UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

CAIO MARTIM BARROS

AVALIAÇÃO DE TÉCNICAS DE GERAÇÃO DE DADOS SINTÉTICOS TABULARES EM TAREFAS DE CLASSIFICAÇÃO

FORTALEZA

2023

CAIO MARTIM BARROS

AVALIAÇÃO DE TÉCNICAS DE GERAÇÃO DE DADOS SINTÉTICOS TABULARES EM TAREFAS DE CLASSIFICAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. César Lincoln Cavalcante Mattos

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B275a Barros, Cario Martim.
Avaliação de técnicas de geração de dados sintéticos tabulares em tarefas de classificação / Cario Martim Barros. – 2023.
104 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia de Computação, Fortaleza, 2023.
Orientação: Prof. Dr. César Lincoln Cavalcante Mattos.

1. Dados sintéticos tabulares. 2. Aprendizado de máquina. 3. IA generativa. 4. Adição de dados. 5. Synthetic data vault (SDV). I. Título.

CDD 621.39

CAIO MARTIM BARROS

AVALIAÇÃO DE TÉCNICAS DE GERAÇÃO DE DADOS SINTÉTICOS TABULARES EM TAREFAS DE CLASSIFICAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. César Lincoln Cavalcante Mattos (Orientador)
Universidade Federal do Ceará - UFC

Prof. Dr. João Paulo do Vale Madeiro
Universidade Federal do Ceará - UFC

Prof. Dr. João Paulo Pordeus Gomes
Universidade Federal do Ceará - UFC

AGRADECIMENTOS

Ao Prof. Dr. César Lincoln Cavalcante Mattos, pela orientação, ensinamentos e paciência.

“Nós somos o que fazemos repetidamente. Excelência, então, não é um ato, mas um hábito.”

(Aristóteles)

RESUMO

A geração de dados sintéticos desempenha um papel crucial em aprendizado de máquina em diversos domínios. Este estudo explora o Synthetic Data Vault (SDV), uma ferramenta que utiliza métodos de aprendizado de máquina para produzir dados sintéticos de alta qualidade. Para isso, desenvolve-se uma pipeline de dados completa, a fim de assegurar a persistência automatizada e correta dos dados de treino e teste, que abrange a geração, aplicação e avaliação dos dados sintéticos em algoritmos de Machine Learning (ML) para classificação. A eficácia destas etapas são verificadas por meio de comparações entre diferentes estratégias para as etapas de criação de modelos geradores de dados sintéticos e seu uso em modelos de classificação, utilizando métricas adequadas para cada abordagem. Dessa forma, ao empregar o SDV em diferentes conjuntos de dados, esta pesquisa evidencia a relevância e a utilidade da criação e uso de dados artificiais, possibilitando análises mais robustas e modelos de aprendizado de máquina mais precisos. Assim, o SDV oferece uma solução promissora para enfrentar o problema de escassez, balanceamento e privacidade de dados, impulsionando a pesquisa e as aplicações na área.

Palavras-chave: Synthetic Data Vault (SDV). Dados Sintéticos Tabulares. Aprendizado de Máquina. IA Generativa. Adição de Dados.

ABSTRACT

The generation of synthetic data plays a crucial role in machine learning across various domains. This study explores SDV, a tool that utilizes machine learning methods to produce high-quality synthetic data. To achieve this, a complete data pipeline is developed to ensure the automated and accurate persistence of training and testing data, covering the generation, application, and evaluation of synthetic data in ML algorithms for classification. The effectiveness of these steps is verified through comparisons between different strategies for creating synthetic data generator models and their use in classification models, using metrics suitable for each approach. By employing SDV on different datasets, this research highlights the relevance and utility of creating and using artificial data, enabling more robust analyses and more accurate machine learning models. Thus, SDV provides a promising solution to address the challenges of data scarcity, imbalance, and privacy, driving research and applications in the field.

Keywords: Synthetic Data Vault (SDV). Synthetic Tabular Data. Machine Learning. Generative AI. Data Augmentation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
CDF	Cumulative Distribution Function
CLI	Command Line Interface
CSV	Comma Separated Values
CTGAN	Conditional Tabular GAN
DNN	Deep Neural Network
GANs	Generative Adversarial Networks
GNN	Generative Neural Network
JSON	JavaScript Object Notation
ML	Machine Learning
NLP	Natural Language Processing
SDM	Synthetic Data Metrics
SDtypes	Synthetic Data Types
SDV	Synthetic Data Vault
SMOTE	Synthetic Minority Oversampling Technique
t-SNE	t-Distributed Stochastic Neighbor Embedding
TVAE	Time-Varying Autoencoder
TVD	Total Variation Distance
VAEs	Variational Autoencoders

SUMÁRIO

1 INTRODUÇÃO

A recorrente ausência de dados em diversos contextos restringe o desenvolvimento de modelos de ML e dificulta a obtenção de informações precisas e relevantes sobre os resultados produzidos Li *et al.* (2023). Além disso, o desbalanceamento de dados emerge como um obstáculo importante, comprometendo a performance dos algoritmos de classificação em realizar previsões precisas, uma vez que certas classes ou grupos possam ter poucos exemplos ou estão muito correlacionados com outros dados.

Outra problemática relevante é a imposição de restrições de acesso aos dados por empresas e instituições, como comumente reportado em artigos médicos Guillaudeux *et al.* (2023), o que em consequência dificulta o compartilhamento de dados e informações, prejudicando o desenvolvimento e validação de modelos de ML em ambientes reais.

Adicionalmente, a demanda por um volume significativamente maior de dados do que os disponíveis para atividades, como a criação de cenários de teste na área de segurança cibernética, representa uma barreira à obtenção de modelos mais robustos e precisos (LI *et al.*, 2023).

Portanto, para enfrentar as problemáticas descritas, este estudo explora métodos para a geração de dados sintéticos com aplicações reais em experimentações. Por meio de estudos científicos na área, elas são focadas utilizando um conjunto de bibliotecas disponíveis graças ao SDV Patki *et al.* (2016), o qual é uma ferramenta promissora para gerar dados sintéticos de alta qualidade por meio de técnicas de aprendizado de máquina e modelos estatísticos.

1.1 Desafios para a Geração de Dados Sintéticos

Como mencionado em Rodrigues (2021), Clarke (1976), a geração de dados sintéticos é um problema que pode ser dividido em dois subproblemas distintos. Contudo,

com base em novos estudos, pode-se adicionar mais dois subproblemas para enfatizar os desafios para essa geração, os quais são:

1. **Fidelidade aos dados reais:** A produção dos dados sintéticos necessita estar condizente aos valores do dados reais que são inseridos como exemplos de entrada. Ademais, essa produção tem que representar outros valores diferentes do que existem nos reais (RODRIGUES, 2021).
2. **Lógica dos dados sintéticos frente a realidade:** Em diversos contextos, existem colunas e consequentemente dados colunares que precisam ser gerados de uma forma específica, como as datas. Normalmente o tratamento das datas requer ajustes no sentido de dependência temporal com o presente ou com outras datas existentes no conjunto de dados.
3. **Privacidade dos dados:** Os dados sintéticos precisam, em diversos contextos, como em conjuntos de dados de saúde Röglin *et al.* (2022), garantir a anonimização desses dados reais sensíveis. Estudos avançados garantem melhorias nesse desafio utilizando Generative Adversarial Networks (GANs) (BEAULIEU-JONES *et al.*, 2019).
4. **Segurança e criação de cenários:** Dados sintéticos são utilizados para diversos motivos, uma delas é fornecer dados de teste em cenários de segurança cibernética. Aqui, eles são gerados de forma a garantir a identificação de vulnerabilidades, fortalecer sistemas e desenvolver estratégias de defesas eficazes (KEYRUS, 2023).

1.2 Objetivos

Este trabalho tem como objetivo geral avaliar o uso do SDV em diversos contextos (categoria, estrutura e contexto informacional do dado), descrevendo seus recursos e funcionalidades. Além disso, objetiva a explicação sobre como são gerados os dados por meio do SDV a partir da investigação sobre diversos modelos de aprendizado

de máquina para produzir os dados artificiais, incluindo a geração estatística usando o GaussianCopula Houssou *et al.* (2022), a abordagem sobre redes neurais baseadas em Variational Autoencoders (VAEs) utilizando o Time-Varying Autoencoder (TVAE) Kühnel *et al.* (2023) e GANs Goodfellow *et al.* (2020) como Conditional Tabular GAN (CTGAN) Badu-Marfo *et al.* (2020) e abordagem mista entre Cópula e CTGAN com CopulaGAN.

1.2.1 Objetivos Específicos

O projeto visa construir uma *pipeline* completa para o gerenciamento dos dados, desde a extração e análise até o treinamento, validação e avaliação das métricas Kühnel *et al.* (2023) relevantes implementadas e fornecidas pelo SDV. Assim, para garantir a validade e efetividade dessa abordagem, o melhor modelo gerado é utilizado em tarefas de classificação utilizando modelos de ML, a fim de comparar o desempenho em três estratégias: com a adição dos novos dados gerados, apenas dados sintéticos ou utilizando SMOTE Chawla *et al.* (2002), que é discutido na seção 3.4.1.

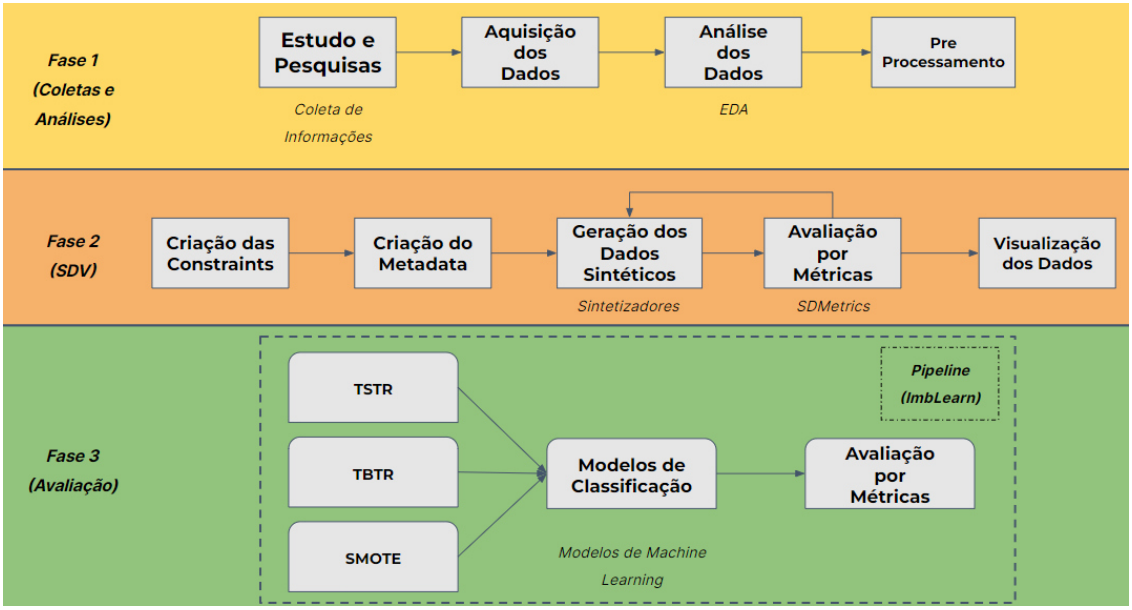
De modo geral, os objetivos especifícos são:

1. Analisar a geração de dados sintéticos por diversos modelos de ML, a partir do SDV, nomeados de sintetizadores. Essa investigação é realizada por métodos de distribuição de valores, gráficos, tabelas e métricas.
2. Criar um *pipeline* de dados para otimizar processos, facilitar a geração dos dados e persistir os dados sintéticos para os modelos de classificação de forma simples, automatizada e correta.
3. Utilizar dados sintéticos em diversas estratégias para abordagem de modelos de classificação com algoritmos de ML, a fim de analisar o desempenho das estratégias por meio do uso desses novos dados.

1.2.2 Planejamento Estratégico da Pesquisa

Essa seção se refere a como é projetada e desenvolvida todo o escopo do projeto. Ela é dividida em três fases para facilitar a compreensão de como os processos são realizados.

Figura 1 – Apresentação do Fluxo de Tarefas (Roadmap) da Pesquisa. Diagrama ilustrativo que constitui-se em três fases, para delimitações e criações de objetivos claros.



Fonte: Criado pelo próprio autor.

Na Figura 1, mostra-se o desenvolvimento do projeto separado em três fases:

1. **Fase 1:** É feita a coleta dos conjuntos de dados e análise eles para identificação inicial sobre o contexto dos dados, como: pre-processamento, checagem de valores que se afastam significativamente dos padrões estatísticos (outliers) e captação, substituição ou remoção de linhas nulas.
2. **Fase 2:** Uso do SDV, inicia-se pela criação das constraints, que são restrições lógicas (regras determinísticas dos dados), criação do metadata que é a estrutura padrão da ferramenta para identificação dos atributos do

conjunto de dados e por fim a geração dos dados sintéticos e sua análise.

3. **Fase 3:** Aplicação dos dados sintéticos em projetos reais de classificação com ML pela sua utilização para avaliação e efetivação. Uso de diversas estratégias para avaliar os modelos, identificar padrões e melhores desempenhos. Essa fase consiste na construção de um pipeline da biblioteca ImbLearn (LEMAîTRE *et al.*, 2017), para persistir os dados de forma correta e que não ocorra vazamento dos dados durante os processos.

1.3 Justificativa da Pesquisa

Em síntese, ao utilizar o SDV em conjuntos de dados, busca-se impulsionar a pesquisa e as aplicações na área, permitindo a realização de análises mais robustas e a criação de modelos de aprendizado de máquina mais precisos. A qualidade dos dados sintéticos são avaliadas através de métricas eficazes, por meio do Synthetic Data Metrics (SDM), e a comparação entre diferentes modelos possibilitará a identificação do melhor entre eles, a partir do contexto dos dados. Para certificar a fabricação dos dados artificiais, esses dados são introduzidos em modelos de ML em tarefas de classificação em experimentação diversas para demonstrar que são possíveis gerar dados concisos e que melhoram o desempenhos de tarefas de classificação, além de performar em outros âmbitos importantes na área de estudo. Dessa forma, garante-se que o SDV surge como uma solução promissora para enfrentar as limitações impostas pelas problemáticas mencionadas e aprimorar o desempenho geral das aplicações de ML.

1.4 Estrutura dos Capítulos

Os capítulos seguintes estão estruturados de forma que o Capítulo 2, apresenta a fundamentação teórica, ou seja, explicação dos contextos, termos de pesquisa e os conceitos que são a base para a elaboração dos experimentos de geração dos dados

sintéticos. No Capítulo 3 tem-se a metodologia, com o intuito de evidenciar a abordagem de pesquisa, instalação, configuração e utilização do SDV e seus recursos, como também as formas para geração dos dados sintéticos e as estratégias que são utilizadas para analisar a produção dos dados artificiais.

No Capítulo 4, encontram-se os conjuntos de dados utilizados e suas devidas características, os resultados obtidos com os experimentos realizados nesta pesquisa, como a comparação entre os diversos sintetizadores para produção dos dados sintéticos para identificação do melhor (TVAE, CTGAN, CopulaGAN ou GaussianCopula) e resultados sobre as abordagens das experimentações sobre tarefas de classificação com ML, como pode ser visto na subseção 1.2.1, pela utilização de métricas específicas para cada caso testado.

Por fim, as conclusões finais são apresentadas no Capítulo 5, assim como a identificação de melhorias e sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O presente capítulo tem como objetivo fornecer a base teórica que serviu de embasamento para a criação deste trabalho. Nele, são discutidos os seguintes tópicos essenciais: revisão da literatura sobre termos relevantes, análise dos desafios e métodos relacionados à geração de dados sintéticos em formato tabular, bem como a introdução ao SDV e seus recursos e por conclusão as maneiras de avaliação dos dados sintéticos e algoritmos de classificação com uso de métricas específicas.

2.1 Revisão Bibliográfica

A geração de dados sintéticos é um processo matemático e estatístico realizado por modelos de ML que são treinados usando um conjunto de dados correspondente. Os dados sintéticos são informações geradas artificialmente que podem ser usadas no lugar de dados históricos reais para treinar modelos de Artificial Intelligence (AI) quando os conjuntos de dados reais carecem de qualidade, volume, variedade, segurança ou privacidade, como mencionado na seção 1.1.

Os dados sintéticos podem ser gerados de diversas formas. Com o avanço de estudos e tecnologias, uma alternativa bastante utilizada em termos recentes são por simulações ou algoritmos de computador como uma alternativa mais econômica aos dados reais, ou seja, utilizando conceitos de ML pela utilização de Cópulas Elidan (2013) ou com algoritmos mais complexos envolvendo redes neurais (como VAEs e GANs). Contudo, existem também outras formas menos custosas e complexas que envolvem: métodos lineares, reamostragem de dados, balanceamento de dados e tratamento de dados por Natural Language Processing (NLP).

O aprendizado de máquina, ou ML, é um método usado para planejar modelos complexos e algoritmos que se prestam a fazer previsões. No modelo de aprendizado supervisionado, que é o método a ser utilizado neste projeto, o algoritmo é treinado com

um conjunto de dados rotulados, com exemplos de entradas e saídas desejadas Ludemir (2023). Essa atividade supervisionada tem o objetivo de construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não rotulados. Um dos objetivos, como já mencionado, é justamente analisar os acertos e erros desses classificadores após experimentações de dados reais utilizando dados sintéticos, por meio de métricas de avaliação de modelos de classificação.

Ademais, quando mencionado sobre algoritmos complexos, estes estão comumente baseados em modelos estatísticos, pois fornecem um papel primordial na construção de dados sintéticos de alta qualidade e utilidade, pois garantem uma estrutura matemática e probabilística para a geração de informações fictícias que se assemelham aos dados reais.

Esses modelos estatísticos abrangem uma variedade de técnicas, como Cópulas, modelos lineares generalizados e redes neurais generativas (GANs). Esses modelos são utilizados para capturar a complexa estrutura e dependências presentes nos dados reais, permitindo, assim, que sejam criadas amostras sintéticas que preservem as características essenciais do conjunto de dados original, garantindo a produção de dados para treinamento de dados de ML.

2.2 Métodos para Geração de Dados Sintéticos Tabulares

Na seção 2.1, são introduzidas alguns termos e técnicas para a geração de dados sintéticos tabulares. Este capítulo será descrito, de modo explicativo, como funciona a produção dos dados artificiais por meio de dados históricos.

2.2.1 Métodos Lineares

Os métodos lineares são uma das técnicas mais simples e comuns para a geração de dados sintéticos. Eles envolvem a criação de um modelo linear que descreve

a relação entre as variáveis nos dados originais. Uma vez que o modelo é construído, ele pode ser usado para gerar novos dados que seguem a mesma distribuição. No entanto, esses métodos têm limitações significativas. Eles pressupõem que os dados seguem uma distribuição normal e que as variáveis são independentes umas das outras. Além disso, eles não são capazes de capturar relações complexas ou não lineares entre as variáveis.

2.2.2 Reamostragem dos Dados (Resampling)

A técnica de reamostragem é outra abordagem comum para a geração de dados sintéticos. Ela envolve a criação de novos conjuntos de dados ao reamostrar aleatoriamente os dados originais. Isso pode ser feito de várias maneiras, incluindo a amostragem com reposição (também conhecida como bootstrap) ou sem reposição. A reamostragem pode ser uma ferramenta poderosa para a geração de dados sintéticos, pois ela permite preservar as propriedades estatísticas dos dados originais enquanto introduz alguma variabilidade. No entanto, assim como os métodos lineares, a reamostragem tem suas limitações. Ela pode não ser adequada para conjuntos de dados com estruturas complexas ou quando os dados não são independentes e identicamente distribuídos.

2.2.3 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN), ou mais conhecido pelo termo inglês NLP, é uma subárea da AI que se concentra na interação entre computadores e linguagem humana. Ele pode ser usado para gerar dados sintéticos tabulares ao transformar texto em uma forma estruturada que pode ser facilmente manipulada e analisada. Por exemplo, existem técnicas avançadas de NLP, como modelos de linguagem baseados em redes neurais (VERO *et al.*, 2023; FONSECA; BACAO, 2023), podem ser usados para gerar texto realista que pode então ser transformado em dados tabulares.

2.2.4 SMOTE (Synthetic Minority Oversampling Technique)

O Synthetic Minority Oversampling Technique (SMOTE) Chawla *et al.* (2002) consiste na produção de dados sintéticos para tratar conjunto de dados desequilibrados. Ele é muito abordado quando existe a necessidade de solucionar problemas em que a classe minoritária tem uma representação significativamente menor do que a classe majoritária. Dessa maneira, o SMOTE atua no balanceamento dessas classes, criando dados sintéticos para a classe de menor quantidade de exemplos a partir de uma combinação ponderada de pares de amostras próximas. Dessa forma, o SMOTE tem se mostrado eficaz em melhorar o desempenho de modelos de aprendizado de máquina em situações em que o desequilíbrio de classe é um desafio, tornando-se uma ferramenta importante na produção de dados sintéticos para tarefas de classificação e análise.

