University College London

Department of Computer Science

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Computational Finance/Financial Risk Management, University College London

Application of Generative Adversarial Networks on lending dataset

Candidate Number

PJYR4

Academic Supervisor

Prof. Tomaso Aste

Department of Computer Science

University College London

September 12, 2021

Abstract

Class imbalance is a common problem in data that impedes the predictive performance of classification algorithms. In the task of loan default prediction, the impact of the imbalance on classifiers can often be economically costly. Oversampling methods are commonly used to deal with unbalanced datasets, and a large number of linear interpolation and KNN-based oversampling methods such as SMOTE, ADASYN and their variants are constantly proposed by scholars. However, they have an inherent disadvantage in dealing with high-dimensional and complex datasets. Deep learning networks can model complex data well, and models based on generative adversarial networks(GANs) have made relatively significant progress in generating tabular data (e.g., database tables). Research at this stage has generally focused on the use of GAN for generating tabular data as novel oversampling tools, however this often requires complex structures and extensive hyper-parameter tuning. As an exploration of the application of GAN to unbalanced datasets, this paper proposes a framework that combines GANs with traditional oversampling methods. We compare our framework with five resampling methods and the results demonstrate that it can lead to better stability of the classifier in the presence of insufficient data.

Acknowledgments

I would like to express my sincere gratitude to anyone who helped me in this project.

Thanks to my academic supervisor, Aste Tomaso, for his professional guidance and valuable pieces of advice from every meeting all through the project. Without his support, this would not have been possible.

Thanks to my parents for their long-lasting spiritual encouragement and financial support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Accurate prediction of default events has been an important theme for lenders since the advent of lending in history [1]. Banks and other lenders have tried various methods to avoid default events through the mechanism of multiple approvals and guarantors. However, as the demand for lending has grown, the costs and errors associated with having humans involved in the process have severely limited the development of lending [2]. In line with the trend of big data, the increased computing power and the rapid development of machine learning have led to an interest in further reducing the risk of default. At the same time, the automation of the credit review process has helped to avoid discrimination and corruption, making access to lending more equitable for everyone [3].

The difficulty for applying machine learning algorithms to the task of loan default detection, in addition to access to user information, is more tricky in dealing with data imbalance. Although defaults occur in all types of lending, they are still a small probability event. This lead to a significant imbalance between defaulted and non-defaulted samples in the historical data. For the classification algorithms, the imbalance can lead to a tendency to classify the samples into major classes on the one hand, i.e. to classify defaulting users as non-defaulting users. On the other hand the defaulted samples are not sufficient to provide enough classification information.

This leads to poor predictive performance of the classification algorithm in general [4].

The most common method of dealing with unbalanced datasets is oversampling, where a balanced dataset is synthesised artificially by generating more samples from the minority class [5]. Traditional oversampling methods, such as SMOTE and ADASYN, rely heavily on spatial interpolation and KNN algorithms, however it has been shown that not work well on high-dimensional datasets with large imbalance rates [6].

Generative Adversarial Networks (GANs) [7] are a powerful technique for deep generation. Given a training set, this technique learns to generate new data with the same statistics as the training set. GANs are trained to learn image data by transforming the pictures into blocks of pixels (numerical data), whereas if one considers tabular data as a two-dimensional matrix, there is no essential difference between these two kinds of data. With the advances made by GAN in generating image data, more and more research is experimenting with GANs to generate tabular data, with the expectation that networks of this structure will be applied to the processing of unbalanced datasets. The main challenges in using GANs to generate tabular data are as follows:

1. Real datasets often contain different data types, i.e. discrete and continuous variables, the GAN model must apply different activation functions to the output.

2. Unlike image data, values in tabular data tend to only approximately obey a complex non-Gaussian distribution, which makes min-max transformation lead to the problem of gradient vanishing [8].

3. The high imbalance in the categorical variables makes the training process of GAN more prone to mode collapse. Missing minor categories tend to cause only a small loss to the overall distribution of the data and are therefore difficult to be detected by the discriminator.

Some GAN-based models used to generate tabular data, such as MedGAN and CTGAN, largely address these problems by increasing the number of discriminators and pre-processing the data, while the PacGAN architecture avoids the mode collapse that occurs when GANs are trained with tabular data [9, 10]. Several studies have designed MedGAN and CTGAN as novel oversampling tools based on their remarkable progress in generating tabular data, but such approaches currently have significant disadvantages compared to traditional oversampling models. GANs are inherently difficult to train, and most GAN-based oversampling models are even more complex in structure and require extensive hyperparameter tuning. And even so, most of the GAN-based oversampling models are not significantly better than traditional oversampling methods.

1.2 Objectives and achievements

The objective of this dissertation is to make a new exploration of the application of GANs to the processing of imbalanced data, outperform traditional oversampling models without requiring too much hyperparameter tuning. We first investigated the validity of the GAN-based model as an oversampling method, and then designed a new framework, CTGANS, combining GAN with a traditional oversampling model. In particular, we focus on the shortcomings of traditional oversampling models, namely the risk of processing only minority classes and interpolating high-dimensional data [11], while retaining the advantages of GAN in learning the joint distribution of the original data. Two classification algorithms and three metrics are used to validate the effectiveness and stability of our framework on unbalanced datasets with different sample sizes.

Our proposed framework outperforms traditional resampling methods such as SMOTE on peer-to-peer (P2P) lending dataset [12]. Compared to using GAN as an oversampling tool, CTGANS has more obvious advantages and does not require much hyperparameter tuning work, which greatly reduces the time cost associated with training. The application of CTGANS to the P2P dataset is just one example, and in fact it can be widely applied to other unbalanced datasets due to its full consideration of numerical and categorical variables. Its modularity could allow its performance to be enhanced with the proposal of more efficient GANs for generating tabular data. However the specific effects on other datasets still needs to be verified.

1.3 Structure

In Chapter 2 we review previous literature on GANs for generating tabular data and on the application of GANs to unbalanced datasets. In Chapter 3 we present the motivation and rationale for designing CTGANS, and the metrics we used to evaluation the gap between the synthetic and original data. In Chapter 4 we provide an analysis of the generative performance of CTGAN, as well as a comparison of the performance of CTGANS and other resampling methods on the two classification algorithms, explaining the phenomena of interest. The last chapter makes conclusions and provides the further research directions.

Chapter 2

Literature Review

2.1 GAN-based tabular data generation

Although methods of abstracting multiple columns of tabular data into a joint probability distribution have been practised for many years, for example with Bayesian networks [13] and spatial decomposition trees [14] for modelling discrete and continuous variables respectively. However, the interpretability of the synthetic data generated by these models is limited by the type of distribution and computational problems, making it difficult to be applied in practice [8]. It has become increasingly popular to design the structure of a GAN so that it learns the joint distribution of the data [15].

Based on the achievements of GAN in the generation of image data, in the study [16], its authors try to generate synthetic data by following a traditional GAN. They designed the table-GAN, combining multiple rows of one-dimensional data into a two-dimensional matrix, processing by 2D convolution similar to image data. Although many of the GAN-based models generate images that are already very fine, applying this approach to tabular data is still considered inappropriate. Firstly, there is a high correlation between adjacent pixels in image data, but this property is rare in tabular data [17]. In image data, the pixel values obey a Gaussian-like distribution, and so can be passed in Gaussian noise to the generator so that the generator can learn the generative principles inherent in the image more quickly

[18]. However, tabular data tends to obey a more complex distribution, which suggests that a dedicated architecture for tabular data is necessary, and that it is reasonable to design a more complex structure for the GAN-based model for tabular data than for the GAN-based model applied to images. Secondly, for the category variables, the method of label encoding used by the table-GAN model also proved to be theoretically flawed [10].

Current research into making GANs suitable for generating tabular data has focused on two areas; one is to increase the number of discriminators, i.e. to capture certain features in the data more specifically by introducing more discriminators [19]. The other approach is to pre-process the incoming data so that its features are more easily learned by the generator [8].

MedGAN [19] was originally designed to perform image to image translation in the field of medical image analysis. However, recent research has shown that MedGAN also has a significant advantage over traditional GANs in generating tabular data by using Wasserstein GAN gradient penalties [20]. MedGAN consists of three neural networks, compared to the traditional GAN structure, MedGAN incorporates a Feature Extractor. It is pre-trained to extract deeper features from the data to calculate the loss of the generator. This allows the synthetic data to be consistent with the real data at a deeper level, avoiding mode collapse. Although the three neural networks impose more time cost on MedGAN, many studies have shown that MedGAN does have better stability compared to other GAN-based models [21].

The CTGAN [8] allows the GAN model to better learn the features of the data by pre-processing the numerical and categorical variables in the data separately. Unlike the MedGAN model, CTGAN does not capture features through an additional pre-trained auto-encoder, but instead separates features in continuous and discrete variables through a variational Gaussian mixture model and one-hot encoding, respectively. This has the advantage of preserving the unsupervised features of GAN while avoiding the time cost of introducing more neural networks.

2.2 GAN-based oversampling method

With the development of GAN-based tabular data generator, more and more researchers are applying GANs as novel oversampling method, making the comparison between GAN-based and traditional oversampling methods a focus of attention. In [22], the authors compare the performance of the GAN-based oversampling method in predicting credit card defaults with the traditional oversampling method SMOTE. As with the SMOTE method, the GAN was trained through minority class to perform the oversampling, and the neural network was subsequently trained using both sets of oversampled data and tested on the same test set. Although the results in [22] show that the GAN-based mothod is slightly better than SMOTE in terms of sensitivity and specificity, SMOTE has a clear advantage in terms of accuracy and precision. Also, their analysis of the results lacked a comparison of recall, which is a very important metric for credit card default prediction type tasks. A more important issue is that the suitability of data generated by a neural network structure generator for training a feed-forward neural network remains to be demonstrated. We will discuss this issue further in Chapter 4.3.

