PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Validação de modelos computacionais: um estudo integrando *Generative Adversarial Networks* e Simulação a Eventos Discretos

Gustavo Teodoro Gabriel

Itajubá, julho de 2022

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Gustavo Teodoro Gabriel

Validação de modelos computacionais: um estudo integrando *Generative Adversarial Networks* e Simulação a Eventos Discretos

Tese submetida ao programa de Pós-Graduação em Engenharia de Produção como parte dos requisitos para obtenção do Título de Doutor em Ciências em Engenharia de Produção.

Área: Engenharia de Produção

Orientador: Prof. Dr. José Arnaldo Barra Montevechi

Coorientador: Prof. Dr. Fabiano Leal

Julho de 2022
Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Gustavo Teodoro Gabriel

Validação de modelos computacionais: um estudo integrando *Generative Adversarial Networks* e Simulação a Eventos Discretos

Tese aprovada por banca examinadora em 18 de julho de 2022, conferindo ao autor o título de Doutor em Ciências em Engenharia de Produção.

Banca Examinadora:

Prof. Dr. Anderson Rodrigo de Queiroz (North Carolina Central University)

Prof. Dr. Fernando Augusto Silva Marins (UNESP)

José Henrique de Freitas Gomes (UNIFEI)

Prof. Rafael de Carvalho Miranda (UNIFEI)

Prof. Dr. Fabiano Leal (coorientador - UNIFEI)

Prof. Dr. José Arnaldo Barra Montevechi (orientador)

Julho de 2022
Itajubá

DEDICATÓRIA

À Deus, aos meus pais Valéria e Adauto, à minha irmã Júlia e aos meus familiares e amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, aos meus pais Valéria e Adauto e a minha irmã Júlia, que mesmo de longe sempre me apoiaram e me deram suporte. Agradeço aos meus avós, pelos conselhos e por passar de forma simples a tranquilidade que eu preciso.

Agradeço ao professor José Arnaldo Barra Montevechi pela orientação, confiança e por proporcionar inúmeras oportunidades nesses quatro anos de pesquisa do doutorado. Também não posso deixar de agradecer ao professor Fabiano Leal pela coorientação e parceria desde os tempos de graduação.

Agradeço ainda aos professores e colegas de trabalho do grupo de pesquisa NEEAD que sempre estiveram dispostos a ajudar, contribuindo pela melhoria do trabalho.

Não posso deixar de agradecer aos meus amigos e companheiros de laboratório, que me ajudaram com críticas, sugestões tanto no trabalho como em artigos. Quero agradecer principalmente ao Afonso no qual seguimos o doutorado em colaboração e aos amigos Carlos Henrique e João Victor que trabalhamos diretamente ao longo desses quatro anos de pesquisa.

Agradeço também aos meus amigos de longa data, principalmente ao Faca Amolada que, mesmo de longe sempre os considero. Aos amigos da graduação e mestrado que continuaram comigo e que me apoiaram. Também agradeço a República Tijolinho e aos agregados que passaram a maior parte do tempo dividindo os momentos bons e ruins ao longo desses anos.

Agradeço à FlexSim pela oportunidade e crescimento proporcionado para atingir aos objetivos do trabalho. Sou muito grato pela confiança, parceria e aprendizados.

Agradeço a CAPES pelo apoio financeiro dado a pesquisa durantes os quatro anos e meio de trabalho.

Enfim, sou muito grato pelo apoio dado por essas pessoas, não somente para a contribuição pessoal, mas também profissional. Obrigado!

EPÍGRAFE

“Não precisam que me digam

De que lado nasce o sol

Porque bate lá meu coração”

Belchior

RESUMO

A validação de modelos computacionais de Simulação a Eventos Discretos (SED) é primordial para o sucesso do projeto, pois é a partir dela que se garante que o modelo simulado corresponde ao sistema real. Apesar disso, não é possível assegurar que o modelo represente 100% o sistema real. A literatura sugere várias técnicas de validação, porém é preferível o uso de testes estatísticos pois eles apresentam evidências matemáticas. Entretanto, existem limitações, pois testam média ou desvio padrão de forma individual, sem levar em consideração que os dados podem estar dentro de uma tolerância pré-estabelecida. Pode-se utilizar as *Generative Adversarial Networks* (GANs) para treinar, avaliar, discriminar dados e validar modelos de SED. As GANs são duas redes neurais que competem entre si, sendo que uma gera dados e a outra os discrimina. Assim, a tese tem como objetivo propor um método de validação de modelos computacionais de SED para avaliar uma ou mais métricas de saída, considerando uma tolerância para a comparação dos dados simulados com os dados reais. O método proposto foi dividido em duas fases, onde a primeira, denominada "Fase de Treinamento", tem como objetivo o treinamento dos dados e a segunda, "Fase de Teste", visa discriminar os dados. Na segunda fase, é realizado o Teste de Equivalência, o qual analisa estatisticamente se a diferença entre o julgamento dos dados está dentro da faixa de tolerância determinada pelo modelador. Para validar o método proposto e verificar o Poder do Teste, foram realizados experimentos em distribuições teóricas e em um modelo de SED. Assim, as curvas com o Poder do Teste para a tolerância real de 5.0%, 10.0% e 20.0% foram geradas. Os resultados mostraram que é mais eficiente o uso do conjunto de dados que apresenta uma amostra maior na "Fase de Teste" e é mais adequado o conjunto de tamanho amostral menor na "Fase de Treinamento". Além disso, a confiança do Poder do Teste aumenta, apresentando intervalos de confiança menores. Ainda, quanto mais métricas são avaliadas de uma só vez, maior deve ser a quantidade de dados inseridos no treinamento das GANs. O método ainda sugere classificar a validação em faixas que mostram o quão válido o modelo é: Muito Forte, Forte, Satisfatória, Marginal, Deficiente e Insatisfatória. Por fim, o método foi aplicado em três modelos reais, sendo dois deles na área de manufatura e um na área da saúde. Concluiu-se que o método proposto foi eficiente e conseguiu mostrar a o grau de validação dos modelos que representam o sistema real.

Palavras-chave: Simulação a Eventos Discretos, GANs, Inteligência Artificial, Validação de Modelos Computacionais, Teste de Equivalência.

ABSTRACT

Computer model validation of Discrete Event Simulation (DES) is essential for project success since this stage guarantees that the simulation model corresponds to the real system. Nevertheless, it is not possible to assure that the model represents 100% of the real system. The literature suggests using more than one validation technique, but statistical tests are preferable. However, they have limitations, since the tests usually test the mean or standard deviation individually, and do not consider that the data may be within a pre-established tolerance limit. In this way, Generative Adversarial Networks (GANs) can be used to train, evaluate and discriminate data and validate DES models, because they are two competing neural networks, where one generates data and the other discriminates them. The proposed method is divided into two phases. The first is the "Training Phase" and it aims to train the data. The second, the "Test Phase" aims to discriminate the data. In addition, in the second phase, the Equivalence Test is performed, which statistically analyze if the difference between the judgments is within the tolerance range determined by the modeler. To validate the proposed method and to verify the Power Test, experiments were carried out in continuous, discrete, and conditional distributions and in a DES model. From the tests, the Power Test curves were generated considering a real tolerance of 5.0%, 10.0% and 20.0%. The results showed that it is more efficient to use the dataset that presents larger sample in the “Test Phase” while the set with smaller sample size needs to be used in the “Training Phase”. In addition, the confidence of the Power Test increases with big higher dataset in first phase, presenting smaller confidence intervals. Also, the more metrics are evaluated at once, the greater the amount of data inputted in the GANs' training. The method suggests classifying a validation based on the achieve tolerance: Very Strong, Strong, Satisfying, Marginal, Deficient and Unsatisfying. Finally, the method was applied to three real models, two of them in manufacturing and the last one in the health sector. We conclude that the proposed method was efficient and was able to show the degree of validation of the models that represent the real system.

Keywords: Discrete Event Simulation, GANs, Artificial Intelligence, Computer Model Validation, Equivalence Test.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

AM – Aprendizagem de Máquina

AS – Aprendizado Supervisionado

AnS – Aprendizado não Supervisionado

ApR – Aprendizado por Reforço

AWS – *Amazon Web Service*

C2ST – *Classifier Two-Sample Tests*

cGANs – Conditional *Generative Adversarial Network*

DC – Dados de Comparação

DL – *Deep Learning*

DR – Dados Reais

DS – Dados Simulados

DT – Dados de Treinamento

GANs – *Generative Adversarial Network* – Redes Adversárias Generativas

IA – Inteligência Artificial

IoT – Internet of Things – Internet das Coisas

k-NN – *k-nearest neighbors*

LOS – *Length of Stay* – Tempo de permanência

PCA – *Principal Component Analysis*

PP – Pergunta de Pesquisa

RNA – Redes Neurais Artificiais

RSL – Revisão Sistemática da Literatura

SED – Simulação a Eventos Discretos

TEDDP – Teste de Equivalência para a Diferença de Duas Proporções

TOST – *Two One-Sided Test*

TPM – Tempo Porta-Médico

TPT – Tempo Porta-Triagem

V&V – Verificação e Validação

WSC – *Winter Simulation Conference*

SUMÁRIO

1. INTRODUÇÃO

1.1. Contextualização

Modelos de Simulação a Eventos Discretos (SED) surgiram na área militar e desde então são utilizados para a implementação de novos sistemas e melhorias de processo, análise de capacidade, alocação de recursos, auxílio na tomada de decisão, entre outros. Scheidegger *et al.* (2018) identificaram mais de 20 áreas nas quais a SED foi utilizada, como manufatura, transporte, serviços e saúde. Além disso, com o avanço da indústria 4.0, os modelos vêm se tornando cada vez mais sofisticados e sendo utilizados em tempo real ou quase-real (RODIČ, 2017; SANTOS *et al.* 2020).

Esses modelos desenvolvidos são frequentemente usados em situações que envolvem grande risco pessoal ou financeiro (SOKOLOWSKI e BANKS, 2010). Por essa razão, modeladores e tomadores de decisão se preocupam com a confiabilidade e seus resultados (SARGENT, 2015). Para Popovics, Pfeiffer e Monostori (2016), as decisões devem ser tomadas com base em resultados de simulação corretos, e os modelos só trazem resultados efetivos se correspondem de fato ao sistema simulado. Para isso, os métodos existentes na literatura para realizar um projeto de simulação sugerem a etapa de validação do modelo computacional.

De fato, Montevechi *et al.* (2015) identificaram os principais métodos utilizados em projetos de simulação e, entre oito frameworks, apenas um não faz menção a esta etapa. Harrel *et al.* (2012) afirmam que a validação computacional é essencial para garantir que o modelo simulado represente o sistema real, uma vez que o processo de simulação é propenso a erros. Além disso, se o modelo construído não refletir a realidade, os experimentos realizados no modelo de simulação não darão uma resposta segura para o sistema real (BANKS e CHWIF, 2011).