Em suma, como citado por Houssou *et al.* (2022), o SMOTE pode ser generalizado pela equação 2.1 abaixo, em que consiste na consideração de uma amostra x_i e por consequência uma nova amostra será gerada considerando seus k vizinhos mais próximos, na maneira de:

- Calcular a diferença entre o vetor de características da amostra em consideração e um dos k vizinhos mais próximos, x_{ki}.
- Multiplicar a diferença por um número aleatório γ entre 0 e 1 e adicione isso ao vetor da amostra para gerar uma nova amostra x_{new}. A equação 2.1 abaixo enfatiza a representação da geração desse novo dado.

$$x_{\text{new}} = x_i + \gamma(x_{ki} - x_i) \tag{2.1}$$

2.2.5 Redes Neurais

As redes neurais artificiais, ou Artificial Neural Network (ANN), são sistemas de computação com nós interconectados que funcionam como os neurônios do cérebro humano. Usando algoritmos, elas podem reconhecer padrões escondidos e correlações

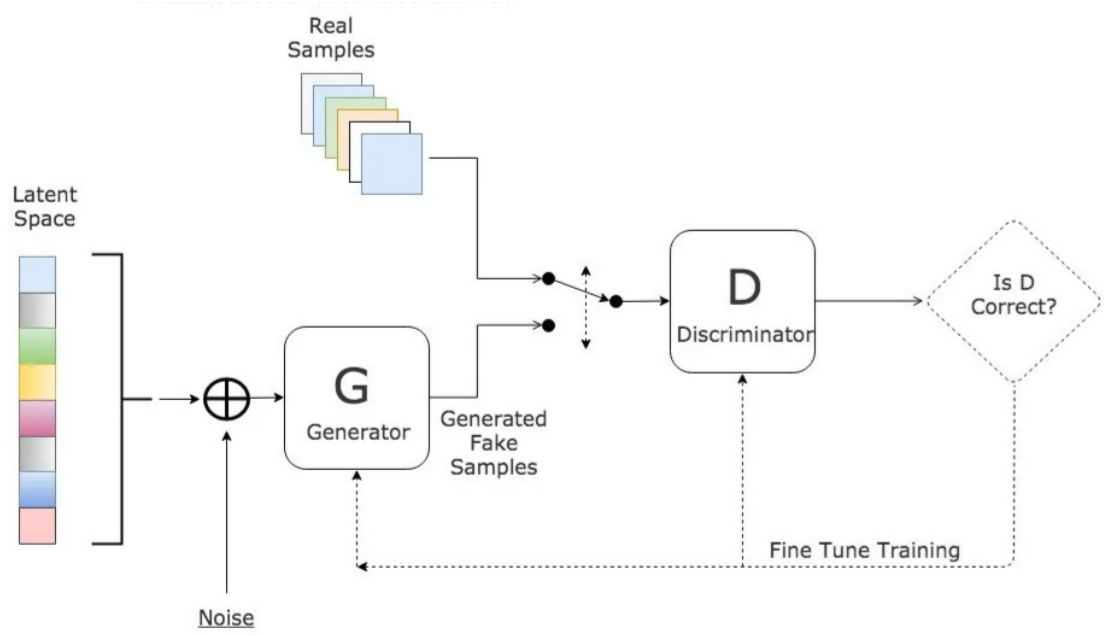
em dados brutos, agrupá-los e classificá-los. Em adição, as redes neurais profundas, ou Deep Neural Network (DNN), que envolve uma construção de uma arquitetura mais elaborada, com adição de novas camadas intermediárias com o propósito de captar relações ainda mais complexas dos dados.

Nesse sentido, o uso de Generative Neural Network (GNN) é uma das melhores opções para a produção de novos dados e sua abordagem está presente nas experimentações dessa pesquisa. As GANs e VAEs empregam GNN para aprender e imitar a complexa distribuição dos dados originais, a fim de criar novas amostras que se assemelhem às amostras reais.

2.2.5.1 Generative Adversarial Networks (GANs)

A técnica de uso de GANs se baseia em um modelo de aprendizado de máquina que consiste em duas redes neurais em competição: um gerador e um discriminador, como pode ser analisado na Figura 2. O gerador cria dados sintéticos, enquanto o discriminador avalia se esses dados são reais ou sintéticos. A competição entre essas duas redes leva ao refinamento contínuo dos dados gerados, tornando-os cada vez mais semelhantes aos dados reais Goodfellow *et al.* (2020), criando, assim, dados sintéticos de alta qualidade.

Figura 2 – Representação de uma Generative Adversarial Network (GAN), em que consiste na amostragem de dados com diferenças para a geração de novos dados, a partir da rede neural geradora e, sua concorrente, a discriminadora para distinguir os dados reais e sintéticos, melhorando continuamente os dados gerados.



Fonte: Hui (2018)

A fórmula matématica abaixo 2.2 é uma equação sobre a função objetivo do discriminador, a fim de otimizar o treinamento de uma GANs. Na equação 2.2, D(x) é a probabilidade do discriminador (D) classificar a entrada (x) como real, G(z) é a saída amostral do gerador G quando a entrada é z, $E_{x\sim p_{\mathrm{x}}(x)}[\log D(x)]$ é a expectativa sobre os dados reais e $E_{z\sim p_z(z)}[\log(1-D(G(z)))$ é a expectativa sobre os dados gerados.

$$G_{\min}(D,G) = E_{x\sim p_{\mathrm{x}}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{2.2}$$

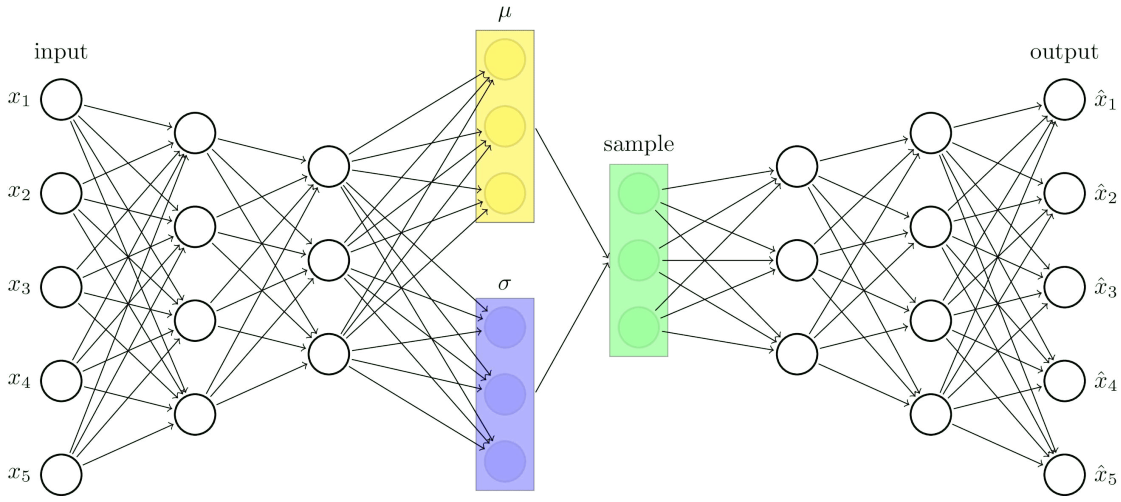
Nessa equação 2.2, se busca maximar a saída D(x) para 1, que é o discriminador, enquanto minimiza o valor de D(G(z)), que é a função que o gerador almeja diminuir, construindo o método de treinamento desse modelo baseado no contraste no

intuito de aprender a produzir os dados que mais se assemelham aos reais. Se esse treinamento cessa, é referente a ter atingido o equilíbrio de Nash, em que ocorre quando o discriminador não consegue mais distinguir entre probabilidades de amostras reais ou falsas (BADU-MARFO *et al.*, 2020).

2.2.5.2 Variational Autoencoders (VAEs)

VAEs, ou Autoencoders Variacionais, representam uma técnica avançada na produção de dados sintéticos que se baseia em redes neurais. Os VAEs são projetados para aprender a representação latente dos dados, permitindo assim a geração de amostras sintéticas que seguem as distribuições dos dados reais. A principal inovação dos VAEs é a capacidade de gerar dados que não apenas se assemelham aos originais, mas também ocupam posições específicas no espaço de características latentes, tornando-os particularmente úteis na geração de dados sintéticos que podem ser ajustados para atender a diferentes necessidades de análise ou pesquisa.

Figura 3 – Representação de uma Variational Autoencoder (VAE). A Figura ilustra a construção de uma arquitetura que se baseia na codificação dos dados de entrada e sua decodificação a partir do treinamento a fim de minimizar as diferenças entre a entrada real e sua decodificação.



Fonte: Riebesell (2023)

Na Figura 3 acima, a representação de um Variational Autoencoder (VAE) introduz dois vetores, média (μ) e desvio padrão (σ), que servem como parâmetros de variáveis aleatórias. Esses parâmetros permitem amostragem estocástica, resultando em variações nas codificações, mesmo para entradas idênticas. O μ controla o centro da codificação, enquanto o σ regula sua dispersão. Isso cria espaços latentes suaves localmente, ideal para interpolação entre classes, mas também permite separação de classes, influenciando diretamente na geração de novos samples e no grau de variação das codificações Shafkat (2018), o que é crucial para o desempenho do modelo. Essa flexibilidade beneficia a reconstrução eficiente de dados de treinamento.

2.2.6 Cópulas

Outro método para a fabricação desses dados são por meio do uso de Cópulas. As Cópulas funcionam como funções matemáticas que separam a distribuição individual de cada variável da estrutura de dependência conjunta entre elas, no sentido de modelar e capturar relações complexas entre variáveis. Isso possibilita que os modelos de Cópulas reproduzam eficientemente as correlações presentes nos dados originais, enquanto as distribuições individuais podem ser ajustadas separadamente. A incorporação de Cópulas na produção de dados sintéticos permite aos pesquisadores preservar relações intervariáveis essenciais e, simultaneamente, gerar amostras que mantenham as características das distribuições individuais.

Em modelos de ML, como tratado em Elidan (2013), este projeto envolveu a execução de uma tarefa semelhante de síntese de dados, com o propósito de contrastar o uso de Cópulas em relação a outras abordagens de geração de dados, tais como SMOTE e autoencoders. Assim, os resultados da pesquisa mostram que as cópulas e suas variações, como a Gaussiana, superou consideravelmente as outras abordagens, as quais preservaram melhor a estrutura de dependência e distribuições marginais dos conjuntos de dados originais.

2.2.7 Ferramentas para Geração de Dados Sintéticos Tabulares

Com o entendimento sobre os métodos para a geração de dados sintéticos, é importante ressaltar como esses métodos são utilizados e quais são suas melhores usos e aplicações em pesquisas. Existem diversas ferramentas que utilizam os métodos explicados 2.2, todavia existem ferramentas que são mais conhecidas no mercado por diversos fatores: reconhecimento, facilidade de implementar, recursos e funcionalidades, código aberto, entre outros fatores.

Nesse sentido, destacam-se:

1. **Data Augmentation com NLP**: Uma das formas mais recorrentes e fáceis de utilizar NLP são utilizando *frameworks* ou bibliotecas prontas para essa finalidade. Assim, facilita o *Data Augmentation*, ou adição de dados, utilizando bibliotecas em Python como: NL-Augmenter GEM-benchmark (2021), NLPAug Ma (2019), TextAugmenter dsfsi (2019) e TextAttack QData (2020). Onde, em praticamente todas essas mencionadas, possuem tarefas e transformações em comum, restando o usuário a utilizar da forma correta para o propósito final.
2. **Synthesized.io:** É uma plataforma de geração de dados sintéticos impulsionada por Application Programming Interface (API). Ela oferece uma maneira rápida de criar dados confiáveis para aprendizado de máquina, desenvolvimento e teste de aplicativos com arquivos de configuração fáceis de usar.
3. **YData-synthetic:** É uma biblioteca em Python para geração de dados sintéticos que utiliza redes neurais profundas para aprender as propriedades dos dados reais e criar dados sintéticos realistas.
4. **Synthetic Data Vault (SDV):** É uma ferramenta OpenSource (código aberto) que consiste em um conjunto de bibliotecas em Python, onde

contém as principais formas para a geração de dados sintéticos e diversas funcionalidades e recursos a fim de produzir dados artificiais de qualidade.

Portanto, são várias as vantagens ao optar pelo SDV em comparação com outras ferramentas, incluindo::

1. **Código aberto, comunidade ativa e disponível gratuitamente:** Garante acesso a diversos recursos de maneira gratuita. Por ser código aberto, permite evoluções e atualizações para manter o desenvolvimento do repositório a fim de melhores desempenhos na criação de dados artificiais.
2. **Diversas formas de sintetizadores:** Existem diversas formas de construção de dados, utilizando as melhores e mais conhecidos métodos para isso, além de poder ajustar os hiperparâmetros para melhorias de desempenhos e vistoria do melhor sintetizador para as características específicas do conjunto de dados.
3. **Métricas de avaliação utilizando SDM:** Possuem diversas maneiras de analisar e validar os dados sintéticos gerados, importante para efetivar o uso da ferramenta.
4. **Capacidade de criação de restrições (constraints):** É possível gerar dados alinhados com o que é esperado, como mencionado em alguns desafios 1.1, como fidelidade de dados e lógica de geração.
5. **Privacidade dos dados:** Anonimizar os dados sensíveis, importante para o não compartilhamento de informações relevantes dos dados históricos originais.
6. **Capacidade de gerar dados de diversas formas:** No SDV, não existe somente a criação de dados sintéticos por tabelas únicas. Pode-se utilizar para criação de tabelas múltiplas como também para séries temporais.

2.3 O Synthetic Data Vault (SDV)

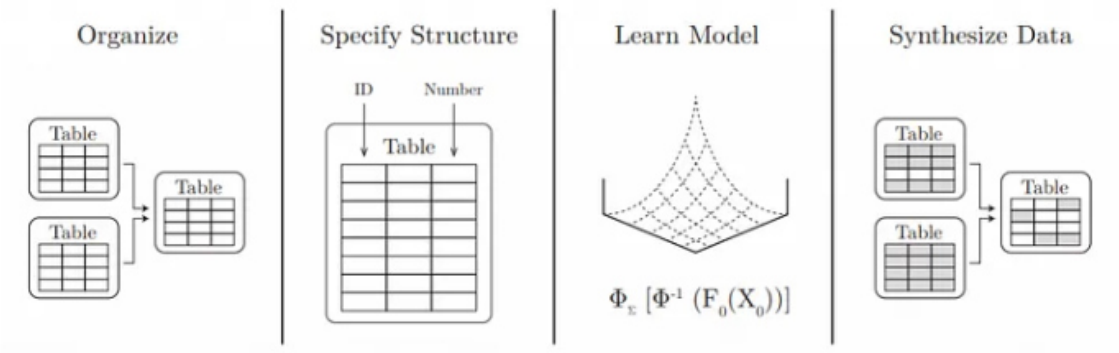
De acordo com o próprio GitHub <https://github.com/sdv-dev/SDV> da ferramenta do SDV (PATKI *et al.*, 2016):

> The Synthetic Data Vault (SDV) is a Python library designed to be your one-stop shop for creating tabular synthetic data. The SDV uses a variety of machine learning algorithms to learn patterns from your real data and emulate them in synthetic data.

Portanto, pela na citação acima, o SDV é conjunto de bibliotecas em Python, desenvolvido por membros do laboratório de pesquisa do MIT, em que contém as principais formas para a geração de dados sintéticos e diversas funcionalidades e recursos com efeito de fabricar dados fictícios.

A Figura 4 generaliza o fluxo de trabalho do SDV. O usuário coleta e formata os dados, especifica a estrutura e os atributos dos dados, executa o sistema de modelagem e, em seguida, usa o modelo treinado para sintetizar novos dados (PATKI *et al.*, 2016).

Figura 4 – Funcionamento do Synthetic Data Vault (SDV). Consiste na organização estrutural da tabela ou multi-tabelas, criação do metadata para especificação dos atributos, entrada dos dados para o sintetizador para criação dos modelos sintetizadores e por fim sua produção de dados sintéticos.



Fonte: Patki *et al.* (2016)

Pela 2.3, em complemento com 2.2.7 e com a documentação disposta no GitHub (<https://github.com/sdv-dev/SDV>), o SDV propõe:

- Criar dados sintéticos usando ML. O SDV oferece diversos modelos, desde métodos estatísticos clássicos (GaussianCopula) até métodos de Deep Learning (CTGAN). Gere dados para tabelas individuais, tabelas interconectadas ou tabelas sequenciais.
- Pré-processar, anonimizar e definir restrições de negócios (constraints). Ademais, permite controlar o processamento de dados para melhorar a qualidade dos dados sintéticos. Possibilidade de escolher entre diferentes tipos de anonimização e definir regras de negócios na forma de restrições lógicas.
- Avaliar e visualizar os dados reais frente aos dados sintéticos usando diversas métricas.
- Diagnosticar problemas e criar um relatório de qualidade para obter visões adicionais.

2.3.1 Principais Recursos e Funcionalidades do SDV

Pela apresentação do SDV em 2.3 e pelas qualidades da ferramenta exposta em 2.2.7, esta subseção resume e qualifica os principais recursos e finalidades do SDV

- **Tipos de dados suportados:** O SDV pode lidar com diferentes tipos de dados, dependendo da estrutura e da natureza dos dados originais. Os tipos de dados suportados são:
  - **Single table (Tabela única):** Dados que consistem em uma única tabela, onde cada linha representa uma observação independente e cada coluna representa uma variável ou característica.
  - **Multi table (Tabela múltipla):** Dados que consistem em várias tabelas relacionadas entre si por meio de chaves primárias e estrangeiras. Um exemplo para esse formato é a utilização de banco de dados.
  - **Sequential (Séries temporais):** Dados que consistem em sequências

ordenadas de eventos ou observações ao longo do tempo.

- **Conjunto de Dados (datasets):** O SDV fornece datasets para cada tipo de dado pelo módulo *sdv.datasets.demo* para experimentações rápidas e entedimento do desenvolvimento e aplicação da ferramenta. Todavia, também é possível adicionar conjunto de dados do domínio do usuário.
- **Pré-processamento:** Antes de gerar os dados sintéticos, é necessário realizar algumas etapas de pré-processamento para preparar os dados originais para o SDV. Essas etapas são:
  - **Criação do metadata:** O metadata é um objeto que contém informações sobre a estrutura e o tipo dos dados originais, tais como o nome das tabelas e colunas, as chaves primárias e estrangeiras, os tipos de dados e os domínios válidos.
  - **Criação das constraints (opcional):** As constraints são restrições lógicas que devem ser satisfeitas pelos dados sintéticos, tais como valores únicos, intervalos válidos, dependências funcionais ou relações entre tabelas. É cxdes3w definida como uma solução para regras determinísticas, ou regras de negócios.
- **Anonimização dos Dados:** Uma das principais vantagens do SDV é que ele permite anonimizar os dados originais, removendo ou substituindo as informações sensíveis ou identificáveis dos indivíduos ou entidades presentes nos dados. Isso é configurado por meio do metadata, definindo a variável "*pii*".
- **Sintetizadores:** Os sintetizadores são os modelos de ML que o SDV usa para aprender as propriedades estatísticas e relacionais dos dados originais e gerar os dados sintéticos. O SDV oferece diferentes sintetizadores para cada tipo de dado suportado, tais como:
  - **Single table (Tabela única):** O SDV oferece quatro sintetizadores para

dados de uma única tabela: GaussianCopula (e em adição GaussianCopula* (FastML)), CTGAN, CopulaGAN e TVAE. Esses sintetizadores usam diferentes técnicas para modelar a distribuição conjunta das variáveis da tabela e gerar amostras sintéticas a partir dela 2.2.

  - **Multi table (Tabela múltipla):** O SDV fornece um sintetizador específico para analisar correlações entre tabelas, chamado HMASynthesizer.
  - **Sequential (Séries temporais):** Para os dados sequenciais, o SDV utiliza o sintetizador chamado PARSynthesizer. Ele utiliza técnicas captar a dinâmica temporal dos dados e gerar sequências sintéticas com a mesma frequência e duração dos dados originais.
  - Ademais, para cada sintetizador, é possível realizar melhorias com ajustes de hiperparâmetros. Além disso, o SDV fornece outros sintetizadores para cada tipo de dados denominado de *Enterprise*, sendo uma plataforma para empresas e consequentemente paga, com mais recursos especifícos para tratamentos dos dados originais.

- **Geração de Dados Sintéticos por Amostragem Condicional:** É possível gerar amostragens de dados sintéticos a partir da criação dos sintetizadores, como também é possível gerar amostras de dados passando parâmetros condicionais. Por exemplo, gerar dados sintéticos cujo uma coluna "A"deve ter um valor específico.
- **Avaliação e visualização dos dados sintéticos:** Após gerar os dados sintéticos, é importante avaliar e visualizar os resultados para verificar se eles preservam as propriedades estatísticas e relacionais dos dados originais e se satisfazem as constraints (restrições lógicas) definidas. O SDV oferece um framework para analisar, visualizar e diagnosticar esta produção, a partir do SDM.

2.3.2 Algoritmos para a Geração de Dados Sintéticos Tabulares por meio do SDV (Sintetizadores)

Esta seção foca na explicação dos principais sintetizadores para a construção de novos dados, utilizando técnicas de ML a partir de métodos estatísticos e redes neurais. Os sintetizadores no SDV são técnicas de ML envolvendo redes neurais que utilizam dados históricos para produzir novos dados distintos e coerentes, introduzido na seção 2.3.2.

2.3.2.1 GaussianCopula

O sintetizador GaussianCopula é baseado em funções Cópula Elidan (2013), que são funções matemáticas que permitem descrever a distribuição conjunta de múltiplas variáveis a partir das suas distribuições marginais. O objetivo desse sintetizador, então, é transformar as variáveis originais em variáveis padronizadas com distribuição normal usando a Cumulative Distribution Function (CDF), e depois ajustar uma distribuição normal multivariada aos dados transformados.

A CDF é uma função matemática que descreve a probabilidade acumulada de uma variável aleatória tomar um valor menor ou igual a um valor específico. Ela, basicamente, é usada para transformar variáveis em uma distribuição normal padrão. Na equação 2.3, enuncia a fórmula para realizar essas transformações necessárias:

$$F(x) = \int_{-\infty}^{x} f(t)dt \tag{2.3}$$

onde f(t) é a função densidade de probabilidade de X e o intervalo de integração é de $-\infty$ a x. Esta equação expressa a CDF como a integral da função densidade de probabilidade até o valor especificado.