The paper [23] also explores the differences in performance between GAN-based oversampling methods and traditional oversampling methods. Unlike [22], the authors not only tested on 12 real-world datasets and 10 synthetic datasets, but also compared more traditional oversampling methods, such as variants of SMOTE. The results show that GAN-based oversampling methods completely outperform other oversampling methods in terms of the F-measure, G-mean and AUC metric. Although such results are encouraging, the paper lacks some details in the design of the model and has some limitations in the analysis of the results. Firstly, the 22 datasets used in the paper contain only numerical variables, yet categorical variables are very common in tabular data. Secondly, the analysis of the results does not distinguish between the real dataset and the synthetic dataset, but rather their average. A noteworthy point is that synthetic datasets create different Gaussian clusters on the vertices of a hypercube [24], whereas typically the generator is a seemingly random synthetic sample generated based on Gaussian noise. In this case, it may be easier for the generator to learn the principles of synthetic data generation [18]. This may allow subsequent classification models to obtain scores on these datasets that far exceed those of real-world data, resulting in a huge increase in overall scores. This interpretation could equally explain the third point, namely that based on the number of layers of the generator published in the paper it can be found that no hyperparameter tuning is required for artificially generated datasets, whereas for different real-world datasets a large amount of hyperparameter tuning is often required, with the number of units and layers varying for each different dataset.

In [25], the authors compared four different structures of GAN-based oversampling methods with traditional oversampling methods. The results show that the advantages of the traditional oversampling methods are still evident. The three oversampling methods ROS, ADASYN and SMOTE performed best in terms of AUC, Recall and Accuracy, respectively. In contrast, only the WCGAN outperformed the traditional oversampling model on the F1-Score. Although this is only the performance on one dataset, a summary of the literature shows that most GAN-based oversampling models perform close to, or even slightly worse than, traditional oversampling methods [22, 25]. Specifically, for multiple datasets, GAN models generally perform less consistently than traditional oversampling methods [26]. For a single dataset, the structure of the GAN can be designed to outperform traditional oversampling methods [27, 28]. However, given the difficulty and time cost of training GAN models, the rationality of applying GAN as an oversampling method has been questioned. Meanwhile, more applications of GAN to the task of processing unbalanced datasets still need to be explored.

Chapter 3

Methodology

3.1 Traditional Resampling Method

Unbalanced datasets usually cause bias to the classification model, manifested by the model returning the majority class for all input samples [29]. There are two main methods of dealing with unbalanced datasets, oversampling and undersampling. Undersampling methods balance the data set by reducing the sample from the majority classes. However, undersampling is rarely used in practice because it actually reduces the information of the majority classes in the dataset [30]. For example, if we have a dataset of 10,000 data, which contains 100 data of class 1 and the rest of the data of class 0. In this case, if we undersample, we would have to remove at least 9,800 data from the data to achieve balance. This process would result in a large amount of information being lost for the majority class, making the model more error prone.

However there are still some undersampling models that are considered worth trying, such as the NearMiss [31]. The principle of this undersampling method is to increase the gap, while decrease the similarity between the minority and majority classes. This is done by removing samples that are close to each other and samples that are close to the minority class from the majority class. In Figure 3.1 we show an example of undersampling by applying the Near Miss method. In fact, it is common for there to be partial duplicates in the data, and increasing the distance between the minority and majority classes of data can also improve the stability of the classification model. However, it is not clear how much data from the majority class is appropriate to remove. This results in that if the data is very poorly balanced, then to achieve balance, undersampling has to remove too much data from the majority class, making the performance of the classification algorithm unstable [32].

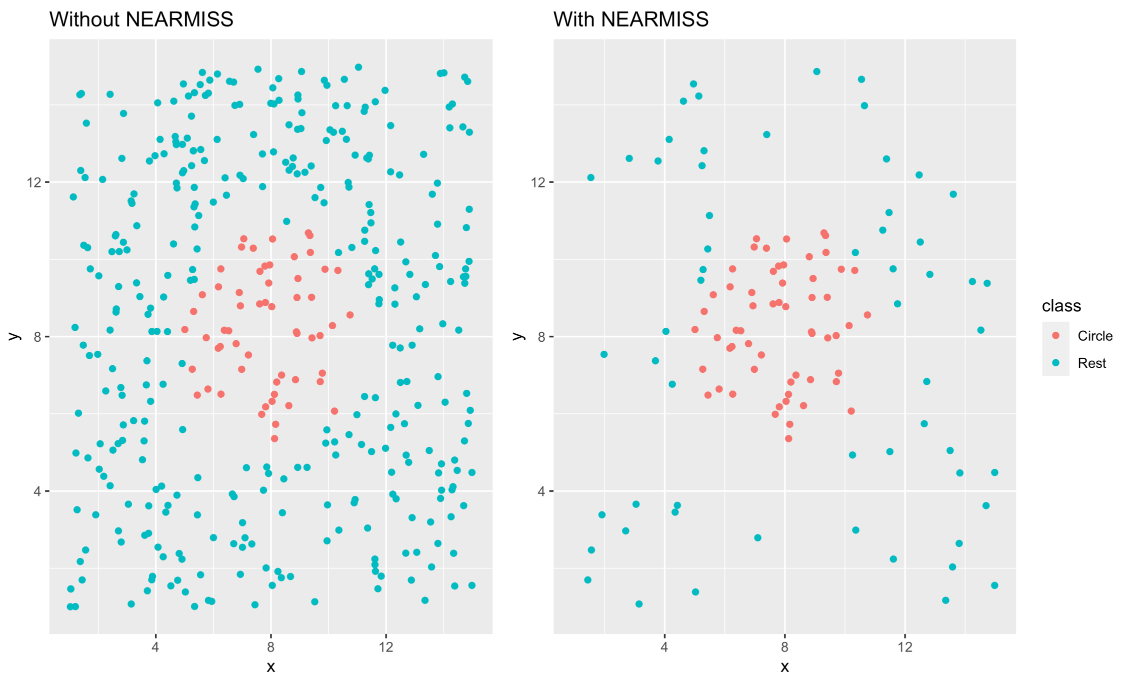

Figure 3.1: Example of NearMiss method for processing unbalanced dataset

The synthetic minority over-sampling technique (SMOTE) [6] method has been one of the most commonly used oversampling methods when dealing with unbalanced datasets, due to its ease of implementation and stability of results. SMOTE uses linear relationships between samples from minority class to generate synthetic samples. The process is as follows: first take a sample from the minority class and consider its k nearest neighbours (in the feature space). Then take the vector between one of these k neighbours and the current sample and multiply this vector by a real number between (0, 1). Finally add this vector to the current dataset to create a new synthetic sample. This is repeated until the dataset is balanced. As shown in Figure 3.2.

The principle and process of the Adaptive synthetic sampling approach for imbalanced learning (ADASYN) and SMOTE methods are essentially the same, the only difference being that a random noise is added to these points after the synthetic samples have been created. This improves the realism of the synthetic data, i.e. not all synthetic samples are linearly correlated with the original data, but there are certain variances [33].

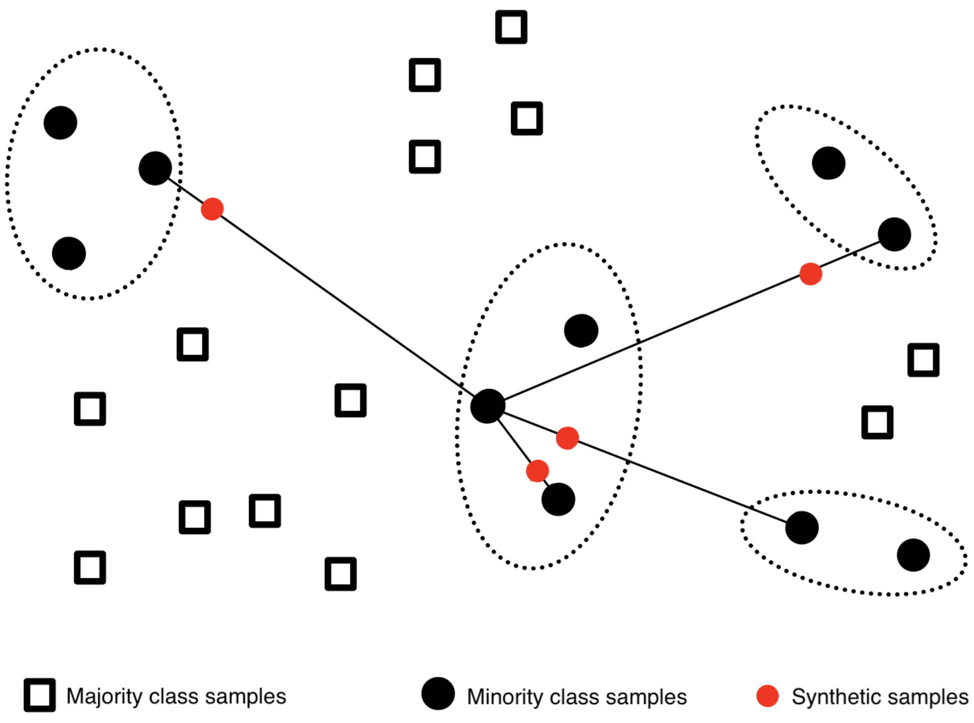

Figure 3.2: Schematic diagram of the SMOTE method for generating minority class data

3.2 Data Processing

The dataset we use in this paper comes from LendingClub (www.lendingclub.com), where the data was collected between 2007 and 2018. The dataset consists of over two million pieces of data with 29 variables, including 7 categorical variables and 23 numerical variables, each of which has the meaning are shown in Table A.1 in the Appendix.