Nesse contexto, Zeigler e Nutaro (2016) destacaram que a validade é o grau em que um modelo representa fielmente sua contraparte do sistema. Os processos de validação têm como objetivo a quantificação da precisão do modelo, comparando os resultados da simulação com resultados experimentais ou operacionais no mundo real (THACKER *et al.*, 2004), determinando se o modelo representa adequadamente o sistema para o propósito desejado (LAW, 2015; SARGENT e BALCI, 2017). Com isso, mesmo que não se possa garantir que um modelo seja 100% válido, a validação é parte crucial para que os resultados obtidos do modelo de simulação sejam confiáveis (KIESLING, LÜTHI e KHAYARI, 2005; CHWIF, MUNIZ e SHIMADA, 2008; OLSEN e RAUNAK, 2015).

Nesse sentido, processos de Verificação e Validação (V&V) foram desenvolvidos para minimizar esses erros e fazer modelos úteis para atender às finalidades para as quais foram criados (LEAL *et al.*, 2011). Enquanto a verificação do modelo visa garantir que as lógicas do modelo computacional e sua implementação estejam corretas, a validação do modelo tem como objetivo a comprovação de que um modelo computacional possui uma faixa satisfatória de precisão consistente com o propósito pretendido sobre o domínio da aplicabilidade planejada (SARGENT, GOLDSMAN e YAACUB, 2016).

O processo de validação não é uma tarefa simples. Apesar do presente estudo focar na validação do modelo computacional, Robinson (2002), Balci (2010) e Leal *et al.* (2011) afirmam que a validação é de grande importância em toda a etapa da simulação. Para a validação computacional, ela deve ocorrer de forma cíclica (POPOVICS, PFEIFFER e MONOSTORI, 2016), contínua (SARGENT, 1986; KLEIJNEN, 1995; BALCI, 2010; WANG, 2013) e iterativa (TSIOPTSIAS; TAKO e ROBINSON, 2016) ao longo do desenvolvimento do modelo (FOURES; ALBERT e NKETSA, 2013). Parte do modelo deve ser construído, verificado e validado antes de prosseguir. Banks e Chwif (2011) sugerem que os modelos devem ser construídos dos detalhes mais simples para os mais complexos, facilitando assim a validação e evitando retrabalho (reduz o tempo de modelagem) quando os modelos se tornam complexos e os erros são mais difíceis de serem encontrados. Além disso, quando os erros são detectados mais cedo assegura-se uma melhor qualidade ao projeto (BALCI, 2010; FOURES; ALBERT e NKETSA, 2013). De fato, várias versões do modelo são realizadas antes de se chegar à versão final, uma vez que problemas podem aparecer ao longo de sua construção (SARGENT, 2013).

Ainda, há duas abordagens para a validação computacional, onde ela pode ser realizada para sistemas reais e sistemas hipotéticos. No primeiro caso, a validação é baseada na comparação do modelo construído no computador e as medições realizadas no sistema físico. Nessa abordagem, deve-se explorar, de forma mais completa possível, as saídas dos modelos reais com os modelos simulados. Nos sistemas hipotéticos, o sistema modelado é criado apenas para um *design* e geralmente não é possível obter um satisfatório grau de confiança do modelo (SADOUN, 2000; SARGENT, 2013). Modelos hipotéticos são mais difíceis de serem validados, pois é necessário utilizar apenas da experiência do modelador, ou de técnicas de análises de sensibilidade. Portanto, o foco do trabalho é a validação de modelos que representam os sistemas reais.

1.2. Justificativa e Relevância

Apesar dos *frameworks* sugerirem a etapa de validação do modelo computacional, Brade e Lehmann (2002) destacam que, infelizmente, a maior parte do processo de validação é incompleto e fragmentário. Por muitas vezes, os resultados dessa etapa não são bem estruturados e nem bem documentados, deixando lacunas e falta de evidências de como foram realizados. Para que esse problema possa ser resolvido, a literatura apresenta uma vasta lista de técnicas subjetivas e objetivas para garantir que os resultados do modelo de simulação sejam equivalentes aos do sistema real. Porém, a aplicação dessas técnicas ao longo do ciclo de vida de um projeto de simulação pode ser demorada e dispendiosa (CHWIF, MUNIZ e SHIMADA, 2008; SARGENT, 2013). Na prática, sob pressão de tempo para concluir um estudo de simulação, a validação e a sua documentação são sacrificadas primeiro (BALCI, 1994) e, como consequência, todas as etapas seguintes podem não responder à realidade (BANKS e CHWIF, 2011). Esse problema pode se tornar cada vez mais grave, na medida em que modelos de simulação têm se tornando mais complexos (HOLLMANN, CRISTIÁ e FRYDMAN, 2014).

Segundo Sargent (2013), uma das dificuldades para as atividades de validação se deve ao fato de que infelizmente não há um conjunto de testes específicos que possam ser facilmente aplicados para determinar a "correção" de um modelo. Além disso, não existe algoritmo para determinar quais técnicas ou procedimentos usar (SARGENT e BALCI, 2017). De acordo com Tsioptsias, Tako e Robinson (2016), a falta de validação perfeita ou única nos modelos leva os modeladores a realizar um *trade-off* de validade. Não existe um padrão de quais variáveis do modelo selecionar para o processo de V&V, uma vez que, segundo Raunak e Olsen (2014), cada elemento da simulação é um candidato a validação.

Para analisar e verificar como a etapa de validação computacional está consolidada na literatura, foi realizada uma Revisão Sistemática da Literatura (RSL). Para isso, uma busca exploratória foi realizada nas bases de dados *Scopus*, *Science Direct* e *Web of Science*. Optou-se por essas bases pois são consideradas as maiores (ALRABGHI e TIWARIA, 2015) e que possuem uma alta taxa de citação (FRANCESCHINI; MAISANO; MASTROGIACOMO, 2014). A pesquisa pelos artigos foi realizada no dia 15 de janeiro de 2020, nas bases citadas anteriormente Três grupos de termos foram definidos para a busca e estão apresentados na Tabela 1.1. O primeiro grupo restringe a pesquisa apenas a SED. O segundo evidencia que os artigos devem apresentar, em seu escopo, como a validação computacional foi inserida no projeto. Por fim, o terceiro grupo faz referência aos artigos que expõem a parte teórica da validação computacional. As palavras de busca utilizadas poderiam estar contidas no título,

resumo ou palavras-chaves. Foi utilizado o conector booleano *OR* para palavras que estão entre um mesmo grupo e o conector *AND* para palavras que estão em grupos diferentes. Não houve restrição de data para a pesquisa dos artigos.

Tabela 1.1 - Grupos de termos para a busca nas bases de dados

Simulação	**Validação**	**Teoria**
Discrete Event Simulation	*computer model validation* *model valid** *validation and verification* *verification and validation* *V&V*	*state of art* *framework* *overview* *conduct* *process* *history* *literature review* *model*

Foram encontrados 191 artigos na base de dados *Scopus*, 7 artigos na base de dados *Web of Science*, 74 na base *Science Direct* e 25 artigos relacionados a outras fontes, totalizando 297 artigos para análise. Foram detectados 56 artigos duplicados. Sendo assim, 241 artigos passaram para a etapa de leitura de título e resumo. Após a etapa de rastreamento e a identificação dos artigos que apresentaram os critérios de inclusão e exclusão, 190 artigos foram excluídos, passando 51 (21.1%) para a fase de leitura integral. Na fase de leitura integral, 22 artigos foram excluídos, sendo que 3 não foram encontrados disponíveis em sua versão total, 5 não apresentava aplicação da SED, 12 não responderam a nenhuma pergunta definida na etapa de planejamento e 2 eram parte de um capítulo de livro. Dessa maneira, foram selecionados 29 artigos para a etapa de síntese qualitativa e análise dos dados. A metodologia, análise e resultados podem ser encontrados no Apêndice A.

Como resultado, a literatura apresenta cerca de 75 técnicas para auxiliar o modelador na etapa (SARGENT e BALCI, 2017) que são divididas em técnicas subjetivas e objetivas. As técnicas subjetivas são baseadas na experiência do tomador de decisão que conhece o modelo de simulação. Desse modo, técnicas subjetivas dificilmente garantem os resultados obtidos nos experimentos de simulação (LEAL *et al.*, 2011). Eles são considerados como a única possibilidade quando nenhuma abordagem objetiva pode ser aplicada (WANG, 2013) e são recomendadas quando sistemas hipotéticos são simulados. Por outro lado, técnicas objetivas são preferidas, uma vez que usam métodos matemáticos ou estatísticos para comparar modelos simulados com sistemas observáveis. Nesse sentido, dados reais são necessários. Segundo Balci (1994) e Wang (2013), as técnicas objetivas fornecem evidências únicas, mas seu uso requer experiência e conhecimento mais profundo por parte do modelador.

Quando o modelador usa técnicas objetivas, Sargent (2013) sugere que o intervalo de confiança e testes estatísticos sejam utilizados. Se o modelo simulado for validado por apenas uma saída, ele pode ser executado usando estatísticas univariadas podendo ser testes paramétricos (*1-sample t*, *2-sample t*, teste F) ou testes não paramétricos (1 Wilcoxon *sample*, *Mann-Whitney*), dependendo da disponibilidade de dados. Sargent (2013) afirma que, se as variáveis de interesse são aleatórias, os testes avaliam se a média ou variância do modelo de simulação e do sistema real são iguais. Além disso, Leal *et al.* (2011) propõem uma diretriz para comparar intervalos de confiança de acordo com as propriedades dos dados.

Apesar da literatura apresentar vários testes estatísticos de comparação de média e variância, conforme mostrado no estudo de Sargent e Balci (2017), Sargent (2015) indica que esses testes usam apenas um único ponto ao invés de um intervalo, ou consideram intervalos de forma indireta. Assim, os modeladores assumem que o sistema que está sendo modelado é um sistema observável e que os dados necessários para a validação podem ser obtidos e coletados. Dessa maneira, o autor criou um teste onde uma faixa de precisão ou tolerância é levada em consideração nos intervalos de confiança. Sargent (2015) criou esse teste por acreditar que o modelo de simulação nunca é validado na sua totalidade e há faixas de tolerância em que o modelo pode estar contido e ainda ser considerado válido. A tolerância é importante porque os resultados podem variar em uma faixa aceitável (limite superior e inferior) previamente definida pela equipe do modelador e pelo cliente. Sargent *et al.* (2016) apontam que a tolerância deve ser definida em um estágio inicial no projeto de simulação.