Por fim, a aplicação em função da distribuição normal multivariada é parametrizada por uma matriz de correlação, que captura as dependências lineares entre as

variáveis. Para gerar dados sintéticos, basta amostrar da distribuição normal multivariada e aplicar a função CDF das distribuições marginais originais às amostras (HOUSSOU *et al.*, 2022).

2.3.2.2 GaussianCopula: configuração FastML*

O GaussianCopula é um sintetizador rápido se for comparado aos demais modelos oferecidos pelo SDV, na maioria dos conjuntos de dados testados. Além disso, ele possui uma versão adaptada chamada *FastML*, que são ajustes nos hiperparâmetros do GaussianCopula. A diferença entre eles ocorre na utilização da distribuição normal, no reforço da utilização de máximos e mínimos de intervalos e da aproximação dos valores numéricos. Dessa forma, na própria documentação do SDV Patki *et al.* (2016), o GaussianCopula com a pré-configuração FastML é indicado como utilização padrão pois é um modelo acessível, que exige de menos conhecimento sobre os dados e que tem um ótimo desempenho.

Em conclusão, quando for citado a utilização do GaussianCopula com a configuração FastML, será indicada na pesquisa como **GaussianCopula***.

2.3.2.3 Conditional Tabular GAN (CTGAN)

O sintetizador CTGAN é baseado em redes adversariais generativas (GANs) 2.2.5.1, que são modelos de aprendizado profundo que consistem em duas redes neurais: um gerador e um discriminador. O gerador tem como objetivo produzir dados sintéticos que sejam indistinguíveis dos dados reais, enquanto o discriminador tem como objetivo distinguir entre os dados reais e os sintéticos. Os dois redes competem entre si em um jogo minimax, até que o gerador consiga enganar o discriminador.

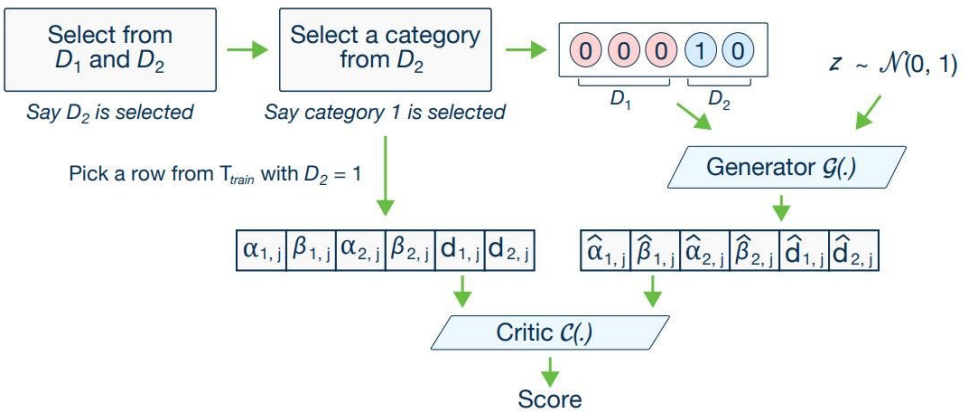
O CTGAN é uma variação do modelo GAN original, que incorpora algumas modificações para lidar melhor com dados tabulares, por conta dos tratamentos em dife-

renciações que existem em dados tabulares: discretos, contínuos, categóricos e booleanos, como distribuições não gaussianas e multimodais e colunas discretas desbalanceadas. Entre essas modificações estão (XU *et al.*, 2019):

1. **Normalização Específica por Modo (Mode-specific Normalization):** Para lidar com colunas contínuas de dados complexos, o CTGAN utiliza uma normalização específica por modo. Isso envolve identificar os diferentes modos em uma coluna, usando um modelo estatístico para amostrar valores com base nessas distribuições.
2. **Gerador Condicional e Treinamento por Amostragem (Conditional Generator and Training-by-Sampling):** O CTGAN introduz um gerador condicional que é treinado para produzir amostras que correspondam à condição fornecida e aprende a representar a distribuição condicional real dos dados. Assim, ele cria amostras com base em uma condição específica e garante que todas as categorias de atributos discretos sejam amostradas de forma equilibrada durante o treinamento.
3. **Estrutura de Rede (Network Structure):** Para capturar as relações entre as colunas de dados, o CTGAN utiliza redes totalmente conectadas em seu gerador e crítico. Ambas têm duas camadas ocultas totalmente conectadas. No gerador, são aplicadas ativação ReLU e normalização em lote, enquanto no crítico, é usada a função leaky ReLU e Dropout em cada camada oculta.
4. **Arquitetura:** O CTGAN é treinado usando a perda WGAN (Wasserstein GAN) com penalização de gradiente e otimizador Adam com uma taxa de aprendizado específica.

A Figura 5, criada pelos autores do CTGAN Xu *et al.* (2019), representa a arquitetura para a modelagem dos dados tabulares e geração dos dados sintéticos por meio das adaptações de GANs introduzida pelo pontos descritos acima.

Figura 5 – Representação de um Conditional Tabular GAN (CTGAN). É um modelo derivado da GAN e entre suas diferenças está a dependência condicional das colunas do dataset.



Fonte: Xu *et al.* (2019)

2.3.2.4 CopulaGAN

CopulaGAN é uma variação do CTGAN que utiliza a transformação baseada na CDF que as GaussianCopulas 2.3.2.1 aplicam para facilitar a tarefa do CTGAN de aprender os dados. A ideia é transformar as variáveis originais em variáveis padronizadas com distribuição uniforme usando a função inversa da CDF, e depois aplicar o modelo CTGAN aos dados transformados. Para gerar dados sintéticos, basta amostrar do modelo CTGAN e aplicar a função CDF das distribuições marginais originais às amostras.

2.3.2.5 Time-Varying Autoencoder (TVAE)

O sintetizador TVAE é baseado em VAEs 2.2.5.2, que são modelos de aprendizado profundo que consistem em duas redes neurais para modelar a probabilidade dos dados Xu *et al.* (2019): um encoder e um decoder. O encoder tem como objetivo mapear os dados de entrada em um espaço latente de menor dimensão, que são dados

codificados como pontos ou distribuições. O decoder tem como objetivo reconstruir os dados de saída a partir do espaço latente. O objetivo é aprender uma representação compacta e informativa dos dados, que possa ser usada para gerar novos dados.

O TVAE incorpora modificações para lidar melhor com dados tabulares. Entre essas modificações estão: o uso de uma camada de *embedding* (ou camada de incorporação) para codificar as variáveis categóricas, o uso de uma rede condicional para permitir a geração de dados com condições pré-definidas e o uso de uma perda de tripla para preservar a similaridade dos dados. Também utiliza o otimizador Adam nos treinamentos e uma taxa de aprendizagem de $1e^{-3}$

2.3.3 Metodologias para Avaliação dos Dados Sintéticos e Classificadores

Esta seção explica e justifica a maneira de como é avaliada e visualizada a criação de novas amostras artificiais. O SDM é uma biblioteca de código aberto para avaliação de dados sintéticos tabulares, criado pelo próprio SDV. Além da análise dos dados sintéticos, avalia-se também a eficácia da sua geração em tarefas de classificação com ML. Esta avaliação abrange o desempenho dos modelos em diversas estratégias, utilizando métricas como accuracy, precision, recall, f1-score e ROC-AUC para medir a eficácia e o desempenho.

2.3.3.1 SDMetrics

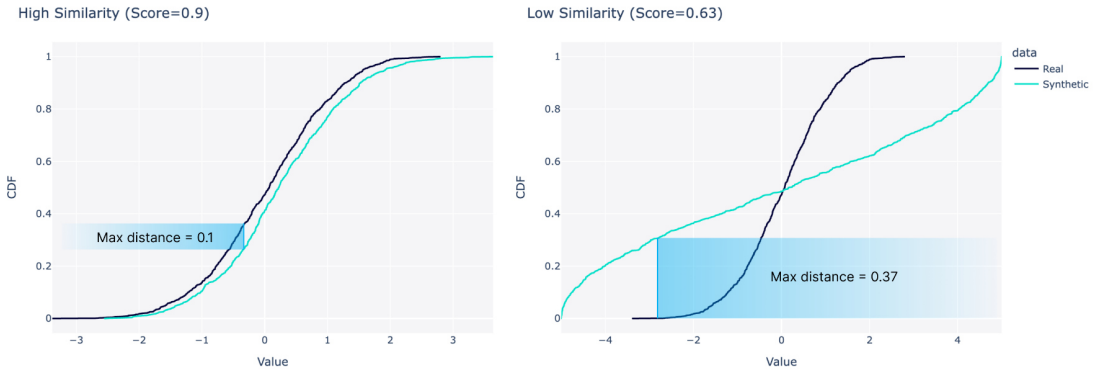
Como mencionado na seção acima, o SDM fornece diversas formas de avaliar cada tipo de dado gerado, utilizando métricas diferentes para ser efetiva nessa avaliação. Assim, são duas formas mais usuais de utilizar a API do SDM, pelo uso dos métodos: *evaluate_quality* e *run_diagnostic*.

- **evaluate_quality:** O relatório de qualidade, (*evaluate_quality*), captura os formatos das colunas, as tendências dos pares de colunas e a cardina-

lidade. Ele avalia a qualidade dos seus dados sintéticos em termos de formatos de colunas e correlações. Sobre os dados avaliados, são utilizados KSComplement para o tipo numerical e datetime e TVComplement para boolean e categorical.

- **KSComplement:** Avalia a semelhança entre uma coluna real e uma coluna sintética em termos de formatos da coluna utilizando a estatística de Kolmogorov-Smirnov. Ela é calculada pela distribuição numérica em sua função de distribuição acumulada CDF. O valor da estatística de Kolmogorov-Smirnov é a maior diferença entre as duas CDFs, que varia entre 0 a 1, com o valor invertido no SDM (DATACEBO, INC., 2023).

Figura 6 – Representação da utilização da métrica do Synthetic Data Metrics: KSComplement. As Figuras representam a utilização da estatística de Kolmogorov-Smirnov e sua distribuição numérica convertida pela CDF, representada por essas distribuições. A estatística KSComplement é a diferença máxima entre as curvas.



Fonte: DataCebo, Inc. (2023)

A Figura 6 exemplifica essa estatística, onde quanto maior o valor retornado, melhor é o dado sintético criado.

- **TVComplement:** Avalia a semelhança entre uma coluna real e uma coluna sintética em termos de formatos da coluna utilizando a Total Va-

riation Distance (TVD). O escore que representa essa diferença é dado pela equação 2.4, que calcula a dissimilidade entre as distribuições de frequências:

$$\text{TVD(R, S)} = \frac{1}{2}\dot{\sum}_{w\in\Omega}|R_w - S_w|. \tag{2.4}$$

Ela compara as diferenças entre as frequências reais e sintéticas R_w e S_w, em que "w"é o valor das colunas correspondentes, Ω representa todos as possibilidades de valores da coluna e, por fim, quanto menor TVD(R, S), mais semelhantes são as colunas.

- **CorrelationSimilarity:** Essa métrica avalia a correlação entre pares de colunas numéricas e computa a similaridade entre os valores dos dados reais e sintéticos. Pode ser utilizada por dois coeficientes: Pearson e Spearman para calcular essa correlação. No projeto, é utilizado o padrão Pearson.

  A equação 2.5 apresenta o cálculo sobre a métrica:

$$\text{score} = 1 - \frac{1}{2}\sum|S_{A,B} - R_{A,B}|. \tag{2.5}$$

  Pela equação abordada 2.5 DataCebo, Inc. (2023), S e R são valores dos dados reais e sintéticos. A e B representa as colunas do dataset. O escore fica entre -1 e +1 que indica a taxa de correlação entre esses dados.

- **ContigencySimilarity:** Essa métrica avalia a correlação entre pares de colunas categóricas e computa a similaridade entre os valores dos dados reais e sintéticos. A correlação, então, é representada pela equação 2.6 abaixo:

$$\text{score} = 1 - \frac{1}{2} \sum_{\alpha \in A} \sum_{\beta \in B} |R_{\alpha,\beta} - S_{\alpha,\beta}|. \tag{2.6}$$

Pela equação abordada 2.6 DataCebo, Inc. (2023), α e β representam as possibilidades do dados categóricos nas colunas A, B e R, S representam suas frequências frente a essas respectivas colunas, cujo valores de escore variam entre 0 e 1.

- **run_diagnostic:** Verifica se as linhas sintéticas são cópias puras dos dados reais, se os dados sintéticos cobrem todo o intervalo de valores e se os dados sintéticos aderem aos intervalos originais. Sobre os dados avaliados, são utilizados RangeCoverage para o tipo numerical e datetime e CategoryCoverage para boolean e categorical.
  - **RangeCoverage:** Esta métrica avalia se uma coluna sintética abrange a faixa completa de valores presentes em uma coluna real. Dessa forma, o escore para o RangeCoverage é dada pela equação 2.7:

$$\text{score} = 1 - \left[\max\left(\frac{\min(s) - \min(r)}{\max(r) - \min(r)}, 0 \right) + \max\left(\frac{\max(r) - \max(s)}{\max(r) - \min(r)}, 0 \right) \right]. \tag{2.7}$$

A fórmula 2.7 DataCebo, Inc. (2023) calcula o quão próximos os valores mínimos e máximos de s estão dos valores mínimos e máximos reais em r, sendo r e s são as colunas reais e sintéticas, respectivamente.

  - **CategoryCoverage:** Esta métrica avalia se uma coluna sintética abrange todas as categorias possíveis presentes em uma coluna real. A representação dessa métrica é dada pela equação 2.8 DataCebo, Inc. (2023):

$$\text{score} = \frac{C_s}{C_r}. \tag{2.8}$$

Logo, ela calcula o número de categorias únicas, c, presentes na coluna real r. Em seguida, calcula o número dessas categorias presentes na coluna sintética s. A métrica retorna a proporção de categorias reais que estão nos dados sintéticos representada na equação

2.3.3.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

O t-Distributed Stochastic Neighbor Embedding (t-SNE) é um algoritmo de visualização de dados de altas dimensões. Nele, se utiliza um kernel Gaussiano para converter pontos em altas dimensões para probabilidades de conexões (P) e um kernel t-Student, com um grau de liberdade, para representar as probabilidades de conexões entre os pontos em baixas dimensões (Q), no espaço mapeado (MAATEN; HINTON, 2008).

Nessa pesquisa, o t-SNE é utilizado para visualizar a distribuição dos dados sintéticos frente aos dados reais para ter uma noção visual se a geração dos dados sintéticos se aproxima e é coerente com os dados reais, pois é importante que seja.

2.3.3.3 Métricas para Avaliação de Classificações em Machine Learning

Para contextualizar, uma tarefa de classificação em ML envolve a atribuição de rótulos ou categorias a objetos ou instâncias com base em suas características ou atributos. Dessa forma, existe o uso de um conjunto de dados, esse conjunto de dados é tratado e manipulado para servir como entrada para modelos de ML, sendo a coluna alvo (target) do dataset separada das demais colunas. Por fim, o modelo é treinado e testado para identificar se sua previsão coincide com o dado real. Existem várias métricas para poder avaliar a eficácia dos modelos de ML.

Nesse sentido, após a criação dos novos dados e sua validação por meio do SDM, abordagens são realizadas por meio de tarefas de classificação com ML para avaliar o uso dos dados sintéticos a partir de métricas conhecidas.

Para facilitar o entendimento sobre as métricas, a tabela abaixo mostra a base sobre o contexto das métricas que são utilizadas para avaliar os algoritmos de classificação, que é por meio da matriz de confusão sobre a expectativa da realidade versus acerto.

Tabela 1 – Explicação sobre Matriz de Confusão. Indicadores sobre suas classificações para construção de métricas sobre a ótica de Previsão versus. Realidade.

		Previsão	
		Sim	Não
Realidade	Sim	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Fonte: Elaborado pelo autor.

- **Acurácia (Accuracy):** A acurácia, representada na equação 2.9, mede a proporção de previsões corretas em relação ao número total de previsões. Essa métrica dá margem para identificar o quão preciso o modelo é, na ótica sobre as previsões corretas no geral, em que sua equação 2.9 é dada por:

$$\text{Accuracy} = \frac{\text{VP} + \text{VN}}{\text{Total}} \tag{2.9}$$

  O escore accuracy, portanto, é uma fração sobre a soma dos Verdadeiros Positivos com Verdadeiros Negativos sobre o Total de amostras.

- **Precisão (Precision):** A precisão, representada na equação 2.10, mede a proporção de previsões positivas corretas em relação ao número total de previsões positivas feitas pelo modelo. Dessa forma, é uma análise

sobre a qualidade das previsões positivas pelo modelo, identificando quantas previsões positivas realmente eram corretas. Assim, seu cálculo é baseado na equação 2.10:

$$\text{Precision} = \frac{\text{VP}}{\text{VP} + \text{FP}} \tag{2.10}$$

Dessa forma, precision é calculada pela fração entre os Verdadeiros Positivos sobre a soma entre Verdadeiros Positivos mais Falsos Positivos.

- **Revocação (Recall):** A revocação, representada na equação 2.11, mede a proporção entre verdadeiros positivos em relação ao número de positivos reais, ou seja, quantos casos positivos foram detectados pelo modelo. A representação do recall, ou revocação, é dada pela equação 2.11:

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{FN}} \tag{2.11}$$

Portanto, a métrica recall é composta pela fração entre os Verdadeiros Positivos sobre a soma entre Verdadeiros Positivos mais Falsos Negativos.

- **F1-Score:** O F1-score, representada na equação 2.12 abaixo, é a média harmônica entre a precisão (precision) e a revocação (recall):

$$\begin{aligned} \text{F1-Score} &= 2 \cdot \frac{\mathit{Precision} \cdot \mathit{Recall}}{\mathit{Precision} + \mathit{Recall}} \\ \text{F1-Score} &= 2 \cdot \frac{\text{VP}}{2 \cdot \text{VP} + \text{FP} + \text{FN}} \end{aligned} \tag{2.12}$$

O f1-score é uma métrica utilizada para análise que precisam considerar a qualidade das previsões positivas e a capacidade do modelo de identificar os casos positivos.

- **ROC-AUC:** A curva ROC (Receiver Operating Characteristic) é uma representação gráfica de uma área que indica a capacidade de um modelo em distinguir classes positivas e negativas. Nesse sentido, a ROC-AUC avalia o desempenho geral de um modelo a partir do uso das equações 2.13:

$$\begin{aligned} \text{TVP} &= \frac{\text{VP}}{\text{VP}+\text{FN}} \\ \text{TFP} &= \frac{\text{FP}}{\text{FP}+\text{VN}} \\ \text{ROC-AUC} &= \int_0^1 \text{TVP}(f)\,\text{d}(\text{TFP})(f) \end{aligned} \tag{2.13}$$

  A equação 2.13 representa o cálculo da área sobre a curva ROC-AUC, que é uma integral da TVP (Taxa de Verdadeiros Positivos) em função da Taxa de Falsos Positivos (TFP) sobre o limite de classificação que varia entre 0 e 1, cujo valor de ROC-AUC esteja mais próximo de 1, melhor o modelo é capaz de distinguir entre classes positivas e negativas.

Em resumo, o uso dessas estatísticas são para verificar se modelos de ML que foram treinados por meio de inferências conjuntas por estratégias distintas desempenham valores melhores a essas métricas quando comparados a experimentações sem o uso das amostras sintéticas. Assim, é importante entender cada métrica pois, a depender do conjunto de dados utilizado, é pertinente realizar a avaliação com base em uma métrica específica, embora todas sejam calculadas durante as rotinas de classificação.

3 METODOLOGIA

Este capítulo se concentra na contextualização, abordagem e solução da problemática sobre a pesquisa realizada, a qual consiste na geração de dados sintéticos a partir da obtenção de conjuntos de dados diversificados em questões de tamanhos de linhas e colunas, tipo e atributo dos dados e contextos abordados.

Dessa forma, o objetivo dessa seção é fornecer informações detalhadas sobre como o SDV é implementado para a criação de novos dados, como também mencionar quais métodos são aplicados e avaliados dentro dessas implementações realizadas. Além disso, mencionar sobre quais foram as estratégias abordadas para evidenciar o uso das amostras artificiais geradas.

Em suma, o capítulo é dividido em: abordagem de pesquisa; instalação, configuração e utilização do SDV e seus recursos, bem como as formas para geração dos dados sintéticos e as estratégias que são utilizadas para analisar a produção dos dados artificiais.

3.1 Abordagem da Pesquisa

Como mencionado na introdução 1, onde se destaca a importância da geração dos dados sintéticos frente a ausência de dados reais por diversos fatores, nele também ressalva a falta de estudos ou pesquisas concretas sobre as melhores formas de criação de dados sintéticos para experimentações em tarefas de ML.

Nesse intuito, realiza-se um trabalho exploratório a fim de identificar estudos parecidos e criar hipóteses sobre o real impacto que a criação dos dados sintéticos pode realizar a respeito do conjunto de dados (dataset). Em função disso, inicia-se a coleta de estudos diversos sobre formas de geração de dados sintéticos e na coleta de dados para a realização de experimentações, para o propósito de efetivar o que foi mencionado nos objetivos da pesquisa, na seção 1.2 e 1.2.1.

Assim, a conduta sobre a pesquisa envolve uma abordagem quantitativa, onde são utilizados sete conjuntos de dados de diferentes contextos, como de saúde, ramo imobiliário, grau de satisfação, qualidade de vinhos e classificaçao de grãos de arroz com diferentes colunas e tamanhos de linhas com o intuito de demonstrar que diferentes experimentações em tarefas de classificação utilizando algoritmos de ML trazem melhorias para as estratégias que são abordadas com o uso desses dados sintéticos.