3.2.1 Categorical variable

Although researchers have been using GANs to generate tabular data for a long time, the use of GAN-based models to generate tabular data containing categorical variables was deliberately avoided in the early days, because the training of neural networks required that both generators and discriminators be fully differentiable [7]. However, this limitation of GANs is gradually being overcome with the introduction of a number of methods.

The most intuitive approach is to perform a one-hot encoding of each column of the categorical variables and then apply the activation function softmax to the output of the generator. One-hot encoding avoids the numerical relationships arise from encoding categorical variables to integers by separating each of the classes in the categorical variables [16]. The training process is then guaranteed to be differentiable by means of an activation function. However, such an approach has the obvious disadvantage of making the data generated by the softmax function easily recognisable by the discriminator. For example [0.2, 0.7, 0.1] corresponds to the real data [0, 1, 0]. The generator correctly generates the synthetic data we need, but the discriminator can easily distinguish the real data from the synthetic data by the integer nature of the real data, even though it does not learn any deeper connections between the data. Although this problem can be avoided to some extent by adding noise to the real data, in many cases it is necessary to add very strong noise to be effective [11].

Gumbel-softmax [34] is an improvement on the above method that polarises the probability of a high probability event occurring by adding noise that obeys the Gumbel distribution to the logarithm of the softmax probabilities, while ensuring that the process is differentiable. The formula for Gumbel-softmax is as follows :

$$\text{Gumbel-softmax } (x_i) = \frac{\exp\left((x_i + g_i)/\tau\right)}{\sum_{j=1}^{k} \exp\left((x_j + g_j)/\tau\right)} \quad \text{for} \quad i = 1, \ldots, k \tag{3.1}$$

where x_i corresponds to the log probability of the i-th categorical variable, g_i follows the $Gumbel(0,1)$ and τ is the "temperature" parameter. When $0 < \tau < 1$, the expectation of the Gumbel-softmax random variable is close to the expectation of a categorical variable with the same logits. When $\tau > 1$, the expectation of the Gumbel-softmax random variable is close to the Uniform distribution.

3.2.2 Numerical variable

Although the generation of numerical variables is relatively straightforward, there are still some issues to consider. Numericial variables contain two types of data, discrete variables and continuous variables. Discrete variables include the duration of the loan, the number of properties, etc. Continuous variables contain the number of loans, fico score, etc. Similar to the treatment of categorical variables, discrete variables can be processed by one-hot encoding. For continuous variables, to avoid mode collapse during GAN training, we use mode-specific normalizaiton [8]. Unlike the approach of the MedGAN model, we do not wish to identify deep-level features in the data by pre-training a neural network. Although deeper neural network results are better at capturing features deeper in the data, they also introduce an element of uncontrollability that makes it difficult to guarantee that the generator will learn enough features under unsupervised conditions.

Mode-specific normalization [8] is a method for identifying the number of modes in a continuous variable and separating these modes. The number of modes in a continuous variable is estimated by using the variational Gaussian mixture model (VGM) [35], then the probability of each mode corresponding to each value in the continuous variable is found, and finally the corresponding probability is derived by normalisation. Specifically, for the continuous variable C_i, the VGM finds three modes: α_1, α_2 and α_3 for example, then the learned Gaussian mixture is $P_{C_i}(c_{i,j}) = \sum_{k=1}^{3} \mu_k N(c_{i,j}; \alpha_k, \phi_k)$ for each $c_{i,j}$ in C_i, where μ_k are the weight and ϕ_k are the standard deviation of the modes. For each mode the probability density $\rho_k = \mu_k N(c_{i,j}; \alpha_k, \phi_k)$. Finally, we can represent $c_{i,j}$ as a one-hot vector [0,1,0] and the corresponding value for each modes are represented as $\beta_{i,j} = \frac{c_{i,j} - \alpha_2}{4\phi_2}$. The mode-specific normalization approach adds interpretation and stability to the model compared to the feature extractor in the MedGAN model.

3.3 GAN-based model

3.3.1 Traditional GAN

Generative adversarial networks (GANs) [36] are a modelling strategy that learns through two neural networks, a discriminator (D) and a generator (G), playing against each other. In this case, they are integrated into a network for training, the generator is responsible for generating synthetic data close to the real data and the discriminator tries to separate real data and synthetic data. The two models improve their performance in a confrontation until the synthetic data generated by the generator is indistinguishable from the real data. GAN is now gaining popularity as a method for modelling complex data distributions, with impressive results, especially in the field of images generation [37].

The generator defined as $G : Z \to X$ where Z is the noise space and X is the synthetic data space. And the discriminator defined as $D : X^* \to [0, 1]$, where X^* is a mixture of real and synthetic data. The synthetic data X from generator aim to capture the distribution of the real data, and the output of discriminator represents the probability that the incoming data is synthetic data.

From a mathematical perspective, the two neural network G and D play a two-player minimax game with the following value function:

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{x \sim p_{data}}[\log D(x)] + \mathop{\mathbb{E}}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.2}$$

where z and x are samples from the noise space and real data distribution respectively.

3.3.2 Conditional GAN

According to Equation 3.2, it can be found that the only purpose of the GAN model during training is to make the synthetic data as close as possible to the real data. This results in the GAN being too free in the direction of learning [38]. In fact in many cases, especially for unbalanced datasets, we not only need the GAN to learn the distribution of the real data, but also the number of occurrences of the unbalanced classes in the synthetic data to be the same as in the real data. Therefore in order to constrain the GAN, the method of conditional GAN is proposed as an extension of the traditional GAN.

In a conditional GAN (cGAN) [39], the decisions of the generator and discriminator are not only based on the noise space Z, but also include an additional information set I. I can represent, for example, a class label or a certain categorical feature [40]. This process is equivalent to turning an unsupervised GAN model into a semi-supervised model. The results of many studies confirm that this approach is notably effective in generating unbalanced datasets compared to traditional GANs.

Similar to the training structure of traditional GAN, cGAN defined by using the information set I as a conditional variable. Now the the generator and discriminator are defined as $G : Z \times I \to X$ and $D : X^* \times I \to [0, 1]$ respectively. And the corresponding value function $V(D, G)$ is given below:

$$\begin{aligned} \min_G \max_D V(D,G) =& E_{x,i \sim p_{\text{data}}\ (x,i)}[\log D(x,i)] + \\ & E_{z \sim p_z(z), i \sim p(i)}[\log(1 - D(g(z,i),i))] \end{aligned} \tag{3.3}$$

where z, x and i are samples from the noise space, real data distribution and information set respectively.

Unlike traditional probability distribution models, GANs are known as invisible probability distributions. This means that the training process does not result in a density function that can actually be computed, but rather a simulation procedure that is used to generate new data points. This results in GANs being very difficult to train and mode collapse occurs from time to time. This is manifested by the generator learning to generate samples based on some of the features in the real data and missing many other patterns, making its performance not reflect any advantage over traditional probabilistic models [41]. And in many cases, finding mode collapse is often more difficult than dealing with it. In order to address this problem, many scholars have modified the structure of GAN in terms of loss functions, activation functions and optimisation algorithms [42]. However, this has not completely solved the mode collapse problem of GAN models, and the training of GANs still requires a lot of tuning work.

3.3.3 CTGAN

Conditional tabular generative adversarial network (CTGAN) [8] is the GAN-based model proposed by Xu in 2018, to generate tabular data. We have presented the data processing part in chapter 3.2.1 and 3.2.2. Its complete structure is shown in the Figure 3.3 below:

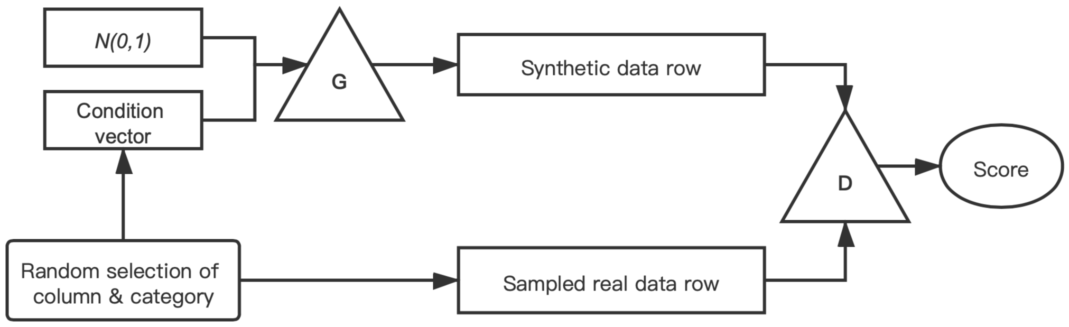

Figure 3.3: The architecture of CTGAN, G and D represent the generator and discriminator respectively

In addition to the treatment of numerical and categorical variables, CTGAN uses recent advances in GAN training, including the use of the Wasserstein-GAN structure and the PacGAN framework [43].

Wasserstein-GAN (WGAN) achieves the objective of making the distribution of the synthetic data close to the true distribution by using the Wasserstein Distance [44] as the loss function. The advantages of the Wasserstein Distance over traditional measures of distribution differences, such as Kullback-Leibler (KL) Divergence and Jensen-Shannon (JS) Divergence, are as follows [45]:

- Gives natural distances between discrete and continuous distributions.
- Gives not only a measure of distance but also a scheme for converting an existing distribution into a target distribution.
- The geometric characteristics of the distribution itself can be maintained when converting a distribution to a target distribution.

The expression for using the Wasserstein distance as a loss function to measure the distance $L(p_a, p_b)$ between the distribution p_a and p_b is as follow:

$$L\left(p_a, p_b\right) = W\left(p_a, p_b\right) = \max_{w \in W} \mathbb{E}_{x \sim p_a}\left[f_w(x)\right] - \mathbb{E}_{z \sim p_a(z)}\left[f_w\left(g_\theta(z)\right)\right] \tag{3.4}$$

where f_w is a K-Lipschitz continuous function parameterized by w.

