UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Assessing VTE Risk in Cancer Patients using Deep Learning Synthetic Data Generation and Domain Adaptation Techniques

Author:
Sergi BECH

Supervisor:
Dr. Oriol PUJOL
Barbara LOBATO

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 30, 2023

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Assessing VTE Risk in Cancer Patients using Deep Learning Synthetic Data Generation and Domain Adaptation Techniques

by Sergi BECH

Venous thromboembolism (VTE) poses a significant health risk for cancer patients. In this thesis, we address the challenge of predicting VTE occurrence in cancer patients by employing state-of-the-art methods and exploring the potential of deep learning and synthetic data generation techniques.

We present a Python implementation of the current state-of-the-art method for VTE prediction in cancer patients. This serves as a benchmark for our subsequent investigations.

Building upon this foundation, the investigation focuses into the application of deep learning synthetic data generation methods to assess the risk of future treatments and medication for preventing VTE in cancer patients. Utilizing a small dataset comprising genetic and clinical variables, we extensively explore and compare the performance of state-of-the-art generative deep learning models specifically designed for tabular data.

Notably, we adopt the CopulaGAN architecture to generate synthetic tabular data, which is subsequently utilized to train a deep learning-based classifier using domain adaptation techniques to fine-tuned the model with real data. The resulting model outperforms current state-of-the-art medical scores in accurately assessing VTE risk. Furthermore, the Precision-Recall curve derived from our model offers enhanced flexibility in selecting optimal operational points for VTE risk assessment.

By combining the power of deep learning and synthetic data generation, our research contributes to the advancement of VTE risk prediction in cancer patients. The proposed methodology demonstrates promising results and paves the way for improved patient care and personalized treatment strategies.

Furthermore, we introduce target encoding in the architecture of the conditional tabular generative adversarial network (CTGAN) to handle better large categorical variables.

We believe that our findings have significant implications for the field of oncology and hold great potential for enhancing patient outcomes and reducing the burden of VTE in cancer care.

Acknowledgements

I would like to express my gratitude to my family, especially my parents, for their support and encouragement throughout this journey. Their belief in me and constant motivation have been absolutely invaluable.

I am immensely grateful to my supervisors, Dr. Oriol Pujol and Barbara Lobato, for their guidance, expertise, and mentorship. Their invaluable insights and encouragement have played an important role in developing this research work.

Furthermore, I extend my thanks to Dr. Jose Manuel Soria for granting permission to use the data, which made this research possible.

This work would not have been possible without the support and contributions of these individuals and organizations. I am truly indebted to them for their support and guidance throughout this research endeavor.

Contents

Chapter 1

Introduction

1.1 Motivation

Cancer, being one of the most prevalent and life-threatening diseases in contemporary times, poses significant challenges to healthcare systems worldwide. In 2020 alone, there were over 19 million newly diagnosed cases of cancer and approximately 10 million cancer-related deaths globally (Sung et al., 2021). Nonetheless, significant advancements in oncologic treatments have transformed the outlook for many cancer types, rendering them as chronic conditions or even curable in some cases (Pituskin, 2022).

It is important to recognize that cancer patients face an elevated risk of developing venous thromboembolism (VTE) compared to the general population (Blom et al., 2005). VTE encompasses conditions such as deep vein thrombosis (DVT) and pulmonary embolism (PE) and can have severe consequences if left untreated. Fortunately, the use of anticoagulants has proven highly effective in preventing VTE in cancer patients, and improved anticoagulant options have become available in recent years (Desai et al., 2020).

However, it is crucial to acknowledge that cancer patients also face an increased risk of bleeding complications (Al-Samkari and Connors, 2019). This poses a significant clinical challenge as healthcare professionals must strike a delicate balance between the benefits of administering anticoagulants to prevent VTE and the potential risks of severe bleeding.

Moreover, as the life expectancy of chronic cancer patients continues to improve due to advancements in treatment, the likelihood of experiencing VTE at some point during their journey increases as well (Mahajan et al., 2022). Consequently, the clinical management of cancer patients becomes even more complex, as the potential for both VTE and bleeding complications must be carefully considered and addressed.

Addressing these intricate challenges requires a comprehensive understanding of the interplay between cancer, VTE, and anticoagulant therapy. By further exploring the dynamics of these factors, we can develop strategies and interventions that optimize the clinical management of cancer patients, ensuring both their safety and the prevention of life-threatening complications.

1.1.1 Machine Learning attempts to predict VTE

The first attempt to create a tool able to classify cancer patients by VTE risk prior to anti-cancer treatment is the Khorana Risk Score (KRS) (Khorana et al., 2008), which utilises five clinical features and assigns a score depending on their observation. Patients are then classified into low, intermediate or high risk. Despite its modest performance, the KRS has become the reference score and many of the current scores in clinical practice are modified versions of this.

With the increase in accessibility to genetic analysis and the evidences that support the role of genetics in VTE (Zöller et al., 2015), (Zöller, 2019), new scores based on combining genetic variables and phenotypic information have been developed. The first of such scores was TiC-Onco score (Muñoz Martin et al., 2018). This score has been recently replaced by ONCOTHROMB score (Muñoz et al., 2023), which establishes the current state-of-the-art and outperforms KRS when access to genetic information is available.

Due to the costs and potential impact of these studies, the creation and validation of these scores usually entails small datasets. For example, the validation of the ONCOTHROMB score only uses a few hundreds of patients. Although using the correct methodologies can lead to models with a certain degree of generalization, the small amount of samples combined with the large dimensionality (imposed by the combination of phenotypic and genetic features) certainly hinders predictive models from achieving the highest performance.

The machine learning community has recognized the challenges posed by limited data in conjunction with the curse of dimensionality. To address this issue, dimensionality reduction and feature selection techniques have been widely employed. However, with the emergence of deep learning techniques and the growing demand for extensive data to train complex models, there has been a resurgence of methodologies focused on generating new samples. Synthetic data generation has garnered significant attention as it overcomes certain limitations of noise-based models and interpolation methods like SMOTE (Chawla et al., 2002). Within this trend, a multitude of approaches have emerged to generate synthetic tabular data, leveraging existing architectures originally designed for generating images. These approaches encompass variational autoencoders (VAEs) (Wan, Zhang, and He, 2017), generative adversarial networks (GANs) (Abedi et al., 2022), diffusion models (Azizi et al., 2023), and more.

1.2 Objective

The objectives of the thesis are the following ones:

- We aim to replicate the results presented in the paper by Muñoz et al., 2023 to establish our benchmark. This involves meticulously reproducing the findings to validate their accuracy and reliability. Once we have successfully replicated the paper's results, our focus shifts towards optimizing and enhancing those findings.

- The main goal is to generate synthetic data from the original dataset, encompassing all patient variables and their corresponding risk allele counts. To achieve this, we will employ deep learning techniques to develop synthetic data that faithfully captures the statistical characteristics of the original dataset.

- With the generated synthetic data in hand, in the subsequent analysis, our aim is to substitute the logistic regression model employed in the original paper with a more suitable model for handling larger datasets. We intend to evaluate the performance of this new model and compare it against the benchmark. This comparison will enable us to determine if the new model yields improved results.

- We will delve into the theoretical details of tabular Generative Adversarial Networks (GANs) and assess the performance of several state-of-the-art methods for generating synthetic data applicable to our dataset. This analysis will allow us to compare different approaches and identify the most effective ones. This knowledge will enable us to suggest and implement improvements that enhance the performance of tabular GANs.

Overall, our objective is to reproduce the results presented in the aforementioned paper, contribute to optimize the VTE risk assessment by using synthetic data and understand all the fundamentals of the GANs for tabular data.

1.2.1 Tools

To achieve our goals, we will use Python as our primary programming language and leverage deep learning libraries such as TensorFlow (Abadi et al., 2015), Keras (Chollet et al., 2015) and SDV (Team, 2022) to implement deep learning models.

For efficient data processing and comprehensive statistical analysis, we will employ established machine learning libraries such as Pandas and Scikit-learn (Pedregosa et al., 2011).

1.2.2 Contributions

In this thesis, we provide the following contributions:

- A very detailed study, along with a Python reproduction of the ONCOTHROMB12-01 (ONCOTHROMB) score paper.
- We explore the creation of tabular synthetic data compatible with features in ONCOTHROMB score with the goal of inducing a learning bias in such model. To that end, a variant of Conditional Tabular GAN — a CopulaGAN — is used to generate a large dataset of compatible data with the real ONCOTHROMB cohort. Then, using transfer learning techniques (Iman, Arabnia, and Rasheed, 2023), a pretrained model with synthetic data is fine-tuned to the real data. The results show that the obtained model improves not only the generalization of the score but also the range of operational points in the Precision-Recall curve. This is of particular interest, as it allows to improve the evaluation of the two risks involved, namely complications due to VTE and those due to anticoagulant treatment.
- We suggest a slight adjustment to the CTGAN architecture to effectively manage large categorical variables using target encoding. This modification has demonstrated promising results; however, it is a preliminary approach, and further experimentation is required to validate its effectiveness.

The code pertaining to this thesis can be accessed on GitHub at the following URL: `https://github.com/sergibech/Master_thesis`

1.2.3 Contents

This thesis is structured in the following manner: Chapter 1 provides a concise introduction and rationale for the problem at hand, highlighting the potential of artificial intelligence techniques. In Chapter 2, the relevant theoretical background and concepts that will be referenced throughout the thesis are presented. Moving on, Chapter 3 offers an overview of all the conducted experiments, emphasizing the objectives and providing a general outline of the steps and methods employed in each one. Subsequently, Chapters 4, 5, 6 and 7 deep dive into the experimental details, ensuring the reproducibility of each experiment and presenting the results through tables and graphics. Lastly, Chapter 8 encapsulates the conclusions drawn from our work and outlines potential future research directions.

Chapter 2

Theoretical Background

2.1 Assessing VTE risk using ONCOTHROMB score

The ONCOTHROMB score was created using both genetic and clinical features with the aim to predict whether a patient diagnosed with cancer will develop VTE within the next six months since the diagnosis. The cohort of patients used to train this classifier is described in Muñoz Martin et al., 2018. The predictive features used to train a logistic regression are: VTE risk according to tumor type as defined in the KRS, stage, whether the body mass index (BMI) is greater than 25 Kg/m2, and a Genetic Risk Score (GRS) that condenses the number of risk alleles of 9 Single Nucleotide Polymorphisms (SNPs) known to be associated with VTE. The genotyping and data collection procedures were performed using the protocols established in the Spanish ONCOTHROMB 12-01 study Muñoz et al., 2023.

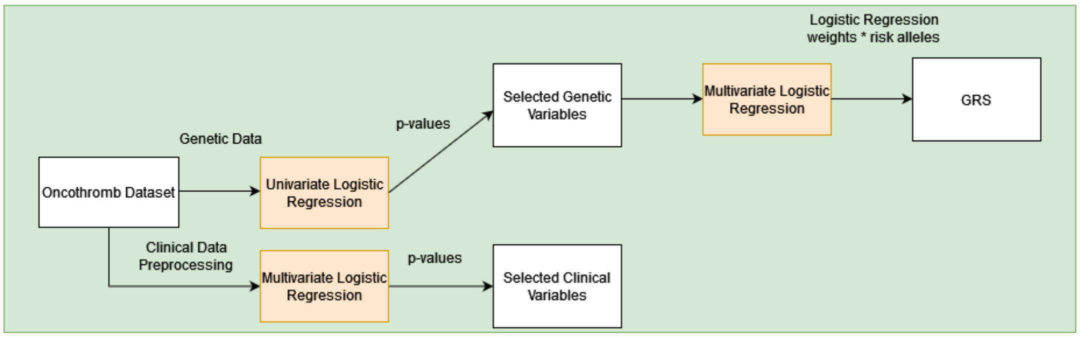

Logistic Regression
weights * risk alleles
Selected Genetic Variables
Multivariate Logistic Regression
GRS
p-values
Genetic Data
Oncothromb Dataset
Univariate Logistic Regression
Clinical Data Preprocessing
Multivariate Logistic Regression
p-values
Selected Clinical Variables

FIGURE 2.1: ONCOTHROMB score preprocessing pipeline.

The steps in which the score was built are described in Muñoz et al., 2023 and showed in Figure 2.1. The selection of the genetic variables is done using an univariate logistic regression, and variables with a p-value ≤ 0.25 were chosen. Next, a multivariate logistic regression was performed using the number of risk alleles for the selected genetic variants. The weight of each genetic variant was multiplied by the number of risk alleles, thus generating a GRS for each patient. On the other hand, a multivariate logistic regression was also performed with the aim of selecting the clinical variables, and those with p-value ≤ 0.25 were kept. If any missing values were found for the selected variables for a patient, that patient was excluded from the analysis. Finally, a logistic regression is performed to predict VTE using the selected clinical variables and the GRS.