Testes univariados são recomendados quando o objetivo do projeto é avaliar apenas uma saída. Além disso, se o projeto pretende avaliar mais métricas, todas elas precisam ser validadas. Nesse sentido, mais de um teste deve ser realizado. Balci (1994) e Sargent *et al.* (2016) afirmam que técnicas multivariadas devem ser aplicadas se os resultados forem correlacionados. Portanto, os intervalos de confiança simultâneos devem ser utilizados para mostrar como as variáveis se comportam no modelo. Sargent e Balci (2017), em uma análise da prática da validação nos últimos anos mostra que técnicas multivariadas como intervalos de confiança simultâneos, teste *2-sample* T^2 *de Hotteling* e MANOVA podem ser encontrados na literatura. Apesar disso, o mais utilizado é o teste *2-sample de Hotteling* T^2 que, segundo Balci e Sargent (1984), testa a equivalência das médias de duas populações normais variadas.

Embora seja possível realizar os testes estatísticos da literatura, eles apresentam algumas limitações. Se o modelador aplicar testes univariados para cada métrica avaliada, o erro Tipo II começa a aumentar, uma vez que a significância do modelo começa a diminuir. O erro ocorre quando aceitamos um modelo inválido e é considerado um risco do construtor de

modelos. O erro do Tipo II é o mais sério porque ações desastrosas podem acontecer, uma vez que a tomada de decisão baseada no modelo inválido pode levar ao colapso do sistema real (BALCI e SARGENT, 1981; BALCI, 1994). Os testes multivariados também apresentam limitações. O *teste de Hotteling* T^2 testa a equivalência de médias de duas populações normais (BALCI e SARGENT, 1982) e não se preocupa com o desvio padrão.

Considerando os problemas apresentados, o trabalho propõe o uso de *Deep Learning* (DL) para validar modelos de SED. O DL foi introduzido no Aprendizado de Máquina (AM) na década de 1980 e depois foi usado para Redes Neurais Artificiais (RNA) (SCHIMIDHUBER, 2015). Os métodos DL são uma evolução das RNAs, apresentando várias camadas que são capazes de aprender dados que contêm vários níveis de abstração (LECUN *et al.*, 2015). Ao utilizar as GANs é possível validar modelos de SED que simulam sistemas reais e que apresentam muitos dados disponíveis, algo que vem sendo cada vez mais possível com a digitalização das indústrias e a inserção do grande volume de dados disponível pela implementação da Indústria 4.0.

Uma vez que os testes estatísticos avaliam apenas a média ou o desvio das distribuições de forma independente, pode-se utilizar um tipo de RNA, sendo ela as *Generative Adversarial Networks* (GANs) que é um algoritmo de ML usado para treinar, avaliar e discriminar dados. De acordo com Ma *et al*. (2021), as GANs e suas variações proporcionam as aplicações mais interessantes em ML. Elas foram desenvolvidas por Goodfellow *et al*. (2014) e são modelos de aprendizagem profunda. São duas RNAs que competem entre si, onde uma gera dados e a outra os discrimina. O Gerador *G* gera dados e amostras falsas para enganar a rede discriminadora. O Discriminador *D* diferencia se os dados gerados são falsos ou verdadeiros. Após o treinamento, a rede *G* passa a produzir amostras semelhantes aos dados reais. Portanto, essa estrutura permite a geração de novas amostras sintéticas que se ajustem à verdadeira população a partir da qual foi gerada. O uso de GANs é um dos campos de pesquisa mais importantes em inteligência artificial (PAN *et al.*, 2019), pois é possível observar comportamentos não lineares e dependência entre as variáveis presentes no conjunto de dados (GOODFELLOW *et al.*, 2014; AGNESE *et al.*, 2020).

1.3. Definição do problema

Como abordado anteriormente, os testes convencionais da literatura permitem avaliar se a média, mediana ou desvio padrão de uma amostra são estatisticamente iguais às de outras. Além disso, alguns testes necessitam que os dados apresentem algumas premissas, como tamanho de amostra, seguir uma distribuição normal, entre outras. Apesar disso, sabe-se que as

saídas de alguns sistemas não se adequam a essas exigências e assim violam os pressupostos necessários para a realização da comparação dos dados. Diante disso, Zhang, Patuwo e Hu (1998) descrevem que as redes neurais conseguem se adaptar aos dados inseridos no algoritmo e modelá-los independentemente das relações, sejam elas lineares ou não, eliminando assim, a necessidade de se seguir pressupostos estatísticos. De fato, Brownlee (2020) afirma que as GANs se adaptam a qualquer tipo de dados, sejam eles reais, inteiros, booleanos ou categóricos codificados.

Além dos testes já citados, há os testes de Equivalência, onde é possível comparar se a média de uma amostra é estatisticamente igual a um alvo ou a uma outra amostra mais ou menos uma variação estipulada pelo tomador de decisão. O teste consegue captar apenas a média dos dados e não o comportamento deles como um todo. Uma maneira de resolver esse problema é utilizar o Teste de Equivalência para a Diferença de Duas Proporções (TEDDP). Apesar disso, é necessário que haja uma classificação dos dados, que podem ser realizadas pelas GANs.

Assim, a presente tese contribui com a literatura ao integrar as GANs com um teste estatístico que considera uma tolerância para validar modelos de simulação. Testes tradicionais requerem pressupostos estatísticos e, na maioria das vezes não consideram a diferença entre o conjunto de dados do modelo simulado e o sistema real. Ao utilizar a técnica proposta, é possível validar modelos de SED reais que apresentam coleta de dados com alto volume, considerando tolerância nas saídas e aumento a confiabilidade do modelo.

Portanto, diante dos problemas apresentados, e considerando múltiplas saídas do modelo simulado e uma tolerância nas saídas, o trabalho visa responder se: “É possível desenvolver um método onde uma ou mais métricas de validação podem ser validadas conjuntamente considerando uma faixa de precisão entre as saídas e diminuindo a probabilidade de se validar um modelo não válido (Erro Tipo II)?”

1.4. Objetivos

O trabalho tem como objetivo geral propor um método de validação de modelos computacionais de SED que permita avaliar uma ou mais métricas de saída, aumentando a confiabilidade dos testes e, ao mesmo tempo, considerando uma tolerância para a comparação desses resultados. Para isso, serão utilizadas as GANs para o treinamento dos dados e o discriminador para julgar os dados simulados e reais. Elas são utilizadas pois os dados inseridos não precisam obedecer a premissas de normalidade que testes estatísticos convencionais exigem. Além disso, o presente trabalho tem como objetivos específicos:

- Testar o método proposto em distribuições teóricas contínuas e discretas;
- Testar o método proposto em um modelo de SED teórico;
- Avaliar o poder do teste para as distribuições e modelo testado;
- Propor uma escala de validação de modelos computacionais de acordo com a tolerância considerada entre dados reais e simulados;
- Validar três modelos reais com o método proposto utilizando as GANs.

1.5. Delimitações do estudo

A pesquisa se delimita em utilizar as GANs como meio de treinamento de dados e o seu discriminador para julgar os dados treinados e dados comparados. Além disso, admite-se que os dados inseridos já estão preparados, ou seja, não existem dados faltantes e, quando utilizados para avaliar mais de uma característica, eles apresentam a mesma quantidade em cada atributo. Para a realização do Teste de Equivalência, considera-se que a as tolerâncias utilizadas são únicas e o intervalo deve ser simétrico.

Os modelos de simulação utilizados no presente trabalho já foram validados em estudos realizados previamente. Dessa maneira, assume-se que todas as etapas de construção do modelo vindas anteriormente foram realizadas conforme métodos de construção de modelo de simulação presentes na literatura. A modelagem conceitual, coleta de dados, construção do modelo computacional já estão verificadas e validadas.

1.6. Estrutura do trabalho

O presente trabalho encontra-se estruturado em seis capítulos. O capítulo1, apresentado anteriormente, mostrou uma introdução do tema, contextualizando o uso da validação e a seu estado da arte em modelos de SED justificando a escolha do tema. Além disso, apresentou o problema de pesquisa e os objetivos do trabalho. O Capítulo 2 apresenta a fundamentação teórica para o entendimento dos conceitos utilizados, sendo discutidos a Simulação a Eventos Discretos, a Inteligência Artificial, as *Generative Adversarial Networks* e Testes Estatísticos. No Capítulo 3 encontra-se a classificação da pesquisa e o método de pesquisa utilizado. O Capítulo 4 mostra a explicação do método proposto. O Capítulo 5 apresenta a validação do método por meio de experimentos em distribuições estatísticas teóricas e objetos de estudo para a validação de modelos de SED. Por fim, no capítulo 6 são apresentadas as conclusões, nas quais são explanadas a síntese dos resultados, limitações da pesquisa, sugestões para trabalhos futuros e as considerações finais, seguidas pela bibliografia e os Apêndices.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os aspectos gerais da simulação computacional com foco na Simulação a Eventos Discretos. Assim, são apresentadas as suas definições, seus elementos, vantagens e desvantagens. Além disso, destaca-se os aspectos e a importância da validação computacional e as recomendações para conduzir essa etapa presente na literatura. Ainda, é explicado como a Inteligência Artificial (IA) pode ser uma ferramenta integrada à Simulação e como pode ser utilizada para melhorias e apoio à tomada de decisão. Dentro das possibilidades e algoritmos da IA, é explicado o funcionamento das GANs e das cGANs (utilizadas em dados condicionais) e seu papel na validação do modelo computacional. Por fim, os testes estatísticos utilizados são demonstrados para realizar a comparação dos dados simulados com os dados reais.

Para a elaboração da fundamentação teórica e apresentação dos conceitos listados acima, foram analisados e estudados artigos de periódicos e de congressos nacionais e internacionais, principais livros da área da simulação, teses e dissertações.

2.1. Definições gerais de simulação

Segundo Banks *et al.* (2010) a simulação é a imitação de um sistema ou processo do mundo real ao longo de um tempo, sendo realizada manualmente ou em computador. Assim, ela é considerada a criação de uma história artificial de um sistema para que se possa observar e extrair inferências sobre as características do modelo simulado.

De acordo com Harrel, Ghosh e Bowden (2012), a simulação é a reprodução de um processo dinâmico utilizando um modelo computacional. Através desse modelo, é possível avaliar, medir e aperfeiçoar as métricas de qualquer sistema. Para Negahban e Yilmaz (2014) a simulação consegue desenvolver qualquer tipo de modelo seja ele real ou hipotético. Law (2015) diz que a finalidade dos experimentos é verificar como o sistema real reage as situações de contorno e possíveis mudanças realizadas por um longo ou curto espaço de tempo.

Montevechi *et al.* (2007) defendem que a simulação é a importação de problemas reais para um ambiente virtual e controlado. É a partir do ambiente controlado que se estuda as mudanças e os fatores envolvidos no processo sem gerar elevados custos e danos físicos ao processo real. Ahmed, Scoble e Dunbar (2016) compartilham da mesma ideia, dizendo que a simulação vem se tornando uma técnica popular para a representação de sistemas complexos e a análise de cenários permite o controle sobre o custo e tempo.