In contrast to the traditional task, in training WGAN, the task of the discriminator is no longer to distinguish between the real data and the synthetic data, but to find the parameters of the f_w function so that the distribution of the synthetic data becomes closer to the distribution of the real data.

The PacGAN [43] framework is an approach designed for the mode collapse problem. By modifying the structure of the discriminator so that it makes decisions based on packaged multiple samples of the same class. This amplifies the distance between the distribution of data generated by a generator with mode collapse and the real distribution, making it more penalised.

3.4 Application of GAN-based models on unbalanced datasets

3.4.1 Background to GAN-based oversampling methods

The motivation for applying GANs to the task of processing unbalanced datasets stems from their promising progress in generating tabular data, yet the suitability of GANs as an oversampling method remains controversial. While related research has confirmed that some GAN constructs such as cWGAN [10] and GOGAN [26] do outperform traditional oversampling methods, the improvement has not been significant. Also these models are often designed only for processing specific types of data. In fact, if a model is designed according to a researcher inspired by a specific theory, then its positive results are meaningful. Conversely, if a researcher designs a large number of models with different structures for a particular dataset, and makes a large number of parameter tunings, then the reliability of the final observed performance is significantly reduced [46].

3.4.2 Disadvantages of Traditional & GAN-based oversampling methods

The disadvantages of the oversampling method come first from the data itself. When the imbalance rate of the data is large, the need for minority class sample generation increases. At this point, the disadvantages of the oversampling method are magnified, especially when the minority class samples do not provide sufficient information. Specifically, the sparse nature of the high-dimensional space makes it possible that samples generated by SMOTE based on KNN and spatial interpolation, for example, although appearing to be very close to the real data in all dimensions, do not actually have a positive impact on the classification algorithm when applied, and even mislead it [47].

The main drawback of traditional oversampling methods is that the learning for unbalanced datasets rests on samples from the minority class. For improving the performance of classification algorithms on unbalanced datasets, especially for the task of binary classification, the treatment of majority class is also necessary. This is because the outliers that confuse the classification algorithm are also likely to be present in the majority class [48]. However, existing oversampling methods do not support such an operation, as generating majority class sample would further increase the imbalance rate of the dataset. This requires the oversampling method to generate more samples, which further magnifies its drawbacks.

The disadvantages of GAN-based oversampling methods come from the fact that training neural networks require a relatively large dataset, otherwise the structure of the neural network is not guaranteed to be stable enough, leading to a large bias in the data it generates [49]. In many cases, the minority class is not sufficient to provide enough data for neural network training, not to mention that it does not even provide enough classification information. Thus GAN-based is less stable than traditional oversampling methods.

3.4.3 Our Method: CTGANS

To overcome the disadvantages of traditional oversampling methods, while applying the advances made by GAN in generating tabular data to the treatment of unbalanced datasets, we propose to combine GAN with traditional oversampling methods to improve the performance of the classification algorithm without excessive parameter tuning. This is achieved by first generating a certain amount of data from a trained CTGAN model, which has the same imbalance rate as the original data. Traditional oversampling methods are then applied to a mixture of the original data and the synthetic data generated by CTGAN. Because of the proven excellent performance of SMOTE, in this paper, we use a combination of CTGAN and SMOTE, which we refer to as CTGANS.

The role of CTGAN in this framework is to weaken the effect that outliers in the data have on the classifier.The Wasserstein Distance in the CTGAN model ensures that it does not learn outliers in the data during the learning process [50]. The SMOTE method in turn weakens the impact of anomalous samples generated by the CTGAN model due to insufficient data.

One noteworthy issue is how much synthetic data is appropriate to generate via CTGAN. While this will inevitably need to be determined through multiple attempts, we believe that the number generated should not exceed half the number of samples from the original data, otherwise the amount of the minority class sample that need to be generated by SMOTE will be too large.

3.4.4 Generative performance metrics

Given that there is no universally accepted method of measuring similarity between datasets, in this paper we focus on three aspects of the similarity between the synthetic data generated by the CTGAN model and the real data.

First, the distribution of each variable between the synthetic and real data should be an intuitive measure. For numerical variables, we used kernel density estimates with Gaussian kernels and a bandwidth of 0.02, as in Figure 4.1. For categorical variables, we used the frequencies of all classes in each categorical variable as vertical coordinates, as in Figure 4.2. To make the results more straightforward, we calculated the mean squared error (MSE) [51] and the maximum distance (d_∞) between the probability density functions (PDFs) of the real and synthetic data, respectively. Assume that y and y^* is a vector of PDF values of the synthetic data and original data, respectively, the expression of MSE and d_∞ as follow:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i^* - y_i)^2 \tag{3.5}$$

$$d_\infty = max\{|y_i - y_i^*||\text{i} = 0, \text{...}, \text{n}\} \tag{3.6}$$

where n is the number of y.

Secondly, we use the standard deviation to measure the degree of variation and dispersion in the synthetic data, as shown in Table 4.2. As described in Section 3.4, we expect that the GAN model will not learn outliers in the data during training, which will manifest itself in the synthetic data having a lower standard deviation than the real data. However there are still many cases where the standard deviation of the data generated by the GAN model will increase, for example when new outliers are generated due to features in the data being incorrectly learned.

Thirdly, we analysed the correlation coefficients between the variables in the synthetic data and original data. By ranking the absolute values of the subtracted variables correlation matrices of the synthetic and original data, we can obtain the difference in the correlation between the variables of the synthetic and real data, as shown in Figure 4.3. In general, if the correlation between the two sets of data is less than 0.1, it can be assumed that no correlation exists [52]. Therefore for features with a correlation difference of less than 0.1, we consider CTGAN to have generated the feature reasonably well.

3.5 Evaluation Framework

In order to fully compare the performance of CTGANS in processing unbalanced datasets, we used datasets processed by multiple resampling methods as the training set for the classification algorithms, and fixed a 2000-row dataset from the entire data as the test set. In this paper, we used two classification algorithms, Ridge

Logistic Regression (RLR) and Multi-layer Perceptron (MLP), and three metrics, Accurary, Recall rate and F1-score [53], which will be described in chapter 3.5.1 and 3.5.2. In the meantime, we have tested the results on original datasets with different sample sizes (100, 500, 1000, 5000, 10000, 50000). Note that the imbalance rate was set to 10% (close to the original data) for the original datasets, and the number of samples in the test set was kept constant across training sets, i.e. 2000 rows of data.

Regarding the resampling methods, in addition to the three resampling methods SMOTE, ADASYN and NearMiss introduced in chapter 3.1, we also applied CTGAN as an oversampling method to compare with CTGANS. Also, we set two baselines for the various resampling methods. One is the Random Oversampling (ROS) method, which balances the dataset by randomly duplicating samples from minority classes. We use this method as a minimum standard for oversampling methods; an oversampling method would not make any sense if it did not perform as well as ROS. As an upper bound for the classification algorithm, we obtained balanced datasets with different sample sizes from the entire dataset by random sampling, i.e. the same number of default and non-default samples. Because it contains the classification information for the minority class that is missing from the training set, all oversampling methods should not perform better than the balanced dataset. However, in this paper we use a credit card default dataset, i.e. a binary classification task. In this case, since the original dataset contains more samples of the majority classes than the balanced dataset, the performance of the balanced dataset is not guaranteed to be a strict upper bound on the performance of the oversampling method.

3.5.1 Classification algorithm

For a set of sample $x^{(i)}, y^{(i)}$, $i = 1, ...N$, $x^{(i)}$ represent the variables of the $i - th$ sample and $y^{(i)}$ represent its label. The original linear regression [54] model finds the weights $\hat{\beta}$ by minimising the Residual Sum of Squares (RSS):

$$RSS = \sum_{i=1}^{N} \left(\sum_{j=0}^{p} \hat{\beta}_j x_j^{(i)} - y^{(i)} \right)^2 \tag{3.7}$$

where p is the number of variables. For data with a large number of variables, the overly complex structure of linear regression can easily lead to overfitting. Regularisation is a way of adjusting the complexity of the model to help find the model that provides the best out-of-sample performance. In this paper, we reduce the size of the model parameters by adding a penalty term (l2 penalty). At this point, the objective function ($J(\beta)$) to be minimised becomes:

$$J(\beta) = \sum_{i=1}^{N} \left(\sum_{j=1}^{p} \hat{\beta}_j x_j^{(i)} - y^{(i)} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{3.8}$$

where lambda controls the extent to which the penalty term affects the linear regression weights, which decrease (numerically) as lambda increases. The output of linear regression represents the probability of a sample belonging to a label, which can be transformed into a logistic regression model by setting the threshold, i.e. Ridge Logistic Regression (RLR).

Algorithm	Hyperparameter	Value
Ridge Logistic Regression	Penalty parameter	0.1
Multi-layer Perceptron	Learning rate	0.001
	Activation	ReLU
	Hidden layer number	1
	Hidden layer size	30
	Penalty parameter	0.001
	Solver	Adam
	Epoches	300

Table 3.1: Hyperparameter settings for the employed classification algorithm

We also use a Multi-layer Perceptron (MLP) [55] as a classifier, which has a more complex structure than logistic regression. The MLP contains at least three layers, namely an input layer, a hidden layer and an output layer. The use of a non-linear activation function allows it to distinguish non-linearly separable data [56]. Similar to logistic regression, we also have a regularisation term added to the loss function to prevent overfitting. The hyperparameter settings for both classification algorithms are shown in Table 3.1.

3.5.2 Evaluation Metrics

We evaluate the performance of trained classifiers for loan defaults prediction with three metrics: Accuracy, Recall and F-measure.