2.2 Deep generative models

Deep generative models use deep learning techniques to generate synthetic data as similar as possible to real-world data. One of the most common types of deep generative models is the Generative Adversarial Network (GAN) (Goodfellow et al., 2014). These models where first developed to generate synthetic images, but several types of GANs were specifically developed for tabular data (Abedi et al., 2022).

2.2.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are a fundamental framework in deep learning, consisting of two key components: a generative model G and a discriminative model D. The generative model G aims to learn the underlying data distribution and generate synthetic samples that closely resemble real data. This is achieved through a non-linear mapping function, often implemented using deep learning typical architectures such as a multi-layer perceptron or using convolutional layers. On the other hand, the discriminative model D assesses the likelihood that a given sample originates from the training data rather than being generated by G.

The generator, denoted as $G(z; \phi_g)$, learns to map a prior noise distribution $p_z(z)$ to the data space in order to generate data samples x. The discriminator, $D(x; \phi_d)$, computes a probability score indicating whether the input data x is from the training data or the generator's distribution p_g. Note that, ϕ_g and ϕ_d represent the parameters of the generator (G) and the discriminator (D) networks, respectively. The generator and the discriminator are both trained simultaneously like a two-player min-max game, so the objective function of the GANs is:

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

The discriminator's objective is to maximize the objective function by correctly classifying real samples with a high probability (i.e., assigning a high value to $\log D(x)$) and generated samples with a low probability (i.e., assigning a low value to $\log(1 - D(G(z)))$). By doing so, the discriminator becomes more effective at distinguishing between real and generated samples.

On the other hand, the generator's objective is to minimize the objective function by generating samples that can fool the discriminator into classifying them as real. The generator aims to produce samples that result in a high value for $\log(1 - D(G(z)))$, indicating a low probability of being generated. By minimizing this term, the generator becomes more proficient at generating realistic samples that are more likely to be classified as real by the discriminator.

2.2.2 Conditional Generative Adversarial Networks

Conditional Generative Adversarial Networks (cGANs) (Mirza and Osindero, 2014) are an extension of GANs that incorporate additional conditions to guide the generation process. In cGANs, the generator is conditioned on extra input y, such as class labels or other auxiliary information, in addition to random noise. Figure 2.2 shows the architecture of a naive cGAN. The conditioning is performed by feeding y into the both the discriminator and generator as additional input layer. In the generator, the input noise $p_z(z)$ and y are combined in a joint hidden representation. The objective function is:

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

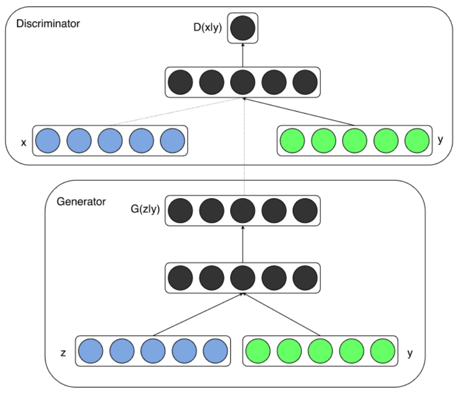


FIGURE 2.2: Conditional adversarial net from Mirza and Osindero, 2014.

The flexibility of the adversarial training framework allows for many ways to encode and incorporate the class labels into the discriminator and generator models. A good practice for a representation is explained in the paper (Denton et al., 2015) which proposes using an embedding layer followed by a fully connected layer with a linear activation that scales the embedding to the size of the image before concatenating it in the model as an additional channel or feature map.

The conditional nature of cGANs allows for a wide range of applications, including image synthesis, image-to-image translation, text-to-image synthesis, and more. By conditioning the generator on specific information, cGANs provide a powerful framework for controlled and tailored data generation tasks, with applications in computer vision, natural language processing, and other domains.

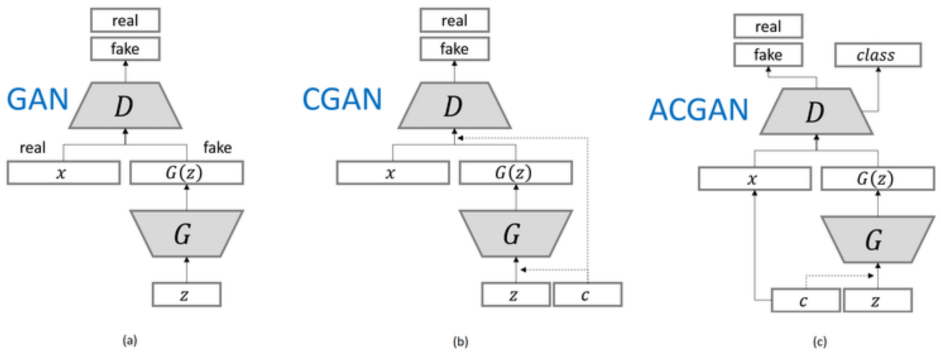


FIGURE 2.3: Comparison of the different GANs frameworks to generate synthetic data. Figure (a) represents the general architecture of the unconditional GAN. Figure (b) shows the cGAN architecture. Figure (c) shows the AC-GAN architecture.

2.2.3 Auxiliary Classifier Generative Adversarial Networks

The Auxiliary Classifier Generative Adversarial Network (AC-GANs) (Odena, Olah, and Shlens, 2017) combines the class conditional GAN with an auxiliary decoder that is tasked with reconstructing class labels. This architecture extends the conditional GAN by modifying the discriminator to predict the class label of an image instead of taking it as input. This change stabilizes training, enables the generation of high-quality large images, and ensures that the latent space representation is not dependent on the class label.

This extension is done by modifying the discriminator to contain an auxiliary decoder network that outputs the class label for the training data. In the AC-GAN, each generated sample is associated with a specific class label y in addition to the noise input z. The generator G utilizes both the class label and noise to produce fake images $X_{fake} = G(z|y)$. In practice, the discriminator and auxiliary classifier can be implemented as a single neural network model with two outputs. The discriminator D provides two probability distributions: one over the sources of the data (real or fake) and another over the class labels. The objective function for AC-GANs can be written as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x,y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z,c)))] + \lambda\mathcal{L}_{\text{class}}(D,G)$$

The AC-GAN objective function combines two main components: the log-likelihood of the correct source and the log-likelihood of the correct class. The first two components of the formula are identical to those in conditional GANs, capturing the discriminator's ability to correctly classify the source of real and generated samples.

The third term, represented by $\mathcal{L}_{\text{class}}(D,G)$, constitutes the classification loss. It incentivizes the discriminator to accurately classify both real and generated samples into their respective classes. The weight parameter λ determines the relative importance of the classification loss compared to the adversarial loss.

It is important to note that the specific form of the classification loss, $\mathcal{L}_{\text{class}}(D,G)$, can vary depending on the problem and the specific type of classification task at hand. Commonly used classification loss functions include cross-entropy or hinge loss, involving a comparison between predicted class labels and true class labels.

Overall, the AC-GAN objective function extends the conditional GAN framework by incorporating the log-likelihood of the correct class, encouraging the model to generate samples that not only resemble the real data source but also match the desired class labels. Figure 2.3 illustrates the distinctions among the three types of GANs presented. It is important to note that the main difference between the naive GAN architecture (Figure 2.3 (a)) is that its generator is not conditioned on any vector, whereas both cGAN and AC-GAN generators (Figure 2.3 (b) and Figure 2.3 (c)) are conditioned with class labels to generate images specific to the corresponding class. Additionally, the AC-GAN differs from cGAN in that its discriminator predicts the class label of the generated image, which is then compared with the class of the real image

2.3 GANs for tabular data

Tabular GANs, also known as GANs for tabular data, are a variation of GANs specifically designed to generate structured tabular data. While traditional GANs are primarily used for generating images, text, or audio, tabular GANs adapt to the unique challenges of generating structured data in a tabular format, such as spreadsheets, databases or tables.

Generating tabular data with GANs presents several challenges compared to other data types. The key obstacles include:

1. High Dimensionality: Tabular data often has a high number of features or columns, which can make learning the underlying patterns and generating coherent data more difficult.

2. Categorical and Numerical Variables: Tabular data can contain a mix of categorical and numerical variables. Categorical variables require special handling to ensure proper generation and preservation of their properties, such as one-hot encoding or embedding techniques.

3. Data Distribution and Relationships: Tabular data may have complex distributions and intricate relationships between variables. Capturing these dependencies accurately during the training process is crucial for generating realistic and meaningful samples.

4. Missing Data and Outliers: Real-world tabular data often contains missing values or outliers. Generating synthetic data that handles missing data appropriately and does not produce unrealistic outliers is a challenge for tabular GANs.

5. Highly imbalanced categorical columns. Missing a minor category only causes tiny changes to the data distribution that is hard to be detected by the discriminator. Imbalanced data also leads to insufficient training opportunities for minor classes.

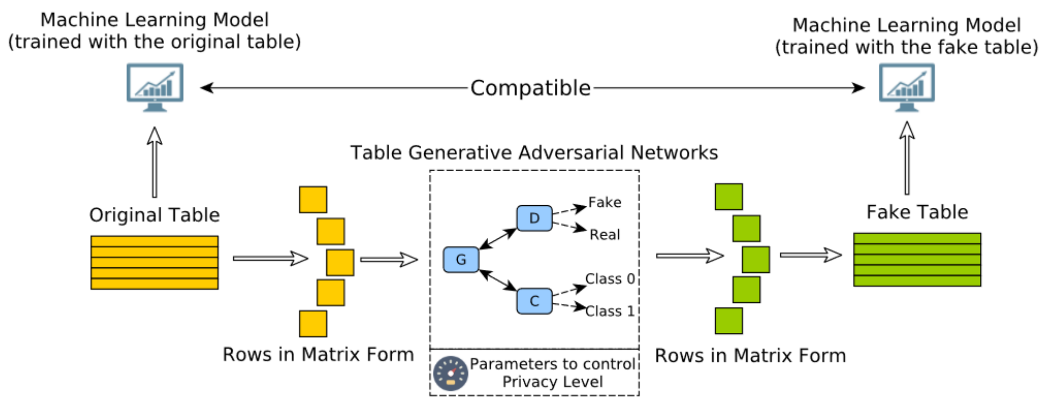

Machine Learning Model (trained with the original table)
Compatible
Machine Learning Model (trained with the fake table)
Table Generative Adversarial Networks
Original Table
Fake Table
D
Fake
Real
G
C
Class 0
Class 1
Rows in Matrix Form
Parameters to control Privacy Level
Rows in Matrix Form

FIGURE 2.4: Overall workflow of the TableGAN (Park et al., 2018).

2.3.1 TableGAN

TableGAN, as introduced by Park et al., 2018, is a specialized method for generating synthetic tables that encompass categorical, discrete, and continuous values. It builds upon the Deep Convolutional GAN (DCGAN) architecture proposed by Radford, Metz, and Chintala, 2016, but introduces three neural networks: a generator, a discriminator, and an additional classifier network. The inclusion of the classifier network plays a crucial role in maintaining the semantic integrity of synthetic records. By predicting labels for these records, the classifier network ensures

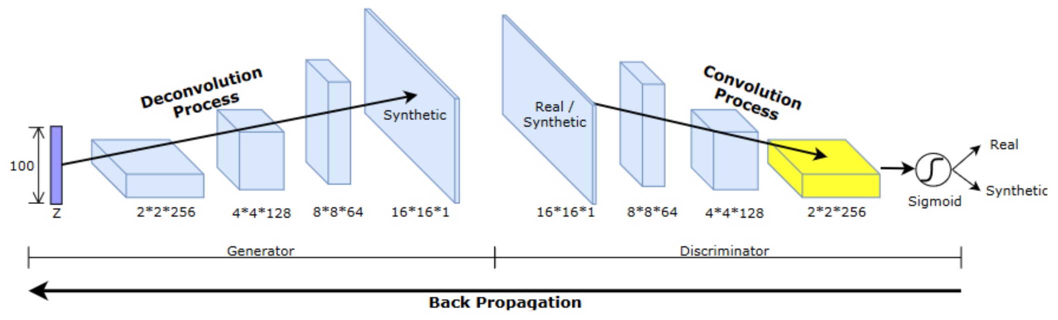


FIGURE 2.5: TableGAN architecture. Note that the classifier is omitted due to space limitations but shares the same neural network structure as the discriminator (Park et al., 2018).

that only valid records are generated, preventing the generation of records that are not possible with the original data. During the training process, the classifier network learns the semantics from the original table, enhancing the plausibility of the generated records and minimizing the chances of fabricated tables being detected. Figure 2.4 illustrates the general workflow of TableGAN, while Figure 2.5 provides a detailed depiction of the generator and discriminator architectures. TableGAN architecture includes a generator and a discriminator. The generator performs deconvolution operations to generate records, while the discriminator uses convolution operations to classify records as real or fake. The final loss, obtained after applying sigmoid activation, is used to update the generator. The dimensions of the input latent vector (z) and intermediate tensors should consider the number of attributes present in the table.