O modelo de simulação computacional deve capturar e representar com fidelidade o mesmo sistema em condições reais (CHWIF e MEDINA, 2015). Apesar disso, os elementos envolvidos em cada projeto podem variar, uma vez que dependem do propósito e como a pesquisa é realizada (ALRABGHI e TIWARI, 2013).

A principal finalidade da simulação é responder questões do tipo "*what if*", ou seja, "o que aconteceria se" (BANKS *et al.* 2010). As experimentações respondem à pergunta e auxiliam na análise de cenários, propondo as melhorias para o processo real (ROBINSON, 2014; JU *et al.*, 2015; CHWIF e MEDINA, 2015). Atieh *et al.* (2016) afirmam que a análise de cenário e a experimentação fortalecem a visualização de todas as áreas que devem ser melhoradas e não somente avalia a diferença relativa entre o processo em seu estado atual e as modificações sugeridas. Laurindo *et al.* (2019) ainda dizem quem as perguntas do tipo "o que aconteceria se" podem ser respondidas mesmo sem a necessidade de um sistema real. Dessa forma, a simulação é utilizada para antecipar os acontecimentos de sistemas e processos que ainda são projetos, ajudando na tomada de decisão.

AbouRizk (2010) conclui que a simulação é uma boa alternativa quando se requer uma solução integrada ou é necessário a flexibilização da lógica de construção do modelo. Assim, a utilização de animações computacionais facilita a visualização dos processos, as análises de gargalo, fluxo, processamento e a possibilidade de otimização dos modelos. A simulação vem sendo empregada em diversos campos como sistemas de manufatura, transportes, setores públicos, saúde, militar, recursos naturais, entretenimento, entre outros (BANKS *et al.*, 2010 e LAW, 2015).

2.1.1. Simulação a Eventos Discretos

Os sistemas e processos atuais necessitam de experimentações para testar possíveis melhorias. Quando possível, Law (2015) sugere que estas experimentações sejam feitas através de um modelo, sendo ele físico ou matemático. Apesar de eficientes, modelos físicos como protótipos, miniaturas e objetos não são de interesse da pesquisa operacional. Modelos matemáticos representam o sistema de forma lógica e como eles se comportam ao sofrerem uma variação. Law (2015) os classifica em modelos analíticos e simulação. Os modelos analíticos apresentam soluções exatas através de equações e relações matemáticas. À medida que esses problemas começam a ficar complexos e apresentar variações, onde não é possível utilizar equações e funções, é necessário utilizar a simulação (LAURINDO *et al.*, 2019). Kleijenen (2009) diz que ela é utilizada quando não se quer ou não se consegue fazer um modelo matemático adequado. Hillier e Lieberman (2010) confirmam que a simulação pode ser

utilizada para investigar quase todos os tipos de sistemas estocásticos, sendo assim uma técnica versátil e de grande aplicabilidade.

Banks *et al.* (2010), Chwif e Medina (2015) e Law (2015) classificam os modelos de simulação em três dimensões: determinísticos e estocásticos; estáticos e dinâmicos; discretos e contínuos. Assim, quando não há evidências de aleatoriedade ou probabilidades, os modelos são classificados como estocásticos, caso contrário é um modelo determinístico. Modelos estáticos são modelos que não se alteram com o passar do tempo também denominado de Simulação de Monte Carlo. Por outro lado, os modelos dinâmicos levam em consideração as mudanças ocorridas no sistema ao longo do tempo. Assim, se a evolução ocorrer em espaço de tempo e eventos finitos, é considerado um modelo discreto. Caso contrário, se a evolução ocorrer continuamente e com taxas de mudança ao longo do tempo o modelo é considerado contínuo. A Figura 2.1 mostra a divisão de um sistema.

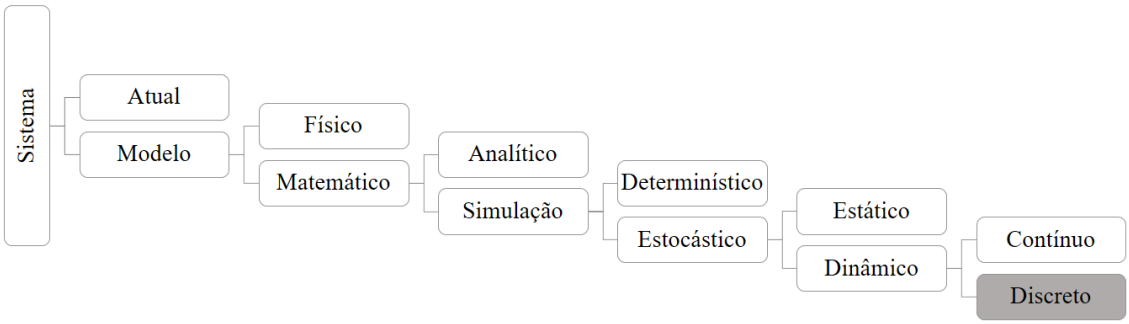

Figura 2.1 - Divisão de um sistema

Fonte: adaptado de Banks *et al.* (2010), Chwif e Medina (2015) e Law (2015)

O presente trabalho tem como objetivo os modelos discretos, onde os eventos ocorrem em espaços finitos de tempo, ou seja, o foco está na Simulação a Eventos Discretos. Portanto, de acordo com Garani e Adam (2008), Yuriy e Varena (2008) e Bateman *et al.* (2013), a SED é definida como a simulação em que as variáveis mudam instantaneamente em um espaço de tempo, ocorrendo em um único momento e orientada por eventos. Assim, o modelo apresenta as suas mudanças de forma aleatória, onde os tempos dos eventos são os pontos do tempo simulado (SCHRIBER, BRUNNER e SMITH, 2017). A SED incorpora efeitos da variabilidade do sistema ao utilizar distribuições de probabilidades para caracterizar as variáveis do sistema de incerteza (BOTÍN, CAMPBELL e GUZMÁN, 2015).

Skoogh, Johansson e Stahre (2012) afirmam que a SED é uma ferramenta promissora para planejar, projetar e melhorar fluxos de processos. Jahangirian *et al.* (2010) dizem que ela é apropriada para níveis táticos e operacionais nas tomadas de decisão. Segundo os autores, ela

tende a ser conveniente para as análises detalhadas dos processos, utilização dos recursos, enfileiramento e análises a curto prazo.

Um aspecto chave na SED é descrição do estado do sistema, o qual inclui valores das variáveis dos processos, como a distribuição de probabilidade de chegadas das entidades, durações dos eventos, status dos eventos e os recursos necessários (DONG *et al*. 2012). De fato, a SED é utilizada em sistemas onde há redes de filas e atividades em que as entidades competem por recursos (MORADI, NASIRZADEH e GOLKHOO, 2015). Assim, Robinson (2014) conclui que ela é eficiente para modelar sistemas em que ocorrem filas devido a taxa de chegada em um local ser maior que a taxa de processamento da próxima atividade.

Jahangirian *et al.* (2010) realizaram um estudo mostrando que a SED é o tipo de simulação mais utilizada dentre os sistemas dinâmicos, baseado em agentes e híbridos. Alrabghi e Tiwari (2015) confirmam que a SED predomina, sendo utilizada tanto sozinha, quanto cominada a outras técnicas. Ainda, Scheidegger *et al.* (2018) concluem que a SED, apesar de ter surgido em meados dos anos 1970 juntamente com os sistemas dinâmicos, é o tipo de simulação que mais apresenta contribuições desde então. Em seu estudo, os autores identificaram mais de 20 áreas das quais ela foi aplicada para o auxílio na tomada de decisão.

2.1.2. Elementos da SED

A SED apresenta alguns elementos característicos. De acordo com Harrel, Ghosh e Bowden (2012) e Schriber, Brunner e Smith (2017) os elementos são: entidades, recursos, elementos de controles e operações. Os elementos definem quem, o quê, quando, onde e como ocorre o processo no modelo computacional. Os elementos variam de acordo com a pesquisa a ser realizada. Assim, busca-se obter um equilíbrio entre o comportamento observado no sistema real e as simplificações realizadas no modelo computacional (ALRABGHI e TIWARI, 2013).

As entidades, de acordo com Harrel, Ghosh e Bowden (2012) são as unidades que se locomovem e são processadas ao longo do sistema simulado. Scheidegger *et al.* (2018) salientam que entidades são todos aqueles elementos que esperam por um serviço ou por um processamento. Entidades externas são criadas e movimentadas pelo modelador, enquanto as entidades internas são criadas em manipuladas pelo próprio software de simulação (SCHRIBER, BRUNNER e SMITH, 2017). As entidades são representadas por clientes, pacientes, documentos, peças, produtos, veículos e projetos.

Os recursos são elementos que fornecem serviços e instalações de suporte para que as operações possam ocorrer. Geralmente as entidades competem pelo seu uso, uma vez que eles são limitados em termos de capacidades (HARREL, GHOSH e BOWDEN, 2012; SCHRIBER,

BRUNNER e SMITH, 2017). Scheidegger *et al.* (2018) consideram como recursos os funcionários, médicos, operadores, trabalhadores, servidores, veículos e equipamentos.

Para Harrel, Ghosh e Bowden (2012) e Schriber, Brunner e Smith (2017), os elementos de controle indicam as lógicas de programação que regem o modelo simulado. Dessa maneira, elementos de controle são contadores, valores de dados do sistema, expressões aritméticas e booleanas.

Por fim, a operação é cada etapa realizada no sistema, podendo ser feita pela ou na própria entidade enquanto está sendo processada (HARREL, GHOSH e BOWDEN, 2012; SCHRIBER, BRUNNER e SMITH, 2017). Na operação há o local onde as entidades são geradas e alocadas no sistema (geralmente no início do processo) e o local onde elas são removidas do modelo (SCHEIDEGGER *et al.*, 2018).

2.1.3. Vantagens e Desvantagens da SED

A literatura apresenta muitas vantagens ao se utilizar a SED como ferramenta de apoio para a tomada de decisão. Banks *et al.* (2010) afirmam que, ao modelar um sistema utilizando a SED, o sistema real não necessita de interrupção para analisar e explorar as regras de decisões, fluxos de informações e procedimentos organizacionais; *layouts* físicos e sistemas de transporte são testados sem comprometer os recursos; as hipóteses de viabilidade de implementação no sistema podem ser testadas; as variáveis do sistema são estudadas e são extraídas análises em relação a importância de cada uma delas; o tempo pode ser expandido ou comprimido para a investigação dos fenômenos e perguntas do tipo “o que acontece se” são respondidas.

Law (2015) e Jahangirian *et al.* (2010) ainda dizem que a SED tem a capacidade de prover uma visualização ampla e completa dos sistemas, permitindo que os tomadores de decisões obtenham os resultados antes de sua implementação. Ainda, é possível que os modeladores projetem experimentos empíricos que seriam caros ou difíceis de serem executados no sistema real ou com pessoas.