- Accuracy: proportion of examples correctly quantified. Accuracy is a metric of the performance of the model to correctly predict the overall data:

$$Accuracy = \frac{TP + TN}{N}$$

- Recall: proportion of positives that are correct identified. Recall is a metric of the performance of the model to correctly predict default events:

$$Recall = \frac{TP}{FN + TP}$$

- F-measure [53]: calculated from the precision and recall of the test, where precision is the number of true positive results dealing with the number of all positive results. The general form is as follow:

$$F_\beta = (1 + \beta^2) \frac{\text{precision} * \text{recall}}{(\beta^2 * \text{precision}) + \text{recall}}$$

  and

$$Precision = \frac{TP}{TP + FP}$$

  where β is the weighting parameter. In this paper we have used the F_1 score ($\beta = 1$), i.e. precision and recall have the same weight.

These three metrics were chosen for two main reasons. First, all three metrics are well suited to evaluate the performance of binary classification algorithms on unbalanced datasets. Secondly, for unbalanced datasets, it is often the minority class that is of more interest to us. For example, in the problem of predicting credit card defaults, the type II error is often much more costly than the type I error. This can be interpreted as the cost of incorrectly predicting a normal user as a defaulting user may simply be an increase in supervision of that user or a restriction on the number of loans they can take out. The cost of predicting a defaulting user as a normal user is a direct financial loss. Obviously the platform or the bank will want to avoid the second type of error as much as possible. This is why recall tends to take precedence over accuracy in such problems. However, if the entire focus is on controlling the recall rate, and any windfall by the user is identified as a default operation, then the trust of users in the platform is lost, resulting in potential financial losses, which could be equally devastating for the platform. The introduction of F-measure is therefore also necessary. It is worth noting that the balance of how recall and accuracy should be controlled needs to be analysed on a case-by-case basis. The benefit of the F-measue F_β is that the weighting on precision and recall can be dynamically assigned by changing the value of β.

3.6 Implementation

The code for the models covered in this article was written in Python 3.8. The implementation of the classifier and resampling method was based on the Python library Scikit-Learn [57] version 0.24.2 and Imbalanced-Learn [58] version 0.8.0. Our CTGANS and CTGAN was written in PyTorch [59] version 1.7.0 and SDV [60] version 0.12.0.

Chapter 4

Result and Discussion

4.1 Generative performance

4.1.1 Kernel Density Estimation

We first show the distribution of the twelve variables of the synthetic and original data, as Figures 4.1, 4.2. This includes the six most important numerical variables

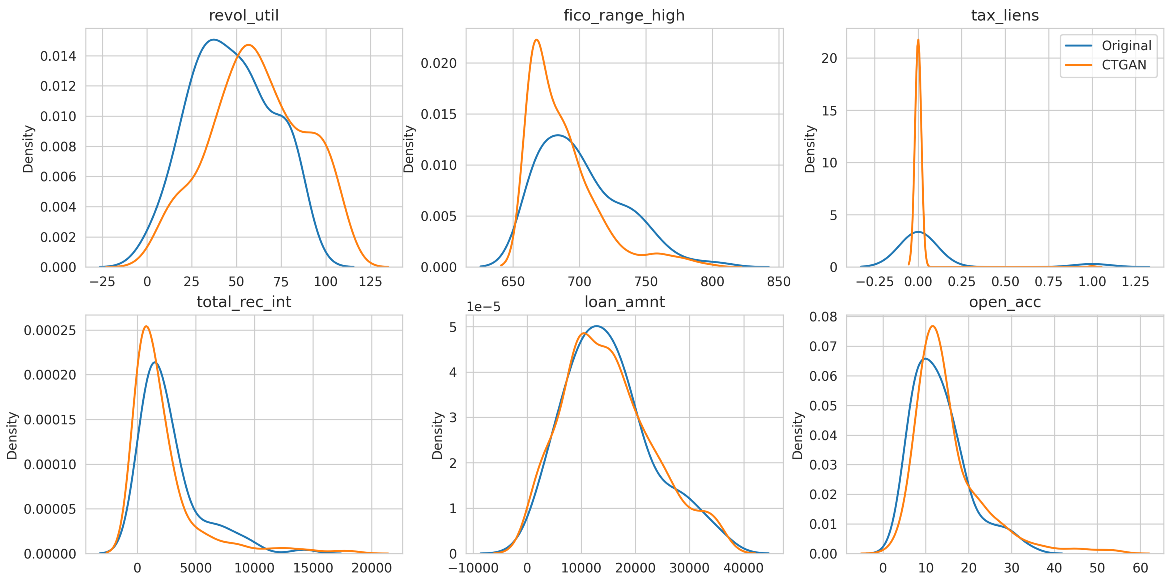

Figure 4.1: Gaussian Kernel density estimates plot of six numerical variables in the Lending dataset, where the six numerical variables correspond to the top six variables of feature importance, as shown in Figure A.1 in Appendix.

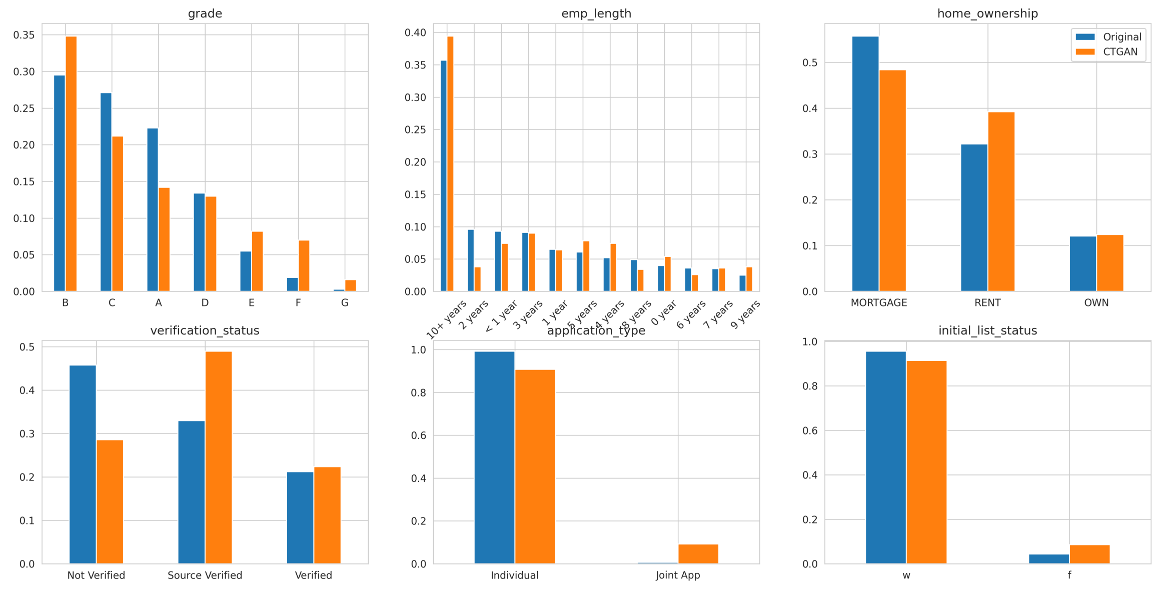

Figure 4.2: Frequency distribution plots for categorical variables, comparing the real and generated distributions of the categorical variables in the Lending dataset.

and six of all seven categorical variables. The analysis of these two plots shows that in most cases the distribution of the variables of the synthetic data generated by the CTGAN model is consistent with that of the original data. However, there are still variables that do not match the original data, such as "fico_range_high" and "tax_liens", and CTGAN fails to generate numerical classes that do not occur frequently. However, such a problem is not commonly found in categorical variables; instead, CTGAN pays sufficient attention to infrequent categories, for example in the "application_type" column. For the categorical variables, the CTGAN model generated more average values in each variables of the synthetic data compared to the real data.

4.1.2 Statistical Metrics

Based on the standard deviation of the variables in the synthetic and original data presented in Table 4.1, it can be seen that the standard deviation of the data generated by the CTGAN model is overall smaller than that of the original data, which is consistent with our hypothesis. However, whether this allows the original dataset to be enhanced and thus improve the performance of the classification algorithm still needs further validation.

Variable	revol_util	fico_range_high	tax_liens	total_rec_int	loan_amnt	open_acc
Original	23.41	33.58	0.31	2818.53	8725.50	5.60
CTGAN	26.34	26.43	0.06	3026.01	8217.38	8.19
Variable	**delinq_2yrs**	**total_acc**	**pub_rec**	**term**	**installment**	**revol_bal**
Original	0.89	11.96	0.54	11.06	251.58	20321.14
CTGAN	0.31	11.56	0.28	10.48	245.45	17383.88

Table 4.1: Standard deviation of the variables in original data and in the synthetic data generated by CTGAN

According to Table 4.2, we can see that in most cases the difference between the synthetic data and the original data is within acceptable limits, however, there are still some variables where there is a significant difference, such as ”pub_rec”. The impact of these variables on the classifiers will be analysed further in section 4.4.

