


FIGURE B.13: LDA00 in News

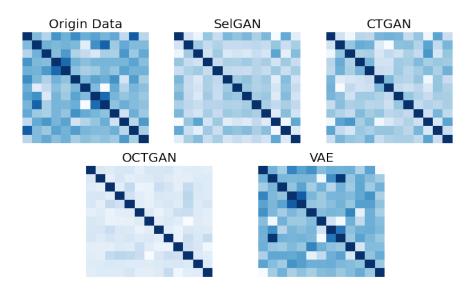


FIGURE B.14: Correlation Heap Map for Covertype

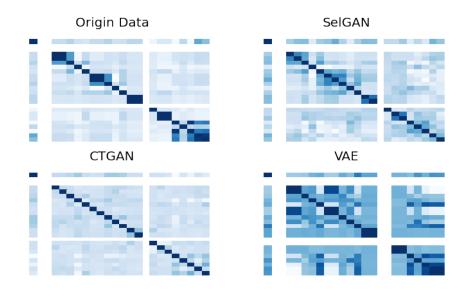


Figure B.15: Correlation Heap Map for Ticket

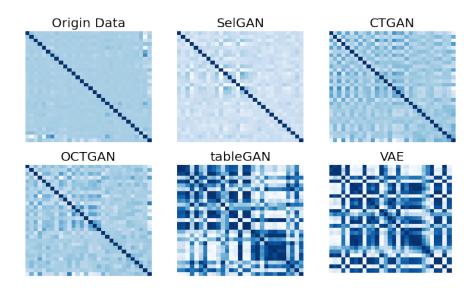


FIGURE B.16: Correlation Heap Map for Credit

Bibliography

- [1] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):1–41, 2017.
- [2] Jérémie Sublime and Ekaterina Kalinicheva. Automatic post-disaster damage mapping using deep-learning techniques for change detection: Case study of the tohoku tsunami. Remote Sensing, 11(9), 2019. ISSN 2072-4292. doi: 10.3390/rs11091123. URL https://www.mdpi.com/2072-4292/11/9/1123.
- [3] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015. URL https://cpang4.github.io/gan/.
- [4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. Advances in neural information processing systems, 27, 2014.
- [5] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. arXiv preprint arXiv:1811.11264, 2018.
- [6] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. Advances in Neural Information Processing Systems, 32, 2019.
- [7] Jayoung Kim, Jinsung Jeon, Jaehoon Lee, Jihyeon Hyeong, and Noseong Park. OCT-GAN: neural ode-based conditional tabular gans. CoRR, abs/2105.14969, 2021. URL https://arxiv.org/abs/2105.14969.
- [8] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. CoRR, abs/1806.03384, 2018. URL http://arxiv.org/abs/1806.03384.
- [9] Aoting Hu, Renjie Xie, Zhigang Lu, Aiqun Hu, and Minhui Xue. Tablegan-mca: Evaluating membership collisions of gan-synthesized tabular data releasing. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications

Bibliography 53

Security, CCS '21, page 2096–2112, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384544. doi: 10.1145/3460120.3485251. URL https://doi.org/10.1145/3460120.3485251.

- [10] Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and V. S. Subrahmanian. Faketables: Using gans to generate functional dependency preserving tables with bounded real data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2074–2080. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/287. URL https://doi.org/10.24963/ijcai.2019/287.
- [11] Daphne Koller and Nir Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009.
- [12] Dana H Ballard. Modular learning in neural networks. In Aaai, volume 647, pages 279–284, 1987.
- [13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.
- [14] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/arjovsky17a.html.
- [15] Karim Armanious, Chenming Jiang, Marc Fischer, Thomas Küstner, Tobias Hepp, Konstantin Nikolaou, Sergios Gatidis, and Bin Yang. Medgan: Medical image translation using gans. Computerized medical imaging and graphics, 79:101684, 2020.
- [16] Adriel Cheng. Pac-gan: Packet generation of network traffic using generative adversarial networks. In 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pages 0728–0734. IEEE, 2019.
- [17] Wentao Wu, Jeffrey F Naughton, and Harneet Singh. Sampling-based query reoptimization. In Proceedings of the 2016 International Conference on Management of Data, pages 1721–1736, 2016.
- [18] Yannis E. Ioannidis and Viswanath Poosala. Histogram-based solutions to diverse database estimation problems. *IEEE Data Eng. Bull.*, 18(3):10–18, 1995.
- [19] Zhenjie Zhang, Yin Yang, Ruichu Cai, Dimitris Papadias, and Anthony Tung. Kernel-based skyline cardinality estimation. In Proceedings of the 2009 ACM SIG-MOD International Conference on Management of data, pages 509–522, 2009.

Bibliography 54

[20] Yaoshu Wang, Chuan Xiao, Jianbin Qin, Rui Mao, Makoto Onizuka, Wei Wang, Rui Zhang, and Yoshiharu Ishikawa. Consistent and flexible selectivity estimation for high-dimensional data. In *Proceedings of the 2021 International Conference* on Management of Data. ACM, jun 2021. doi: 10.1145/3448016.3452772. URL https://doi.org/10.1145%2F3448016.3452772.

- [21] Ju Fan, Junyou Chen, Tongyu Liu, Yuwei Shen, Guoliang Li, and Xiaoyong Du. Relational data synthesis using generative adversarial networks: A design space exploration. *Proc. VLDB Endow.*, 13(12):1962–1975, jul 2020. ISSN 2150-8097. doi: 10.14778/3407790.3407802. URL https://doi.org/10.14778/3407790.3407802.
- [22] Spartan documentation. https://dashboard.hpc.unimelb.edu.au/. Accessed: 2022-05-20.