TableGAN adopts the loss function from the DCGAN in order to train the discriminator but employs two additional specialized loss functions for the generator, namely information loss and classification loss, which are tailored to the table synthesis process and contribute to the generation of realistic synthetic tables. The information loss function objective is to compare the first-order and second-order statistics (i.e., mean, standard deviation) of the features of real and synthetic records. This loss is also controlled using hinge in order to balance the trade off between privacy level and more similar statistics of fake and real data.

The classification loss measures the discrepancy between the label of a generated record and the label predicted by the classifier trained on real data. In cases where there are multiple labels, the classifier neural network can be expanded to enable multi-task learning. This involves incorporating multiple final sigmoid activations that share intermediate layers. Each sigmoid activation is trained to predict a specific label based on the shared intermediate layers, allowing for the classification of multiple labels simultaneously.

2.3.2 Conditional Tabular GAN

Conditional tabular GAN (CTGAN) (Xu et al., 2019) proposes to use a conditional generator and new techniques in order to overcome the need to simultaneously model discrete and continuous columns, the multi-modal non-Gaussian values within each continuous column, and the severe imbalance of categorical columns.

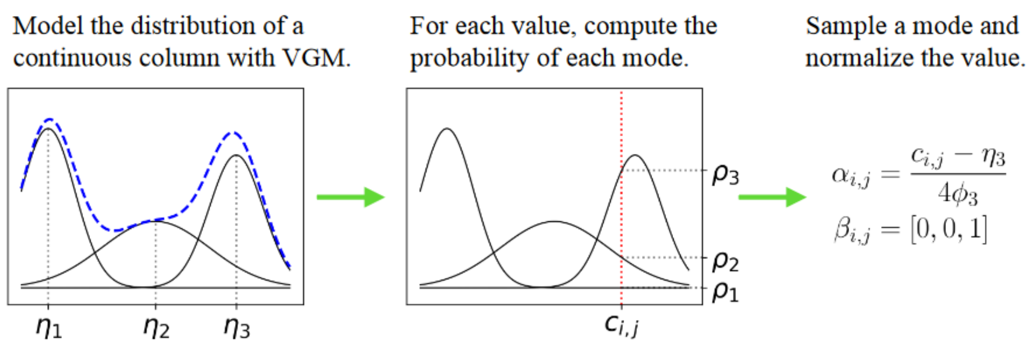

Model the distribution of a continuous column with VGM.
For each value, compute the probability of each mode.
Sample a mode and normalize the value.
η_1
η_2
η_3
$c_{i,j}$
ρ_3
ρ_2
ρ_1
$\alpha_{i,j} = \frac{c_{i,j} - \eta_3}{4\phi_3}$
$\beta_{i,j} = [0, 0, 1]$

FIGURE 2.6: An example of mode-specific normalization from Xu et al., 2019.

Mode-specific Normalization:

Continuous values with arbitrary distributions are challenging to represent directly, unlike discrete values that can be easily encoded as one-hot vectors. To address this issue and handle columns with complex distributions, mode-specific normalization is introduced. Figure 2.6 illustrates the workflow of this technique applied to a continuous column (C_i).

The first step involves utilizing a variational Gaussian Mixture model (VGM) to estimate the number of modes (m_i) for each continuous column and fit a Gaussian mixture. Next, Kernel Density Estimation (KDE) is employed to estimate the probability densities.

For each value ($c_{i,j}$) in the continuous column, the probability densities (ρ_i) of $c_{i,j}$ coming from each mode are computed. This step provides insights into which mode each value is likely to belong to.

The final step entails sampling one mode based on the probability density and using the sampled mode to normalize the value. In figure 2.6, for example, the third mode is selected based on probability densities, resulting in the representation of $c_{i,j}$ using a one-hot encoder vector ($\beta_{i,j} = [0, 0, 1]$), indicating the third mode. Additionally, a scalar value $\alpha_{i,j} = \frac{c_{i,j} - \eta_3}{4\phi_3}$ is calculated, where ϕ_3 represents the standard deviation of the third mode. This scalar value provides a normalized representation of the original value relative to the selected mode.

To represent a single row of data, the approach involves concatenating the representations of continuous values with the one-hot encoding vectors of the discrete columns.

In other words, for each continuous value, its corresponding representation (the mode-specific normalization discussed earlier) is computed. These representations are concatenated together. Similarly, for each discrete column, the discrete value is encoded using a one-hot encoding scheme, resulting in a binary vector representing the category of the discrete value. This concatenated representation captures both the continuous information and the discrete category information for the row.

Conditional Generator and Training-by-Sampling:

Class imbalance poses a problem where the training data lacks sufficient representation of the minority class. When training if data is randomly sampled, the rows belonging to the minority category are underrepresented, leading to potential issues in training the generator effectively. The solution presented is formed by three

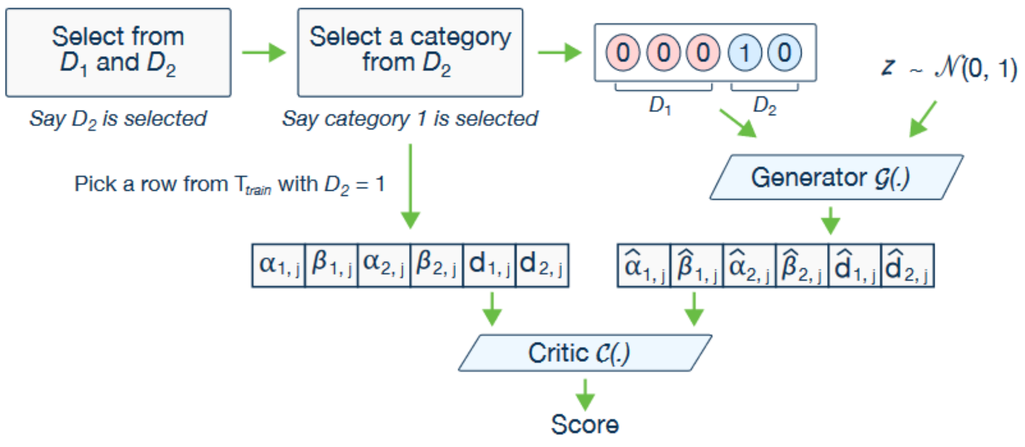

Select from D_1 and D_2
Say D_2 is selected
Select a category from D_2
Say category 1 is selected
0 0 0 1 0
D_1
D_2
$z \sim \mathcal{N}(0, 1)$
Pick a row from T_{train} with $D_2 = 1$
Generator $\mathcal{G}(.)$
$\alpha_{1,j}$ $\beta_{1,j}$ $\alpha_{2,j}$ $\beta_{2,j}$ $d_{1,j}$ $d_{2,j}$
$\hat{\alpha}_{1,j}$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

FIGURE 2.7: Example of conditional generator workflow from Xu et al., 2019.

key elements: the conditional vector, the generator loss an the training-by-sampling method. The conditional generator can generate synthetic rows conditioned on one of the discrete columns. With training-by-sampling, the conditional 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 shows the scheme of a concrete sampling of this technique.

The generator takes the conditional vector, which specifies the desired category to sample, as input. To ensure that the generator produces samples matching this condition, its loss is modified by incorporating cross-entropy between the desired one-hot encoding and the generated one-hot encoding. Consequently, as the training progresses, the generator gradually learns to replicate the given condition accurately in every generated row.

The last mechanism introduced is the training-by-sampling. The purpose of this method is to properly sample the conditional vector and training data in order to help the model evenly explore all possible values in discrete columns. In order to do that a discrete column is randomly selected with equal probability. Then, a probability mass function (PMF) is constructed across the range of values of the column selected, such that the probability mass of each value is the logarithm of its frequency in that column. A value is selected randomly according this PMF and the conditional vector of this selection is created as input to the generator.

Network Structure:

The generator and discriminator are composed of fully-connected layers. The synthetic row is generated using the function *tanh* to generate the scalar values α_i and gumbel softmax (Jang, Gu, and Poole, 2017) to generate the indicators β_i and the discrete values. The model is trained using Wasserstein distance with gradient penalty (Gulrajani et al., 2017) and Adam as optimizer. The PacGAN framework (Lin et al., 2018) with 10 samples in each pac is used in order to help the GAN to represent all the modes or diverse patterns present in the real data distribution. This technique refers to the grouping of multiple generated samples into a single pac (pack) so during training, instead of generating one sample at a time, the generator generates 10 samples in a pack. The discriminator is then trained to distinguish between real data and these packed samples.

2.3.3 Tabular Variational Autoencoder

The Tabular Variational Autoencoder (TVAE) is also discussed in the work by Xu et al., 2019. To make it suitable for tabular data, the same preprocessing techniques as those used in CTGAN are applied. The TVAE utilizes two neural networks to model the distributions and trains them using the evidence lower-bound (ELBO) loss (Kingma and Welling, 2022).

The decoder neural network generates a joint distribution assuming that $\alpha_{i,j}$ follows a Gaussian distribution with varying means and variances and all $\beta_{i,j}$ and $d_{i,j}$ variables follow a categorical Probability Mass Function (PMF). The encoder architecture is similar to a conventional VAE (Kingma and Welling, 2022).

2.3.4 CTAB-GAN

CTAB-GAN (Zhao et al., 2021) tries to improve the previous tabular data synthesizers by addressing some of the limitations: encoding mixed data type of continuous and categorical variables, efficient modeling of long tail continuous variables and increased robustness to imbalanced categorical variables along with skewed continuous variables. This model combine the strengths of prior art, such as classifier loss, information loss, effective encoding of the multi-modal complex distributions, and conditional vector.

To solve the problem of imbalanced datasets the conditional generator and the training-by-sampling methods from the CTGAN are used. However, to improve the quality of the generator information loss and classification loss terms are added. The information loss measures statistical differences between synthetic and real data, while the classification loss helps to increase the semantic integrity of synthetic data by adding an auxiliary classifier in parallel to the discriminator in order to predict a label for each synthetic sample as in TableGAN.

Finally, the mode-specific normalization from CTGAN is also used to encode the continuous variables. This technique is also extended to work with mixed types, that is variables that contains discrete and continuous values. The generator and discriminator follow a similar structure as the TableGAN using convolutional layers. To address the challenge of encoding continuous values in the tail of long tail distributions a logarithm transformation is applied to the initial data. This pre-processing step helps alleviate the difficulty faced by VGM in encoding such values.

2.3.5 CopulaGAN

The CopulaGAN implemented in Team, 2022 is an extension of the CTGAN that combines the architecture of the CTGAN with a preprocessing part using Gaussian Copula transformations (a copula is a function that represents the correlation or dependence between variables with independence of their marginal distributions). The process consists of the following steps:

1. Determine the distribution of each non-categorical variable: Before applying the Gaussian Normalizer, it is necessary to determine the distribution of each non-categorical variable in the input dataset. Common distributions for continuous variables include Gaussian (normal), exponential, uniform, etc. The choice of distribution depends on the characteristics of the variable and the underlying data.

2. Gaussian Normalizer transformation: Once the distribution of each variable is identified, the Gaussian Normalizer can be applied. The Gaussian Normalizer transforms each variable to a standard normal space, where the variable follows a standard Gaussian (normal) distribution with mean 0 and standard deviation 1. First, the Cumulative Distribution Function (CDF) of the corresponding distribution for each variable is applied. The CDF converts the values to their respective percentiles within the distribution. The transformed values obtained from the CDF are then passed through the inverse CDF (also known as the quantile function or percent-point function) of a standard normal distribution. This step maps the transformed values to the standard normal distribution. The result of this transformation is that the variables will have a mean of 0 and a standard deviation of 1, which simplifies the modeling process for CTGAN.

3. Fit CTGAN with the transformed table.

4. Sample using CTGAN.

5. Reverse the transformation using CDF and inverse CDF: After sampling from CTGAN and obtaining synthetic data, you need to reverse the previous transformation to obtain values in the original scale. This involves applying the CDF of a standard normal distribution and then inverting the CDF of the distribution that corresponds to each variable.

The code for CopulaGAN and Gaussian Normalizer can be found in the sdv library Team, 2022.