3.2 Instalação e Configuração do SDV

A instalação do SDV é simples. Necessita-se de poucas instalações de bibliotecas do Python e é importante, a efeito de estudos, experimentações e resultados vigentes, replicar as mesmas versões dos pacotes devido a ser uma ferramenta Open-Source que está em constante evolução e, portanto, é passível de mudanças em todo o contexto de desenvolvimento.

Neste âmbito, o próprio autor desta pesquisa experimentou essas mudanças de versões. Como a ferramenta foi utilizada por um período suficientemente longo, implementações e métodos foram alterados ao longo do tempo e códigos tiveram que ser adaptados de acordo com a versão. Por exemplo, as restrições 3.3.2 e algoritmos adicionais para a geração de dados sintéticos **??** foram incluídos.

Os experimentos foram feitos em notebooks pelo *JupyterLab*, na versão 3.10.9 do Python.

Tabela 2 – Bibliotecas e Versões em Python Utilizadas na Pesquisa. Pela criação do ambiente virtual Conda, existe uma forma simples para replicar o projeto com as versões indicadas.

Pacote	**Versão**
ctgan	0.7.3
imbalanced-learn	0.10.1
matplotlib	3.7.0
nb-black	1.0.7
numpy	1.23.5
pandas	1.5.3
ydata_profiling	v4.5.1
plotly	5.15.0
sdv	1.2.0
seaborn	0.12.2
scikit-learn	1.2.1
ucimlrepo	0.0.3

Fonte: Elaborado pelo autor.

Dessa forma, a 2 referencia os pacotes e versões que são utilizadas na pesquisa. No repositório do GitHub Barros (2023), que pode ser utilizado para realizar a replicação do projeto, existe um arquivo de formato *txt* chamado *requirements.txt*. Uma recomendação para garantir o sucesso das instalações e replicação do projeto é utilizando a referência do código que está no apêndice, por Command Line Interface (CLI), em que consiste na criação de um ambiente virtual utilizando o Anaconda (Conda). No apêndice A, mostra a execução desse ambiente virtual.

3.3 Geração dos Dados Sintéticos pelo Synthetic Data Vault (SDV)

Este capítulo recorre a geração dos dados sintéticos a partir do uso do SDV. Neste recurso, é importante ressaltar as etapas que retrocedem a criação dos novos dados, em que existe um "pre-processamento"desses dados reais para o tratamento e geração coerente dos dados sintéticos. Dessa forma, é importante definir a construção do metadados, *constraints* e os algoritmos de ML que serão utilizados para finalmente obter os novos dados.

3.3.1 Criação dos Metadados

o SDV utiliza um componente chamado *Metadata*, ou Metadado em português, que contém informações detalhadas sobre os dados originais e os dados sintéticos. O Metadata é muito importante para documentar, validar e processar a geração de dados sintéticos. Dessa forma, o Metadata é uma estrutura da forma dicionário em Python, mais propriamente descrito como um JavaScript Object Notation (JSON), que contém os tipos de dados para cada coluna do conjunto de dados (dataset) reais. Por fim, é importante destacar sobre o Metadata esses pontos abaixo:

- **Descrição dos Dados Reais e Sintéticos:** O Metadata mantém a estrutura sobre os dados que são passados. Dessa forma, é importante configurar-lo da forma coerente ao que se espera sobre a geração dos dados sintéticos. Assim, é imprescindível observar qual tipo do dado colunar que precisa ser gerado para que se mantenha da mesma forma.
- **Validação dos Dados Sintéticos:** Consequência do ponto anterior, a geração dos dados sintéticos manterá o tipo e formato do dado da forma como foi construído o Metadata.
- **Privacidade e Anonimização:** Existem 6 opções para o Synthetic Data Types (SDtypes), que são formatos possíveis para armazenar as informações sobre o contexto de cada coluna. Dentre elas, estão: *boolean*, *categorical*, *datetime*, *numerical*, *id* e *other* (DOCS, 2023).
- **Melhoria do Processamento dos Dados:** Quando construído o Metadata com as informações do SDtypes, é possível melhorar o processamento e diminuir drasticamente o tempo para a geração desses dados. Um exemplo disso é quando uma coluna é convertida para "categorical"no lugar de "object", ou quando uma coluna do tipo "numerical"utiliza um *Int8* em alternativa a *Int64* Docs (2023), o que diminui consideravelmente

a alocação de memória do dado e consequentemente melhora a velocidade da criação dos modelos.

Como mencionado acima, o SDtypes é muito importante para o tratamento do Metadata. Então abaixo está descrito como pode ser realizada essa configuração, de acordo com a própria documentação do SDV (DOCS, 2023):

1. **Boolean:** Configuração padrão para valores True e False, apenas utiliza-se *boolean* no Metadata.
2. **Categorical:** Configuração padrão para valores do tipo string, apenas utiliza-se *categorical* no Metadata.
3. **Datetime:** Campo das datas. Neste é possível adicionar uma chave-valor "datetime_format"(pela ideia do JSON), para definir o formato do tipo "date"sintéticos que serão criado, de acordo com o *strftime*.
4. **Id:** Contexto de índices. Neste é possível adicionar uma chave-valor "regex_format"por meio de expressões regulares para definir a forma de como os índices são representados nos dados originais.
5. **Numerical:** Neste é possível configurar o 'computer_representation' para essas opções: 'Float', 'Int8', 'Int16', 'Int32', 'Int64', 'UInt8', 'UInt16', 'UInt32', 'UInt64'.
6. **Other:** Outros formatos de texto, string, como: 'address', 'phone_number', 'ssn', 'email'. Também é possível informar por meio da chave 'pii' se a coluna em questão é formada por dados sensíveis e no caso precisa gerar dados anonimizados.

3.3.2 Criação das Regras Determinísticas por Constraints

O SDV tem um recurso chamado *Constraints*, que são formas de restringir a geração do dado sintético no intuito de modelar-lo. Esse recurso é uma ferramenta poderosa, pois pode garantir o sucesso da geração do dado sintético no sentido de

regras de negócios ou regras determinísticas, porque as constraints, ou restrições, estão diretamente relacionadas ao Metadata 3.3.1, em que garante na própria criação dos modelos de ML, a forma e estrutura de como o dado de saída deve ser.

Sobre as restrições, pode-se falar que possuem 9 restrições pré-definidas, que basta instanciar e utilizá-las, mas também o SDV permite restrições manuais, permitindo regras bem mais complexas. Das 9 constraints existentes, as figuras 7 e 8 mostram como estão sendo definidas e como servem para os modelos, sendo diferenciadas pela dependência colunar: se a constraint depende somente da coluna que está sendo configurada, ela é chamada de *'Single Column Constraint'*, caso contrário, *'Multi Column Constraint'*, a qual existe uma dependência com outras colunas para realizar a regra determinística.

Figura 7 – Implementação de Regras Determinísticas (Constraints): Classes destinadas à Aplicação em Colunas Únicas. Existem soluções pré-configuradas para a criação de regras determinísticas em colunas, a fim de restringir ou facilitar a geração de novos dados em colunas independentes.

Constraint Class	Description	Example
Positive	All the values in the column must be >0	The `room_rate` must be positive
Negative	All the values in the column must be <0	The `credit_balance` must be negative
ScalarInequality	All the values in the column have a fixed lower or upper bound	All `checkin_date` values must occur on or after Jan 1, 2020
ScalarRange	All the values in the column have fixed lower and upper bounds	All values in `amenities_fee` must be between 0 and 500.00
FixedIncrements	All the numerical values are increments of a whole number	All values in `salary` must be divisible by 1000

Fonte: SDV (2023).

Figura 8 – Implementação de Regras Determinísticas (Constraints): Classes destinadas à Aplicação em Múltiplas Colunas. Existem soluções pré-configuradas para a criação de regras determinísticas entre colunas, a fim de restringir ou facilitar a geração de novos dados em colunas dependentes de outras.

Constraint Class	Description	Example
FixedCombinations	No shuffling is allowed other than what's already observed in the data	The `city` and `country` values cannot be shuffled to create new permutations.
OneHotEncoding	The original data columns represent a one hot encoding scheme	Exactly 1 of the following columns has a `1` in each row: `not_subscribed`, `basic_subscriber`, `premium`
Inequality	The value in one column must always be greater than the other	The `checkout_date` must always be after the `checkin_date`
Range	The value in one column is bounded by the values in other columns	The `parent_age` must be in between `child_age` and `grandparent_age`
* ChainedInequality	A chain of 2 or more columns in an inequality	`purchase_date` < `start_date` < `end_date` < `expiration_date` < `termination_date`

Fonte: SDV (2023).

3.3.2.1 Adição das Constraints nos Modelos de Geração de Dados Sintéticos

Como mostrado em 3.3.2, existem nove constraints pré-definidas. No caso do projeto em questão, foram-se usadas somente essas para restringir a geração dos dados. Todavia, caso necessário, é possível criar novas regras customizadas.

A efeito de demonstração, o código abaixo ressalta como é realizado a construção de uma constraint, em que é definido qual constraint deseja utilizar, no caso *ScalarRange* 7. Essa constraint foi utilizada no experimento do conjunto de dados

(dataset) 1 **??**.

No código sobre as constraints, que pode ser analisada no Apêndice A, o sintetizador para a geração dos dados sintéticos é instanciado, que no caso é o CTGAN, e adiciona aos parâmetros a devida constraint. Assim, garante que na geração dos dados sintéticos, a coluna "age"terá apenas valores entre 29 e 77 (idade).

3.3.3 Criação dos Dados Sintéticos por Diferentes Algoritmos de Machine Learning (Sintetizadores)

Após configurar o metadata 3.3.1 e as constraints que são referidas nas seções 3.3.2 e 3.3.2.1, a parte de pré-processamento dos dados é concluída e pode-se partir para a geração dos dados sintéticos a partir de diferentes modelos de ML. Como mencionado na seção 2, existem diversas formas de geração dos dados pelo SDV, elas são:

1. CopulaGAN
2. GaussianCopula
3. GaussianCopula*: GaussianCopula com mudanças nos hiperparâmetros (FastML)
4. TVAE
5. CTGAN

Portanto, a ideia é testar cada modelo de geração, chamado de *sintetizador*, para analisar qual é o melhor modelo que se ajusta aos dados reais, utilizando métricas estatísticas e visualizações das variações entre dados reais e sintéticos por meio do SDM (DATACEBO, INC., 2023).

3.4 Estratégias sobre os Dados Sintéticos gerados para a Avaliação em modelos de Classificação

Essa seção explicita a forma de como os dados sintéticos impacta e valida os objetivos da pesquisa. Existem duas formas de validar a eficiência dos dados sintéticos com SDV em tarefas de classificação, utilizando modelos simples de ML, entre elas estão:

1. **TBTR:** Treinar com os dados balanceados, adicionando dados sintéticos no treino e testar com os dados reais Lee *et al.* (2021), Lee *et al.* (2022), Yoon *et al.* (2019), Esteban *et al.* (2018). Essa estratégia consiste em entender se a adição de novos dados auxiliam na melhoria do desempenho dos modelos de classificação.
2. **TSTR:** Treinar apenas com dados sintéticos, testar com dados reais. Essa estratégia consiste em entender se a geração dos dados sintéticos são suficientemente bons para uma proposta de gerar dados que são difíceis de serem coletados ou extraídos por parte da segurança e privacidade dos dados.

Além dessas duas estratégias, existem mais duas que servem para comparar e certificar a qualidade dos dados, que são:

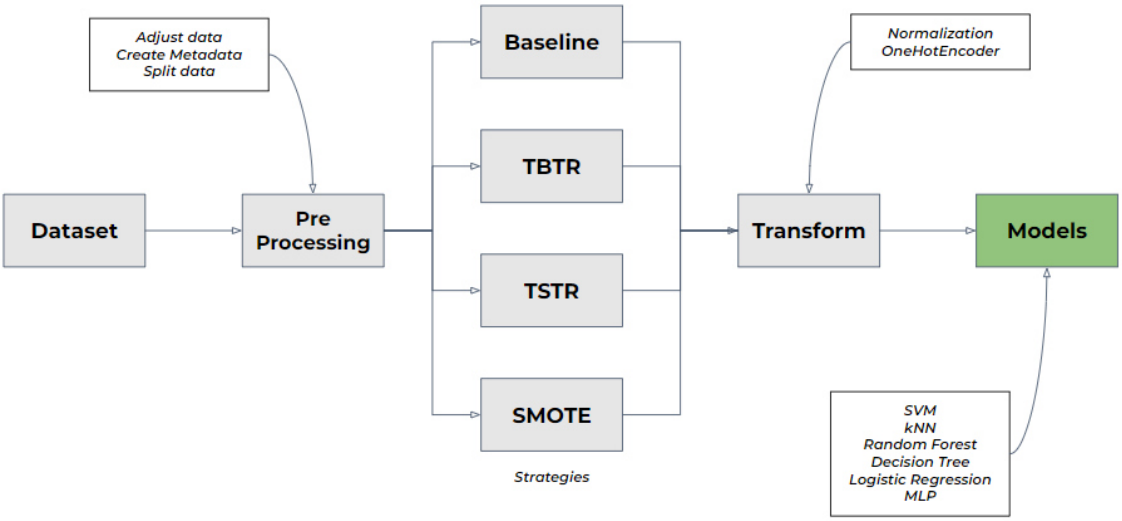
1. **Baseline:** Treinar com dados reais, testar com dados reais. Modelo base, serve como ponto inicial para comparação com as estratégias restantes.
2. **SMOTE:** Treinar com os dados balanceados, adicionando dados sintéticos no treino utilizando a técnica SMOTE e testar com os dados reais.

Utilizando as métricas como *accuracy*, *f1score*, *recall*, *precision*, *roc-auc* como validadores, citado na seção 2.3.3.3, é possível entender e analisar se os dados criados se ajustam aos modelos.

3.4.1 Execução das Estratégias pelo uso de Pipeline

Sobre a temática que envolve a geração de dados sintéticos e sua utilização em modelos de classificação, é de extrema importância que o treinamento e teste desses dados seja feita de forma padronizada e que não exista o vazamento de dados de treino para teste, para que os dados persistam da forma correta. Além disso, pela utilização de dados sintéticos, também é importante ressaltar que sua utilização se deve apenas no conjunto de dados de treino, tendo em vista o escopo do projeto em entender se a geração de dados é coerente e se desempenham papel importante em ambientes de modelos de classificação.

Figura 9 – Diagrama do Processo de Classificação pelo uso de Estratégias utilizando Pipeline. O uso do Pipeline persiste os dados da forma correta entre as etapas do processo para que o treino, teste e análise de métricas sejam coerentes e funcionais.



Fonte: Elaborado pelo próprio autor.

Pela Figura 9, percebe-se o processo geral de funcionamento da pesquisa. Primeiramente inicia-se pela utilização de um conjunto de dado (dataset). Nele, existe estudo para entender seus padrões, melhorias dos dados como preprocessamento para

limpar dados, retirar linhas duplicadas, substituir valores nulos, engenharia de características e após isso a divisão do dataset em X, y que é como os modelos de classificação recebem os dados.

Em seguida, após preparação dos dados por meio do preprocessamento e com o Metadata do SDV criado, a partir da Fase 2 da seção 1.2.2, o método está pronto para ser consumido pela pipeline. A pipeline é uma forma de automatizar o fluxo de dados e processos dentro do contexto de classificação. Nesse âmbito, a pipeline utilizada é por meio da biblioteca do ImbLearn Lemaître *et al.* (2017), que é aproveitada por conta de ser ter um método de balanceamento de dados, que ocorre no SMOTE oriundo do ImbLearn e adaptado sua classe e método de oversampling para o SDV com as estratégias TSTR e TBTR.

Caso as estratégias sejam TBTR ou TSTR, existe uma classe adaptada do SMOTE, chamado de wrapper, que serve como "fit_resample". Para simplificar a compreensão da pipeline nessas duas estratégias, a enumeração das etapas abaixo apresenta o processo de maneira mais clara:

1. Recebe os dados e o metadata, realiza preprocessamento se necessário e divide os dados entre X, y em features e target, respectivamente. Esses dados serão divididos em cinco folds (pastas), que serão utilizados como treino e teste no processo de classificação com modelos de ML.
2. Estratégia escolhida: TBTR e TSTR. Após a escolha, os dados de entrada junto ao metadata fazem a criação do modelo sintetizador de dados sintéticos.
3. A partir do sintetizador criado, realiza o balanceamento das classes no conjunto de treino de acordo com a estratégia usada.
4. Com o conjunto de dados de treino balanceado, realiza a etapa de transformação dos dados. Normalização com StandardScaler ou MinMaxScaler para atributos numéricos e OneHotEncoder para atributos categóricos.

5. Dados padronizados, balanceados e efetivos, inicia-se o processo de classificação para os modelos citados na Figura 9 em models. Nessa etapa, utiliza-se GridSearchCV para ajustes dos melhores hiperparâmetros e cinco pastas de divisão dos dados.
6. Por fim, coleta as métricas de desempenho, como accuracy, precision, recall, f1-score e ROC/AUC para avaliar e retorna em uma tabela com as médias e desvios padrões ao longo das cinco pastas.

Ademais, caso a estratégia seja Baseline ou SMOTE, os passos para a construção desses modelos são parecidos aos da estratégias acima. Para o Baseline, como é apenas treinar com dados reais e testar com dados reais, as etapas que consistem no uso dos sintetizadores e balanceamento das classes não são realizadas. Para o SMOTE, o processo que envolve a utilização de sintetizadores e seu balanceamento para classe minoritária é substituída pelo próprio método de SMOTE.

4 RESULTADOS

Este capítulo refere-se a apresentação dos resultados do projeto e consequentemente dos objetivos. Os resultados são divididos entre as abordagens, que foram referenciadas no roadmap em Figura 1, como também nos objetivos 1.2.1: resultados vigentes sobre as métricas para a geração dos dados sintéticos e resultados sobre a classificação de tarefas de ML, utilizando estratégias diferentes.

Pela metodologia descrita na abordagem 3.1, são utilizados cinco conjunto de dados (datasets) diferentes, em formato Comma Separated Values (CSV), os quais são descritos na próxima seção 4.1, com intuito de avaliar e validar a geração dos dados sintéticos, assim como a melhoria de desempenho por meio da análise de métricas sobre essas atividades de ML.

4.1 Conjunto de Dados

Essa seção refere-se à caracterização dos conjuntos de dados utilizados.

1. **Conjunto de dados Heart Cleveland (1):** O conjunto de dados (dataset) 1 é sobre dados clinícos para medições de doenças cardíacas, onde o objetivo é classificar o atributo *condition* para 0 ou 1 a fim de identificar se os pacientes tem doenças ou não. O conjunto de dados (dataset) é proveniente do UCI Machine Learning, um repositório de dados renomados Repository (2023), porém foi adaptado por conta de algumas manipulações erradas, sendo assim utilizado da referência (CHERNGS, 2023).
2. **Conjunto de dados Credit Fraud (2):** Esse dataset R (2023) é utilizado para detecção de fraudes em cartões de créditos. Seus atributos são relacionados a informações sobre a compra pelo cartão. Possui 1,000,000 de linhas com 8 colunas, sendo a coluna alvo 'fraud' um dado binário

para identificar se é um caso fraudulento ou não.

3. **Conjunto de dados Airline (3):** O conjunto de dados 3 Klein (2023) é utilizado para prever o grau de satisfação do cliente em um voo. Possui 129,880 linhas com 25 colunas, sendo a coluna alvo "satisfaction"um dado binário para identificar o nível de satisfação do cliente.
4. **Conjunto de dados Employee (4):** O conjunto de dados 4 Elmetwally (2023) contém informações sobre os funcionários de uma empresa, incluindo formação educacional, histórico profissional, dados demográficos e fatores relacionados ao emprego. Esse dataset possui 4,653 linhas com 9 colunas, sendo a coluna alvo "LeaveOrNot"representando se o funcionário continua na empresa ou não, para analisar a retenção dos colaboradores.
5. **Conjunto de dados Hotel Reservations (5):** Esse conjunto de dados Raza (2023) possui originalmente 36275 linhas para 13 colunas. A tarefa de classificação consiste em um alvo também binário "booking_status"para identificar correlações e causas para o cancelamento ou não de reservas de hotéis.
6. **Conjunto de dados Wine (6):** Esse conjunto de dados Cortez Paulo e Reis (2009) é bem conhecido no contexto de ML e foi escolhido por razões de ter somente dados numéricos contínuos, uma característica diferente do que existem nos outros 5 conjunto de dados testados. Nele, houve uma adaptação para manter a mesma estrutura sobre a coluna alvo, para ser binário. Assim, adaptou-se a coluna "quality"para 1, se a qualidade for acima de 6, e 0 caso a qualidade seja abaixo disso. Ademais, conjunto de dados é composto por 1599 linhas para 12 colunas.
7. **Conjunto de dados Rice (7):** Novamente um conjunto de dados famoso Cinar e Koklu (2019), o intuito de utilizar esse dataset vem para comple-

mentar a ideia do conjunto de dados 6, wine, pois neste em sua grande maioria é composta por dados numéricos contínuos e o restante inteiros, auxiliando as futuras hipóteses sobre a geração dos dados sintéticos e modelos de classificação. Por fim, o conjunto de dados possui 3810 linhas para 8 colunas e sua coluna alvo é "Class".

4.2 Resultados para Geração dos Dados Sintéticos

Essa seção refere-se a avaliação da geração dos dados sintéticos pela criação de sintetizadores, mostrada na seção 2.3.2. Para essa análise, é utilizada as métricas comentadas na seção 2.3.3.1 e sua visualização da geração dos dados perante aos dados reais pelo t-SNE, comentado na seção 2.3.3.2.