Variable	revol_util	fico_range_high	tax_liens	total_rec_int	loan_amnt	open_acc	delinq_2yrs
MSE	2.37e-06	2.40e-05	1.19e-06	3.75e-09	1.44e-11	2.36e-4	0.77
d_∞	4.253-03	4.08e-03	2.50e-03	1.30e-4	4.26e-06	3.72e-2	0.24
Variable	**total_acc**	**pub_rec**	**term**	**installment**	**revol_bal**	**annual_inc**	**Mean**
MSE	1.09e-04	1.00	3.59e-04	1.09e-07	2.35e-11	1.06e-11	0.14
d_∞	1.39e-02	1.43	2.03e-02	5.90e-04	8.93e-06	6.13e-06	0.13

Table 4.2: Results of the metrics (MSE and d_∞) of distance on the distribution of synthetic and real data

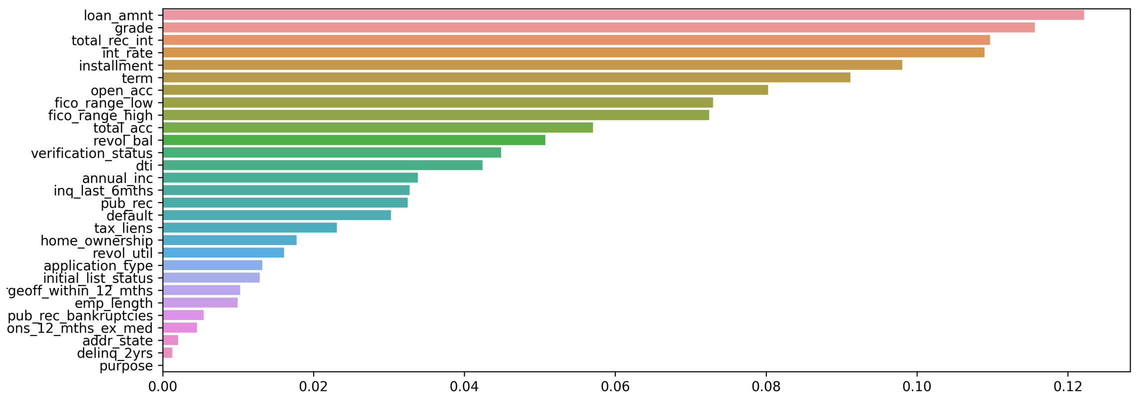

Figure 4.3: Difference in correlation coefficients between real data and synthetic data generated by CTGAN trained with a training set of 10,000 samples.

4.1.3 Correlation

Figure 4.3 shows the correlation gaps between all 29 features in the synthetic data and the original data. Most of the gaps between the features generated by CTGAN and the original data are acceptable, but there are still some, such as "loam_amnt" and "grade", that have more significant gaps with the real data.

In general, the CTGAN succeeded in generating a complex dataset containing both numerical and categorical variables, with most of the variables in the original data being correctly modelled. However, it is undeniable that there are still a small number of variables that have not been generated reasonably well, i.e. their distribution differs from that of the original data. This suggests that mode collapse, although somewhat avoided with the application of multiple methods, still exists. In the next section, we will specifically analyse the impact of these variables on the oversampling method and the classification algorithm.

4.2 Experimental Result

By applying four oversampling methods, one undersampling method and CTGANS to the original dataset, we obtained six balanced datasets. At the same time we extracted a balanced dataset from the entire data with the same sample size as the original dataset. We used these seven datasets together with the original dataset as training sets to train two classification algorithms, Ridge Logistic Regression and Multi-layer perceptron, and tested them on a fixed test set (for different sample sizes of original data). We conducted the above experiments on the original dataset with different sample sizes and the results are shown in Tables 4.3, 4.4 and 4.5.

For the CTGANS, to determine the number of synthetic samples that needed to be generated using CTGAN, we tested it on the original dataset with a sample size of 10,000, and the results showed that the classification algorithm performed best when the synthetic data was about a quarter of the original data, in line with our assumptions in chapter 3.4.3. To ensure the reliability of the model, we maintained this ratio for the rest of the experiments. Other hyperparameters settings are shown in Table A.2 in appendix.

Table 4.3: Performance of the original data without any resampling methods, random replication of the minority class and balanced dataset on the two classification algorithms

N_g	Accuracy			Recall			F1-score		
	Original	ROS	Balanced	Original	ROS	Balanced	Original	ROS	Balanced
100	0.79, 0.79	0.71, 0.78	0.76, 0.75	0.31, 0.30	0.24, 0.09	0.39, 0.39	0.34, 0.34	0.28, 0.14	0.40, 0.38
500	0.81, 0.81	0.68, 0.79	0.74, 0.73	0.05, 0.20	0.51, 0.20	0.16, 0.25	0.10, 0.28	0.39, 0.27	0.25, 0.33
1000	0.83, 0.83	0.67, 0.77	0.73, 0.72	0.04, 0.07	0.59, 0.28	0.28, 0.43	0.07, 0.12	0.44, 0.32	0.39, 0.48
5000	0.87, 0.87	0.67, 0.77	0.70, 0.70	0.01, 0.16	0.68, 0.32	0.48, 0.64	0.02, 0.23	0.44, 0.35	0.55, 0.63
10000	0.88, 0.88	0.69, 0.75	0.69, 0.71	0.02, 0.18	0.69, 0.36	0.56, 0.63	0.03, 0.25	0.46, 0.36	0.62, 0.66
50000	0.90, 0.89	0.68, 0.74	0.69, 0.72	0.01, 0.19	0.72, 0.52	0.63, 0.68	0.01, 0.26	0.46, 0.43	0.67, 0.71

* Each block in the table contains two pieces of data, the one on the left is the performance of logistic regression and the one on the right is the performance of Multilayer perceptron. N_g is the sample size of the Original dataset.

Table 4.4: Performance of the dataset processed with three traditional oversampling methods (SMOTE, ADASYN, NearMiss) on two classification algorithms.

N_g	Accuracy			Recall			F1-score		
	SMOTE	ADASYN	NM	SMOTE	ADASYN	NM	SMOTE	ADASYN	NM
100	0.74, 0.79	0.74, 0.79	0.60, 0.56	0.21, 0.10	0.22, 0.11	0.48, 0.67	0.23, 0.15	0.24, 0.17	0.31, 0.36
500	0.72, 0.79	0.70, 0.79	0.48, 0.47	0.46, 0.20	0.47, 0.18	0.67, 0.69	0.38, 0.26	0.37, 0.24	0.32, 0.33
1000	0.69, 0.75	0.69, 0.79	0.55, 0.51	0.55, 0.26	0.56, 0.21	0.58, 0.65	0.41, 0.29	0.40, 0.28	0.33, 0.33
5000	0.69, 0.77	0.69, 0.78	0.50, 0.45	0.68, 0.29	0.67, 0.28	0.73, 0.76	0.45, 0.32	0.46, 0.33	0.35, 0.34
10000	0.70, 0.79	0.70, 0.75	0.50, 0.44	0.69, 0.29	0.68, 0.33	0.75, 0.80	0.46, 0.32	0.46, 0.39	0.36, 0.35
50000	0.68, 0.75	0.68, 0.78	0.48, 0.42	0.71, 0.35	0.71, 0.49	0.80, 0.84	0.46, 0.39	0.46, 0.38	0.37, 0.35

* Each block in the table contains two pieces of data, the one on the left is the performance of logistic regression and the one on the right is the performance of Multilayer perceptron. N_g is the sample size of the Original dataset.

Table 4.5: Performance of datasets processed with GAN-based(CTGAN) oversampling method and CTGANS on two classification algorithms

N_g	Accuracy		Recall		F1-score	
	CTGAN	CTGANS	CTGAN	CTGANS	CTGAN	CTGANS
100	0.58, 0.62	0.63, 0.77	0.24, 0.19	0.48, 0.17	0.24, 0.15	0.37, 0.22
500	0.62, 0.69	0.65, 0.80	0.26, 0.13	0.58, 0.17	0.22, 0.13	0.39, 0.24
1000	0.71, 0.72	0.66, 0.78	0.33, 0.31	0.66, 0.17	0.24, 0.21	0.43, 0.22
5000	0.72, 0.77	0.65, 0.78	0.52, 0.30	0.74, 0.23	0.40, 0.32	0.44, 0.28
10000	0.75, 0.81	0.67, 0.77	0.49, 0.30	0.72, 0.29	0.39, 0.37	0.45, 0.33
50000	0.76, 0.82	0.65, 0.79	0.52, 0.27	0.75, 0.36	0.41, 0.36	0.45, 0.39

* Each block in the table contains two pieces of data, the one on the left is the performance of logistic regression and the one on the right is the performance of Multilayer perceptron. N_g is the sample size of the Original dataset.

By analysing Tables 4.3, 4.4 and 4.5, it can be seen that RLR performs significantly better than MLP on all three metrics for most of the resampling methods, with MLP only outperforming RLR on the original dataset, the balanced dataset and NearMiss. The explanation for the better performance of RLR over MLP is given in chapter 4.3.

Based on the analysis of the performance of the resampling method on RLR (three metrics), we found the following observations:

1. For the original dataset, the performance of RLR is severely affected by the imbalance nature of the dataset. The classification algorithm tends to classify the entire sample as non-defaulted. This makes the Recall rate and F1-score close to 0 when the sample size reaches 50,000. This demonstrates the necessity of resampling methods for dealing with datasets with high imbalance rates.

2. The performance of the three traditional oversampling methods is generally consistent. However, ROS performs slightly better than SMOTE and ADASYN when the number of samples is insufficient, i.e. when the sample size is less than 5000. This is in line with our analysis in chapter 3.4.1, where the oversampling model may be of negative benefit for certain high-dimensional datasets, i.e. the instability of the model has a higher negative impact on the classification algorithm than the benefit it brings due to the excessive amount of data that needs to be generated. For the undersampling method NearMiss, it improves recall at the expense of accuracy. In real life, while a higher recall rate can avoid direct financial losses from defaults, a lower accuracy rate also poses a potential risk to the lending platform.

3. For the balanced datasets, the performance of the RLR model improves as the sample size increases. The accuracy, recall rate and F1-score eventually stabilised at 0.69, 0.63 and 0.67 respectively. Although there was a slight disadvantage in recall rate, the performance was still significantly better than all resampling methods overall. The balanced dataset, on the other hand, did not perform as well as the various oversampling methods when the amount of data was severely insufficient, i.e. when the amount of data was less than 1000. This suggests that when the amount of data is severely insufficient, for binary

classification tasks, it is possible to improve the performance of the classification algorithm by increasing the proportion of a class in the sample and then applying an oversampling model. This is because clearly if the identification of defaulting users is improved, then the misclassification of non-defaulting users will be correspondingly avoided.