Chapter 3

Experimental Setup

3.1 Experiments

In this section, we will provide an overview explanation of the experiments conducted and the objectives associated with each one. We conducted four main experiments as follows:

Firstly, we replicated the paper on the ONCOTHROMB score (Muñoz et al., 2023) using Python. The aim was to reproduce the findings and results presented in the paper.

Next, we conducted a benchmarking study involving different state-of-the-art tabular Generative Adversarial Networks that were adapted to our specific dataset. The purpose of this experiment was to evaluate and compare the performance of these GANs on our dataset.

Then, we employed transfer learning techniques to train a model using synthetic data and subsequently fine-tuned it using real data. This experiment aimed to investigate the effectiveness of transferring knowledge from synthetic data to improve the performance of the model on real data.

In the latest experiment, we have modified the CTGAN architecture to incorporate target encoding. The main objective of this experiment is to enhance the ability of CTGAN to generate large categorical variables effectively.

3.1.1 Dataset

As already mentioned in section 2.1, the dataset used in this study comes from the ONCOTHROMB12-01 cohort, and it comprises clinical and genetic information on 390 patients with 19 clinical features plus 54 genetic features representing the number of risk alleles present in a set of genetic variants known to be linked to VTE. The VTE variable serves as the target, represented as a binary variable indicating whether VTE is present or not for each patient. No imputation is used to deal with the missing values; instead, patients with missing values in key variables are dropped. The same four features used in the original ONCOTHROMB score (mentioned in Chapter 2.1) are the ones used in experiments 3.1.3 and 3.1.4.

3.1.2 Paper reproduction

The objective of this experiment is to replicate the methodology described in the paper and establish a baseline for comparing our own results. Additionally, we aim to provide a Python implementation of the methodology, which was originally implemented using medical software and R. The methodology involves the following steps to derive the ONCOTHROMB score:

1. Genetic Variable Selection: Only the genetic variables are chosen, and p-values are computed to determine the importance of each genetic variable in predicting VTE.
2. Genetic Risk Score Computation: A Genetic Risk Score is calculated for each patient using the selected genetic variables.
3. Clinical Data Selection and Preprocessing: Only the clinical data is selected and preprocessed.
4. Multivariate Logistic Regression: A multivariate logistic regression analysis is conducted with the clinical data to compute p-values and identify the significant clinical variables for our prediction task.
5. Backwards Analysis with AIC: A backwards analysis is performed using Akaike Information Criteria (AIC) to ensure that the selected variables are statistically significant for our prediction task, considering both the clinical variables and the genetic risk score.
6. Logistic Regression Training: Finally, a logistic regression model is trained using the selected variables to predict VTE.
7. Model Evaluation: The performance of the model is evaluated using ROC (Receiver Operating Characteristic) curve and PRC (Precision-Recall Curve).

3.1.3 Tabular GANs performance

In this experiment, we aim to assess the performance of four modern tabular GANs: CTGAN, CTABGAN, TVAE, and CopulaGAN. The primary objective is to investigate their effectiveness in generating synthetic data. Specifically, we generate synthetic data for the variables that were selected after the backwards analysis, ensuring that only the most relevant variables are included.

To evaluate the quality of the synthetic data, we employ a machine learning classifier. The classifier is trained using synthetic data and tested using both the real data and the test generated synthetic data. The performance of the classifier in predicting VTE is then assessed. This evaluation enables us to determine whether the synthetic data performs well in capturing the underlying patterns and characteristics necessary for accurate VTE prediction. By comparing the performance of the classifier on real and synthetic data, we can gain insights into the efficacy of the tabular GANs in generating data that accurately represents the predictive task at hand.

3.1.4 Transferring knowledge from synthetic data to the ONCOTHROMB cohort

In the ONCOTHROMB cohort, each patient contains clinical and genetic information that can be distilled into a small set of aggregated variables. The main hypothesis of this work is that, by means of deep learning generative methods, a large set of compatible data can be synthesized and used to induce a learning bias that can be transferred to the real setting, thus improving the generalization and operational points of the original ONCOTHROMB score.

For that purpose, we first generate synthetic tabular data utilizing generative models. It is crucial to ensure that the synthetic data closely resembles the distribution of real data and, as a result, performs well in a classification task. Therefore,

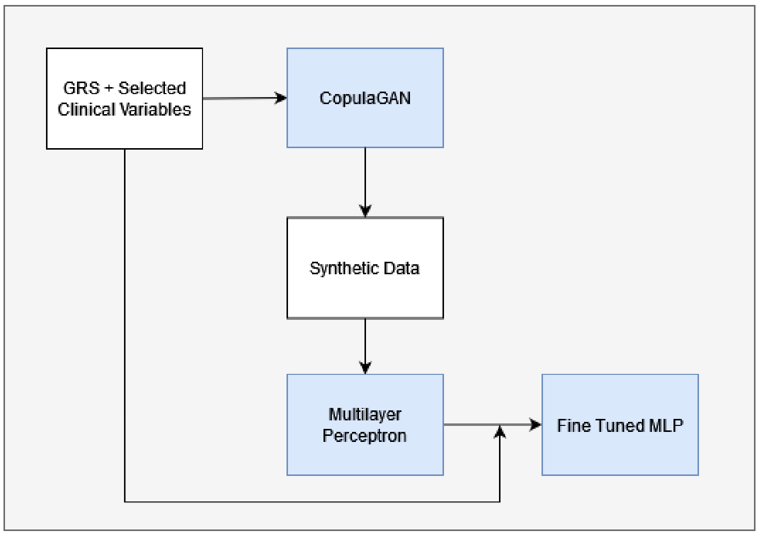


FIGURE 3.1: Methodology proposal

we propose to train a multilayer perceptron (MLP) only with synthetic data and assess its performance. Lastly, a fine-tuning process of the MLP is carried out using real-world data. Figure 3.1 summarizes the proposed methodology.

3.1.5 Enhancing CTGAN with target encoding for efficient modeling of large categorical variables

This experiment is not directly related to VTE, but it aims to enhance the architecture of GANs for tabular data. We observed that CTGAN and other tabular GANs utilize one-hot encoding to represent categorical variables. However, one-hot encoding may not always be the optimal choice for representing categorical variables.

The objective of this experiment is to introduce the capability for CTGAN to encode categorical variables using target encoding. We aim to evaluate whether this change allows CTGAN to synthesize large categorical variables more effectively.

To conduct this experiment, we have decided not to use the ONCOTHROMB dataset, as it does not contain large categorical variables. Instead, we are utilizing the Kickstarter Projects dataset (Kemical, 2020). This dataset consists of crowd-funding projects, and the goal is to predict the funding goal for each project. The categorical features in this dataset include the crowd-funding type, country, state, and currency.

After implementing the necessary modifications to the CTGAN architecture, we proceeded to evaluate the results by generating synthetic data using both one-hot encoding and target encoding. To assess the quality of the generated data, we employed a logistic regression classifier and compared its performance across the synthetic data encoded with different methods, as well as a visual analysis of the generated data compared to real data.

Chapter 4

Reproducing the ONCOTHROMB Score Research Paper: A Comprehensive Analysis and Validation

4.1 Challenges

Reproducing the paper authored by Muñoz et al., 2023 poses to us several challenging obstacles. Firstly, the data provided is not identical to the original, and certain methodological details are missing from the paper. Our main challenge lies in computing the p-values, as the *statsmodels* Python library yields different results compared to their reported values. To address this, we opted to employ a permutation test to compute the p-values and compare all three methods.

Another significant challenge arises after preprocessing the dataset and consolidating the genetic variables into a genetic risk score. This consolidation of nine genetic variables into a single score and the fact that categorical variables have a limited number of categories results in the emergence of duplicate entries in the dataset. To account for these issues, we decided to conduct subsequent experiments using both datasets, one with duplicates and one with duplicates removed. The original paper lacks details on this aspect.

Another disparity we observed is that the p-values table from ONCOTHROMB includes some genetic variables with values below 0.25, yet these variables were not selected. We speculate that this discrepancy could be attributed to null values within those variables or previous clinical knowledge suggesting that they are not associated with VTE.

In conclusion, reproducing this paper is far from a straightforward task, requiring us to make non trivial assumptions and modifications.

4.2 Reproducing the paper

4.2.1 Genetic Risk Score

The first step in our analysis is to select all the genetic variables and calculate the p-values by training a univariate logistic regression model for each variable. This logistic regression model attempts to predict VTE using only one genetic feature at a time. We employ two different approaches to accomplish this task.

The first approach involves using the *statsmodels* Python package to train the logistic regression models. This package also calculates the p-values, which indicate the statistical significance of each variable in predicting VTE. Based on the paper's guidelines, we consider variables with p-values lower than 0.25 to be relevant. However, since the computation of p-values in logistic regression is not extensively documented or explicitly defined in the *statsmodels* package, we employ a more computational approach to compute these p-values.

Our approach is based on performing a permutation test to calculate the p-values. This involves the following steps:

1. We select one genetic variable and train a logistic regression model with VTE as our target variable.
2. We compute the observed AUC-ROC (Area Under the Receiver Operating Characteristic) score, which measures the model's predictive performance.
3. We conduct a permutation test by randomly permuting the target values while keeping the genetic features in the same order.
4. We calculate the AUC-ROC score for the permuted target.
5. We repeat the permutation test 1000 times to obtain a distribution of permuted AUC scores.
6. Compute the p-value as the proportion of permuted AUC values that are as extreme or more extreme than the observed AUC. A small p-value indicates that the observed AUC is unlikely to occur by chance alone, suggesting evidence against the null hypothesis (the hypothesis that there is no significant difference)).

$$p\_value = \frac{\sum_{i=1}^{n\_permutations} (auc\_permuted[i] \geq auc\_obs)}{n\_permutations}$$

7. We repeat this process for all the genetic variables.
8. Finally, we select all genetic variables with p-values lower than 0.25 as they are deemed statistically significant for predicting VTE.

By following these steps, we aim to identify the genetic variables that exhibit a significant association with VTE based on their respective p-values.

The next step in our analysis involves computing the Genetic Risk Score (GRS). To do this, we utilize the number of risk alleles associated with each of the selected genetic features for each patient. We train a logistic regression model using these numbers of risk alleles as the independent variables, with VTE as the target variable.

Once the logistic regression model is trained using the risk alleles, we can compute the GRS for each patient i using the following formula:

$$GRS_i = (C_i \cdot X_i) + b$$

In this equation, the variable C_i represents a vector that holds the coefficients associated with each genetic variable. These coefficients are obtained from a logistic regression model that is trained based on the count of risk alleles. The variable b represents the intercept value derived from fitting this logistic regression model. Lastly, the variable X_i represents a vector that contains the count of risk alleles for each selected genetic variable corresponding to the patient.

By multiplying each coefficient C_i with the respective number of risk alleles for each genetic variable and then adding the intercept value, we obtain the GRS for each patient. The GRS represents a combined score that incorporates the genetic risk associated with multiple genetic variables, as determined by the logistic regression model trained on the risk alleles.

This step allows us to quantify the overall genetic risk for each patient based on the specific combination of risk alleles they possess across the selected genetic variables.

4.2.2 Clinical variables selection

The clinical variables require some preprocessing steps. The different types of tumors need to be classified into categories such as low risk, high risk, and very high risk. This classification allows for better risk stratification based on tumor characteristics. Stomach and pancreas tumors are considered very high risk; lung, lymphoma, gynecological, bladder and testicular are considered high risk; and colorectal low risk.

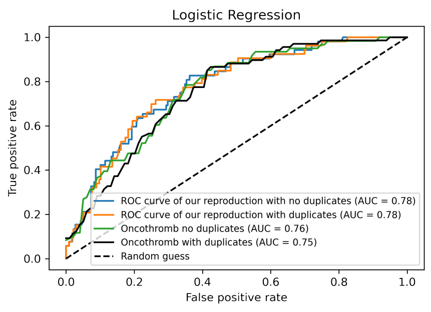
Next, thresholds need to be established for the variables *leukocytes* and *platelets* to determine whether the values are considered risky or not. As per the original paper, a threshold of $350,000$ is set for platelets, while a threshold of $11,000$ is set for leukocytes.

Using these thresholds, values above $350,000$ for platelets are considered risky, indicating a high platelet count. Similarly, values above $11,000$ for leukocytes are considered risky, indicating a high leukocyte count.

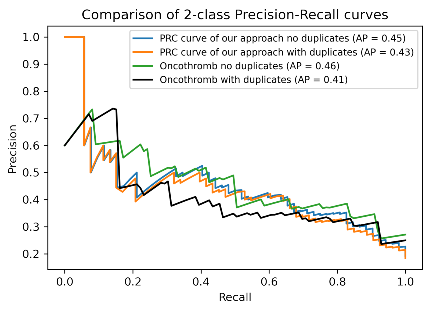
Lastly, the variable *BMI* (Body Mass Index) is also categorized into two classes. One class combines overweight and obese individuals, while the other class combines those who are normal weight or underweight. Additionally, the stage of the cancer is classified into four stages. This categorization provides further insight into the progression and severity of the cancer, enabling a more comprehensive evaluation of the data.