4.2.1 Dataset Heart Cleveland

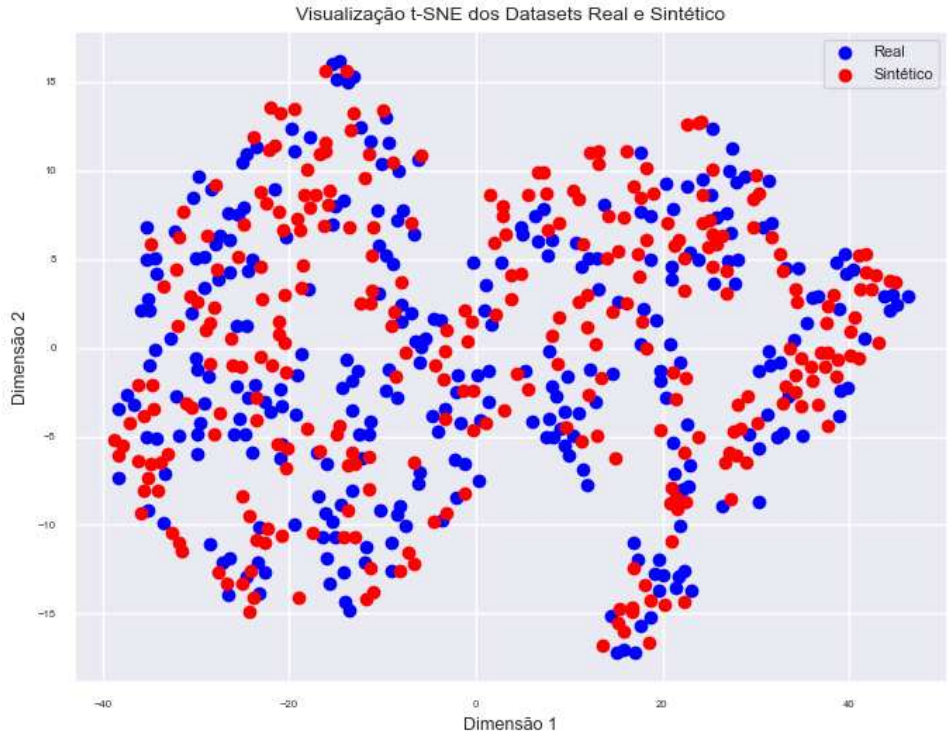
Tabela 3 – Resultados das Avaliações na Geração de Dados Sintéticos através da Utilização de Diferentes Modelos Sintetizadores para o Dataset Heart Cleveland. As métricas utilizadas consistem em técnicas específicas a partir da análise dos atributos existentes no conjunto de dados.

Synthesizer	**Metric**	**Quality Score**
CopulaGAN	KSComplement	0.7597
	CorrelationSimilarity	0.9106
CTGAN	KSComplement	0.8042
	CorrelationSimilarity	0.9064
GaussianCopula*	KSComplement	0.9126
	CorrelationSimilarity	0.9760
GaussianCopula	KSComplement	0.8460
	CorrelationSimilarity	0.9639
TVAE	KSComplement	0.8102
	CorrelationSimilarity	0.9456

Fonte: Elaborado pelo autor.

Pela Tabela 10, percebe-se que o GaussianCopula* teve melhores métricas em ambas utilizadas, em que são baseadas pelos atributos que contém o dataset. Assim, para esse conjunto de dados, o GaussianCopula* será utilizado como sintetizador.

Figura 10 – Visualização da Distribuição entre os Dados Sintéticos e Dados Reais utilizando o Método t-SNE para o Dataset Heart Cleveland. Esse método consiste na redução da dimensionalidade para facilitar a compreensão visual sobre a distribuição dos dados reais e sintéticos.



A Figura 10 demonstra pela utilização do t-SNE a distribuição dos dados entre reais e sintéticos. Percebe-se a similaridade e coerência na construção desses dados.

4.2.2 Dataset Credit Fraud

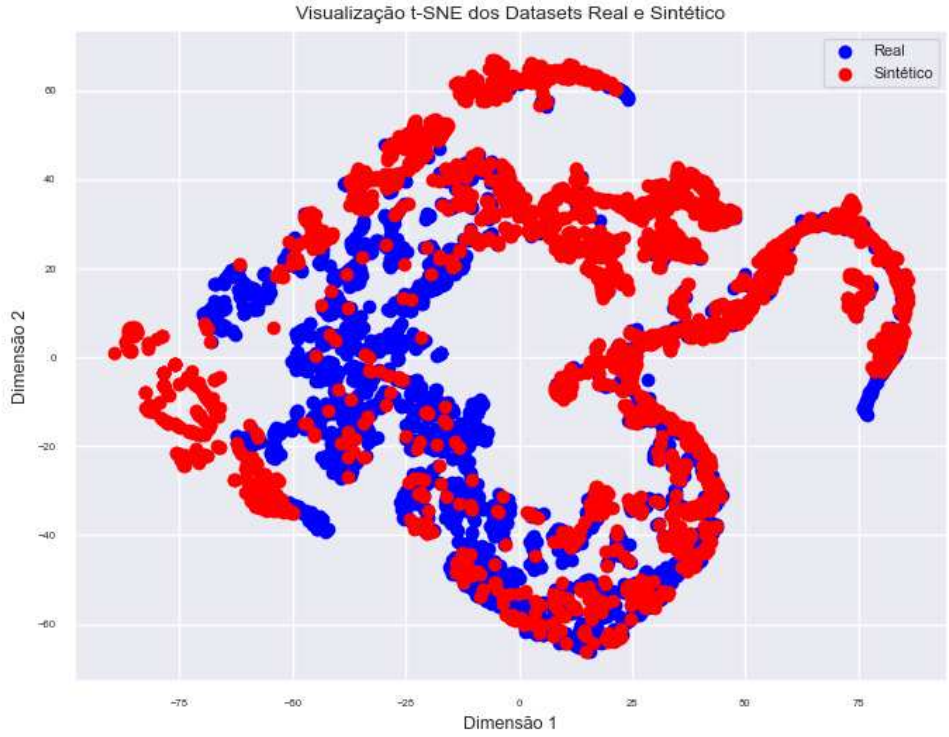
Tabela 4 – Resultados das Avaliações na Geração de Dados Sintéticos através da Utilização de Diferentes Modelos Sintetizadores para o Dataset Credit Fraud. As métricas utilizadas consistem em técnicas específicas a partir da análise dos atributos existentes no conjunto de dados.

Synthesizer	**Metric**	**Quality Score**
CopulaGAN	KSComplement	0.8917
	CorrelationSimilarity	0.9679
CTGAN	KSComplement	0.8997
	CorrelationSimilarity	0.9724
GaussianCopula*	KSComplement	0.8604
	CorrelationSimilarity	0.9821
GaussianCopula	KSComplement	0.6401
	CorrelationSimilarity	0.9750
TVAE	KSComplement	0.8093
	CorrelationSimilarity	0.9628

Fonte: Elaborado pelo autor.

Nesta Tabela 4, o sintetizador CTGAN sobressaiu entre os demais. Porém, realizando testes com a utilização tanto com CTGAN, CopulaGAN e GaussianCopula*, o GaussianCopula* obteve melhores resultados. Aqui, já percebe-se uma tendência em melhores desempenhos a partir da análise da métrica *CorrelationSimilarity*. Outro ponto relevante de enfatizar é que o GaussianCopula* possui um baixo tempo de geração dos dados, o que é bastante importante para o projeto, pois incessantemente o código é treinado e testado em busca de melhores parâmetros e valores.

Figura 11 – Visualização da Distribuição entre os Dados Sintéticos e Dados Reais utilizando o Método t-SNE para o Dataset Credit Fraud. Esse método consiste na redução da dimensionalidade para facilitar a compreensão visual sobre a disposição dos dados reais e sintéticos.



Apesar de bons resultados nas métricas, a Figura 11 não garante uma boa distribuição em seu contexto geral e isso realmente vai impactar nos valores das métricas com o uso das estratégias para os modelos de classificação.

4.2.3 Dataset Airline

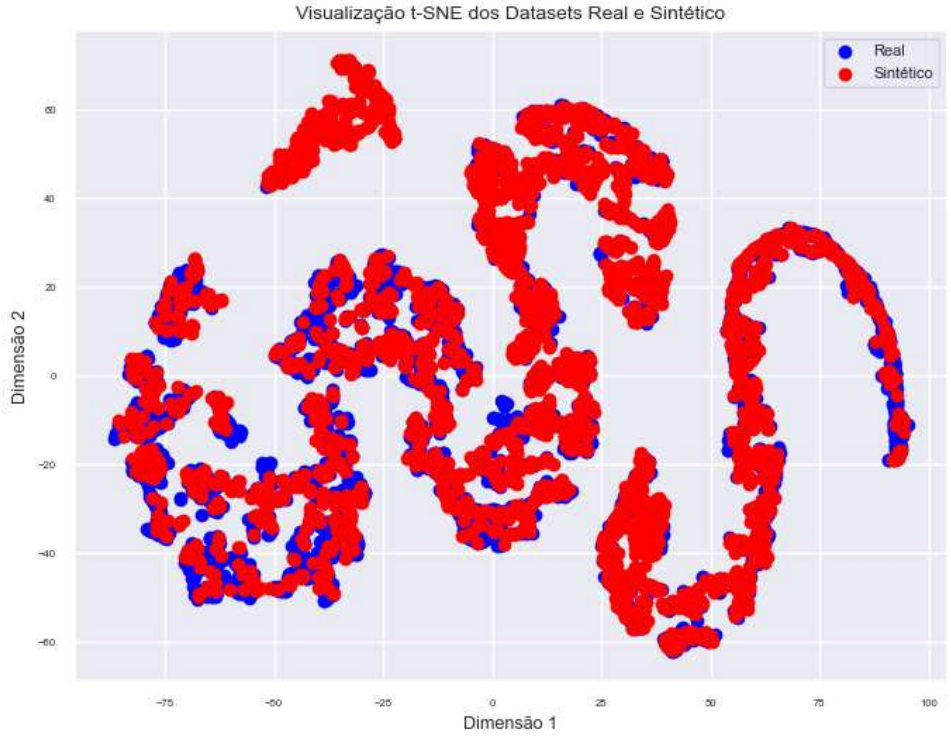
Tabela 5 – Resultados das Avaliações na Geração de Dados Sintéticos através da Utilização de Diferentes Modelos Sintetizadores para o Dataset Airline. As métricas utilizadas consistem em técnicas específicas a partir da análise dos atributos existentes no conjunto de dados.

Synthesizer	**Metric**	**Quality Score**
CopulaGAN	KSComplement	0.8824
	CorrelationSimilarity	0.9201
CTGAN	KSComplement	0.9279
	CorrelationSimilarity	0.9009
GaussianCopula*	KSComplement	0.9143
	CorrelationSimilarity	0.9863
GaussianCopula	KSComplement	0.8199
	CorrelationSimilarity	0.9732
TVAE	KSComplement	0.8547
	CorrelationSimilarity	0.9174

Fonte: Elaborado pelo autor.

A Tabela 5 mantém a concordância sobre o modelo GaussianCopula* ser o melhor nesse caso. Será utilizado esse para a geração dos dados sintéticos para o conjunto de dados Airline.

Figura 12 – Visualização da Distribuição entre os Dados Sintéticos e Dados Reais utilizando o Método t-SNE para o Dataset Airline. Esse método consiste na redução da dimensionalidade para facilitar a compreensão visual sobre a disposição dos dados reais e sintéticos.



Em diferença ao que foi analisado na Figura anterior 11, a Figura 12 está bem mais distribuída e seus dados sintéticos estão de acordo com os dados reais, o que garante em uma melhor produção desses dados e consequentemente em uma melhor análise quando for utilizar os modelos de ML para as tarefas de classificação.

4.2.4 Dataset Employee

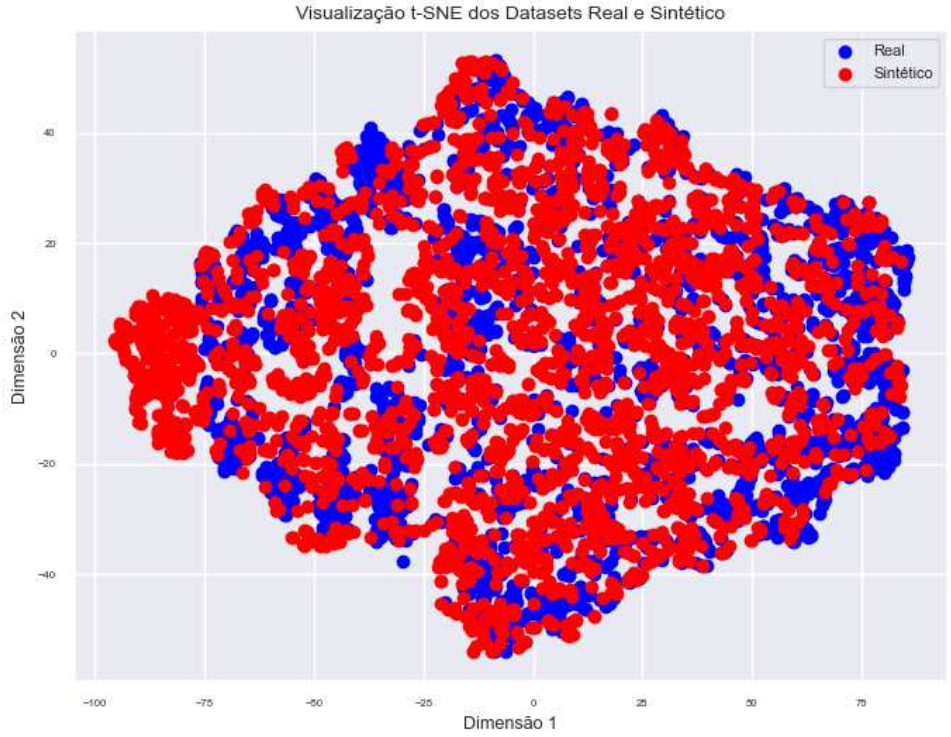
Tabela 6 – Resultados das Avaliações na Geração de Dados Sintéticos através da Utilização de Diferentes Modelos Sintetizadores para o Dataset Airline. As métricas utilizadas consistem em técnicas específicas a partir da análise dos atributos existentes no conjunto de dados.

Synthesizer	**Metric**	**Quality Score**
CopulaGAN	KSComplement	0.8907
	TVComplement	0.9217
	ContingencySimilarity	0.8432
	CorrelationSimilarity	0.9786
CTGAN	KSComplement	0.9273
	TVComplement	0.9671
	ContingencySimilarity	0.8794
	CorrelationSimilarity	0.9750
GaussianCopula*	KSComplement	0.9048
	TVComplement	0.9266
	ContingencySimilarity	0.8356
	CorrelationSimilarity	0.9891
GaussianCopula	KSComplement	0.7593
	TVComplement	0.9471
	ContingencySimilarity	0.7680
	CorrelationSimilarity	0.9725
TVAE	KSComplement	0.8606
	TVComplement	0.9022
	ContingencySimilarity	0.7632
	CorrelationSimilarity	0.9261

Fonte: Elaborado pelo autor.

A Tabela 6 para o dataset Employee possui mais métricas pelo fato de conter atributos numéricos e categóricos. Nele, o CTGAN sobressaiu diante os outros sintetizadores em um score relativamente alto, porém, assim como foi mencionado na seção 4.2.2, o GaussianCopula* possui um menor tempo de processamento e melhores métricas em atividades de classificação.

Figura 13 – Visualização da Distribuição entre os Dados Sintéticos e Dados Reais utilizando o Método t-SNE para o Dataset Employee. Esse método consiste na redução da dimensionalidade para facilitar a compreensão visual sobre a disposição dos dados reais e sintéticos.



Novamente, a distribuição dos dados via t-SNE é satisfatória, apenas na porção à esquerda que os dados distoam um pouco do conjunto de dados reais, o que pode ser analisado por conta do score médio menor que 0.9 que o GaussianCopula* possui nesse caso.

4.2.5 Dataset Hotel Reservations

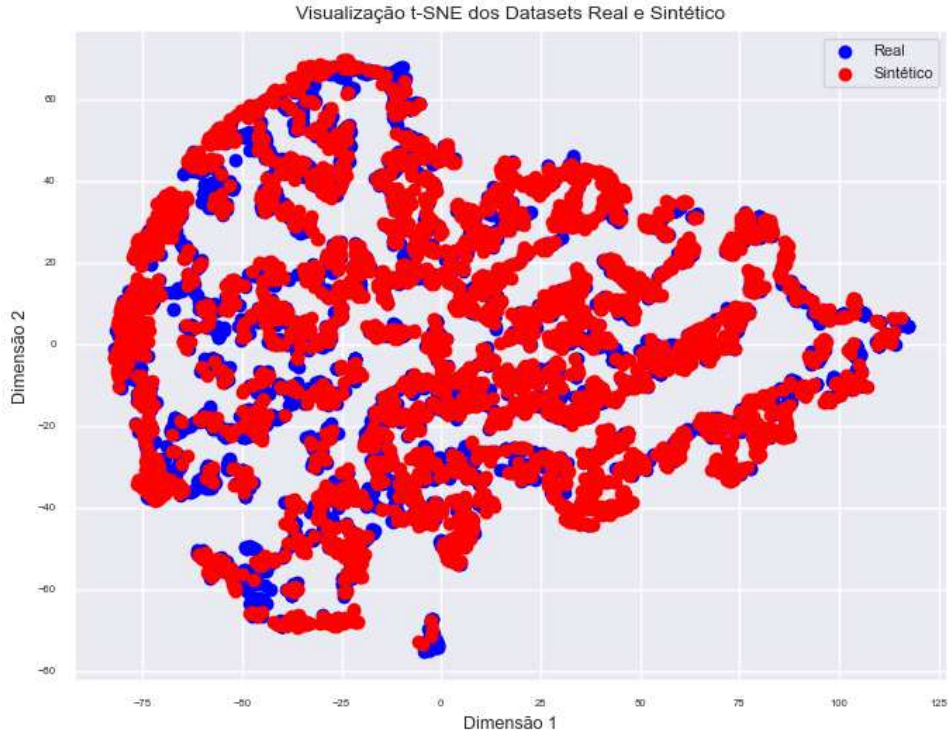
Tabela 7 – Resultados das Avaliações na Geração de Dados Sintéticos através da Utilização de Diferentes Modelos Sintetizadores para o Dataset Hotel Reservations. As métricas utilizadas consistem em técnicas específicas a partir da análise dos atributos existentes no conjunto de dados.

Synthesizer	**Metric**	**Quality Score**
CopulaGAN	KSComplement	0.8750
	CorrelationSimilarity	0.9710
CTGAN	KSComplement	0.9633
	CorrelationSimilarity	0.9768
GaussianCopula*	KSComplement	0.9329
	CorrelationSimilarity	0.9825
GaussianCopula	KSComplement	0.6829
	CorrelationSimilarity	0.9676
TVAE	KSComplement	0.9435
	CorrelationSimilarity	0.9652

Fonte: Elaborado pelo autor.

Nesta Tabela 7, GaussianCopula* possui um melhor desempenho e novamente pode ser analisado que o *CorrelationSimilarity* é maior nesse caso, o que tendencia em uma melhor classificação.

Figura 14 – Visualização da Distribuição entre os Dados Sintéticos e Dados Reais utilizando o Método t-SNE para o Dataset Hotel Reservations. Esse método consiste na redução da dimensionalidade para facilitar a compreensão visual sobre a disposição dos dados reais e sintéticos.



Na Figura 14, como analisado pelos resultados das métricas, a distribuição dos dados sintéticos em comparação com os dados reais conduzem em um formato semelhante devido ao alto valor visto nas métricas da Tabela 7.

4.2.6 Dataset Wine

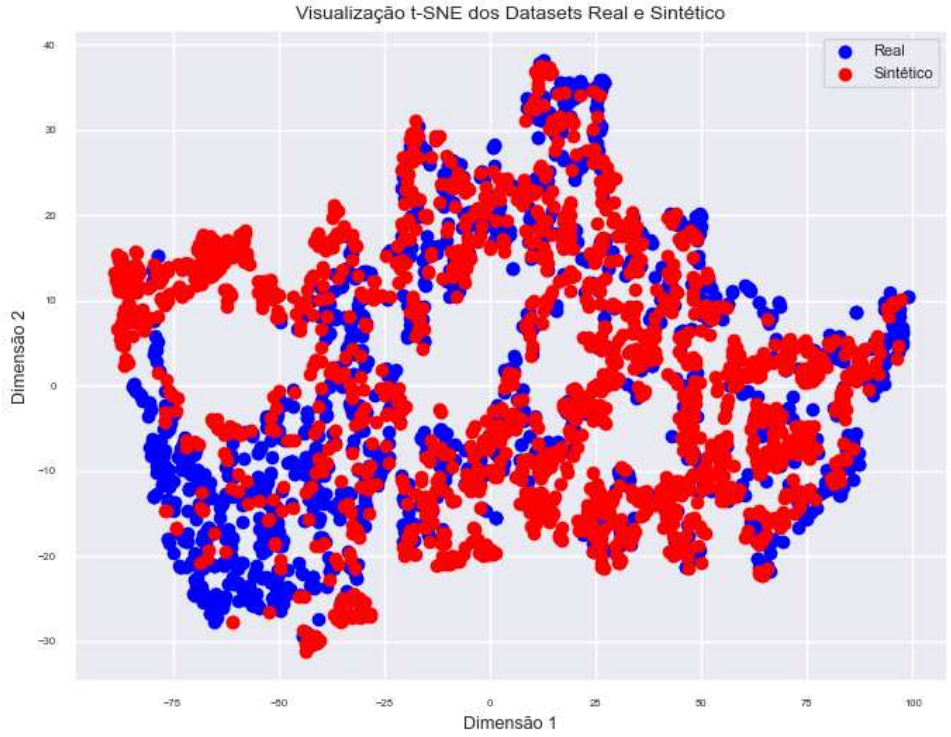
Tabela 8 – Resultados das Avaliações na Geração de Dados Sintéticos através da Utilização de Diferentes Modelos Sintetizadores para o Dataset Wine. As métricas utilizadas consistem em técnicas específicas a partir da análise dos atributos existentes no conjunto de dados.