Devido as suas propriedades, pode-se modelar entidades distintas com características heterogêneas, permitindo assim analisar e rastrear o *status* das entidades e recursos de forma individual (SCHEIDEGGER *et al.*, 2018). Banks *et al.* (2010) e Scheidegger *et al.* (2018) ainda dizem que ela tem capacidade de representar vários fatores do modelo ao mesmo tempo e, devido à competição por recursos e à formação de filas, ela é útil na identificação de gargalos e desempenho do nível de serviço de um sistema.

Assim como todas as técnicas utilizadas na pesquisa operacional, a SED também apresenta desvantagens. Para se conduzir um eficiente projeto de simulação, o modelador deve

ter o conhecimento de um passo a passo e não somente do software de simulação. Além disso, é necessário que o modelador saiba utilizar ferramentas como probabilidade e estatística, *Design of Experiments*, gerenciamento de projetos, entre outras. Portanto, a mão de obra deve ser qualificada e requer profissionais mais experientes (LAW, 2015).

Banks *et al.* (2010) ainda dizem que é necessário um bom treinamento, uma vez que se dois modelos forem construídos por diferentes pessoas, eles podem apresentar resultados semelhantes, mas nunca serão iguais. Ainda, os resultados da simulação podem ser de difícil interpretação, pois as saídas são variáveis aleatórias e a modelagem e análise podem demorar.

Por fim, se o modelo simulado não for validado (foco deste trabalho) as conclusões retiradas da simulação não apresentam nenhuma utilidade, uma vez que não refletem o sistema real. Devido ao grande número de etapas que um projeto de SED deve conter, é possível que não haja tempo suficiente para se realizar um estudo confiável. Dessa maneira, algumas situações, animações e efeitos visuais podem levar aos tomadores de decisão conclusões prematuras, ineficientes e baseada em poucas evidências (CASRSON, 2004).

2.2. Validação de modelos computacionais

Validade é frequentemente considerada como o grau em que um modelo representa fielmente sua contraparte do sistema (ZEIGLER e NUTARO, 2016). Sua avaliação é de suma importância para projetos de simulação, visto que, por natureza, o processo de construção de um modelo é propenso a erros, gerados no processo de tradução do sistema real em um modelo conceitual que será, mais tarde, convertido em um modelo computacional (HARREL *et al.*, 2012).

A validação é uma etapa de suma importância, pois decisões tomadas em cima de um modelo não validado podem levar ao colapso do modelo real (BALCI e SARGENT, 1981; BALCI, 1994). Assim, dois ou três grupos de pessoas devem estar envolvidos na etapa de validação computacional: o time de modeladores, o usuário final do modelo e/o um time independente, sendo que os dois primeiros são obrigatórios. Quando a validação ocorre pelo time de modeladores, o próprio time decide a validade do modelo através de avaliações e testes que são realizados quando o modelo é construído, denominado de auto validação. Por outro lado, quando o modelo é validado pelo usuário final (co-validação), o usuário deve apresentar uma sinergia com a equipe de modeladores e determina o quão satisfatório o modelo é em cada etapa de sua construção (SARGENT, 2013; KAPOOR e SHAH, 2016; YIN e MCKAY, 2018).

Apesar do próprio usuário do modelo ou do time de modeladores poderem realizar a validação, indica-se a validação independente (SARGENT, 2013; KAPOOR e SHAH, 2016;

SARGENT e BALCI, 2017; YIN e MCKAY, 2018). Um terceiro membro, que não faz parte da equipe e que também não é o cliente, realiza, de forma independente a validação. Esse time independente deve ter o conhecimento e o propósito da construção do modelo. A validação independente apresenta uma probabilidade muito maior de outras pessoas aceitarem o modelo e os resultados estarem corretos, evitando assim o erro do tipo II (erro β). Recomenda-se a utilização desse tipo de validação quando se deseja a aceitação pública do modelo e quando o problema apresenta uma situação de risco com alto custo envolvido (SARGENT, 2013).

Esses usuários podem escolher técnicas subjetivas ou objetivas para a validação. Sargent e Balci (2017) afirmam que existem registro de mais de 75 técnicas na literatura, dentre elas, destaca-se a análise de sensibilidade; animação; análise de gráficos (histogramas, *boxplots*, gráficos de dispersão); testes estatísticos (testes t, intervalos de confiança, análise de média, desvio padrão e variância); validação face a face e teste de *Turing*.

Muitos autores as dividem entre técnicas subjetivas e objetivas. As técnicas objetivas são preferidas, uma vez que utiliza de métodos matemáticos ou estatísticas para comparar modelos simulados com sistemas observáveis ou não. Assim, dados reais são requeridos para que procedimentos estatísticos possam ser realizados e determinar a validade do modelo computacional. Segundo Balci (1998) e Wang (2013), as técnicas objetivas fornecem evidências únicas, mas a sua aplicação requer experiência e conhecimento mais aprofundado pelos modeladores.

As técnicas subjetivas são baseadas no julgamento dos tomadores de decisão que conhecem o modelo que está sendo simulado, assim, dificilmente elas garantem os resultados obtidos nos experimentos da simulação (LEAL *et al.*, 2011). Essas técnicas são consideradas como única possibilidade quando nenhum método objetivo é possível de ser aplicado (WANG, 2013). Apesar de ambas as categorias apresentarem vantagens e desvantagens, recomenda-se a utilização das duas para a validação mais completa.

Ao validar o modelo por meio de técnicas objetivas, pode-se cometer três tipos de erros (BALCI e SARGENT, 1981; BALCI, 1994; BANKS e CHWIF, 2011): (i) Erro do tipo I; (ii) Erro do tipo II; (iii) Erro do tipo III. O erro (i) acontece quando um modelo válido é rejeitado; o (ii) acontece quando um modelo inválido é aceito e o erro (iii) compreende em solucionar um problema errado. Existe uma probabilidade associada a cada tipo de erro. Assim, o Erro do tipo I (probabilidade α) é chamado de risco do construtor do modelo e o erro do tipo II (probabilidade β) é chamado de risco do usuário do modelo (BALCI e SARGENT, 1981; BALCI, 1994; SARGENT e BALCI, 2017).

Quando se comete erro (i), o custo do modelo começa a crescer, uma vez que o modelo pode ser totalmente desperdiçado (se nunca for utilizado) ou aumentado de forma desnecessária (se o modelador continuar a construí-lo). Para o erro (ii), ações desastrosas podem acontecer, pois tomadas de decisões baseadas no modelo inválido podem levar ao colapso do modelo real (BALCI e SARGENT, 1981; BALCI, 1994). Assim, o erro do tipo II é considerado o mais grave dentro de um projeto de simulação, pois as tomadas de decisões serão todas baseadas em um modelo que não representa a realidade. Balci e Sargent (1984) e Kleijnen (1995) garantem que é possível diminuir a probabilidade α e β aumentando o número de amostras tanto do modelo real como simulado. Apesar disso, os autores afirmam que há um custo associado a esse aumento, além de ser mais fácil o aumento da amostra no modelo simulado. Por fim, Balci e Sargent (1983) sugerem a criação de gráficos mostrando a relação entre o risco do modelo, a precisão aceitável da validação, o tamanho da amostra e os custos envolvidos na coleta de dados.

2.3. Inteligência artificial

A Inteligência Artificial é um campo que surgiu na ciência da computação e visa que máquinas desenvolvam inteligência parecida com a inteligência humana. Assim, as soluções de problemas que são extremamente difíceis para o cérebro humano se tornam simples para os computadores (AVCI *et al.* 2021). Samuel (1988) diz que a IA é um conjunto de técnicas as quais permitem a construção de equipamentos e sistemas capazes de aprender e aperfeiçoar seus resultados sem intervenção direta de um programador. Russel e Norvig (2020) descrevem quatro categorias de definição da IA. A primeira afirma que as máquinas devem agir humanamente, ou seja, o computador deve: (i) processar uma língua natural, (ii) apresentar conhecimento, (iii) ter um raciocínio automatizado e (iv) apresentar o aprendizado de máquina (*machine learning*) com o objetivo de se comunicar, guardar o que foi aprendido, responder questões, gerar novas conclusões, adaptar a novas circunstâncias e extrapolar padrões. Na segunda, a máquina deve pensar humanamente e, para isso, técnicas da ciência cognitiva são incorporadas no computador. A terceira categoria se refere à máquina pensar racionalmente e por fim, na quarta, ela deve agir racionalmente.

Russel e Norvig (2020) afirmam que houve um crescimento no campo de IA nos últimos anos devido aos avanços computacionais. Além disso, com o surgimento da Indústria 4.0, a IA foi incorporada às fábricas e processos inteligentes, sendo combinada com novas tecnologias como a Internet das Coisas (IoT), computação em nuvens, sistemas cyber-físicos, entre outras (LEE *et al*., 2018). Devido ao vasto campo da IA, Avci *et al.* (2021) mostram que ela está

dividida entre dois campos: IA baseada no conhecimento (*knowledge-based*) e o aprendizado de máquina. A Figura 2.2 mostra como a IA está dividida.

A IA baseada no conhecimento foi realizada com uma lista de declarações do tipo "*if*" e "*else*" codificadas por um especialista humano. Assim, tarefas antes difíceis de serem realizadas por seres humanos puderam ser feitas por computadores. Apesar disso, esses sistemas falharam ao tentar reconhecer rostos humanos, detectar objetos e entender falas, ações comuns para os seres humanos. Dessa forma, o grande desafio enfrentado pelos novos sistemas de IA foi encontrar alternativas para ensinar, para o computador, conhecimento simples e comum ao ser humano (GHAHRAMANI, 2015; GOODFELLOW, BENGIO e COURVILLE, 2016; AVCI *et al.*, 2021).