4. Using CTGAN as an oversampling method performs significantly worse on RLR than the three traditional oversampling models. At a sample size of 50,000, only accuracy improved by 0.08, at the cost of a 0.23 recall rate and 0.05 F1-score loss. We attribute this to the fact that the distribution of some variables in the data generated by CTGAN differed significantly from the original data, and that the generation of too many minority class samples amplified the impact produced by these variables.

5. The CTGANS is a significant improvement over the traditional oversampling model. Although its performance is essentially in line with traditional oversampling methods at 50,000 data, CTGANS perform noticeably better than other resampling methods when the amount of data is insufficient, approaching stability at only 5,000 data. In the process of data updating, CTGANS can compensate to some extent for its disadvantage in training time by reducing the number of training samples.

To sum up, the advantage of CTGANS is in handling datasets with insufficient amount of data. However, when the amount of data is sufficient, unlike processing methods such as feature engineering, CTGANS does not further improve the performance of the classifier compared to traditional oversampling methods.

4.3 Explanation of Anomalies

It is very counter-intuitive that MLPs perform at a huge disadvantage compared to RLR. Because MLPs can be abstracted into more complex logistic regression models, the greater number of neural network units and layers dictates that they can learn more properties from the data than logistic regression models [61]. We have not found an argued explanation for this phenomenon in the past literature. Our conjecture on this is that the neural network is biased in its recognition of default data during the training process. Since the majority of the default data (the minority class) are generated by oversampling method, the MLP identifies the generation patterns during training, so that the decision on the minority class becomes a decision on whether it is synthetic data, similar to the discriminator in GANs. In the test set, on the other hand, all of the data is real data, so the MLP loses the ability to recognise the default data.

We found two pieces of evidence for this conjecture. Firstly in Tables 4.3 and 4.4, there are two datasets on which MLP performs better than RLR, namely the balanced dataset and NearMiss, neither of which contains synthetic data. The next is a comparison between the three oversampling methods. Again as oversampling methods, the three methods perform more or less equally when using RLR, while ROS is better than ADASYN and better than SMOTE when using MLP, which, according to our conjecture, can be explained by the fact that ROS performs better than the other two methods because it does not generate synthetic data. For ADASYN, we believe that this is due to the fact that ADASYN is more complex than SMOTE in the way it generates synthetic data and has some randomness. Therefore, it is more difficult for MLPs to learn the generation method of ADASYN than SMOTE. As the focus of this paper is not on this aspect, we hope to provide a specific analysis of this issue in the future.

4.4 Information Loss

As stated in chapter 4.1, although in most cases the CTGAN succeeded in learning the features of the data and generating synthetic data, it still had shortcomings in the generation of certain variables. Nevertheless, a comparison of Tables 4.3-4.5 shows that overall the CTGANS still performs better than the three oversampling methods. This demonstrates the usefulness of the GAN-based model on unbalanced datasets on the one hand, and the effectiveness of the synthetic data generated by the CTGAN model on the other. To explore the overall impact of the poorly generated variables, we designed the following experiments:

- Prepare a dataset that has been resampled by the CTGANS method, remove one column at a time, then train it through a Ridge logistic regression model and compare it to the performance of the full dataset. If the performance of the RLR becomes better after removing a column, it means that the corresponding variable has a negative impact on the RLR. However, the reverse is not true, and poorer performance after removing a columns does not prove a positive impact of that corresponding variable.

- As the dataset itself may also contain certain outliers that negatively affect the classification algorithm. Therefore, as a comparison, the balanced dataset was chosen. Again, one column was removed in turn and it was observed which column removed had a positive impact on the RLR.

The results are shown in the table below:

	CTGANS	Balanced
features		addr_state
		fico_range_high
	addr_state	annual_inc
	fico_range_high	total_acc
	annual_inc	acc_now_delinq
	total_acc	pub_rec_bankruptcies
	acc_now_delinq	collections_12_mths_ex_med
	open_acc	tax_liens
	loan_amnt	pub_rec
	verification_status	dti
		grade
		purpose

Table 4.6: Variables negatively impacting RLR in CTGANS and Balanced datasets. Sample size of the Original dataset used to train CTGAN and the balanced dataset is 10,000.

Firstly, Table 6 shows that eight variables in the dataset generated by CTGANS had a negative effect on RLR while 12 variables in the balanced dataset. This includes five overlapping variables and, although to varying degrees, it is essentially certain that the cause of the negative effect of these five variables on RLR comes from the data itself. For the three negative variables specific to the CTGANS dataset, we found that they corresponded to three distinct anomalies. For the variable "open_acc", the standard deviation of the data in this variable is significantly larger than that of the original data, as shown in Table 4.1, contrary to our assumption that the standard deviation of the synthetic data is smaller than that of the original data. For the variable "loan_amnt", the correlation coefficient between this variable and the other variables in synthetic data is the most different from the original data (about 0.12), as shown in Figure 4.3, indicating that the data in this variable deviates from the overall data in a more significant way. For the variable "verification_status", it is the only variable in the categorical variable that is clearly different from the distribution of the original data, as shown in Figure 4.2.

On the other hand, the seven variables unique to the balanced data that have a negative impact on the RLR are interpreted as the synthetic data generated by CTGAN weaken the overall impact of the outliers. This is evident for the variable "tax_liens", which is shown in Figure 4.1 to be concentrated around 0, with a standard deviation (0.28) about half that of the variable in the original data . This demonstrates, on the one hand, that certain variables generated by the CTGAN that differ significantly from the original data do not necessarily have a negative impact on the classification model. On the other hand it hints at the difficulty of model optimisation, i.e. increasing the similarity of variables to the original data is both difficult to achieve and at the same time does not guarantee an improvement in classification effectiveness.

Chapter 5

Conclusion and Future Challenge

5.1 Conclusion

We proposed a framework for handling imbalanced datasets based on CTGAN and SMOTE, CTGANS, and compared it with three oversampling methods, an undersampling method and a balanced dataset. We evaluated the P2P lending dataset with two classification algorithms, using three metrics of classification performance. We first evaluated the performance of the CTGAN model for generating tabular data and the validity of the CTGAN-based oversampling approach. The results showed that CTGAN successfully generated a complex dataset containing both numerical and categorical variables, with the distribution of most of the variables largely consistent with the original data, but mode collapse persisted in some variables. This resulted in the performance of the CTGAN-based oversampling method being significantly inferior compared to traditional oversampling methods.

While our method performs largely in line with traditional oversampling methods when the amount of data is sufficient, it has a significant performance advantage when the amount of data is insufficient, using only 10% of the samples to bring the performance of the classification algorithm close to that of other oversampling methods.

However, when comparing the performance of RLR and MLP, we found that the MLP trained on the datasets balanced by oversampling method performed significantly worse than the RLR model on the test set. Our conjecture is that the complexity of the model shifts the focus of learning from the features in the default data to the structure of the data, i.e. the MLP tends to recognise synthetic data in the dataset. Future research could test our conjecture by setting up two test sets, one with real data and one with synthetic data, and by observing the performance of the MLP on both sets.

5.2 Future Challenge

In our analysis of information loss, we found that for variables generated by the CTGAN model, it is not the case that variables that do not match the original data will necessarily have a negative impact on the classification model. Boosting the similarity of the synthetic data generated by the GAN model to the real data does not guarantee an improvement in its overall performance. This uncertainty undoubtedly creates an obstacle to the application of the GAN model to unbalanced datasets.

Future research could work in the direction of increasing the demand for synthetic data generated by the GAN, i.e. in addition to telling the GAN to generate synthetic data close to the real data, it should also tell the GAN how the generated data is to be applied. Similar to the use of the Wasserstein Distance as a loss function in CTGAN, we believe that a classification discriminator could be added to the structure of the GAN to make the synthetic data generated by the generator more suitable for being used to train a classification algorithm. It is also worthwhile to further investigate how to avoid the classifier being trained as a discriminator (same use as in GAN).

Bibliography

[1] J. Turiel and T. Aste, “Peer-to-peer loan acceptance and default prediction with artificial intelligence,” *Royal Society open science*, vol. 7, no. 6, p. 191649, 2020.

[2] A. Byanjankar, M. Heikkilä, and J. Mezei, “Predicting credit risk in peer-to-peer lending: A neural network approach,” in *2015 IEEE symposium series on computational intelligence.* IEEE, 2015, pp. 719–725.

[3] S. Sakprasat and M. C. Sinclair, “Classification rule mining for automatic credit approval using genetic programming,” in *2007 IEEE Congress on Evolutionary Computation.* IEEE, 2007, pp. 548–555.

[4] R. Longadge and S. Dongre, “Class imbalance problem in data mining review,” *arXiv preprint arXiv:1305.1707*, 2013.

[5] Z. Zheng, Y. Cai, and Y. Li, “Oversampling method for imbalanced classification,” *Computing and Informatics*, vol. 34, no. 5, pp. 1017–1037, 2015.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.

[8] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, “Modeling tabular data using conditional gan,” *arXiv preprint arXiv:1907.00503*, 2019.

[9] S. Park and H. Park, “Performance comparison of multi-class svm with oversampling methods for imbalanced data classification,” in *International Conference on Broadband and Wireless Computing, Communication and Applications.* Springer, 2020, pp. 108–119.

[10] J. Engelmann and S. Lessmann, “Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning,” *Expert Systems with Applications*, vol. 174, p. 114582, 2021.

[11] ——, “Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning,” *arXiv preprint arXiv:2008.09202*, 2020.

[12] S. Chen, Q. Wang, and S. Liu, “Credit risk prediction in peer-to-peer lending with ensemble learning framework,” in *2019 Chinese Control And Decision Conference (CCDC).* IEEE, 2019, pp. 4373–4377.

[13] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, “Privbayes: Private data release via bayesian networks,” *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 4, pp. 1–41, 2017.