Synthesizer	**Metric**	**Quality Score**
CopulaGAN	KSComplement	0.7588
	CorrelationSimilarity	0.9026
CTGAN	KSComplement	0.7971
	CorrelationSimilarity	0.9008
GaussianCopula*	KSComplement	0.8828
	CorrelationSimilarity	0.9860
GaussianCopula	KSComplement	0.8241
	CorrelationSimilarity	0.9656
TVAE	KSComplement	0.8990
	CorrelationSimilarity	0.9667

Fonte: Elaborado pelo autor.

Nesta Tabela 8, percebe-se o desempenho de um modelo ainda não enfatizado nessa seção de resultados, que é o TVAE, em que sugere-se que em dados numéricos em predominância, seja um ótimo sintetizador. Porém, com uma diferença mínima, o GaussianCopula* é superior, com média de score de 0.934.

Figura 15 – Visualização da Distribuição entre os Dados Sintéticos e Dados Reais utilizando o Método t-SNE para o Dataset Wine. Esse método consiste na redução da dimensionalidade para facilitar a compreensão visual sobre a disposição dos dados reais e sintéticos.



Pela percepção de boas métricas, com score de 0.934 pelo GaussianCopula*, percebe-se a distribuição efetiva dos dados sintéticos frente a comparação com dados reais na Figura 15, em que, em sua maioria, está bem distribuído ao longo dos dados reais.

4.2.7 Dataset Rice

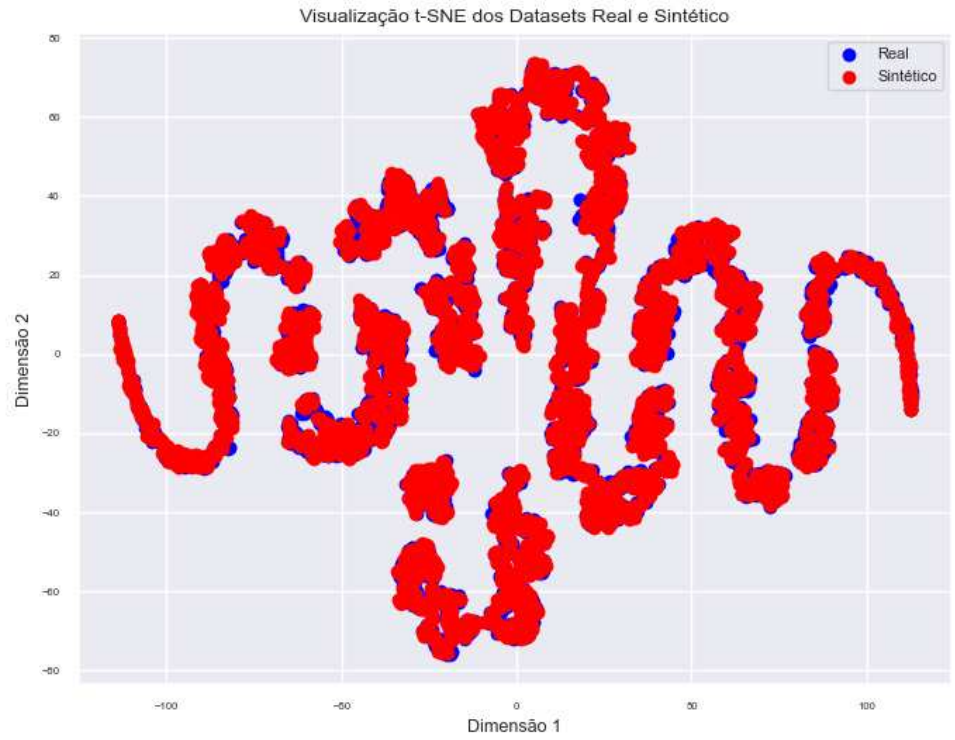
Tabela 9 – Resultados das Avaliações na Geração de Dados Sintéticos através da Utilização de Diferentes Modelos Sintetizadores para o Dataset Rice. As métricas utilizadas consistem em técnicas específicas a partir da análise dos atributos existentes no conjunto de dados.

Synthesizer	**Metric**	**Quality Score**
CopulaGAN	KSComplement	0.7473
	CorrelationSimilarity	0.8867
CTGAN	KSComplement	0.9201
	CorrelationSimilarity	0.9215
GaussianCopula*	KSComplement	0.9428
	CorrelationSimilarity	0.9845
GaussianCopula	KSComplement	0.8381
	CorrelationSimilarity	0.9797
TVAE	KSComplement	0.9474
	CorrelationSimilarity	0.9672

Fonte: Elaborado pelo autor.

Nesta Tabela 9, assim como mencionado na seção anterior 4.2.6, TVAE e GaussianCopula* foram os que obteram melhores métricas para o conjunto de dados, reiterando, assim, a hegemonia do GaussianCopula* para a pesquisa, em que demonstrou efetividade na geração dos dados sintéticos.

Figura 16 – Visualização da Distribuição entre os Dados Sintéticos e Dados Reais utilizando o Método t-SNE para o Dataset Rice. Esse método consiste na redução da dimensionalidade para facilitar a compreensão visual sobre a disposição dos dados reais e sintéticos.



Essa Figura 16 serve como exemplar para o projeto, pois os dados sintéticos estão perfeitamente distribuídos ao longo da distribuição dos dados reais, o que faz dele um destaque.

4.3 Resultados para a Classificação com Dados Sintéticos

Esta seção se refere aos resultados dos modelos de classificação binária utilizando as 4 estratégias abordadas: Baseline, TBTR, TSTR e SMOTE. Para TBTR e TSTR, serão utilizados os melhores sintetizadores da seção 4.2 a fim de equilibrar as

classes.

4.3.1 Dataset Heart Cleveland

Tabela 10 – Desempenho dos Classificadores por Estratégia no Conjunto de Dados Heart Cleveland. Aplicação de métricas para avaliar o desempenho de cada modelo de classificação, empregando estratégias variadas.

Estratégia	**Classificadores**	**Métricas**				
		Accuracy	Precision	Recall	F1 Score	ROC/AUC
Baseline	Decision Tree	0.6872 ± 0.07	0.6565 ± 0.10	0.6745 ± 0.15	0.6589 ± 0.10	0.6853 ± 0.07
	KNN	0.7778 ± 0.05	0.7687 ± 0.09	0.7429 ± 0.07	0.7529 ± 0.07	0.7759 ± 0.05
	Logistic Regression	0.8415 ± 0.06	0.8406 ± 0.11	0.8131 ± 0.06	0.8236 ± 0.07	0.8427 ± 0.06
	MLP	0.8148 ± 0.01	0.8334 ± 0.09	0.7629 ± 0.10	0.7885 ± 0.03	0.8143 ± 0.01
	Random Forest	0.8013 ± 0.03	0.8050 ± 0.07	0.7587 ± 0.06	0.7779 ± 0.02	0.7964 ± 0.03
	SVM	0.8283 ± 0.04	0.8588 ± 0.14	0.7819 ± 0.09	0.8074 ± 0.04	0.8321 ± 0.04
TSTR	Decision Tree	0.7643 ± 0.02	0.7472 ± 0.07	0.7657 ± 0.13	0.7467 ± 0.03	0.7629 ± 0.02
	KNN	0.7642 ± 0.05	0.7724 ± 0.05	0.6922 ± 0.07	0.7295 ± 0.06	0.7579 ± 0.05
	Logistic Regression	0.8415 ± 0.07	0.8587 ± 0.10	0.7687 ± 0.11	0.8099 ± 0.10	0.8359 ± 0.08
	MLP	0.8183 ± 0.02	0.8307 ± 0.06	0.7643 ± 0.08	0.7926 ± 0.03	0.8126 ± 0.02
	Random Forest	0.8283 ± 0.05	0.8491 ± 0.05	0.7675 ± 0.03	0.8060 ± 0.04	0.8225 ± 0.04
	SVM	0.8250 ± 0.04	0.8476 ± 0.10	0.7731 ± 0.08	0.8028 ± 0.04	0.8257 ± 0.04
TBTR	Decision Tree	0.7039 ± 0.05	0.6836 ± 0.09	0.6783 ± 0.14	0.6741 ± 0.08	0.7021 ± 0.06
	KNN	0.7778 ± 0.05	0.7641 ± 0.09	0.7500 ± 0.06	0.7549 ± 0.06	0.7761 ± 0.05
	Logistic Regression	0.8381 ± 0.07	0.8248 ± 0.11	0.8256 ± 0.07	0.8222 ± 0.08	0.8388 ± 0.07
	MLP	0.8114 ± 0.01	0.8114 ± 0.10	0.7839 ± 0.09	0.7898 ± 0.03	0.8132 ± 0.01
	Random Forest	0.8182 ± 0.05	0.8139 ± 0.07	0.7901 ± 0.06	0.7994 ± 0.04	0.8105 ± 0.04
	SVM	0.8384 ± 0.05	0.8576 ± 0.14	0.8105 ± 0.09	0.8221 ± 0.05	0.8433 ± 0.05
SMOTE	Decision Tree	0.6769 ± 0.09	0.6614 ± 0.13	0.6493 ± 0.11	0.6490 ± 0.10	0.6762 ± 0.09
	KNN	0.7744 ± 0.05	0.7590 ± 0.09	0.7500 ± 0.06	0.7523 ± 0.06	0.7729 ± 0.05
	Logistic Regression	0.8481 ± 0.08	0.8254 ± 0.12	0.8566 ± 0.09	0.8367 ± 0.09	0.8516 ± 0.07
	MLP	0.8115 ± 0.03	0.8090 ± 0.08	0.7932 ± 0.10	0.7940 ± 0.02	0.8126 ± 0.03
	Random Forest	0.8147 ± 0.04	0.7982 ± 0.05	0.8020 ± 0.07	0.7983 ± 0.04	0.8114 ± 0.04
	SVM	0.8350 ± 0.06	0.8351 ± 0.12	0.8246 ± 0.10	0.8215 ± 0.06	0.8400 ± 0.06

Fonte: Elaborado pelo autor.

A Tabela 10 mostra os resultados sobre as estratégias. O conjunto de dados utilizado possuem apenas 297 linhas e obteve um resultado melhores de métricas para o SMOTE, para o modelo de classificação Logistic Regression. É importante ressaltar também os resultados do TSTR, que foram bastantes satisfatórios se for comparados às outras estratégias como também o baseline, em que em casos como Random Forest e Decision Tree o TSTR se sobressaiu, o que pode ser um bom resultado para a criação de

dados sintéticos em datasets menores e que devem ser privatizados.

4.3.2 Dataset Credit Fraud

Tabela 11 – Desempenho dos Classificadores por Estratégia no Conjunto de Dados Credit Fraud. Aplicação de métricas para avaliar o desempenho de cada modelo de classificação, empregando estratégias variadas.

Estratégia	**Classificadores**	**Métricas**				
		Accuracy	Precision	Recall	F1 Score	ROC/AUC
Baseline	Decision Tree	0.9986 ± 0.00	0.9922 ± 0.00	0.9920 ± 0.00	0.9921 ± 0.00	0.9957 ± 0.00
	KNN	0.9950 ± 0.00	0.9718 ± 0.01	0.9708 ± 0.01	0.9713 ± 0.01	0.9840 ± 0.01
	Logistic Regression	0.9598 ± 0.00	0.8408 ± 0.01	0.6664 ± 0.02	0.7434 ± 0.02	0.8272 ± 0.01
	MLP	0.9969 ± 0.00	0.9804 ± 0.01	0.9839 ± 0.01	0.9821 ± 0.01	0.9910 ± 0.00
	Random Forest	0.9992 ± 0.00	0.9928 ± 0.00	0.9981 ± 0.00	0.9954 ± 0.00	0.9987 ± 0.00
	SVM	0.9966 ± 0.00	0.9810 ± 0.01	0.9797 ± 0.01	0.9803 ± 0.00	0.9889 ± 0.00
TBTR	Decision Tree	0.9603 ± 0.00	0.6924 ± 0.01	0.9826 ± 0.01	0.8123 ± 0.01	0.9704 ± 0.00
	KNN	0.9498 ± 0.00	0.6390 ± 0.02	0.9801 ± 0.01	0.7735 ± 0.01	0.9635 ± 0.01
	Logistic Regression	0.9313 ± 0.00	0.5646 ± 0.01	0.9345 ± 0.00	0.7038 ± 0.01	0.9327 ± 0.00
	MLP	0.9682 ± 0.00	0.7378 ± 0.02	0.9874 ± 0.00	0.8444 ± 0.01	0.9769 ± 0.00
	Random Forest	0.9694 ± 0.02	0.7576 ± 0.08	0.9580 ± 0.08	0.8460 ± 0.08	0.9643 ± 0.04
	SVM	0.9342 ± 0.02	0.5801 ± 0.07	0.9496 ± 0.03	0.7189 ± 0.06	0.9412 ± 0.02
TSTR	Decision Tree	0.8917 ± 0.00	0.4336 ± 0.01	0.7790 ± 0.02	0.5570 ± 0.01	0.8408 ± 0.01
	KNN	0.8976 ± 0.00	0.4509 ± 0.01	0.7873 ± 0.01	0.5734 ± 0.01	0.8477 ± 0.01
	Logistic Regression	0.9358 ± 0.00	0.5928 ± 0.01	0.8469 ± 0.01	0.6973 ± 0.01	0.8956 ± 0.00
	MLP	0.9353 ± 0.00	0.5949 ± 0.02	0.8180 ± 0.02	0.6885 ± 0.01	0.8823 ± 0.01
	Random Forest	0.9350 ± 0.00	0.5934 ± 0.02	0.8111 ± 0.02	0.6851 ± 0.02	0.8790 ± 0.01
	SVM	0.9356 ± 0.00	0.5917 ± 0.01	0.8479 ± 0.02	0.6969 ± 0.01	0.8960 ± 0.01
SMOTE	Decision Tree	0.9980 ± 0.00	0.9794 ± 0.01	0.9977 ± 0.00	0.9885 ± 0.00	0.9978 ± 0.00
	KNN	0.9941 ± 0.00	0.9524 ± 0.01	0.9813 ± 0.01	0.9666 ± 0.01	0.9883 ± 0.00
	Logistic Regression	0.9338 ± 0.00	0.5734 ± 0.01	0.9503 ± 0.00	0.7152 ± 0.01	0.9413 ± 0.00
	MLP	0.9952 ± 0.00	0.9505 ± 0.01	0.9978 ± 0.00	0.9735 ± 0.01	0.9964 ± 0.00
	Random Forest	0.9984 ± 0.00	0.9839 ± 0.01	0.9976 ± 0.00	0.9907 ± 0.00	0.9980 ± 0.00
	SVM	0.9924 ± 0.00	0.9236 ± 0.01	0.9952 ± 0.00	0.9580 ± 0.01	0.9937 ± 0.00

Fonte: Elaborado pelo autor.

A Tabela para o Credit Fraud 11 não foi muito agradável para as estratégias do TBTR e TSTR, muito por conta dos dados serem muito desequilibrados e, por conta disso, o sintetizador não entender muito bem os padrões para uma classe, pois se foi passado relativamente poucos dados de treino para esse caso. Para enfatizar, existem 45630 para a classe 0 e apenas 4370 para a classe 1, então existe um viés claro para que o modelo não consiga abstrair muito sobre isso.

Um ponto importante de ressaltar é que, se fosse utilizado todos os dados disponíveis, cerca de 1.000.000, o sintetizador resultou em ótimos resultados para as métricas, porém fica inviável de treinar e testar com essa quantidade de dados para a tarefa de classificação, onde existe vários modelos e hiperparâmetros. Nesse sentido, foram-se utilizados 50.000 dados.

Em conclusão, o Baseline com Random Forest obteve melhores métricas no contexto geral e em segundo lugar, bem próximo, esteve também o Random Forest com SMOTE.

4.3.3 Dataset Airline

Tabela 12 – Desempenho dos Classificadores por Estratégia no Conjunto de Dados Airline. Aplicação de métricas para avaliar o desempenho de cada modelo de classificação, empregando estratégias variadas.

Estratégia	**Classificadores**	**Métricas**				
		Accuracy	Precision	Recall	F1 Score	ROC/AUC
Baseline	Decision Tree	0.9398 ± 0.00	0.9401 ± 0.01	0.9200 ± 0.01	0.9299 ± 0.00	0.9375 ± 0.00
	KNN	0.9180 ± 0.00	0.9357 ± 0.00	0.8711 ± 0.01	0.9023 ± 0.00	0.9126 ± 0.00
	Logistic Regression	0.8724 ± 0.01	0.8674 ± 0.01	0.8340 ± 0.01	0.8503 ± 0.01	0.8679 ± 0.01
	MLP	0.9474 ± 0.01	0.9469 ± 0.01	0.9315 ± 0.01	0.9391 ± 0.01	0.9456 ± 0.00
	Random Forest	0.9559 ± 0.00	0.9623 ± 0.00	0.9351 ± 0.01	0.9485 ± 0.00	0.9535 ± 0.00
	SVM	0.9446 ± 0.00	0.9465 ± 0.00	0.9249 ± 0.00	0.9356 ± 0.00	0.9423 ± 0.00
TBTR	Decision Tree	0.9148 ± 0.02	0.8840 ± 0.04	0.9270 ± 0.01	0.9046 ± 0.02	0.9162 ± 0.01
	KNN	0.9024 ± 0.00	0.8939 ± 0.01	0.8797 ± 0.00	0.8867 ± 0.00	0.8998 ± 0.00
	Logistic Regression	0.8613 ± 0.00	0.8277 ± 0.00	0.8597 ± 0.01	0.8434 ± 0.00	0.8611 ± 0.00
	MLP	0.9367 ± 0.01	0.9196 ± 0.02	0.9368 ± 0.01	0.9279 ± 0.01	0.9367 ± 0.01
	Random Forest	0.9476 ± 0.01	0.9375 ± 0.01	0.9423 ± 0.01	0.9399 ± 0.01	0.9470 ± 0.01
	SVM	0.8955 ± 0.04	0.8741 ± 0.04	0.8869 ± 0.04	0.8804 ± 0.04	0.8945 ± 0.04
TSTR	Decision Tree	0.7617 ± 0.00	0.6968 ± 0.00	0.7994 ± 0.01	0.7446 ± 0.00	0.7661 ± 0.01
	KNN	0.7596 ± 0.00	0.6948 ± 0.01	0.7966 ± 0.01	0.7421 ± 0.01	0.7639 ± 0.00
	Logistic Regression	0.8454 ± 0.01	0.8033 ± 0.01	0.8531 ± 0.01	0.8275 ± 0.01	0.8463 ± 0.01
	MLP	0.8445 ± 0.01	0.8137 ± 0.02	0.8337 ± 0.01	0.8233 ± 0.01	0.8434 ± 0.01
	Random Forest	0.8396 ± 0.01	0.7914 ± 0.01	0.8567 ± 0.01	0.8227 ± 0.01	0.8416 ± 0.00
	SVM	0.8480 ± 0.01	0.8077 ± 0.01	0.8531 ± 0.01	0.8298 ± 0.01	0.8485 ± 0.01
SMOTE	Decision Tree	0.9340 ± 0.00	0.9185 ± 0.01	0.9309 ± 0.01	0.9246 ± 0.00	0.9336 ± 0.00
	KNN	0.9161 ± 0.00	0.9067 ± 0.01	0.8992 ± 0.01	0.9029 ± 0.01	0.9141 ± 0.00
	Logistic Regression	0.8626 ± 0.00	0.8275 ± 0.00	0.8638 ± 0.01	0.8453 ± 0.00	0.8628 ± 0.00
	MLP	0.9416 ± 0.01	0.9350 ± 0.02	0.9311 ± 0.02	0.9327 ± 0.01	0.9402 ± 0.00
	Random Forest	0.9493 ± 0.00	0.9384 ± 0.00	0.9454 ± 0.01	0.9419 ± 0.00	0.9489 ± 0.00
	SVM	0.9404 ± 0.00	0.9261 ± 0.01	0.9377 ± 0.01	0.9318 ± 0.00	0.9401 ± 0.00

Fonte: Elaborado pelo autor.

No dataset Airline pela Tabela 12, o algoritmo Random Forest obteve a melhor métrica, com 0.9623 de precision e 0.9535 de ROC/AUC. Destaque também para o TSTR que se tratando somente de dados sintéticos conseguiu obter valores bem satisfatórios.

Nesse contexto de análise, foram-se utilizadas 25976 linhas de dados e não há um desequilíbrio muito relevante nesse dataset, em que existem 14690 para a classe 0 e 11286 para a classe 1.

4.3.4 Dataset Employee

Tabela 13 – Desempenho dos Classificadores por Estratégia no Conjunto de Dados Employee. Aplicação de métricas para avaliar o desempenho de cada modelo de classificação, empregando estratégias variadas.