Next, in line with the methodology outlined in the paper, a multivariate logistic regression analysis is conducted to calculate the p-values and select the clinically significant variables. In this step, we leverage the *statsmodels* package to compute the p-values because we observe that the p-values obtained through the permutation test and *statsmodels* are found to be highly similar.

Once again, variables with p-values lower than 0.25 are chosen, indicating their statistical significance in predicting the outcome of interest. By employing this approach, we can identify and select the clinical variables that demonstrate a strong association with the outcome, based on their respective p-values.



(a) ROC curve



(b) PR curve

FIGURE 4.1: We compared the ROC and PR curves obtained from our replication of the paper. The original ONCOTHROMB score is calculated by considering all the genetic and clinical variables mentioned in the paper. It's worth noting that both approaches include scenarios with duplicated and non-duplicated cases.

4.2.3 Model

To ensure that all the selected clinical variables and the GRS score are appropriate for the analysis, a backward analysis is conducted using the Akaike Information Criterion (AIC). This criterion helps evaluate the goodness of fit of a statistical model while penalizing for model complexity.

To perform the backward analysis using AIC, the following steps are taken:

1. Start with the model that includes all the selected clinical variables and the GRS score.

2. Iteratively remove one variable at a time and calculate the AIC for the reduced model.

3. Compare the AIC of the reduced model with the full model. AIC is calculated by taking twice the negative log-likelihood of the model plus two times the number of parameters in the model.

$$AIC = -2 \cdot \text{LogLikelihood} + 2 \cdot \text{number of parameters}$$

4. If the AIC of the reduced model is lower than that of the full model, indicating a better fit with less complexity, the variable is removed from the model.

5. Repeat steps 2-4 until further removal of variables does not lead to a decrease in AIC.

By employing the backwards analysis with AIC, we ensure that the final model includes only the most relevant variables that provide the best balance of model fit and simplicity.

After selecting all significant variables, we observed that since we select a subset of genetic variables into a single risk score and the clinical variables selected are categorical with few categories, our final dataset contains duplicated rows. The paper does not provide guidance on how to proceed in this scenario. Therefore, we opted to train two versions of the final model: one logistic regression model that excludes the duplicates and another that includes them. The final model is a logistic regression, where the selected variables are used as features to predict the occurrence of VTE.

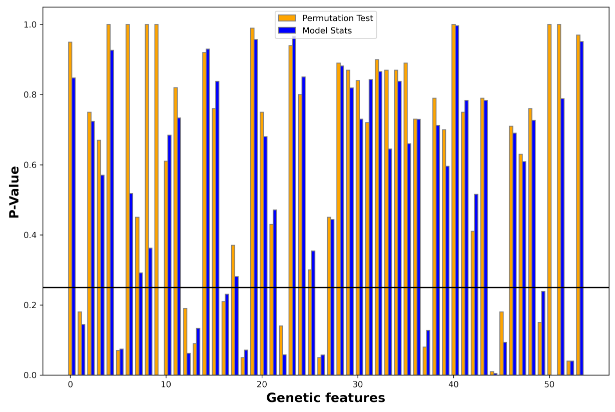


FIGURE 4.2: Comparison of the p-values obtained using the permutation test and the *statsmodels* package.

4.2.4 Results

To assess the effectiveness of our approach, we conducted a comparison between the ROC-AUC and the PR curve of our results and those presented in the original paper. The outcomes are illustrated in Figure 4.1. It is evident that there are some variances between the obtained curves. The ROC curve appears to perform better with our approach, while the PR curve, specifically without duplicates, slightly favors the original implementation described in the paper. It is important to note that even when selecting the exact same variables as mentioned in the paper, the issue of having duplicated rows still persists in the final dataset.

We conducted an analysis and comparison of the selected variables and the computation of p-values for permutation test and *statsmodels*. Figure 4.2 illustrates the p-values for each genetic variable obtained from permutation test and *statsmodels*. The horizontal black line represents the threshold of 0.25, so any variables with p-values below this threshold are considered significant and are the selected ones for computing the GRS. Note that both methods result in the same selection of variables. This can be observed in Figure 4.2, where no p-value is below the threshold for one approach and above for the other. The genetic and clinical variables that were selected are displayed in Table 4.2 and Table 4.1, respectively.

Upon examination of Figure 4.2, we observed that the permutation test and the *statsmodels* approach yield very similar p-values, leading to the selection of the same variables. However, when comparing our selected variables with those in the paper, we identified some differences, as presented in Table 4.2. The permutation test and *statsmodels* selected a larger number of variables compared to the paper's methodology, potentially contributing to the differences observed in Figure 4.1. Note that the variable *rs*6025 is selected, even though its p-value is greater than 0.25. This is because the paper mentions previous medical research that links this genetic variable to VTE. Additionally, we discovered that the variables rs4149755, rs2289252, and rs2036914 have p-values lower than 0.25 for all three approaches. However, the paper did not select these variables, and no further information or reasoning is provided regarding this exclusion.

Furthermore, in terms of the selected clinical variables, Table 4.1 demonstrates variations in the selection of these variables as well. Lastly, Figure 4.3 shows the selected variables used to compute the ONCOTHROMB score.

Paper	Permutation Test	Statsmodels
tnm stage detailed	*tnm stage detailed*	*tnm stage detailed*
bmi category	-	-
primary tumour simplified	*primary tumour simplified*	*primary tumour simplified*
-	*family background vte*	*family background vte*
-	*diabetes mellitus*	*diabetes mellitus*
-	*dyslipidemia*	*dyslipidemia*

TABLE 4.1: Union of selected clinical variables for three different approaches. The "-" symbol indicates that the variable is not selected due to a higher p-value.

Paper	Permutation Test	Statsmodels	P-values
rs4524	rs4524	rs4524	0.106 | 0.141 | 0.062
rs6025	rs6025	rs6025	0.403 | 0.382 | 0.291
rs2232698	rs2232698	rs2232698	0.089 | 0.070 | 0.074
rs2227631	-	-	0.143 | 0.419 | 0.281
rs268	rs268	rs268	0.234 | 0.053 | 0.071
rs11696364	rs11696364	rs11696364	0.015 | 0.010 | 0.004
rs5110	rs5110	rs5110	0.117 | 0.171 | 0.238
rs6003	rs6003	rs6003	0.125 | 0.126 | 0.133
rs169713	rs169713	rs169713	0.072 | 0.092 | 0.128
-	rs3136516	rs3136516	0.267 | 0.230 | 0.220
-	rs4149755	rs4149755	0.162 | 0.188 | 0.058
-	rs2289252	rs2289252	0.035 | 0.051 | 0.057
-	rs6034465	rs6034465	0.471 | 0.226 | 0.093
-	rs2036914	rs2036914	0.061 | 0.034 | 0.040

TABLE 4.2: This table displays the union of the selected genetic variables for all three approaches. The "-" symbol indicates that the variable did not have a p-value lower than 0.25 and thus was not selected. The P-values column represents the respective p-values for each approach, following the same order as the columns: paper, permutation test, and *statsmodels*.

Paper	Permutation Test	Statsmodels
GRS	GRS	GRS
tnm stage detailed	*tnm stage detailed*	*tnm stage detailed*
bmi category	-	-
primary tumour simplified	*primary tumour simplified*	*primary tumour simplified*
-	*family background vte*	*family background vte*

TABLE 4.3: The final variables selected for each approach. It is important to note that the computation of the GRS relies on the selected genetic variables. Therefore, even though the GRS is selected in all cases, the values of the GRS will vary depending on the specific genetic variables chosen. During the backwards analysis using AIC, diabetes mellitus and dyslipidemia were discarded.

Chapter 5

Analysis of Tabular GANs: A Benchmarking Study on the ONCOTHROMB Dataset

5.1 Tabular GANs performance analysis

In this experiment, we trained four state-of-the-art tabular GANs: CTGAN, TVAE, CopulaGAN, and CTABGAN. The first three models were trained using the code provided in the sdv library (Team, 2022), while CTABGAN is based on the paper Zhao et al., 2021 and was adapted to suit our specific problem.

The training dataset for the generative models consisted of five different variables: *VTE*, *GRS*, *TNM stage detailed*, *BMI category*, and *primary tumor simplified*, that is the dataset obtained after the preprocessign applied in Muñoz et al., 2023. It is important to mention that all variables, except for GRS, are categorical, while GRS is a continuous variable.

For consistency, we aimed to maintain a similar setup across all models during training. Therefore, each model was trained for 1000 epochs, utilizing a batch size of 30, and performing 5 discriminator steps for every generator update.

To evaluate the quality of synthetic data, first we perform a visual evaluation of the models by examining the cumulative sums per feature for both real and synthetic data. This helps us visually assess any differences or similarities between the two datasets.

Next, we evaluate the models using logistic regression to determine if the synthetic data is capable of classifying VTE patients as effectively as the real data. To accomplish this, we split the synthetic data into training and testing sets. We then train a logistic regression model using the training data and evaluate its performance using both the test data and the real data.

By employing logistic regression and comparing the performance of models trained on synthetic and real data, we can determine the effectiveness of the synthetic data in accurately classifying VTE patients.

	ONCOTHROMB	CTGAN	TVAE	CopulaGAN	CTABGAN
Test real data non-duplicates	0.46/0.76	0.44/0.76	0.47/0.77	0.48/0.76	0.48/0.77
Test real data duplicates	0.41/0.75	0.48/0.77	0.44/0.77	0.44/0.76	0.44/0.77
Test synthetic data	0.52/0.81	0.52/0.76	0.43/0.79	0.56/0.83	0.58/0.82

TABLE 5.1: Results of AUC-PRC/AUC-ROC for the baseline model and tabular GANs. Models are evaluated using test synthetic data and real data with and without duplicates.

5.1.1 Results

Table 5.1 presents the results obtained from the analysis. It is evident that all methods perform reasonably well in generating synthetic data. CopulaGAN, CTGAN, and CTABGAN exhibit similar performance, while TVAE appears to perform relatively worse. In Figure 5.1, we can observe the cumulative sums per feature while figure 5.2 displays the AUC curve and PRC for CTGAN synthetic data trained on logistic regression. Likewise, Figure 5.3 and 5.4 depict the same experiments conducted using the TVAE architecture. Moving forward, Figure 5.5 and 5.6 showcase the evaluation of CopulaGAN, and finally, Figure 5.7 and 5.8 demonstrate the corresponding graphics for CTABGAN.

Upon examining the distributions of the generated features, it becomes evident that all architectures produce synthetic data that closely resembles the real data. For instance, CTGAN and CopulaGAN display similarities due to using the same architecture. Furthermore, the classification curves exhibit variations, with CTGAN being particularly noteworthy. The test synthetic data generated by CTGAN shows a behavior that closely aligns with the real data. Overall, all architectures appear to achieve the objective of generating synthetic data. However, for our specific problem, CopulaGAN and CTGAN seem to be the best choices.

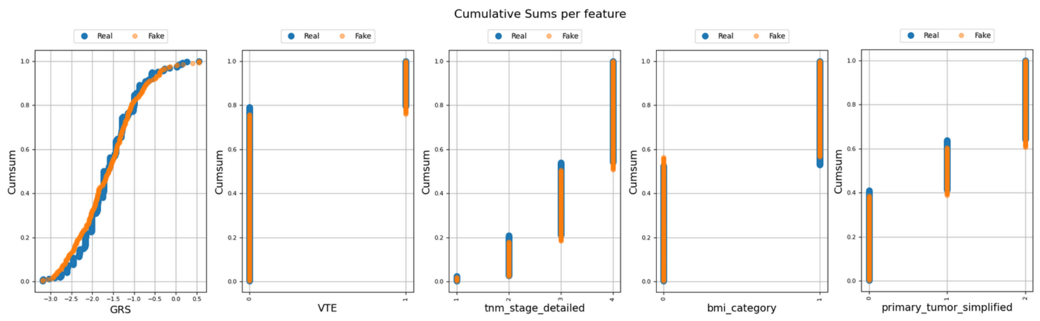
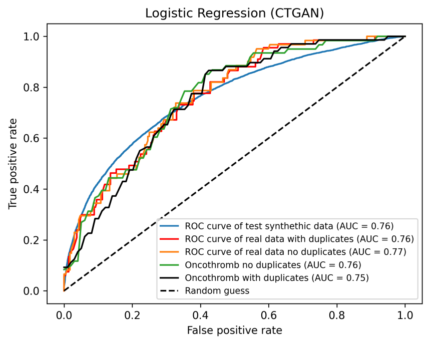
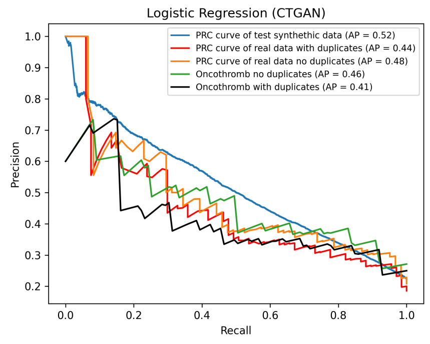


FIGURE 5.1: Cumsum per feature for CTGAN



(a) ROC curve



(b) PR curve

FIGURE 5.2: Performance of CTGAN. The model is evaluated visually and using a logistic trained on synthetic data and tested on both synthetic and real data.