[14] J. P. Reiter, “Using cart to generate partially synthetic public use microdata,” *Journal of Official Statistics*, vol. 21, no. 3, p. 441, 2005.

[15] Y. Pu, S. Dai, Z. Gan, W. Wang, G. Wang, Y. Zhang, R. Henao, and L. C. Duke, “Jointgan: Multi-domain joint distribution learning with generative adversarial nets,” in *International Conference on Machine Learning.* PMLR, 2018, pp. 4151–4160.

[16] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, “Data synthesis based on generative adversarial networks,” *arXiv preprint arXiv:1806.03384*, 2018.

[17] Z. Liu, S. B. Navathe, and J. T. Stasko, “Ploceus: Modeling, visualizing, and analyzing tabular data as networks,” *Information Visualization*, vol. 13, no. 1, pp. 59–89, 2014.

[18] X. Xuan, B. Peng, W. Wang, and J. Dong, "On the generalization of gan image forensics," in *Chinese conference on biometric recognition.* Springer, 2019, pp. 134–141.

[19] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," in *Machine learning for healthcare conference.* PMLR, 2017, pp. 286–305.

[20] M. K. Baowaly, C.-C. Lin, C.-L. Liu, and K.-T. Chen, "Synthesizing electronic health records using improved generative adversarial networks," *Journal of the American Medical Informatics Association*, vol. 26, no. 3, pp. 228–241, 2019.

[21] K. Armanious, C. Jiang, M. Fischer, T. Küstner, T. Hepp, K. Nikolaou, S. Gatidis, and B. Yang, "Medgan: Medical image translation using gans," *Computerized Medical Imaging and Graphics*, vol. 79, p. 101684, 2020.

[22] U. Fiore, A. De Santis, F. Perla, P. Zanetti, and F. Palmieri, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Information Sciences*, vol. 479, pp. 448–455, 2019.

[23] G. Douzas and F. Bacao, "Effective data generation for imbalanced learning using conditional generative adversarial networks," *Expert Systems with applications*, vol. 91, pp. 464–471, 2018.

[24] I. Guyon, "Design of experiments of the nips 2003 variable selection benchmark," in *NIPS 2003 workshop on feature extraction and feature selection*, vol. 253, 2003.

[25] H. Ba, "Improving detection of credit card fraudulent transactions using generative adversarial networks," *arXiv preprint arXiv:1907.03355*, 2019.

[26] F. Zhou, S. Yang, H. Fujita, D. Chen, and C. Wen, "Deep learning fault diagnosis method based on global optimization gan for unbalanced data," *Knowledge-Based Systems*, vol. 187, p. 104837, 2020.

[27] W. Zhang, X. Li, X.-D. Jia, H. Ma, Z. Luo, and X. Li, "Machinery fault diagnosis with imbalanced data using deep generative adversarial networks," *Measurement*, vol. 152, p. 107377, 2020.

[28] S. K. Lim, Y. Loo, N.-T. Tran, N.-M. Cheung, G. Roig, and Y. Elovici, "Doping: Generative data augmentation for unsupervised anomaly detection with gan," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 1122–1127.

[29] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the border: active learning in imbalanced data classification," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 127–136.

[30] S. Nguyen, J. Quinn, and A. Olinsky, "An oversampling technique for classifying imbalanced datasets," in *Advances in Business and Management Forecasting*. Emerald Publishing Limited, 2017.

[31] I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of workshop on learning from imbalanced datasets*, vol. 126. ICML United States, 2003.

[32] T. Thaher, M. Mafarja, B. Abdalhaq, and H. Chantar, "Wrapper-based feature selection for imbalanced data using binary queuing search algorithm," in *2019 2nd international conference on new trends in computing sciences (ICTCS)*. IEEE, 2019, pp. 1–6.

[33] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE, 2008, pp. 1322–1328.

[34] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[35] Y. Anzai, *Pattern recognition and machine learning*. Elsevier, 2012.

[36] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[37] M. Chen, W. Liao, H. Zha, and T. Zhao, "Statistical guarantees of generative adversarial networks for distribution estimation," *arXiv preprint arXiv:2002.03938*, 2020.

[38] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[39] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[40] A. Koshiyama, N. Firoozye, and P. Treleaven, "Generative adversarial networks for financial trading strategies fine-tuning and combination," *arXiv preprint arXiv:1901.01751*, 2019.

[41] D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba, "Seeing what a gan cannot generate," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4502–4511.

[42] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[43] Z. Lin, A. Khetan, G. Fanti, and S. Oh, "Pacgan: The power of two samples in generative adversarial networks," *Advances in neural information processing systems*, 2018.

[44] L. V. Kantorovich, "Mathematical methods of organizing and planning production," *Management science*, vol. 6, no. 4, pp. 366–422, 1960.

[45] G. Montavon, K.-R. Müller, and M. Cuturi, "Wasserstein training of restricted boltzmann machines," *Advances in Neural Information Processing Systems*, vol. 29, pp. 3718–3726, 2016.

[46] C. R. Harvey and Y. Liu, "Backtesting," *The Journal of Portfolio Management*, vol. 42, no. 1, pp. 13–28, 2015.

[47] A. Y.-c. Liu, "The effect of oversampling and undersampling on classifying imbalanced text datasets," Ph.D. dissertation, Citeseer, 2004.

[48] V. Podgorelec, M. Hericko, and I. Rozman, "Improving mining of medical data by outliers prediction," in *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05).* IEEE, 2005, pp. 91–96.

[49] V. Tresp, S. Ahmad, R. Neuneier *et al.*, "Training neural networks with deficient data," *Advances in neural information processing systems*, pp. 128–128, 1994.

[50] C. Frogner, C. Zhang, H. Mobahi, M. Araya-Polo, and T. Poggio, "Learning with a wasserstein loss," *arXiv preprint arXiv:1506.05439*, 2015.

[51] D. M. Allen, "Mean square error of prediction as a criterion for selecting variables," *Technometrics*, vol. 13, no. 3, pp. 469–475, 1971.

[52] E. T. Jaynes, *Probability theory: The logic of science.* Cambridge university press, 2003.

[53] Y. Sasaki *et al.*, "The truth of the f-measure. 2007," *URL: https://www. cs. odu. edu/˜ mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07. pdf [accessed 2021-05-26]*, 2007.

[54] S. Weisberg, *Applied linear regression.* John Wiley & Sons, 2005, vol. 528.

[55] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[56] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 5, no. 4, pp. 455–455, 1992.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[58] G. Lemaître, F. Nogueira, and C. K. Aridas, “Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 559–563, 2017.

[59] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[60] A. Montanez *et al.*, “Sdv: an open source library for synthetic data generation,” Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[61] I. Kurt, M. Ture, and A. T. Kurum, “Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease,” *Expert systems with applications*, vol. 34, no. 1, pp. 366–374, 2008.

Appendix A

Additional information

feautre	explaination
addr_state	The state provided by the borrower in the loan application
annual_inc	The self-reported annual income provided by the borrower during registration.
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrowers credit file for the past 2 years
dti	A ratio calculated using the borrowers total monthly debt payments on the total debt obligations
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
fico_range_high	The upper boundary range the borrowers FICO at loan origination belongs to.
fico_range_low	The lower boundary range the borrowers FICO at loan origination belongs to.
grade	LC assigned loan grade
home_ownership	The home ownership status provided by the borrower during registration. Our values are: RENT
initial_list_status	The initial listing status of the loan. Possible values are W
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
installment	The monthly payment owed by the borrower if the loan originates.
int_rate	Interest Rate on the loan
loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time
open_acc	The number of open credit lines in the borrowers credit file.
pub_rec	Number of derogatory public records
purpose	A category provided by the borrower for the loan request.
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
total_acc	The total number of credit lines currently in the borrowers credit file
total_rec_int	Interest received to date
acc_now_delinq	The number of accounts on which the borrower is now delinquent.

Table A.1: Full name and explanation of the abbreviated features in the Lending dataset

Hyperparameter	Value
Generator learning rate	0.0002
Discriminator learning rate	0.0002
Random sample pass to the Generator	128
Generator weight decay for Adam optimizer	1e-06
Discriminator weight decay for Adam optimizer	1e-06
batch size	25
Epochs	300
Generator output sample size	(256,256)
Discriminator output sample size	(256,256)
Dropout rate	0.2

Table A.2: CTGAN Hyperparameter Setting, referenced to the paper [8].

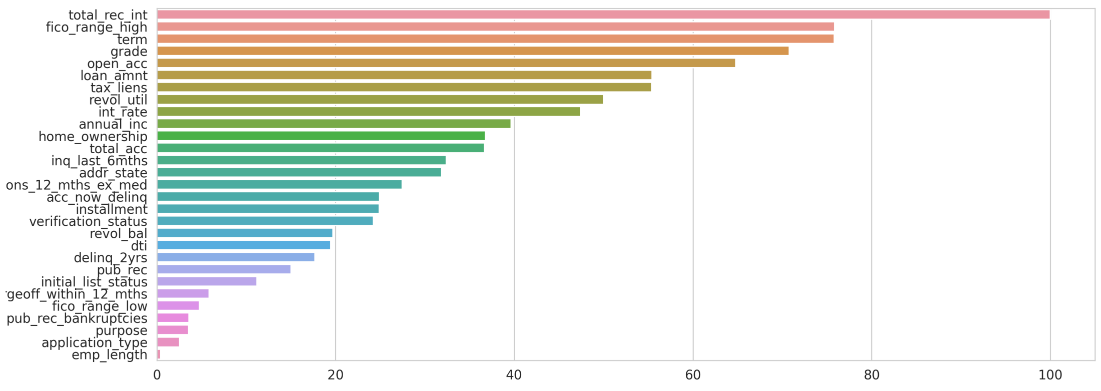

Figure A.1: Feature importance according to the weights of RLR trained on sample size 10000 training set