Estratégia	**Classificadores**	**Métricas**				
		Accuracy	Precision	Recall	F1 Score	ROC/AUC
Baseline	Decision Tree	0.8461 ± 0.01	0.8640 ± 0.04	0.6580 ± 0.02	0.7462 ± 0.01	0.8016 ± 0.01
	KNN	0.8051 ± 0.01	0.7912 ± 0.02	0.5903 ± 0.04	0.6751 ± 0.02	0.7543 ± 0.01
	Logistic Regression	0.7363 ± 0.01	0.6907 ± 0.04	0.4240 ± 0.02	0.5250 ± 0.02	0.6621 ± 0.01
	MLP	0.8478 ± 0.01	0.8659 ± 0.03	0.6607 ± 0.02	0.7492 ± 0.01	0.8031 ± 0.01
	Random Forest	0.8549 ± 0.02	0.8864 ± 0.01	0.6637 ± 0.03	0.7589 ± 0.02	0.8095 ± 0.02
	SVM	0.8448 ± 0.01	0.8760 ± 0.04	0.6393 ± 0.02	0.7390 ± 0.02	0.7959 ± 0.01
TBTR	Decision Tree	0.7771 ± 0.02	0.6804 ± 0.03	0.6647 ± 0.03	0.6721 ± 0.02	0.7505 ± 0.02
	KNN	0.7369 ± 0.01	0.6029 ± 0.02	0.6903 ± 0.03	0.6432 ± 0.01	0.7261 ± 0.01
	Logistic Regression	0.6843 ± 0.02	0.5336 ± 0.03	0.6568 ± 0.02	0.5885 ± 0.02	0.6778 ± 0.02
	MLP	0.8180 ± 0.01	0.7396 ± 0.02	0.7241 ± 0.03	0.7317 ± 0.02	0.7954 ± 0.01
	Random Forest	0.8063 ± 0.01	0.7107 ± 0.04	0.7369 ± 0.03	0.7231 ± 0.02	0.7900 ± 0.02
	SVM	0.6886 ± 0.02	0.5399 ± 0.03	0.6450 ± 0.03	0.5875 ± 0.03	0.6783 ± 0.02
TSTR	Decision Tree	0.5790 ± 0.03	0.4107 ± 0.03	0.5120 ± 0.04	0.4550 ± 0.03	0.5633 ± 0.03
	KNN	0.6026 ± 0.02	0.4373 ± 0.03	0.5391 ± 0.02	0.4825 ± 0.02	0.5875 ± 0.01
	Logistic Regression	0.6624 ± 0.02	0.5067 ± 0.02	0.6249 ± 0.05	0.5594 ± 0.03	0.6534 ± 0.02
	MLP	0.6437 ± 0.03	0.4872 ± 0.05	0.6186 ± 0.04	0.5439 ± 0.04	0.6378 ± 0.03
	Random Forest	0.5947 ± 0.02	0.4346 ± 0.03	0.5811 ± 0.02	0.4962 ± 0.02	0.5918 ± 0.01
	SVM	0.6649 ± 0.01	0.5105 ± 0.01	0.6258 ± 0.02	0.5622 ± 0.01	0.6557 ± 0.01
SMOTE	Decision Tree	0.8261 ± 0.01	0.7723 ± 0.04	0.7045 ± 0.02	0.7359 ± 0.02	0.7973 ± 0.01
	KNN	0.7915 ± 0.01	0.7232 ± 0.03	0.6383 ± 0.03	0.6775 ± 0.02	0.7551 ± 0.01
	Logistic Regression	0.6901 ± 0.01	0.5419 ± 0.02	0.6354 ± 0.04	0.5846 ± 0.02	0.6770 ± 0.01
	MLP	0.8259 ± 0.01	0.7564 ± 0.04	0.7301 ± 0.04	0.7419 ± 0.02	0.8030 ± 0.02
	Random Forest	0.8442 ± 0.01	0.8128 ± 0.01	0.7104 ± 0.03	0.7579 ± 0.02	0.8125 ± 0.01
	SVM	0.8356 ± 0.01	0.7872 ± 0.02	0.7149 ± 0.02	0.7492 ± 0.02	0.8069 ± 0.01

Fonte: Elaborado pelo autor.

Este dataset foi o mais árduo em conseguir métricas suficientemente relevantes para a pesquisa. Tratando-se primeiramente de um dataset com 4653 dados, realizou-se diversos métodos de análise para chegar em valores razoáveis. Nesse contexto, A Tabela 13 demonstra que o SMOTE com Random Forest obteve melhores valores para ROC/AUC, mas Baseline a maior métrica com Random Forest para precision.

Uma observação importante é que o equílibrio das classes melhorou significativamente as métricas para o Logistic Regression, sendo o TBTR que mais aumentou essa métrica. Porém, no geral, TBTR e TSTR não foram boas estratégias.

4.3.5 Dataset Hotel Reservations

Tabela 14 – Desempenho dos Classificadores por Estratégia no Conjunto de Dados Hotel Reservations. Aplicação de métricas para avaliar o desempenho de cada modelo de classificação, empregando estratégias variadas.

Estratégia	**Classificadores**	**Métricas**				
		Accuracy	Precision	Recall	F1 Score	ROC/AUC
Baseline	Decision Tree	0.8689 ± 0.00	0.8179 ± 0.02	0.7726 ± 0.02	0.7942 ± 0.01	0.8442 ± 0.01
	KNN	0.8719 ± 0.00	0.8245 ± 0.01	0.7739 ± 0.01	0.7984 ± 0.00	0.8468 ± 0.00
	Logistic Regression	0.7917 ± 0.00	0.7262 ± 0.01	0.5849 ± 0.00	0.6479 ± 0.01	0.7387 ± 0.00
	MLP	0.8581 ± 0.01	0.7868 ± 0.05	0.7848 ± 0.04	0.7839 ± 0.01	0.8393 ± 0.00
	Random Forest	0.8988 ± 0.00	0.8800 ± 0.00	0.8004 ± 0.01	0.8383 ± 0.00	0.8736 ± 0.00
	SVM	0.8309 ± 0.00	0.8029 ± 0.01	0.6413 ± 0.01	0.7130 ± 0.01	0.7823 ± 0.01
TBTR	Decision Tree	0.8397 ± 0.01	0.7230 ± 0.01	0.8280 ± 0.01	0.7719 ± 0.01	0.8367 ± 0.01
	KNN	0.8331 ± 0.00	0.7220 ± 0.01	0.7980 ± 0.01	0.7581 ± 0.01	0.8241 ± 0.00
	Logistic Regression	0.7682 ± 0.00	0.6214 ± 0.01	0.7489 ± 0.01	0.6792 ± 0.01	0.7633 ± 0.01
	MLP	0.8271 ± 0.01	0.7156 ± 0.02	0.7866 ± 0.03	0.7488 ± 0.00	0.8167 ± 0.00
	Random Forest	0.8749 ± 0.01	0.7920 ± 0.01	0.8383 ± 0.01	0.8145 ± 0.01	0.8655 ± 0.01
	SVM	0.7679 ± 0.00	0.6227 ± 0.01	0.7398 ± 0.01	0.6762 ± 0.01	0.7607 ± 0.00
TSTR	Decision Tree	0.6709 ± 0.01	0.4983 ± 0.01	0.6702 ± 0.02	0.5716 ± 0.01	0.6707 ± 0.01
	KNN	0.6617 ± 0.01	0.4878 ± 0.01	0.6403 ± 0.02	0.5536 ± 0.01	0.6563 ± 0.01
	Logistic Regression	0.7723 ± 0.01	0.6378 ± 0.01	0.7060 ± 0.01	0.6701 ± 0.01	0.7553 ± 0.01
	MLP	0.7700 ± 0.00	0.6289 ± 0.01	0.7287 ± 0.02	0.6748 ± 0.01	0.7595 ± 0.01
	Random Forest	0.7602 ± 0.00	0.6162 ± 0.00	0.7113 ± 0.02	0.6603 ± 0.01	0.7477 ± 0.01
	SVM	0.7717 ± 0.00	0.6380 ± 0.01	0.7011 ± 0.00	0.6680 ± 0.00	0.7536 ± 0.00
SMOTE	Decision Tree	0.8636 ± 0.00	0.7842 ± 0.01	0.8054 ± 0.01	0.7946 ± 0.01	0.8487 ± 0.00
	KNN	0.8613 ± 0.00	0.7654 ± 0.01	0.8317 ± 0.01	0.7971 ± 0.00	0.8538 ± 0.00
	Logistic Regression	0.7721 ± 0.00	0.6291 ± 0.01	0.7415 ± 0.01	0.6807 ± 0.01	0.7642 ± 0.00
	MLP	0.8550 ± 0.01	0.7636 ± 0.03	0.8122 ± 0.04	0.7859 ± 0.01	0.8441 ± 0.01
	Random Forest	0.8947 ± 0.00	0.8416 ± 0.01	0.8364 ± 0.01	0.8389 ± 0.00	0.8798 ± 0.00
	SVM	0.8128 ± 0.00	0.6824 ± 0.01	0.8025 ± 0.01	0.7375 ± 0.01	0.8102 ± 0.00

Fonte: Elaborado pelo autor.

Nesta Tabela 14, foi-se utilizado todo o dataset: 36275 linhas para 13 colunas. Percebe-se pela tabela que o SMOTE em comparação ao Baseline demonstrou muito efetivo, melhorando as métricas para todos os modelos de classificação, com destaque para o Random Forest com 0.8798 de ROC/AUC e 0.8947 de accuracy. Em valores totais, Random Forest em Baseline obteve o maior resultado, com 0.8988 de accuracy.

4.3.6 Dataset Wine

Tabela 15 – Desempenho dos Classificadores por Estratégia no Conjunto de Dados Wine. Aplicação de métricas para avaliar o desempenho de cada modelo de classificação, empregando estratégias variadas.

Estratégia	**Classificadores**	**Métricas**				
		Accuracy	Precision	Recall	F1 Score	ROC/AUC
Baseline	Decision Tree	0.8680 ± 0.02	0.2333 ± 0.32	0.1082 ± 0.15	0.1473 ± 0.20	0.5480 ± 0.07
	KNN	0.8805 ± 0.02	0.5987 ± 0.09	0.3461 ± 0.14	0.4291 ± 0.11	0.6550 ± 0.06
	Logistic Regression	0.8780 ± 0.03	0.6103 ± 0.10	0.2952 ± 0.08	0.3953 ± 0.09	0.6327 ± 0.04
	MLP	0.8818 ± 0.02	0.6333 ± 0.11	0.3391 ± 0.04	0.4401 ± 0.05	0.6532 ± 0.02
	Random Forest	0.8856 ± 0.02	0.7398 ± 0.13	0.2568 ± 0.07	0.3752 ± 0.09	0.6208 ± 0.04
	SVM	0.8737 ± 0.01	0.5440 ± 0.32	0.1127 ± 0.16	0.1630 ± 0.20	0.5527 ± 0.08
TBTR	Decision Tree	0.8368 ± 0.02	0.4365 ± 0.02	0.7007 ± 0.01	0.5377 ± 0.02	0.7793 ± 0.01
	KNN	0.8486 ± 0.03	0.4618 ± 0.11	0.6352 ± 0.07	0.5326 ± 0.09	0.7583 ± 0.04
	Logistic Regression	0.7861 ± 0.03	0.3669 ± 0.07	0.7935 ± 0.04	0.4996 ± 0.07	0.7889 ± 0.03
	MLP	0.8105 ± 0.03	0.3940 ± 0.02	0.7534 ± 0.10	0.5163 ± 0.04	0.7859 ± 0.04
	Random Forest	0.8780 ± 0.04	0.5452 ± 0.09	0.7676 ± 0.07	0.6354 ± 0.08	0.8316 ± 0.05
	SVM	0.7748 ± 0.03	0.3582 ± 0.05	0.8144 ± 0.06	0.4961 ± 0.06	0.7916 ± 0.03
TSTR	Decision Tree	0.6910 ± 0.03	0.2681 ± 0.05	0.7338 ± 0.08	0.3911 ± 0.06	0.7089 ± 0.03
	KNN	0.7017 ± 0.02	0.2778 ± 0.03	0.7500 ± 0.02	0.4046 ± 0.03	0.7219 ± 0.00
	Logistic Regression	0.7824 ± 0.02	0.3643 ± 0.06	0.8208 ± 0.06	0.5025 ± 0.06	0.7982 ± 0.03
	MLP	0.7873 ± 0.05	0.3743 ± 0.06	0.8170 ± 0.07	0.5124 ± 0.06	0.7995 ± 0.05
	Random Forest	0.7911 ± 0.03	0.3766 ± 0.04	0.8131 ± 0.04	0.5137 ± 0.04	0.8005 ± 0.03
	SVM	0.7817 ± 0.02	0.3658 ± 0.05	0.8183 ± 0.05	0.5046 ± 0.05	0.7970 ± 0.03
SMOTE	Decision Tree	0.8349 ± 0.02	0.4271 ± 0.03	0.6326 ± 0.05	0.5094 ± 0.03	0.7496 ± 0.03
	KNN	0.8537 ± 0.02	0.4764 ± 0.09	0.6937 ± 0.07	0.5613 ± 0.08	0.7862 ± 0.04
	Logistic Regression	0.7905 ± 0.03	0.3707 ± 0.03	0.7801 ± 0.11	0.5018 ± 0.05	0.7868 ± 0.07
	MLP	0.8387 ± 0.02	0.4498 ± 0.07	0.6674 ± 0.14	0.5253 ± 0.02	0.7674 ± 0.05
	Random Forest	0.8837 ± 0.03	0.5641 ± 0.08	0.7186 ± 0.11	0.6275 ± 0.07	0.8144 ± 0.06
	SVM	0.8455 ± 0.02	0.4607 ± 0.04	0.7592 ± 0.05	0.5721 ± 0.04	0.8091 ± 0.02

Fonte: Elaborado pelo autor.

A Tabela 15 demonstra que para o conjunto de dados Wine, que apenas accuracy obteve um valor alto, os restantes não, o que pode indicar que o desbalanceamento

das classes é relevante, ainda porque existem poucos dados no geral (1599), onde desses 1382 são para uma classe e 217 para outra.

Nesse sentido, percebe-se que as estratégias TBTR e SMOTE auxiliam para o melhor desempenho dos modelos, aumentando os valores em praticamente todas as métricas analisadas, com o Random Forest para o TBTR sendo o melhor modelo no geral, olhando para a métrica ROC/AUC.

Ademais, ponto bastante positivo para TSTR, em que se demonstra ser uma estratégia com grande potencial de melhorias e que conseguiu métricas melhores do que Baseline usando apenas dados sintéticos.

4.3.7 Dataset Rice

Tabela 16 – Desempenho dos Classificadores por Estratégia no Conjunto de Dados Rice. Aplicação de métricas para avaliar o desempenho de cada modelo de classificação, empregando estratégias variadas.

Estratégia	**Classificadores**	**Métricas**				
		Accuracy	Precision	Recall	F1 Score	ROC/AUC
Baseline	Decision Tree	0.9247 ± 0.01	0.9157 ± 0.02	0.9084 ± 0.02	0.9117 ± 0.01	0.9225 ± 0.00
	KNN	0.9242 ± 0.01	0.9198 ± 0.01	0.9014 ± 0.02	0.9104 ± 0.01	0.9213 ± 0.01
	Logistic Regression	0.9270 ± 0.01	0.9178 ± 0.01	0.9117 ± 0.02	0.9145 ± 0.01	0.9254 ± 0.01
	MLP	0.9278 ± 0.01	0.9205 ± 0.02	0.9098 ± 0.02	0.9148 ± 0.01	0.9253 ± 0.01
	Random Forest	0.9286 ± 0.01	0.9214 ± 0.01	0.9109 ± 0.02	0.9161 ± 0.01	0.9262 ± 0.01
	SVM	0.9292 ± 0.01	0.9219 ± 0.02	0.9114 ± 0.02	0.9164 ± 0.01	0.9268 ± 0.01
TBTR	Decision Tree	0.9081 ± 0.02	0.8875 ± 0.03	0.8989 ± 0.03	0.8929 ± 0.02	0.9069 ± 0.02
	KNN	0.8942 ± 0.02	0.8683 ± 0.02	0.8877 ± 0.02	0.8779 ± 0.02	0.8932 ± 0.02
	Logistic Regression	0.9229 ± 0.01	0.8996 ± 0.02	0.9234 ± 0.01	0.9112 ± 0.01	0.9229 ± 0.01
	MLP	0.9257 ± 0.01	0.9030 ± 0.03	0.9265 ± 0.01	0.9143 ± 0.02	0.9260 ± 0.01
	Random Forest	0.9228 ± 0.01	0.9056 ± 0.01	0.9151 ± 0.02	0.9103 ± 0.01	0.9217 ± 0.01
	SVM	0.9249 ± 0.01	0.9027 ± 0.02	0.9241 ± 0.02	0.9130 ± 0.01	0.9246 ± 0.01
TSTR	Decision Tree	0.9055 ± 0.01	0.8850 ± 0.02	0.8964 ± 0.01	0.8906 ± 0.01	0.9043 ± 0.01
	KNN	0.9005 ± 0.01	0.8791 ± 0.01	0.8896 ± 0.01	0.8843 ± 0.01	0.8992 ± 0.01
	Logistic Regression	0.9247 ± 0.01	0.8985 ± 0.02	0.9294 ± 0.02	0.9135 ± 0.01	0.9251 ± 0.01
	MLP	0.9268 ± 0.01	0.9058 ± 0.02	0.9249 ± 0.01	0.9151 ± 0.01	0.9265 ± 0.01
	Random Forest	0.9244 ± 0.01	0.9067 ± 0.02	0.9181 ± 0.01	0.9123 ± 0.02	0.9234 ± 0.01
	SVM	0.9273 ± 0.01	0.9096 ± 0.02	0.9217 ± 0.02	0.9154 ± 0.01	0.9264 ± 0.01
SMOTE	Decision Tree	0.9262 ± 0.01	0.9071 ± 0.02	0.9227 ± 0.01	0.9146 ± 0.01	0.9258 ± 0.01
	KNN	0.9236 ± 0.01	0.9080 ± 0.01	0.9143 ± 0.02	0.9111 ± 0.01	0.9225 ± 0.01
	Logistic Regression	0.9263 ± 0.01	0.9016 ± 0.01	0.9295 ± 0.01	0.9151 ± 0.01	0.9266 ± 0.01
	MLP	0.9255 ± 0.01	0.9012 ± 0.03	0.9275 ± 0.01	0.9139 ± 0.01	0.9257 ± 0.01
	Random Forest	0.9260 ± 0.01	0.9069 ± 0.01	0.9220 ± 0.01	0.9143 ± 0.01	0.9253 ± 0.01
	SVM	0.9252 ± 0.01	0.8991 ± 0.02	0.9298 ± 0.02	0.9139 ± 0.01	0.9257 ± 0.01

Fonte: Elaborado pelo autor.

A Tabela 16 retrata que deste o Baseline a performance para classificação é altamente positivo, obtendo valores acima de 0.9 em todas as métricas. Dessa forma, não é possível retirar informações relevantes para as estratégias de balanceamento utilizando TBTR e SMOTE, pois ficaram na margem de Baseline. Porém, a estratégia TSTR, utilizando somente dados sintéticos de treino e balanceados, obteve métricas muito boas, ressaltando um ponto importante sobre a questão de privacidade, em que é possível utilizar dados sintéticos nesse caso.

5 CONCLUSÕES E TRABALHOS FUTUROS

Em suma, para esse tipo de classificação binária utilizando SDV, acreditava-se que a sua utilização (TBTR) deveria melhorar as métricas dos modelos de classificação em sua grande maioria, porém foi uma surpresa isso não ter ocorrido, onde se esperava que os métodos envolvendo ML tivesse um melhor desempenho sobre as outras estratégias. Assim, geralmente os melhores modelos foram aqueles utilizando o próprio Baseline: treinar com dados reais, testar com dados reais; ou SMOTE: treinar com dados balanceados utilizando SMOTE, testar com dados reais.

Em suma, inicialmente esperava-se que a utilização da técnica TBTR aprimorasse as métricas dos modelos de classificação na maioria dos datasets utilizados, mas foi surpreendente não observar essa melhoria. Ao contrário das expectativas, os métodos envolvendo ML não demonstraram consistentemente um desempenho superior em comparação com outras estratégias. No geral, os modelos mais eficazes foram aqueles que utilizaram o Baseline (treinamento e teste com dados reais) ou o SMOTE (treinamento com dados balanceados usando SMOTE e teste com dados reais) e para alguns casos o TSTR foi importante ressaltar.

Destaque, nesse âmbito, para o Random Forest, mostrou-se efetivo para todos os datasets e consequentemente assume-se que é um ótimo algoritmo para tratamento de classes desbalanceadas, além de ser um algoritmo relativamente simples e que manteve boas métricas com estratégias distintas, mesmo após balanceamento dos dados.

Outro ponto positivo foi a questão do TSTR, que demonstrou a efetividade em alguns casos para a criação de novos ambientes que a privacidade dos dados são importantes, pois utilizando somente dados sintéticos, conseguiu em alguns casos sobressair sobre as métricas das outras estratégias. Nesse tópico, mostrou-se efetivo nos conjunto de dados Cortez Paulo e Reis (2009), Cinar e Koklu (2019), em que o sintetizador com atributos de conjunto de dados predominantemente contínuos foram satisfatórios no

contexto das métricas, ressaltando a possibilidade do uso de dados sintéticos sob a ótica de privacidade dos dados.

Além disso, TSTR utiliza dados sintéticos em todo seu dados de treino, já o SMOTE, por exemplo, não faz. Necessita-se de ter dados desbalanceados e o SMOTE atua no reequilíbrio delas. Assim, no cenário de privacidade dos dados, o TSTR indica ser relevante.

Dessa forma, existem alguns pontos de melhorias e trabalhos futuros que podem fazer com que a estratégia TBTR e TSTR obtenham ainda mais desempenho e notabilize sobre Baseline e SMOTE para esses casos abordados:

- **Potencial de melhorias pelo ajuste dos hiperparametros:** Assim como o GaussianCopula* foi o sintetizador utilizado pela pesquisa e ele é uma adaptação dos parâmetros do GaussianCopula, existe a possibilidade de existirem melhores sintetizadores com a mudança desses hiperparâmetros.
- **Utilizar outros geradores de dados sintéticos:** No projeto utiliza GaussianCopula, GaussianCopula*, CopulaGAN, CTGAN, TVAE, porém existem diversos outros que podem ser relevantes para esse estudo.
- **Utilizar outros modelos de classificação:** O projeto utiliza 6 algoritmos de classificação distintos, é interessante testar outros mais simples e complexos para entender a tendência dessas métricas com novos algoritmos testados.
- **Abordagem mais minuciosa sobre as saídas dos dados sintéticos:** Entender se cada dado sintético em cada coluna está condizente com o que é esperado ou se existe uma forma de melhorar. Assim como uma abordagem mais minuciosa sobre as métricas de avaliação em relação a utilização da matriz de confusão a fim de entender quais são os casos que o modelo acerta e erra.