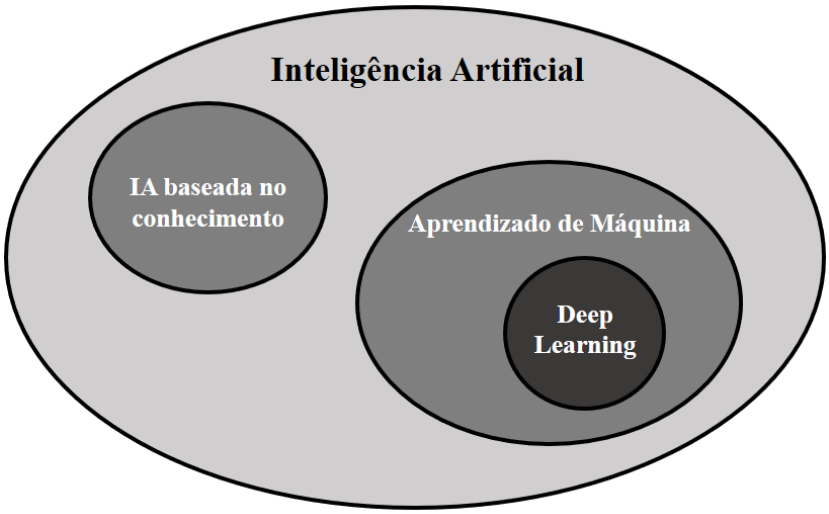

Figura 2.2 - Divisão da Inteligência Artificial
Fonte: Adaptado de Avci *et al.* (2021)

2.3.1. Aprendizagem de Máquina

A Aprendizagem de Máquina (AM) é um subconjunto da IA. Alpayadin (2020) define a AM como o uso de algoritmos que vão se aperfeiçoando com a experiência para prever dados de um futuro próximo. Segundo Angra e Ahuja (2017), AM tem sido os métodos mais eficientes no campo de análise de dados. A sua aplicação está em diversas áreas como engenharia, medicina, estatística, entre outros. Com a AM, é possível gerar e prever dados, identificar objetos em imagens, além de transcrever falas em texto (LECUN, BENGIO e HINTON, 2015).

Raschka, Julian e Hearty (2016) classificam a AM em três diferentes métodos: Aprendizado Supervisionado (AS), Aprendizado não Supervisionado (AnS) e Aprendizado por Reforço (ApR). Jin (2020) afirma que, comparado a outros métodos, o AS é considerado básico, pois os objetivos de aprendizado são estabelecidos pelas pessoas previamente. Assim, o AS tem

como finalidade aprender o relacionamento entre variáveis de entrada e saída através de um conjunto de dados anteriores e atuais previamente rotulados. Dessa forma, os dados rotulados são aqueles que as respostas para as iterações das variáveis já são conhecidas (CHANDRAYAN, 2019; KOTSIANTIS, 2007; DUTTON e CONROY, 1997).

Cui *et al.* (2020) e Saravanan e Sujatha (2018) afirmam que, ao utilizar métodos de AnS, o conjunto de dados de entrada não tem necessidade de apresentar rótulos ou estar classificados *a priori*. Assim, o objetivo é extrair inferências e identificar padrões para determinar clusters e/ou reduzir a dimensionalidade dos dados, como o caso da Análise de Componentes Principais. Jin (2020) conclui que, na prática, o método permite que a máquina aprenda conceitos e conteúdos básicos e tenham liberdade para tirar conclusões a respeito dos dados.

Com os métodos de ApR, há um aprendizado sistemático, no qual dados utilizados anteriormente servem de histórico para determinar os novos. Assim, é desenvolvido um sistema que melhora o seu desempenho com o passar do tempo e com as iterações com o meio ambiente. O modelo aprende e manda uma recompensa ou uma punição para melhorar o aprendizado (RASCHKA, JULIAN e HEARTY, 2016; DUTTON e CONROY, 1997). Dessa forma, os sistemas identificam um comportamento ideal e maximizam o seu desempenho (JIN, 2020).

De acordo com os três tipos de métodos que foram apresentados, ML pode ser utilizado para as seguintes análises de dados (AVCI, 2021; RASCHKA, JULIAN e HEARTY, 2016):

- Classificação: determinar em que categoria as saídas pertencem. A classificação é feita com dados que são considerados discretos;
- Regressão: modelar a relação numérica entre as variáveis de entrada e as variáveis de saída por meio de dados contínuos;
- Previsão: prever dados futuros em um determinado espaço de tempo dado uma série;
- Clusterização: dividir o conjunto de dados em grupos que apresentam características semelhantes de forma que a máquina decide como serão agrupados;
- Redução da dimensionalidade: compactar os dados em um subespaço menor mantendo a maioria das informações relevantes.

2.3.2. *Deep Learning* e Redes Neurais Artificiais

Deep Learning, ou seja, Aprendizado Profundo é um subcampo da AM. Schimidhuber (2015) afirma que o termo foi introduzido a AM nos anos 1980 e em 2000 foi usado para Redes Neurais Artificiais. Assim, o que difere a DL de uma simples RNA é a quantidade de camadas as quais os dados são submetidos. De fato, LeCun *et al.* (2015) diz que os métodos de DL são

compostos por múltiplas camadas que são capazes de aprender recursos de dados com vários níveis de abstração.

As RNAs surgiram para descrever cálculos lógicos, armazenar informações e avaliar parâmetros baseadas nas estruturas celulares do cérebro (MCCULLOCH e PITTS, 1943). Swingler (1996) salienta ainda que as RNAs são algoritmos estatísticos que capturam as entradas, fazem o seu mapeamento e transformam em saídas. Isso significa que as RNAs consistem em um grande número de neurônios que são organizados em camadas. A camada de entrada e saída são mandatórias e, ainda, pode haver diferentes camadas ocultas entre elas (ZHANG, PATUWO e HU, 1998; DOLLING e VARAS, 2002). A Figura 2.3 mostra a estrutura de uma RNA.

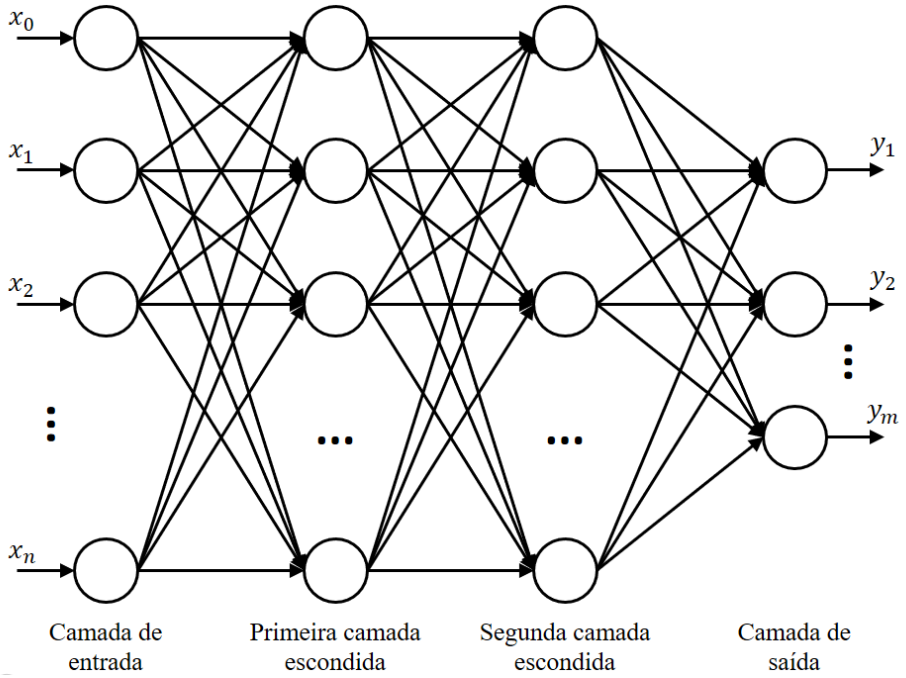

Figura 2.3 - Estrutura de uma RNA

Fonte: Adaptado de Zhang, Patuwo e Hu (1998)

Somers e Casal (2009) afirmam que a primeira camada consiste nas variáveis preditoras ou independentes. A última camada é saída dos dados e representa a variável dependente. Assim, as RNAs são treinadas nas camadas intermediárias para encontrar padrões que são mapeados e levam aos dados de saída. Nas camadas intermediárias, os dados são somados, ponderados e convertidos em funções estatísticas. A função de saída geralmente é uma transferência linear e, a partir do momento que os dados vão sendo processados, os pesos dos coeficientes são ajustados e o erro de previsão reduzido. Os autores ainda concluem que se a transferência estatística for não linear, como por exemplo, uma função sigmoide, as RNAs apresentam padrões complexos de não linearidade.

Zhang, Patuwo e Hu (1998) descrevem que as redes neurais apresentam algumas vantagens em relação à métodos tradicionais de análise, previsão e classificação de dados. A RNA consegue se adaptar aos dados de entrada e modelá-los independentemente das relações lineares ou não lineares, assim, elimina a necessidade de se seguir pressupostos estatísticos. Devido a esse comportamento, a RNA consegue prever e gerar dados futuros mesmo se as informações apresentarem ruídos. Ainda afirmam que elas apresentam formas funcionais mais gerais e flexíveis que os tradicionais métodos estatísticos. Por fim, elas não se baseiam somente em uma fórmula ou equações predefinidas, mas na capacidade de capturar informações ao longo do seu treinamento, fornecendo respostas corretas quando um novo conjunto de dado é testado (OÑA e GARRIDO, 2014).

Apesar da fácil adaptação dos dados a uma regressão, a RNA apresenta algumas desvantagens. Santos *et al.* (2019) afirmam que modelos de RNAs podem ser complexos e assim, se torna difícil determinar os verdadeiros impactos das variáveis de saída. Isso se dá porque quando criadas, os pesos iniciais são gerados de forma aleatória e o treinamento acontece a partir desse ponto, fazendo com que não haja um explicito relacionamento entre essas variáveis (OÑA e GARRIDO, 2014). Além disso, diferentes métodos de RNAs podem gerar modelos que dão importância relativa diferente aos resultados (OLDEN, JOY e DEATH, 2004). Ainda, Hawkins (2004) conclui que se o modelo apresentar *overtraining*, ou seja, ser treinado ao extremo, tem baixo desempenho de previsão.

Minar e Naher (2018) conduziram um estudo onde definem os campos onde as RNAs, mais especificamente a DL, podem ser aplicadas. Eles encontram trabalhos em cerca de 45 campos. Pode-se citar aplicações como classificação de texto, dados e imagens; reconhecimento de falas, imagens, ações humanas, entre outras. Um dos métodos de DL citado pelos autores e com resultados efetivos são as *Generative Adversarial Networks*.

2.4. *Deep Learning* e Simulação

Para Ferreira *et al.* (2020), a integração entre a IA e Simulação é uma tendência que marca a nova era na tomada de decisões. Destaca-se o desenvolvimento de modelos cada vez mais inteligentes, integrados e capazes de fornecer decisões eficientes (RODIČ, 2017; SANTOS *et al.*, 2021). A grande área da IA pode ser entendida como um conjunto de técnicas que reproduzem o comportamento humano por meio de recursos computacionais. Nesse contexto, as técnicas de DL podem ser utilizadas com diversos algoritmos, como redes neurais artificiais, modelos *Fuzzy*, aprendizagem por reforço e meta-heurísticas (FERREIRA *et al.*, 2020).