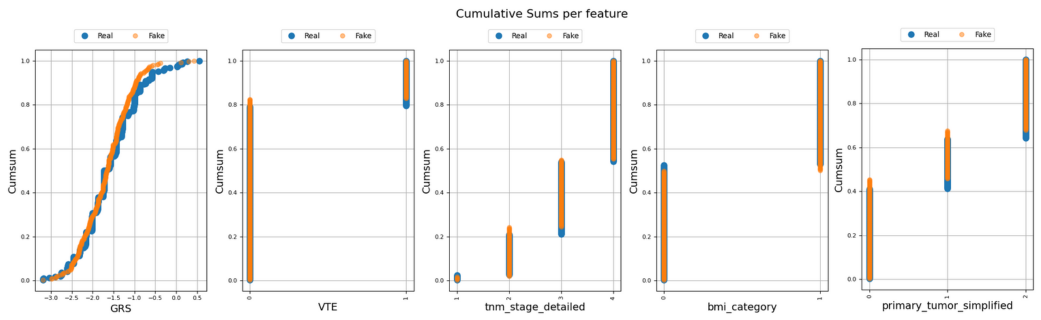
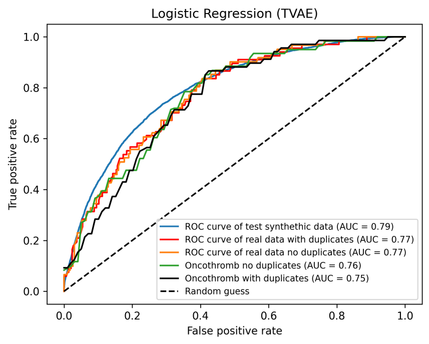
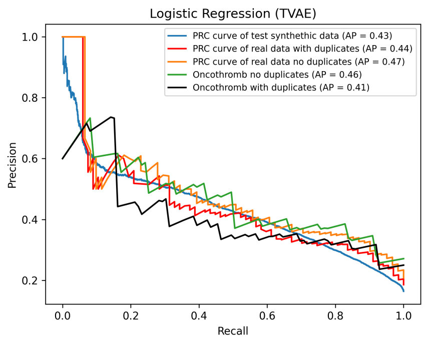


FIGURE 5.3: Cumsum per feature for TVAE.



(a) ROC curve



(b) PR curve

FIGURE 5.4: Performance of TVAE. The model is evaluated visually and using a logistic trained on synthetic data and tested on both synthetic and real data.

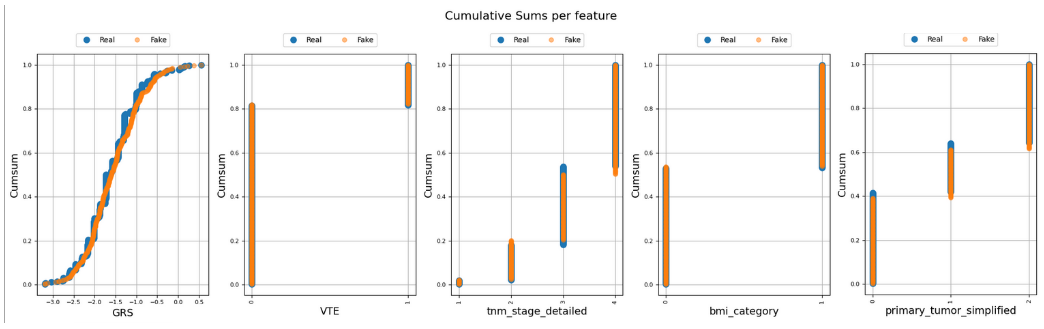
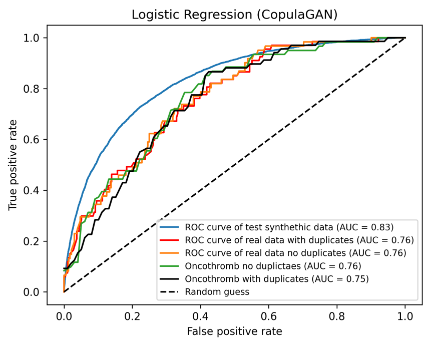
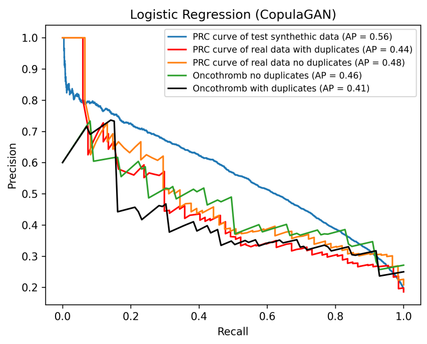


FIGURE 5.5: Cumsum per feature for CopulaGAN.



(a) ROC curve



(b) PR curve

FIGURE 5.6: Performance of CopulaGAN. The model is evaluated visually and using a logistic trained on synthetic data and tested on both synthetic and real data.

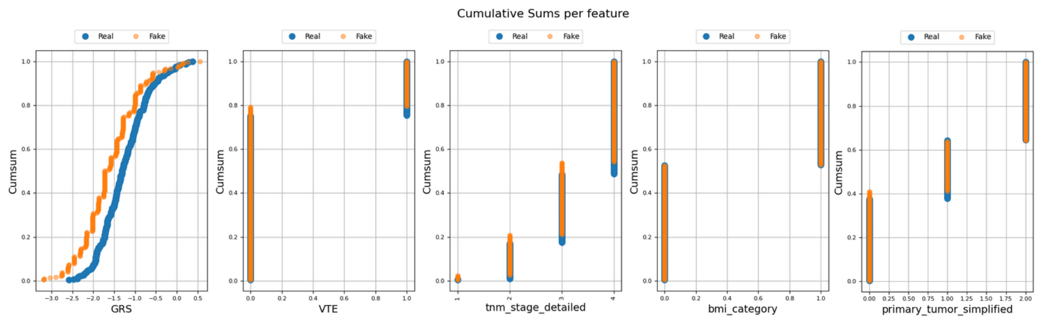
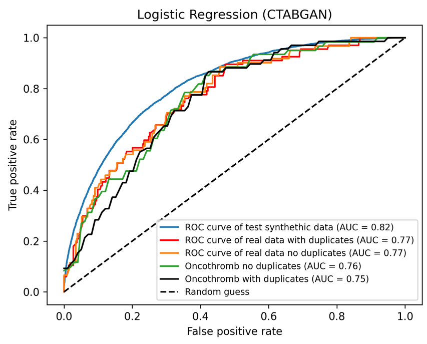
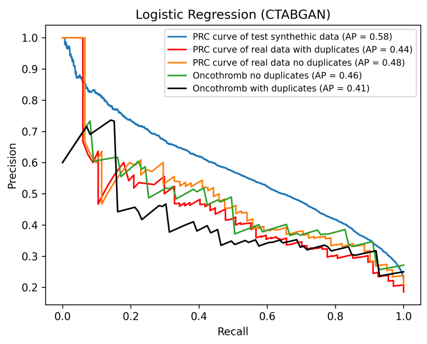


FIGURE 5.7: Cumsum per feature for CTABGAN.



(a) ROC curve



(b) PR curve

FIGURE 5.8: Performance of CTABGAN. The model is evaluated visually and using a logistic trained on synthetic data and tested on both synthetic and real data.

Chapter 6

Assessing VTE in Cancer Patients through Synthetic Data Generation and Transfer Learning

6.1 Synthetic data and domain adaptation

The first step of our proposal is to generate synthetic data samples using the CopulaGAN. As shown in Figure 3.1, we generate synthetic samples containing the GRS and the clinical variables that are selected on Muñoz et al., 2023. The quality of the synthetic data is evaluated both visually and by training classifiers for the prediction of VTE.

The synthetic data generated by the CopulaGAN is used to train a MLP. Then, we proceed to fine-tune the model with real data in order to shift the biases of the classifier. We experiment with various fine-tuning strategies, such as adding more layers to the pretrained model and freezing all layers except for these new ones, as well as retraining all layers for a small number of epochs and several learning rates. Both strategies produced indistinguishable results. For the sake of simplicity, in this thesis we report the results using fine-tuning with reduced learning rates.

6.1.1 Methodology

The first step in this study is to generate synthetic tabular data from the preprocessed dataset, that is using the four selected variables from the original paper Muñoz et al., 2023. A CopulaGAN model is used to generate $100,000$ samples of synthetic data. The CopulaGAN is trained with 5 discriminator updates for each generator update as in Arjovsky, Chintala, and Bottou, 2017; we also set the number of epochs to $1,500$, the batch size to 60, and the rest of parameters are the default ones from Team, 2022.

A MLP is trained with the $100,000$ synthetic data samples with the aim of learning the inductive biases of our problem. The MLP has 3 hidden layers with 128, 64, and 16 neurons, using the Rectified Linear Unit (ReLU) activation function with dropout rates of 0.2 and 0.1 after the first and second layers, respectively. The model was trained using binary cross-entropy as the loss function with Adam optimizer, a learning rate of 1×10^{-3}, and a sigmoid activation function on the output layer. Early stopping was applied by monitoring the validation precision-recall.

Then, domain adaptation techniques are applied to this pretrained deep learning model. Specifically, the model is fine-tuned using real data to adapt and learn its specific patterns. In this step, the architecture of the model remains exactly the same and all layers are trained again, but this time with real data and for a smaller number of epochs and smaller learning rate (1×10^{-4}). To balance the class distribution of

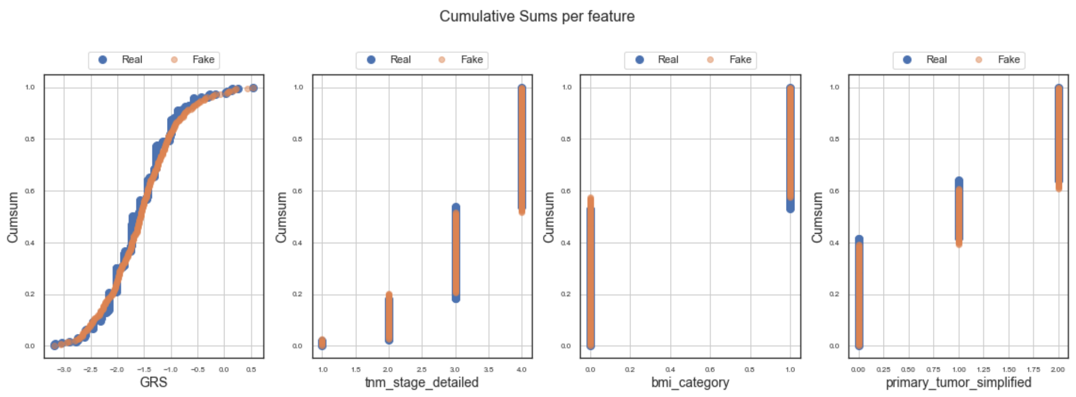


FIGURE 6.1: Cumulative sum per feature of synthethic data and real data

the real data, we computed class weights by dividing the total number of examples by twice the number of negative (or positive) examples. These class weights help to ensure that the model does not become biased towards the majority class during fine-tuning step. The early stopping criteria used during the previous training step is still used for this training as well.

6.1.2 Performance metrics

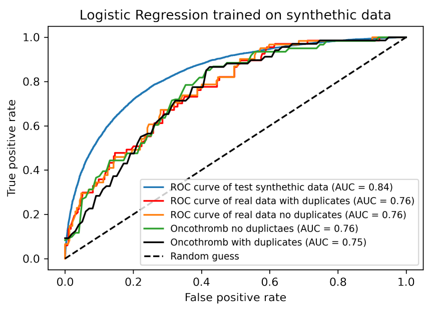
The evaluation of our method is done by comparing the ROC (Receiver Operating Characteristic) curve and PRC (Precision-Recall Curve) with the ONCOTHROMB score's curves. Additionally, we report the area under the curves, AUC-ROC and AUC-PRC, respectively.