- **Maiores conhecimentos sobre os datasets:** Apesar da utilização excessiva dos datasets, existe a possibilidade de incrementação de novas ideias e insights que possam melhorar o desempenho a partir de uma exploração mais detalhada. Outro ponto relevante sobre isso é a questão das Constraints, explicada na seção 3.3.2, em que elas podem ser utilizadas para criação de regras determinísticas em que existe grande possibilidades de melhorar as métricas por conta desse fator.
- **Utilização de MLFlow:** MLFlow é uma tecnologia muito interessante para o propósito do projeto, em que existe essa recorrência em retestar modelos. Essa plataforma é importante nesse sentido, em que guarda informações sobre métricas e hiperparâmetros. Dessa forma, pode-se testar novos casos sem perder os antigos.

REFERÊNCIAS

BADU-MARFO, G.; FAROOQ, B.; PATTERSON, Z. Composite travel generative adversarial networks for tabular and sequential population synthesis. **IEEE Transactions on Intelligent Transportation Systems**, v. 23, p. 17976–17985, 2020. Disponível em: <https://api.semanticscholar.org/CorpusID:215768816>.

BARROS, C. **Repositório deste TCC**. [S.l.]: GitHub, 2023. <https://github.com/caiocmb7/projetos/tree/main/ML%26AI/TCC>.

BEAULIEU-JONES, B. K.; WU, Z. S.; WILLIAMS, C.; LEE, R.; BHAVNANI, S. P.; BYRD, J. B.; GREENE, C. S. Privacy-preserving generative deep neural networks support clinical data sharing. **Circulation: Cardiovascular Quality and Outcomes**, Am Heart Assoc, v. 12, n. 7, p. e005122, 2019.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: synthetic minority over-sampling technique. **Journal of artificial intelligence research**, v. 16, p. 321–357, 2002.

CHERNGS. **Heart Disease Cleveland UCI**. 2023. urlhttps://www.kaggle.com/datasets/cherngs/heart-disease-cleveland-uci.

CINAR, I.; KOKLU, M. **Rice (Cammeo and Osmancik)**. 2019. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5MW4Z.

CLARKE, L. A. A system to generate test data and symbolically execute programs. **IEEE Transactions on Software Engineering**, SE-2, p. 215–222, 1976. Disponível em: <https://api.semanticscholar.org/CorpusID:2940045>.

CORTEZ PAULO, C. A. A. F. M. T.; REIS, J. **Wine Quality**. 2009. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T.

DATACEBO, INC. **Synthetic Data Metrics**. [S.l.], 2023. Version 0.9.3. Disponível em: <https://docs.sdv.dev/sdmetrics/>.

DOCS, S. D. V. **Single Table Metadata JSON**. 2023. Disponível em: <https://docs.sdv.dev/sdv/reference/metadata-spec/single-table-metadata-json>.

DSFSI. **textaugment: Text Augmentation Library**. [S.l.]: GitHub, 2019. <https://github.com/dsfsi/textaugment>.

ELIDAN, G. Copulas in machine learning. In: JAWORSKI, P.; DURANTE, F.; HÄRDLE, W. K. (Ed.). **Copulas in Mathematical and Quantitative Finance**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 39–60. ISBN 978-3-642-35407-6.

ELMETWALLY, T. **Employee Dataset**. [S.l.]: Kaggle, 2023. <https://www.kaggle.com/datasets/tawfikelmetwally/employee-dataset/data>.

ESTEBAN, C.; HYLAND, S. L.; RäTSCH, G. Real-valued (medical) time series generation with recurrent conditional gans. In: **International Conference on Learning Representations**. [s.n.], 2018. Disponível em: <https://paperswithcode.com/paper/real-valued-medical-time-series-generation>.

FONSECA, J.; BACAO, F. Tabular and latent space synthetic data generation: a literature review. **Journal of Big Data**, SpringerOpen, v. 10, p. 115, 2023.

GEM-BENCHMARK. **NL-Augmenter: A Collaborative Repository of Natural Language Transformations**. [S.l.]: GitHub, 2021. <https://github.com/GEM-benchmark/NL-Augmenter>.

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial networks. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 63, n. 11, p. 139–144, oct 2020. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/3422622>.

GUILLAUDEUX, M.; ROUSSEAU, O.; PETOT, J. *et al.* Patient-centric synthetic data generation, no reason to risk re-identification in biomedical data analysis. **npj Digit. Med.**, v. 6, p. 37, 2023.

HOUSSOU, R.; AUGUSTIN, M.-C.; RAPPOS, E.; BONVIN, V.; ROBERT-NICOUD, S. Generation and simulation of synthetic datasets with copulas. **ArXiv**, abs/2203.17250, 2022. Disponível em: <https://api.semanticscholar.org/CorpusID:247839726>.

HUI, J. **GAN — What's Generative Adversarial Networks and its application?** 2018. Acessado em: 13 de Setembro, 2023. Disponível em: <https://jonathan-hui.medium.com/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09>.

KEYRUS. **Desvendando os Dados Sintéticos: o que são e como tirar proveito deles?** 2023. Acessado em: 10 de Setembro de 2023. Disponível em: <https://keyrus.com/br/pt/insights/desvendando-os-dados-sinteticos-o-que-sao-e-como-tirar-proveito-deles>.

KLEIN, T. **Airline Passenger Satisfaction**. [S.l.]: Kaggle, 2023. <https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction>.

KüHNEL, L.; SCHNEIDER, J.; PERRAR, I.; ADAMS, T.; PRASSER, F.; NöTHLINGS, U.; FRöHLICH, H.; FLUCK, J. **Synthetic data generation for a longitudinal cohort study - Evaluation, method extension and reproduction of published data analysis results**. 2023.

LEE, J.; HYEONG, J.; JEON, J.; PARK, N.; CHO, J. Invertible tabular gans: Killing two birds with one stone for tabular data synthesis. In: **Proceedings of the 35th Conference on Neural Information Processing Systems**. [s.n.], 2021. Disponível em: <https://proceedings.neurips.cc/paper/2021/file/22456f4b545572855c766df5eefc9832-Paper.pdf>.

LEE, J.; LEE, C.; PARK, N. Stasy: Score-based tabular data synthesis. 2022. Disponível em: <https://doi.org/10.48550/arXiv.2210.04018>.

LEMAîTRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. **Journal of Machine Learning Research**, v. 18, n. 17, p. 1–5, 2017. Disponível em: <http://jmlr.org/papers/v18/16-365.html>.

LI, J.; CAIRNS, B.; LI, J. *et al.* Generating synthetic mixed-type longitudinal electronic health records for artificial intelligent applications. **npj Digit. Med.**, v. 6, p. 98, 2023.

LUDEMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. **SciELO Brasil**, 2023. Disponível em: <https://doi.org/10.1590/s0103-4014.2021.35101.007>.

MA, E. **nlpaug: Data Augmentation for NLP**. [S.l.]: GitHub, 2019. <https://github.com/makcedward/nlpaug>.

MAATEN, L. v. d.; HINTON, G. Visualizing data using t-sne. **Journal of machine learning research**, v. 9, n. Nov, p. 2579–2605, 2008.

PATKI, N.; WEDGE, R.; VEERAMACHANENI, K. The synthetic data vault. In: **IEEE International Conference on Data Science and Advanced Analytics (DSAA)**. [S.l.: s.n.], 2016. p. 399–410.

QDATA. **TextAttack: A Python framework for adversarial attacks, data augmentation, and model training in NLP**. [S.l.]: GitHub, 2020. <https://github.com/QData/TextAttack>.

R, D. N. **Credit Card Fraud**. [S.l.]: Kaggle, 2023. <https://www.kaggle.com/datasets/dhanushnarayananr/credit-card-fraud/data>.

RAZA, A. **Hotel Reservations Classification Dataset**. [S.l.]: Kaggle, 2023. <https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset/data>.

REPOSITORY, U. M. L. **Heart Disease Data Set**. 2023. urlhttps://archive.ics.uci.edu/dataset/45/heart+disease.

RIEBESELL, J. **TikZ**. 2023. Acessado em: 13 de Setembro, 2023. Disponível em: <https://tikz.net/vae/>.

RODRIGUES, M. F. Geração de dados sintéticos utilizando redes neurais artificiais. In: **Trabalho de Conclusão de Curso (Graduação em Engenharia de Software) - Universidade Federal do Ceará**. [s.n.], 2021. Disponível em: <http://repositorio.ufc.br/handle/riufc/60961>.

RöGLIN, J.; ZIEGELER, K.; KUBE, J.; KöNIG, F.; HERMANN, K.-G.; ORTMANN, S. Improving classification results on a small medical dataset using a gan; an outlook for dealing with rare disease datasets. **Frontier Computer Science** — Computer Vision, v. 4, 2022. Disponível em: <https://doi.org/10.3389/fcomp.2022.858874>.

SDV. **Predefined Constraint Classes**. 2023. urlhttps://docs.sdv.dev/sdv/reference/constraint-logic/predefined-constraint-classes.

SHAFKAT, I. **Intuitively Understanding Variational Autoencoders**. 2018. Disponível em: <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf>.

VERO, M.; BALUNOVIć, M.; VECHEV, M. **Programmable Synthetic Tabular Data Generation**. 2023.

XU, L.; SKOULARIDOU, M.; CUESTA-INFANTE, A.; VEERAMACHANENI, K. Modeling tabular data using conditional gan. In: **Proceedings of the 33rd Conference on Neural Information Processing Systems**. [s.n.], 2019. Disponível em: <https://proceedings.neurips.cc/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf>.

YOON, J.; JORDON, J.; SCHAAR, M. van der. PATE-GAN: Generating synthetic data with differential privacy guarantees. In: **International Conference on Learning Representations**. [s.n.], 2019. Disponível em: <https://openreview.net/forum?id=S1zk9iRqF7>.

APÊNDICE A – CÓDIGOS EM PYTHON IMPORTANTES PARA A PESQUISA

Código-fonte 1 – Criação do Ambiente Virtual

```
# Criar o ambiente virtual conda e ativar-lo.
conda create --name sdv-project python=3.10.9
conda activate sdv-project

# Instalar pacotes
pip install -r requirements.txt
```

Código-fonte 2 – Criação e Adição de Constraints

```
# Cria a estrutura da forma dicionario (JSON) para a Constraint
age_constraint = {
    "constraint_class": "ScalarRange",
    "constraint_parameters": {
        "column_name": "age",
        "low_value": 29.0,
        "high_value": 77.0,
        "strict_boundaries": False,
    },
}

# Instancia um sintetizador e adiciona a constraint
synthesizer = CTGANSynthesizer(metadata)
synthesizer.add_constraints(constraints=[age_constraint])
```

Código-fonte 3 – Evaluate Models para Classificação

```
def evaluation_pipeline(
    X,
    y,
    num_features: list,
    cat_features: list,
    sampling_strategy: str,
    float_to_int_features=None,
    metadata=None,
    target=None,
):
    print(f"Sampling Strategy: {sampling_strategy}")
    np.random.seed(42)

    num_outer_loop_folds = 5
    num_inner_loop_folds = 5
    results = []

    models = [
        (
            "SVM",
            SVC(),
            {"model__C": [0.1, 0.5, 1, 5, 10], "model__kernel":
                ["linear", "rbf"]},
        ),
        (
            "Decision Tree",
            DecisionTreeClassifier(),
            {
                "model__max_depth": [None, 1, 2, 5, 10],
                "model__min_samples_split": [2, 5, 10],
                "model__min_samples_leaf": [1, 5],
```

```
            },
        ),
        (
            "KNN",
            KNeighborsClassifier(),
            {
                "model__n_neighbors": [1, 3, 5, 7, 10],
                "model__weights": ["uniform", "distance"],
            },
        ),
        (
            "Random Forest",
            RandomForestClassifier(),
            {
                "model__n_estimators": [100, 200, 300],
                "model__max_depth": [None, 5, 10],
                "model__min_samples_split": [2, 5],
                "model__min_samples_leaf": [1, 5],
            },
        ),
        (
            "Logistic Regression",
            LogisticRegression(max_iter=1000),
            {
                "model__C": [0.1, 0.5, 1, 5, 10],
                "model__solver": ["liblinear", "sag", "saga"],
            },
        ),
        (
            "MLP",
            MLPClassifier(max_iter=1000),
            {
                "model__hidden_layer_sizes": [(100,), (100, 50)
```

```
                    ],
64                  "model__alpha": [0.0001, 0.001, 0.01],
65              },
66          ),
67      ]
68
69      columns_order = X.columns.tolist()
70
71      num_transformer = ("numerical_standard_scaler",
            StandardScaler(), num_features)
72      transformers = [num_transformer]
73
74      # if necessary
75
76      # cat_transformer = ("categorical_onehot", OneHotEncoder(
            handle_unknown='ignore'), cat_features)
77      # transformers = [cat_transformer, num_transformer]
78
79      for name, model, param_grid in models:
80          print(f"\nTraining Model: {model}")
81          folds = KFold(n_splits=num_outer_loop_folds, shuffle=
                True).split(X, y)
82          for i, (train_index, test_index) in enumerate(folds):
83              print(f"Training fold {i + 1}...")
84
85              X_train, y_train = X.iloc[train_index, :], y.iloc[
                    train_index]
86              X_test, y_test = X.iloc[test_index, :], y.iloc[
                    test_index]
87
88              normalization_step = (
89                  "normalization",
90                  ColumnTransformer(transformers=transformers,
```

```
                remainder="passthrough"),
91          )
92          model_step = ("model", model)
93
94          if sampling_strategy == "TBTR":
95              balance_classes_step = (
96                  "resampling_tbtr",
97                  SDVPipelineTBTR(
98                      metadata=metadata,
99                      target=target,
100                     num_features=num_features,
101                     cat_features=cat_features,
102                     columns_order=columns_order,
103                     float_to_int_features=
                            float_to_int_features,
104                 ),
105             )
106             steps = [balance_classes_step,
                    normalization_step, model_step]
107         elif sampling_strategy == "TSTR":
108             balance_classes_step = (
109                 "resampling_tstr",
110                 SDVPipelineTSTR(
111                     metadata=metadata,
112                     target=target,
113                     num_features=num_features,
114                     cat_features=cat_features,
115                     columns_order=columns_order,
116                     float_to_int_features=
                            float_to_int_features,
117                 ),
118             )
119             steps = [balance_classes_step,
```

```
                normalization_step, model_step]
        elif sampling_strategy == "SMOTE":
            balance_classes_step = ("resampling_smote",
                SMOTE(random_state=42))
            steps = [balance_classes_step,
                normalization_step, model_step]
        elif sampling_strategy == "BASELINE":
            steps = [normalization_step, model_step]

        pipeline = Pipeline(steps=steps)

        clf = GridSearchCV(
            pipeline,
            param_grid,
            cv=num_inner_loop_folds,
            n_jobs=-1,
        )
        clf.fit(X_train, y_train)

        y_pred = clf.predict(X_test)
        accuracy_test = accuracy_score(y_test, y_pred)
        f1_test = f1_score(y_test, y_pred)
        precision_test = precision_score(y_test, y_pred)
        recall_test = recall_score(y_test, y_pred)
        roc_auc_test = roc_auc_score(y_test, y_pred)

        results.append(
            {
                "Classification": name,
                "Accuracy": round(accuracy_test, 4),
                "Precision": round(precision_test, 4),
                "Recall": round(recall_test, 4),
                "F1 Score": round(f1_test, 4),
```

```
150                     "ROC/AUC": round(roc_auc_test, 4),
151                 }
152             )
153
154     mean_df = pd.DataFrame(results).groupby(["Classification"],
            as_index=False).mean()
155     std_df = pd.DataFrame(results).groupby(["Classification"],
            as_index=False).std()
156     results_df = mean_std_results(mean_df, std_df)
157     return results_df
```

Código-fonte 4 – Wrapper para a Estratégia TBTR usando Pipeline

```
# TBTR (Treinar com Dados Balanceados [SDV + Dados Reais] , Testar com Dados Reais)
class SDVPipelineTBTR(BaseSampler):
    _sampling_type = "over-sampling"

    _parameter_constraints = {
        "X": [pd.DataFrame],
        "y": [pd.DataFrame, pd.Series],
        "metadata": [object],
        "target": [str],
        "num_features": [list],
        "cat_features": [list],
    }

    def __init__(
        self,
        metadata,
        target,
        num_features,
        cat_features,
        columns_order,
        float_to_int_features,
    ):
        self.metadata = metadata
        self.target = target
        self.num_features = num_features
        self.cat_features = cat_features
        self.columns_order = columns_order
        self.float_to_int_features = float_to_int_features
        self.synthesizer = SingleTablePreset(self.metadata, name="FAST_ML")
```

```
30          super().__init__()
31
32      def _fit_resample(self, X, y):
33          X = pd.DataFrame(X, columns=self.columns_order).
                reset_index(drop=True)
34          y = pd.DataFrame(y, columns=[self.target]).reset_index(
                drop=True)
35          df_train = pd.merge(X, y, left_index=True, right_index=
                True)
36
37          for col in self.float_to_int_features:
38              X[col] = X[col].astype(int)
39              df_train[col] = df_train[col].astype(int)
40
41          self.synthesizer.fit(df_train)
42
43          class_counts = y[self.target].value_counts()
44          minority_class = class_counts.idxmin()
45          synthetic_samples_needed = class_counts.max() -
                class_counts.min()
46
47          if minority_class == 0:
48              balanced_conditions_0 = Condition(
49                  num_rows=synthetic_samples_needed,
50                  column_values={self.target: 0},
51              )
52              df_synth = self.synthesizer.sample_from_conditions(
53                  conditions=[balanced_conditions_0]
54              )
55          elif minority_class == 1:
56              balanced_conditions_1 = Condition(
57                  num_rows=synthetic_samples_needed,
58                  column_values={self.target: 1},
```

```
            )
            df_synth = self.synthesizer.sample_from_conditions(
                conditions=[balanced_conditions_1]
            )

        X_balanced = pd.concat(
            [
                X.reset_index(drop=True),
                df_synth.drop(self.target, axis=1).reset_index(
                    drop=True),
            ],
            axis=0,
            ignore_index=True,
        )

        y_balanced = pd.concat(
            [y.reset_index(drop=True), df_synth[[self.target]].
                reset_index(drop=True)],
            axis=0,
            ignore_index=True,
        )

        if sparse.issparse(X):
            X_balanced = sparse.vstack([X_balanced], format=X.
                format)
        else:
            X_balanced = np.vstack([X_balanced])

        y_balanced = pd.Series(np.ravel(y_balanced))

        self.X_balanced = X_balanced
        self.y_balanced = y_balanced

```

```
89          X = X_balanced
90          y = y_balanced
91
92          return X_balanced, y_balanced
```

Código-fonte 5 – Wrapper para a Estratégia TSTR usando Pipeline

```
# TSTR (Treinar com Dados Sint ticos, Testar com Dados Reais)
# Dados Sint ticos Balanceados
class SDVPipelineTSTR(BaseSampler):
    _sampling_type = "over-sampling"

    _parameter_constraints = {
        "X": [pd.DataFrame],
        "y": [pd.DataFrame, pd.Series],
        "metadata": [object],
        "target": [str],
        "num_features": [list],
        "cat_features": [list],
    }

    def __init__(
        self,
        metadata,
        target,
        num_features,
        cat_features,
        columns_order,
        float_to_int_features,
    ):
        self.metadata = metadata
        self.target = target
        self.num_features = num_features
        self.cat_features = cat_features
        self.columns_order = columns_order
        self.float_to_int_features = float_to_int_features
        self.synthesizer = SingleTablePreset(self.metadata, name
            ="FAST_ML")
```

```
31          super().__init__()
32
33      def _fit_resample(self, X, y):
34          X = pd.DataFrame(X, columns=self.columns_order).
                reset_index(drop=True)
35          y = pd.DataFrame(y, columns=[self.target]).reset_index(
                drop=True)
36          df_train = pd.merge(X, y, left_index=True, right_index=
                True)
37
38          for col in self.float_to_int_features:
39              X[col] = X[col].astype(int)
40              df_train[col] = df_train[col].astype(int)
41
42          self.synthesizer.fit(df_train)
43
44          class_counts = y[self.target].value_counts()
45          majority_class = class_counts.idxmax()
46          synthetic_samples_needed = class_counts.max()
47
48          balanced_conditions_0 = Condition(
49              num_rows=synthetic_samples_needed,
50              column_values={self.target: 0},
51          )
52
53          balanced_conditions_1 = Condition(
54              num_rows=synthetic_samples_needed,
55              column_values={self.target: 1},
56          )
57
58          df_synth = self.synthesizer.sample_from_conditions(
59              conditions=[balanced_conditions_0,
                    balanced_conditions_1]
```

```
60          )
61
62          X_train_synth = df_synth.drop(self.target, axis=1)
63          y_train_synth = df_synth[self.target]
64
65          if sparse.issparse(X):
66              X_train_synth = sparse.vstack([X_train_synth],
                    format=X.format)
67          else:
68              X_train_synth = np.vstack([X_train_synth])
69
70          y_train_synth = pd.Series(np.ravel(y_train_synth))
71
72          self.X_train_synth = X_train_synth
73          self.y_train_synth = y_train_synth
74
75          X = X_train_synth
76          y = y_train_synth
77
78          return X_train_synth, y_train_synth
```