As técnicas de DL são baseadas na autoaprendizagem do modelo por meio de uma série de camadas de treinamento, uma característica que garante a melhor precisão dos modelos (CORTES *et al.*, 2020). Nesse sentido, nota-se uma crescente adoção de DL em projetos de SED estimulada pelo desenvolvimento de novas tecnologias (DE LA FLUENTE *et al.*, 2018), que podem ser encontradas na construção civil (KARIN e KIM, 2020), manufatura (WU *et al.*, 2020), serviços (DE LA FLUENTE *et al*., 2018), tecnologia da informação (NASCIMENTO *et al.*, 2019), entre outros.

Destaca-se diferentes abordagens considerando a integração entre a IA e a SED. A literatura apresenta trabalhos que exploram o uso de IA para resolver problemas complexos e que utilizam SED como técnica auxiliar para validar as soluções propostas. Nascimento *et al.* (2019) avaliaram, através da SED, métodos de aprendizado por reforço profundo para redes centralizadas de rádios. Resultados promissores foram encontrados com nível de saídas satisfatórios em todos os cenários simulados. Wang *et al*. (2021) desenvolveram e implementaram uma análise de séries temporais e lógica *Fuzzy* para otimizar o deslocamento de pessoas em elevadores de altos prédios. A SED foi utilizada para validar o modelo proposto pelas duas técnicas. Outro estudo, desenvolvido por Karim, Dagli e Qin (2020), utiliza da simulação para validar um método proposto através de uma plataforma robótica remotamente controlada para avaliar inspeção em pontes. A simulação mostrou que a inspeção e manutenção das pontes através do modelo proposto pode reduzir cerca de 80% de horas/homem trabalhadas. Por fim, Wu *et al.* (2020) utilizaram redes neurais e cadeias de *Markov* para analisar um processo em que as tarefas precisam ser feitas várias vezes a fim de minimizar o tempo de ciclo e maximizar o lucro. A SED foi utilizada como ferramenta para a validação do modelo construído com as redes neurais.

Além disso, existem estudos em que o DL é utilizado como técnica auxiliar com o objetivo de promover insumos mais precisos ou mesmo otimizar os experimentos de modelos de SED. De la Fluente, Erazo e Smith (2018) demonstram a integração da SED com um recurso de aprendizado profundo, conhecido como *TensorFlow*, para a tomada de decisão na forma de processos inteligentes. Um processo de aprovação de crédito bancário foi modelado utilizando ML para lidar com mudanças imprevistas que ocorrem no processo de geração de dados simulados no *software* de simulação. Shi *et al.* (2020) utilizam o aprendizado por reforço para programar uma linha de produção automatizada em conjunto com a SED para evitar que os recursos sejam extraídos manualmente e para superar a falta de conjuntos de dados estruturados. Ainda, Cortes *et al.* (2020) implementaram um sistema de tomada de decisão que, com base no ambiente de simulação (software, hardware e lógica de simulação) e DL, identifica a melhor

sincronização e gerenciamento de tempo para realizar um projeto de SED distribuído. Os autores afirmam que a abordagem economiza tempo em experimentação e fornecer melhores experimentos.

Apesar desses estudos, nota-se que há oportunidades na literatura tanto em relação ao uso de outros algoritmos de DL quanto a outros objetivos relacionados à integração de DL e SED e testes estatísticos. Portanto, pode utilizar à validação de modelo de SED com técnicas de DL. Além disso, destacam-se as GANs como uma valiosa técnica de DL com grande potencial para superar os problemas encontrados durante a validação de modelos computacionais. Isso é especialmente válido considerando o contexto da indústria moderna, marcada pela ampla disponibilidade de dados, dado o acelerado desenvolvimento tecnológico, que permite a utilização de técnicas avançadas para melhorar a tomada de decisão (FERREIRA *et al.*, 2020; Santos *et al.*, 2021).

2.5. Generative Adversarial Networks (GANs)

As Redes Adversárias Generativas, que vem do inglês *Generative Adversarial Networks* (GANs) foram propostas por Goodfellow *et al.* (2014) e são caracterizadas como modelos de aprendizagem profunda. Elas são estudadas e utilizadas para gerar imagens falsas, também chamadas sintéticas, mas, significativamente realistas que tem impactado positivamente a área de visão computacional. Atualmente, as GANs recebem atenção por serem empregadas para a geração de manipulação de imagens, vídeos e sons, com o objetivo de gerar mídias falsas (SORIN *et al.*, 2020). Karras *et al.* (2019) mostram em seu estudo que é possível encontrar páginas nsa quais se criam fotos aleatórias de pessoas que na verdade não existem, porém são geradas a partir de GANs.

Assim, segundo Pan *et al.* (2019), as GANs foram criadas a partir da teoria dos jogos, onde duas redes neurais artificiais competem entre si, sendo uma gerando dados, enquanto a outra é considerada discriminadora. A primeira, ou seja, a geradora, objetiva gerar dados e amostras falsas para enganar a rede discriminadora. A segunda, por sua vez, tem o papel de diferenciar se os dados gerados são falsos ou verdadeiros. Com o passar do tempo, o aprendizado e o treinamento das GANs, as redes geradoras começam a produzir amostras que são similares as amostras reais. Portanto, essa estrutura permite gerar novas amostras sintéticas, mas que se enquadram na população verdadeira da qual foi gerada, caracterizando um método AnS (BROWNLEE, 2020). A Figura 2.4 mostra como é o funcionamento das GANs.

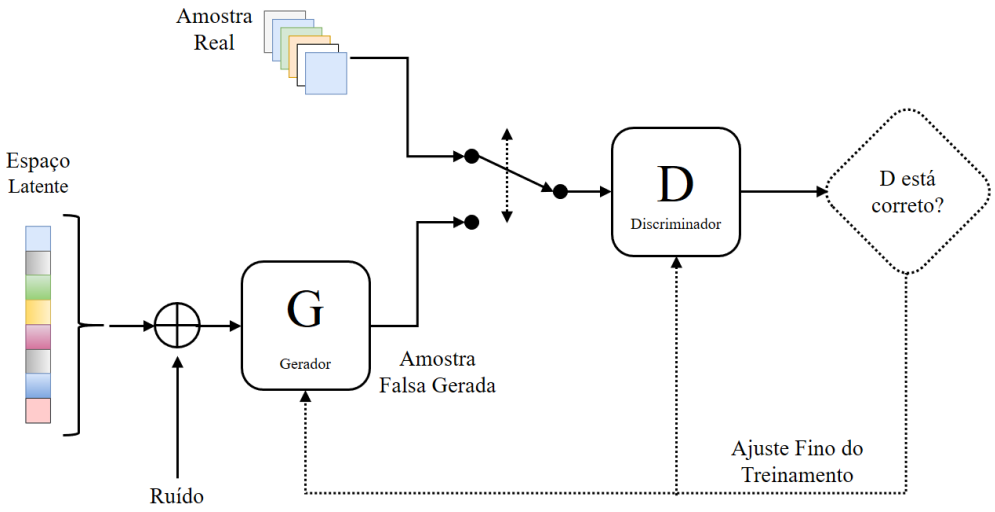

Figura 2.4 - Funcionamento das GANs

Fonte: Adaptado de Pan *et al.* (2019)

Como ilustrado na Figura 2.4, o Gerador *G* gera dados falsos, que se aproximam o tanto quanto possível da distribuição dos dados reais, enquanto o Discriminador $D(x)$ distingue os dados reais dos dados gerados por *G*. Na entrada do Gerador *G*, há um vetor de ruído aleatório Z (geralmente com uma distribuição normal ou uniforme). Esse ruído visa mapear um novo espaço de dados através de *G* para obter uma amostra falsa. Dessa maneira, o vetor *G(z)* é um vetor multidimensional, que carrega informações dos dados gerados baseados na amostra real. O Discriminador *D* recebe amostras reais do conjunto de dados e amostras falsas geradas pelo Gerador *G* e faz uma classificação binária. Assim, a saída de *D* representa a probabilidade de uma amostra ser real (PAN *et al.*, 2019).

Há então uma competição entre as duas redes. Assim, busca-se o equilíbrio entre as duas. Se o Discriminador *D* identifica com sucesso as amostras reais e falsas, ele é recompensado e não sofre nenhuma mudança nos seus parâmetros, enquanto o gerador é penalizado com grandes alterações. Por outro lado, se *G* consegue enganar *D*, ele é recompensado e as alterações devem ser feitas nos parâmetros do Discriminador. Quando atinge o estado ideal, o Gerador *G* consegue gerar réplicas perfeitas e o Discriminador *D* não sabe diferenciá-las. Assim, *D* classifica os dados ao acaso e a sua acurácia gira em torno de 50.0% (BROWNLEE, 2020). Pan *et al.* (2019) concluem que assim o Gerador conseguiu atingir o seu objetivo e aprendeu a distribuição dos dados reais.

Muitos setores vêm utilizando as GANs como meio de geração de dados. Yi, Walia e Babyn (2019) destacam a área médica, utilizando as GANs para sintetizar imagens e aumentar o tamanho das amostras para superar a falta de dados e *overfitting*. Segundo Goodfellow *et al.*

(2014), apesar de receber mais atenção nas áreas médicas e no processamento de vídeos, imagens e sons, as GANs podem ser utilizadas como geração de dados normalmente distribuídos. Os autores defendem que GANs são capazes de aprender distribuições complexas que representam o comportamento de uma dada população e, com isso, gerar novas amostras coerentes com o mesmo comportamento. Assim, a sua utilização é eficiente pois, a partir delas é possível captar comportamentos não lineares e dependência entre as variáveis presentes no conjunto de dados (AGNESE *et al.*, 2020).

2.5.1. Aprendizado das GANs

Goodfellow *et al.* (2014) afirmam que como duas redes adversárias, o Gerador G e o Discriminador D apresentam funções de perda (*loss function*), sendo denominadas de *J(G)* e *J(D)*, respectivamente. O discriminador D é um classificador binário e a função perda é dada pela Equação 1:

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim pdata}\log D(x) - \frac{1}{2}\mathbb{E}_{z}\log\Big(1 - D\big(G(z)\big)\Big) \tag{1}$$

Na Equação 2, *x* representa a amostra real; *z* o vetor aleatório de ruído; *G*(*z*) é o dado gerado pelo gerador e $\mathbb{E}$ é a expectativa dos dados pertencerem ao conjunto. *D*(*x*) indica a probabilidade de *D* discriminar os valores de *x* como dados reais e *D*(*G*(*z*)) é a probabilidade de *D* determinar os dados gerados por *G*. A função perda diz que probabilidades próximas de 1 indicam que os dados são reais, enquanto probabilidade próximas de zero indicam dados falsos. Assim, o objetivo de *D* é determinar corretamente os dados, e então é necessário que *D*(*G*(*z*)) se aproxime de 0, enquanto *G* se aproxime de 1. Brownlee (2020) salienta que dados próximos de 1 indicam melhor desempenho do discriminador. Assim, as duas redes são treinadas ao mesmo tempo e praticam um jogo de *minimax* com o valor da função de *V*(*G*,*D*), conforme mostra a Equação 2:

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{z}(\boldsymbol{z})}\Big[\log\Big(1 - D\big(G(z)\big)\Big)\Big] \tag{2}$$

Ao treinar as redes, elas são capazes de recuperar as distribuições geradoras dos dados se elas se afastam do alvo. Na Figura 2.5, adaptada de Goodfellow *et al.* (2014) mostra-se como os dados se comportam ao longo do treinamento. Ambas são treinadas e atualizadas simultaneamente de acordo com a distribuição discriminativa, ou seja, a função *D*(*x*) (linha azul tracejada). *D*(*x*) distingue entre as amostras da distribuição que gerou os dados (linha pontilhada preta – *p*(*x*)) das amostras geradas pela distribuição generativa (linha sólida verde). A linha horizontal inferior representa o domínio pelo qual z pertence, sendo ele uniforme. As setas com

sentido para cima mostram como os dados são mapeados $x = G(z)$ e são impostos em uma distribuição não uniforme (p_g).