Since as stated in Chapter 4 we observed that, due to the variable selection process conducted in the ONCOTHROMB score, the original data set contains duplicated observations. This may undesirably bias the results. In order to provide a fair comparison, we evaluate all metrics in the dataset with duplicates (as done in the original paper) and also without duplicates.

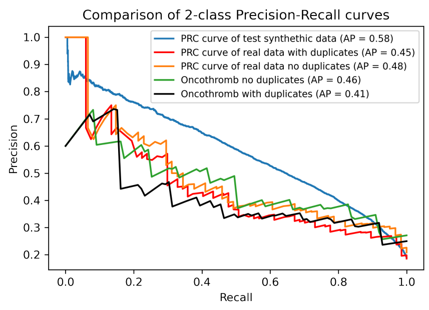
6.1.3 Experiments

The experiments are divided in three blocks:

- **Experiment 1: Synthetic data quality assessment.** The CopulaGAN is trained over all the preprocessed dataset. To assess the quality of the synthetic data, a graphical comparison is performed between the distributions of both synthetic and real data.
- **Experiment 2: Assessment of synthetic data trained classifiers.** Subsequently, the synthetic data was divided into training and test sets, and a logistic regression classifier was trained on this data. This logistic regression is evaluated using the synthetic fake data and also the real data in order to compare it to the baseline. The purpose of this evaluation was to verify that the logistic regression model trained with synthetic data produced similar results to that trained with real data. Afterwards, a MLP consisting of three layers was trained using the synthetic training data, and precision-recall was utilized as the stopping criteria. This model is evaluated with the test data and also with the real data to be compared with the baseline.



(a) ROC curve



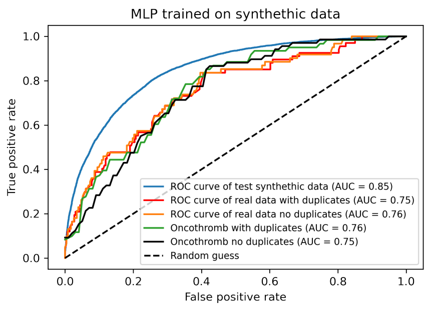
(b) PR curve

FIGURE 6.2: Synthetic and real data evaluation on classification task. The blue curves represents the logistic regression model tested on test synthetic data. The green curves are the ONCOTHROMB model's and the yellow/red curves are from the logistic regression tested on the real data.

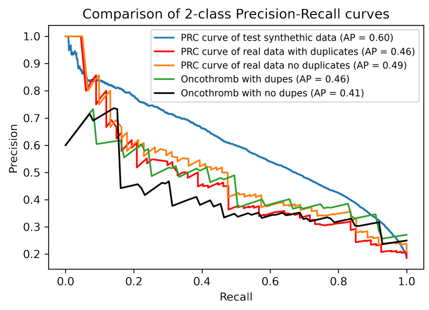
- **Experiment 3: Assessment of the transfer learning approach.** As a last step, the real data is divided into 5 stratified folds and is used to adapt the model to our domain by fine-tunning the MLP.

Experiment 1: Synthetic data qualitative assessment
Synthethic data generated with CopulaGAN is first evaluated by comparing the cumulative sum distribution per feature.
Figure 6.1 suggests that the synthetic data preserves the distribution and patterns of the original data. Moreover, we proceed to evaluate how good is this synthetic data compared to the real data when classifying patients by VTE risk.

Experiment 2: Assessment of synthetic data trained classifiers
The information captured in synthetic data is demonstrated to be useful for classifying real data through the performance of logistic regression trained on synthetic data (Figure 6.2). We observe that the model trained exclusively on synthetic data achieves comparable results to the model trained on the original data. The ROC



(a) ROC curve



(b) PR curve

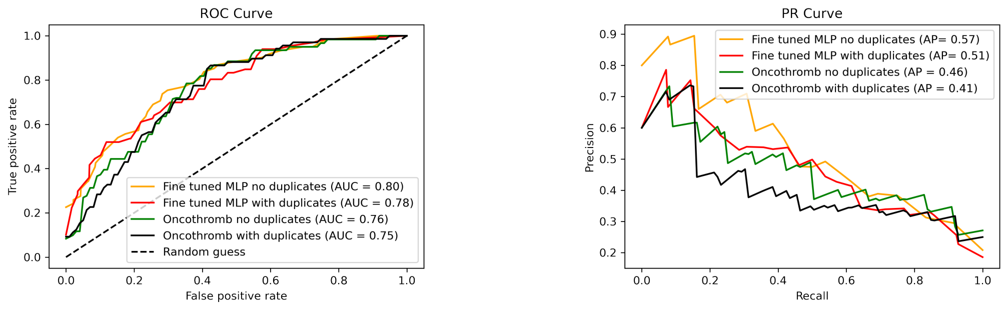
FIGURE 6.3: Synthetic data and real data evaluated on classification task. The blue curves represents the MLP model tested on test synthetic data. The green and black curves are the ones from ONCOTHROMB replications and the yellow and red curves are from the MLP tested on the real data.

curve in Figure 6.2 (a) indicates that our synthetic data still contains valuable information. Similar conclusions can be drawn from Figure 6.2 (b), where synthetic models are on pair with the models trained with real data in terms of PR curve. These results support our hypothesis that the generated synthetic data contains valuable information for making predictions.

Figure 6.3 shows the same kind of evaluation when training a MLP. Again, we observe in both the ROC curve, Figure 6.3 (a), and PR curve in Figure 6.3 (b), that the MLP classifier replicates the findings shown in the former experiment.

Experiment 3: Assessment of the transfer learning approach

As described in the experimental setup, we report the results after using the transfer learning approach. The depicted curves represent the average curves derived from performing a 5-fold cross-validation with stratification, which helps prevent overfitting. Figure 6.4 shows the curves obtained after the fine tuning.



(a) ROC curve

(b) PR curve

FIGURE 6.4: Real data evaluated on classification task. The green and black curves are the one from our ONCOTHROMB replications and the yellow and red curves are from the fine tunned MLP tested on the real data.

When looking at the ROC curves in Figure 6.4 (a) we observe a consistent improvement of the AUC-ROC values compared to the baseline in both settings; i.e. considering duplicates and non-duplicates. This is shown by the clear improvement in the TPR (true positive rates) for the first half of the FPR (false positive rates) in the ROC curves. However, for large values of FPR all methods seems to converge to the same behavior. A more drastic improvement is found in the Precision-Recall curves. Figure 6.4 (b) shows the obtained curves. A detailed examination of these segments according to the two different set-ups, namely *duplicates* and *non-duplicates*, provides evidence that the proposed approach improves the AUC-PRC by 10%. All the relevant figures and its performance metrics are summarized in Table 6.1.

	ONCOTHROMB score	Trained with synthetic data	Fine-tuned with real-world data
Non-duplicates	0.46/0.76	0.49/0.76	**0.57/0.80**
Duplicates	0.41/0.75	0.46/0.75	**0.51/0.78**

TABLE 6.1: Results of AUC-PRC/AUC-ROC for the baseline model, synthetic data, and transfer learning.

The large improvement in AUC-PRC might have important consequences in the clinical practice. If we examine the plots in Figure 6.4 (b) we observe that the fine-tuned approaches clearly improve the operational point range of the score. In the case of considering the original dataset (curves in black and red), we observe that for recalls between 0.2 and 0.8, the proposed methodology achieves important gains. If we focus on the modified dataset (curves green and yellow), we observe the same effect in the range of recalls between 0.0 and 0.65. Again, the proposed method clearly outperforms the baseline. As said earlier, this might have a huge impact in clinical practice as it allows the physician to leverage the different risks involved in the process.

For the sake of discussion, the recall value stands for the rate of VTE detected from all the VTE population. Thus, the larger the recall, the best one can prevent VTE complications by administering anticoagulant treatment. However, as expected, this value trades-off with the precision. In this case, precision is related to the risk created by the administration of the anticoagulant treatment. The operational point of the original ONCOTHROMB article is $(0.8, 0.3)$. This is justified by the flat area in the black precision curve for the range of values between 0.4 and 0.8 recall. Precision of 0.3 means that the original score would recommend to overmedicate 70% of detected patients. The new score proposed in this thesis allows for the exploration and selection of different operational points, with a clear reduction of the risk of overmedication while keeping a large sensitivity value.

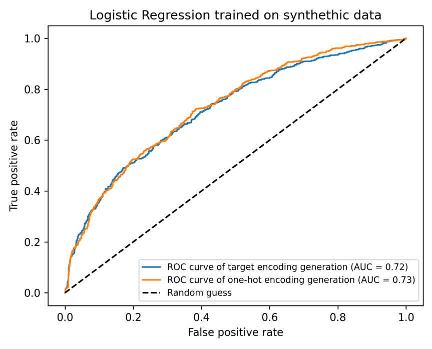
Chapter 7

Enhancing CTGAN with Target Encoding for Efficient Modeling of Large Categorical Variables

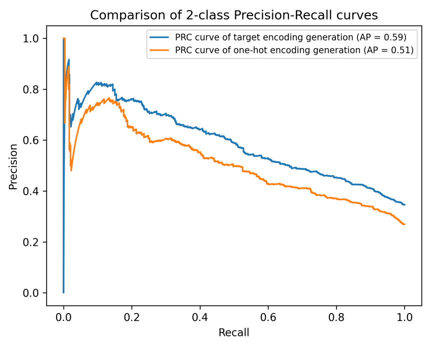
7.1 CTGAN encoding limitation

Having studied and worked extensively with GANs for tabular data, specifically with CTGAN, we have come to the conclusion that there are several avenues of research that can be explored to enhance the generation of synthetic tabular data. Unlike synthetic images, the generation of tabular data is not as advanced and requires further research in order to develop improved models for generating synthetic data. In this thesis we propose enhancing CTGAN architecture through exploring the encoding of categorical variables.

Upon analyzing the architecture, we have observed that all categorical variables are encoded using one-hot encoding. However, this may not be the optimal choice when dealing with categorical variables that have a large number of distinct categories. Therefore, we suggest modifying the architecture to also accommodate target encoding, which can effectively encode categorical data with a substantial number of categories.



(a) ROC curve



(b) PR curve

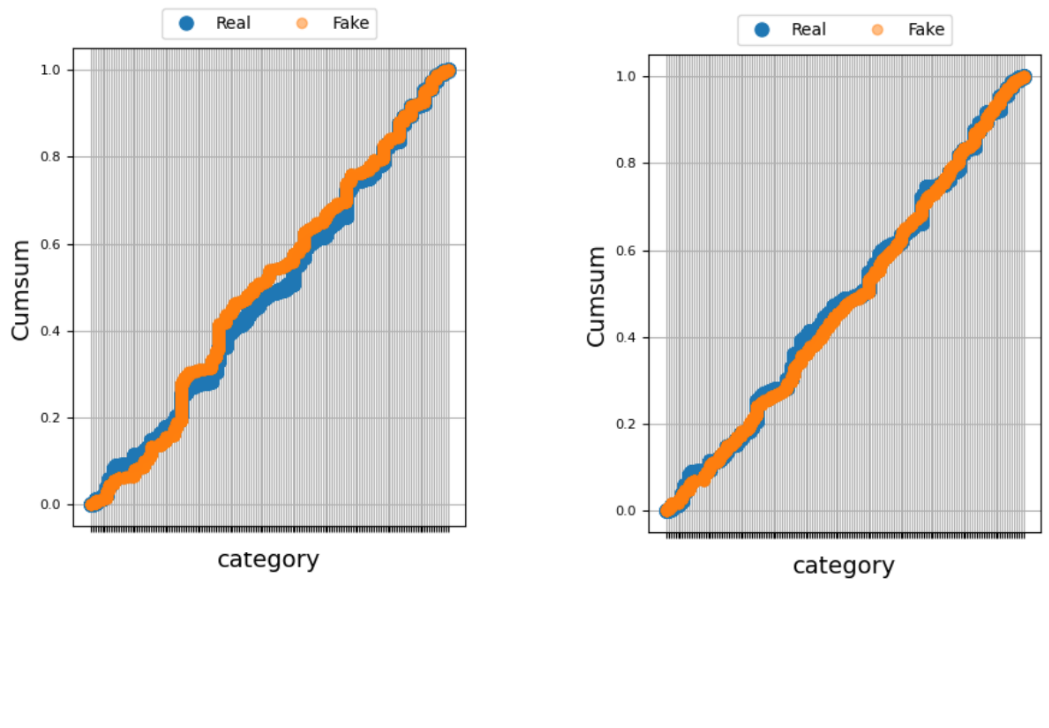
FIGURE 7.1: Target encoding and one hot encoding performance comparison. The orange curves depict the model's performance on test synthetic data generated using one-hot encoding. The blue curves correspond to models trained with target encoding on the category column and one-hot encoding on the other columns.

7.1.1 Target encoding

Target encoding (Micci-Barreca, 2004) is a technique to transform categorical variables into numerical representations based on the target variable. It involves replacing each category in a categorical variable with a value derived from the target variable. This encoding captures the relationship between the category and the target, providing valuable information for predictive modeling.

The process of target encoding involves grouping the data by each unique category, calculating target statistics (such as mean, median, sum, or frequency) within each group, and replacing the original categorical values with the corresponding target statistic. This enables machine learning algorithms to work with categorical variables in a meaningful way.

We made modifications to the code of the CTGAN algorithm provided in the Team, 2022 library to handle categorical data using either target encoding or one-hot encoding. We added input parameters that allow users to choose the encoding type for each categorical variable. When using target encoding, users also need to specify the target variable. These modifications require significant non trivial adjustments to the code to support the utilization of target encoding data. This is because the original code was specifically designed to work with one-hot encoding and all the corresponding data structures are aligned with that encoding scheme. During training, the conditional vector used for sampling category elements still relies on a one-hot encoded vector to determine which category to sample. However, instead of encoding binary values, it now encodes the target encoding values. Additionally, the row representation is altered so that the target encoding variable only requires



(a) One hot encoding (b) Target encoding

FIGURE 7.2: Visual evaluation of category variable for real and fake data. The left plot shows the results trained with one hot encoding and the right plot with target encoding.

a single value for representation, rather than a binary vector. Finally, the reverse transformation from target encoded values to original values is also implemented.

With our modifications, we can now encode the data using target encoding and generate synthetic data that resembles the target-encoded version of the column. We then apply a reverse transformation to recover the initial categories from the encoded values.

7.1.2 Methodology

As the Kickstarter dataset is quite large, we opted to create a subset consisting of 10, 000 random samples. Additionally, we selected a subset of specific columns from the original dataset, namely: *category*, *main category*, *currency*, *pledged*, *goal*, *country*, and *outcome*. This smaller dataset was utilized to train two CTGAN models for 100 epochs. The training process involved a batch size of 60 and 5 discriminator steps.

The first CTGAN model was trained using one-hot encoding for all categorical variables. On the other hand, the second CTGAN model employed the same configuration, except for the category column, which was encoded using target encoding. This approach was chosen due to the category column containing a significantly large number of unique categorical variables.

	One hot encoding	Target encoding
AUC-ROC	**0.73**	0.72
AUC-PRC	0.51	**0.59**

TABLE 7.1: Results of AUC-PRC/AUC-ROC for target encoding and one hot encoding CTGANs.

7.1.3 Results

Figure 7.1 illustrates the performance of both models on a logistic regression-based classification task. In Figure 7.1 (a), the ROC curve of both approaches appears very similar, with the synthetic data trained using one-hot encoding being slightly better by one point. Figure 7.1 (b) demonstrates that the PR curve significantly improves with the synthetic data generated using target encoding. Additionally, the visual evaluation presented in Figure 7.2 suggests that the target encoding approach better resembles the real data.

Table 7.1 presents the AUC-ROC and AUC-PRC values for both approaches. The results indicate that this modification holds promise in improving the synthesis of large categorical variables using CTGAN. However, it is important to note that these findings represent only an initial step, and further experiments with different datasets should be conducted. Additionally, exploring other types of encoding, such as ordinal encoding or quantile encoding (Mougan et al., 2021), could also be considered.

Chapter 8

Conclusions

In this thesis, our first step was to replicate a medical research paper using Python. However, the absence of certain details regarding the methodology posed significant challenges during this process. To overcome some of these obstacles, we proposed an alternative approach for computing p-values. Moreover, we improved the reproducibility of the paper by offering a Python version and providing a comprehensive account of the challenges we encountered, including the decisions made to address duplicated rows. Our methodology yielded results that closely mirrored those reported in the original paper. As a result, we present a Python implementation that can serve as a dependable benchmark for future research investigations.

The main hypothesis of this work is that we can improve the results of the ONCOTHROMB score by using synthetic data. We explored the use of synthetic data generation and transfer learning methodologies for improving the operational points in the PRC and ROC curves of a classifier that predicts the risk of VTE in cancer patients. Results show clear improvements in the AUC-ROC and AUC-PRC. These findings have impact in the clinical practice as it may allow to reduce the risk of complications due to the treatment of VTE.

After conducting an extensive study on the functionality of tabular GANs and comparing various architectures, we have observed that GANs designed for tabular data require further research and improvements in order to attain the same level of reliability as GANs used for image data. Currently, they do not perform as well as GANs for images. In our proposal, we aim to enhance the generation of large categorical data by introducing CTGAN target encoding, which offers the flexibility to choose between target encoding or one-hot encoding for categorical columns. We have found that this modification yields promising results. However, additional experiments and validations are necessary to further investigate its effectiveness.

8.1 Future Work

The next steps in this line of work follow two main avenues. First, the obtained results are very promising but further validation is recommended. Thus, new data is being collected that may further confirm these improvements. Second, we plan on focusing on the influence of the genetic variants in VTE. In this line of thought, we plan on exploring the use of causal machine learning algorithms to better understand the underlying mechanisms of cancer in VTE.

We have identified several future research lines that can enhance the synthetic data generation. Our plans involve conducting further research expanding upon the target encoding study. Additionally, we propose investigating mechanisms to better handle outliers and exploring the potential of employing ensemble techniques to improve the generation of synthetic data. Another exciting area for research is the exploration of diffusion models, which have shown promising results in generating synthetic data (Kotelnikov et al., 2022).

Bibliography

Abadi, Martín et al. (2015). "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". In: *Software available from tensorflow.org*.

Abedi, Masoud et al. (2022). "GAN-Based Approaches for Generating Structured Data in the Medical Domain". In: *Applied Sciences* 12.14, p. 7075. DOI: 10.3390/app12147075.

Al-Samkari, Hanny and Jean M Connors (2019). "Managing the competing risks of thrombosis, bleeding, and anticoagulation in patients with malignancy". In: *Hematology 2014, the American Society of Hematology Education Program Book* 2019.1, pp. 71–79. DOI: 10.1182/bloodadvances.2019000369.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). *Wasserstein GAN*. arXiv: 1701.07875 [stat.ML].

Azizi, Shekoofeh et al. (2023). *Synthetic Data from Diffusion Models Improves ImageNet Classification*. DOI: 10.48550/arXiv.2304.08466. arXiv: 2304.08466 [cs.CV].

Blom, Jeanet W. et al. (Feb. 2005). "Malignancies, Prothrombotic Mutations, and the Risk of Venous Thrombosis". In: *JAMA* 293.6, pp. 715–722. ISSN: 0098-7484. DOI: 10.1001/jama.293.6.715. eprint: https://jamanetwork.com/journals/jama/articlepdf/200333/joc42103.pdf.

Chawla, N. V. et al. (2002). "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16, pp. 321–357. DOI: 10.1613/jair.953. URL: https://doi.org/10.1613%2Fjair.953.

Chollet, François et al. (2015). "Keras". In: *GitHub repository*.

Denton, Emily et al. (2015). *Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks*. arXiv: 1506.05751 [cs.CV].

Desai, Ruchi et al. (2020). "Efficacy and safety of direct oral anticoagulants for secondary prevention of cancer associated thrombosis: a meta-analysis of randomized controlled trials". In: *Scientific Reports* 10.1, p. 18945. DOI: 10.1038/s41598-020-75863-3.

Goodfellow, Ian J. et al. (2014). *Generative Adversarial Networks*. arXiv: 1406.2661 [stat.ML].

Gulrajani, Ishaan et al. (2017). "Improved training of wasserstein GANs". In: *Advances in Neural Information Processing Systems*.

Iman, Mohammadreza, Hamid Reza Arabnia, and Khaled Rasheed (2023). "A Review of Deep Transfer Learning and Recent Advancements". In: *Technologies* 11.2, p. 40. DOI: 10.3390/technologies11020040.

Jang, Eric, Shixiang Gu, and Ben Poole (2017). *Categorical Reparameterization with Gumbel-Softmax*. arXiv: 1611.01144 [stat.ML].

Kemical (2020). *Kaggle: Kickstarter projects*. Accessed: 20-October-2020. URL: https://www.kaggle.com/kemical/kickstarter-projects.

Khorana, Alok A et al. (2008). "Development and validation of a predictive model for chemotherapy-associated thrombosis". In: *Blood, The Journal of the American Society of Hematology* 111.10, pp. 4902–4907. DOI: 10.1182/blood-2007-10-116327.

Kingma, Diederik P and Max Welling (2022). *Auto-Encoding Variational Bayes*. arXiv: 1312.6114 [stat.ML].

Kotelnikov, Akim et al. (2022). *TabDDPM: Modelling Tabular Data with Diffusion Models*. arXiv: 2209.15421 [cs.LG].

Lin, Zinan et al. (2018). “Pacgan: The power of two samples in generative adversarial networks”. In: *Advances in Neural Information Processing Systems*.

Mahajan, Anjlee et al. (2022). “The incidence of cancer-associated thrombosis is increasing over time”. In: *Blood advances* 6.1, pp. 307–320. DOI: 10.1182/bloodadvances.2021005590.

Micci-Barreca, Daniele (2004). “A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems”. In: *SIGKDD Explor. Newsl.* 6.1, pp. 27–32. DOI: 10.1145/980972.980994. URL: https://doi.org/10.1145/980972.980994.

Mirza, Mehdi and Simon Osindero (2014). “Conditional Generative Adversarial Nets”. In: *CoRR* abs/1411.1784. arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

Mougan, Carlos et al. (2021). *Quantile Encoder: Tackling High Cardinality Categorical Features in Regression Problems*. arXiv: 2105.13783 [cs.LG].

Muñoz, Ana et al. (2023). “A Clinical-Genetic Risk Score for Predicting Cancer-Associated Venous Thromboembolism: A Development and Validation Study Involving Two Independent Prospective Cohorts”. In: *J Clin Oncol*. DOI: 10.1200/JCO.22.00255.

Muñoz Martin, Andres J et al. (2018). “Multivariable clinical-genetic risk model for predicting venous thromboembolic events in patients with cancer”. In: *British journal of cancer* 118.8, pp. 1056–1061. DOI: 10.1038/s41416-018-0027-8.

Odena, Augustus, Christopher Olah, and Jonathon Shlens (2017). *Conditional Image Synthesis With Auxiliary Classifier GANs*. arXiv: 1610.09585 [stat.ML].

Park, Noseong et al. (2018). “Data synthesis based on generative adversarial networks”. In: *Proceedings of the VLDB Endowment* 11.10, pp. 1071–1083. DOI: 10.14778/3231751.3231757. URL: https://doi.org/10.14778%2F3231751.3231757.

Pedregosa, Fabian et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pituskin, Edith (2022). “Cancer as a new chronic disease: Oncology nursing in the 21st Century”. In: *Canadian OnCOlOgy nursing JOurnal* 32.1, p. 87.

Radford, Alec, Luke Metz, and Soumith Chintala (2016). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. arXiv: 1511.06434 [cs.LG].

Sung, Hyuna et al. (2021). “Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries”. In: *CA: A Cancer Journal for Clinicians* 71.3, pp. 209–249. ISSN: 0007-9235. DOI: 10.3322/caac.21660.

Team, SDV Development (2022). *SDV: Synthetic Data Vault*. https://github.com/sdv-dev/SDV. Version 0.11.0.

Wan, Zhiqiang, Yazhou Zhang, and Haibo He (2017). “Variational autoencoder based synthetic data generation for imbalanced learning”. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7. DOI: 10.1109/SSCI.2017.8285168.

Xu, Lei et al. (2019). *Modeling Tabular data using Conditional GAN*. DOI: 10.48550/arXiv.1907.00503. arXiv: 1907.00503 [cs.LG].

Zhao, Zilong et al. (2021). *CTAB-GAN: Effective Table Data Synthesizing*. arXiv: 2102.08369 [cs.LG].

Zöller, Bengt et al. (2015). “Family history of venous thromboembolism as a risk factor and genetic research tool”. In: *Thrombosis and haemostasis* 114.11, pp. 890–900. DOI: 10.1160/TH15-04-0306.

Zöller, Bengt (Nov. 2019). “Genetics of venous thromboembolism revised”. In: *Blood* 134.19, pp. 1568–1570. ISSN: 0006-4971. DOI: 10.1182/blood.2019002597. eprint: https://ashpublications.org/blood/article-pdf/134/19/1568/1505335/bloodbld2019002597c.pdf.