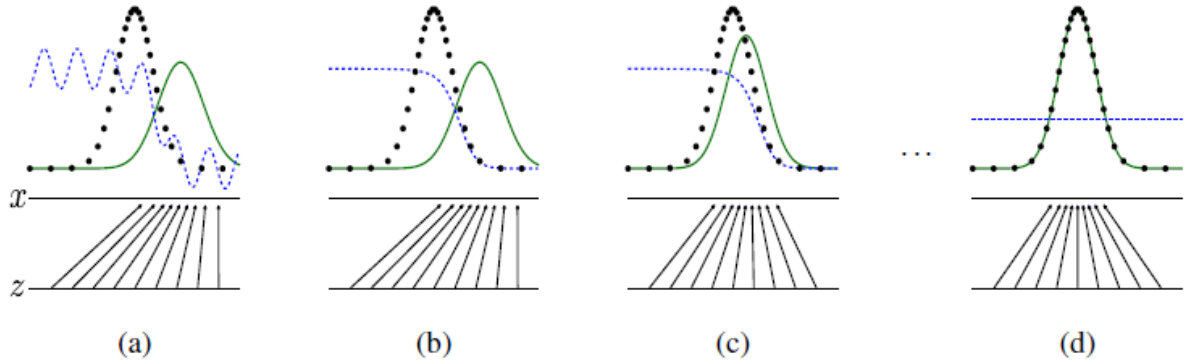

Figura 2.5 - Comportamento dos dados ao longo do treinamento das GANs

Fonte: Goodfellow *et al.* (2014)

A Figura 2.5 (a) mostra duas redes adversárias que estão próximas a convergência, o seja, p_g é semelhante a p_{data} e o Discriminador D ainda é parcialmente preciso. No *loop* interno de treinamento do algoritmo, D é treinado para discriminar as amostras geradas das amostras reais, convergindo $D(x)$ (Figura 2.5 (b)). Assim, há a atualização dos parâmetros do Gerador G, e o gradiente D mostra quais regiões $G(z)$ classifica os dados com maior probabilidade, conforme mostra a Figura 2.5 (c). Se G e D apresentarem capacidade suficiente de aprendizado após rodadas de treinamento, também chamada de épocas, eles atingirão um ponto em que ambos não poderão melhorar uma vez que $p_g = p_{data}$ (Figura 2.5 (d)). Dessa forma, o discriminador não diferencia as duas distribuições, ou seja, $D(x) = 0.5$.

2.5.2. Avaliação das GANs

A literatura apresenta algumas maneiras de avaliar o comportamento das GANs, porém não existe um consenso em quais desses métodos são melhores e quais são os piores (PAN *et al.* 2019). Apesar disso, Brownlee (2020) e Borji (2019) classificam essas avaliações em duas grandes categorias: avaliações qualitativas e avaliações quantitativas.

As avaliações qualitativas geralmente estão baseadas nas percepções humanas, ou seja, as imagens geradas pelas GANs são avaliadas de forma subjetivas. Além disso, pode-se considerar enviesada, pois depende da percepção de cada um, além de apresentar uma alta variabilidade, o que faz necessário que mais pessoas avaliem uma mesma imagem (SALIMANS, *et al.* 2016; BORJI, 2019). Esse tipo de avaliação é importante quando se utiliza geração de dados que são de fácil percepção das pessoas, como imagens, textos e até mesmo sons. Como o objetivo do trabalho é a geração de dados que são pertencentes a um determinado

objeto e não dados perceptíveis, fica mais difícil de analisar através de medidas qualitativas, embora dados até três dimensões possam ser analisados visualmente.

As avaliações quantitativas são mais robustas por apresentarem um cálculo de um valor específico utilizado para resumir a qualidade dos dados gerados (BROWNLEE, 2020). Borji (2019) e Brownlee (2020) citam mais de 20 técnicas encontradas. Apesar de inúmeras técnicas, elas visam avaliar se o gerador está gerando dados compatíveis com a amostra real. Para o presente trabalho, além do gerador gerar dados semelhantes aos dados reais, é importante que o discriminador também seja avaliado, pois é a partir dele que os dados do modelo de simulação poderão ser testados.

Assim, para medir a acurácia do Gerador *G*, pode-se utilizar o Teste de Classificação de Duas Amostras – *Classifier Two-Sample Test* (C2ST) baseado na técnica do *k-NN* (DAVID e OQUAB, 2017, CAI *et al.* 2019). O *k-NN* classifica os dados baseado na observação dos *k* vizinhos mais próximos (COVER e HART, 1967) e tenta separar o conjunto dos dados reais dos conjuntos dos dados gerados pelas GANs. Se os dados gerados são tão precisos e perfeitamente iguais aos dados reais, o algoritmo *k-NN* classifica cada observação de forma aleatória e a precisão dos dados gerados pelas GANs (A_c) é de 50.0%. Por outro lado, se os dados sintéticos não forem realistas, o classificador pode separar facilmente as observações, e a (A_c) esperado chega a 100.0%. Dessa maneira, o *k-NN* classificar os dados em 50.0% significa dizer que os dados sintéticos gerados pelas GANs são iguais aos dados inseridos. Por outro lado, se a classificação for 100.0%, significa que os dados gerados pelas GANs não refletem em nada (0.0%) aos dados inseridos para o seu treinamento. Para o trabalho, a classificação do *k-NN* deve estar entre 50.0% e 52.5%, o que significa que os dados sintéticos representam, no mínimo 95.0% dos dados inseridos.

Por outro lado, para medir a acurácia do discriminador *D*, pode-se utilizar a capacidade de classificação dos dados. Segundo Goodfellow *et al.* (2014) e Pan *et al*. (2019), quando amostras são perfeitas, o discriminador não sabe identificá-las e essa probabilidade é ao acaso, tendo 50.0% de chances de os dados serem verdadeiros ou falsos. Entretanto, como existe um jogo duplo das redes, nem sempre é necessário gerar dados que conseguem chegar a esse nível precisão (BRONWLEE, 2020). Apesar disso, para o propósito do presente trabalho, é necessário que essa acurácia gire em torno de 45.0% a 55.0% e estejam estabilizados, uma vez que os dados testados, devem se parecer com o real para que o modelo de simulação seja validado.

2.5.3. Artifícios para gerar GANs eficientes

Alguns parâmetros devem ser levados em consideração ao construir uma GAN. Primeiramente é necessário definir o espaço latente do modelo. Isso significa a quantidade de inputs que serão fornecidos para as redes. Além disso, é preciso determinar a quantidade de camadas escondidas (*layers*), juntamente com quantos neurônios (*Density*) presentes em cada uma. Outros parâmetros importantes são o tipo de algoritmo a ser utilizado para a otimização, podendo ser o chamado *Adam* ou o *RSMprop*, a taxa de aprendizado (α) de cada uma das redes, o decaimento do aprendizado (*Decay rate*), a função de ativação e o uso de uma suavização ao classificar os dados (*label smoothing*). Assim, com esses parâmetros, nem sempre é fácil atingir a performance de treinamento excelente (PAN *et al.* 2019).

Para resolver esses problemas, a literatura sugere alguns artifícios que são eficientes e faz com que as GANs sejam bem treinadas sem que haja colapsos ou não estabilidade nas funções de perda. Primeiramente, mais camadas escondidas permitem que as redes apresentem melhores acurácias em relação aos dados gerados. Também, a quantidade de neurônios influencia na qualidade da saída dos dados. Recomenda-se utilizar mais camadas e neurônios para o Discriminador *D*, uma vez que ele necessita de um aprendizado melhor para classificar os dados, no caso de uma validação de dados externos.

Além disso, quando o gerador colapsa, significa que o gradiente do Discriminador está apontando muitos pontos semelhantes para uma mesma direção. Assim, o gerador começa a gerar dados iguais e o discriminador acha que as suas definições estão altamente realistas. Para resolver esse problema, Salimans *et al.* (2016) propõem a utilização de discriminação em pequenos lotes. Portanto, o discriminador recebe várias amostras em pequenos lotes combinados ao invés de uma amostra isolada.

Salimans *et al.* (2016) também sugerem a suavização ao classificar os dados. Os autores afirmam que a classificação é feita pela função de ativação de saída na qual o valor 0 é um dado falso e o valor 1 um dado real. Propõe-se assim, a utilização de valores como 0.1 para os dados falsos e 0.9 para os dados reais. Dessa forma, o discriminador não é incentivado a escolher uma classe incorreta, mas reduz a confiança ao classificar os dados (GOODFELLOW, 2016). Denton *et al.* (2015) utilizaram o artifício pela primeira vez e conseguiu amostras melhores e de forma mais eficiente. Apesar disso, Goodfellow (2016) conclui que ainda não está claro o porquê deste artifício funcionar e elevar a qualidade dos dados produzidos.

Heusel *et al.* (2017) propõem o uso de diferentes taxas de aprendizados entre o discriminador e o gerador para garantir o equilíbrio entre os dois. O uso dessas duas diferentes

taxas recebeu o nome de *Two Time-scale Update Rule* (TTUR), ou seja, regra de atualização em duas escalas de tempo. Sugere-se o uso de taxas maiores para o discriminador, uma vez que ele deve aprender a discriminar os dados mais rápido do que o gerador a gerar novas amostras. Juntamente com as taxas de aprendizado, é necessária uma taxa de decaimento para que as GANs se estabilizem mais rápido.

As funções de ativação são equações matemáticas que determinam a saída de uma rede neural. A função é anexada a cada um dos neurônios e determina se ele deve ser ativado ou não (HAYKIN *et al.*, 1998). São encontrados alguns tipos de função de ativação na literatura, como identidade, sigmoide, tangente hiperbólica, unidade linear unificada (ReLU), unidade linear unificada *Leaky* (*Leaky* ReLU), entre outras. Apesar disso, Radford, Metz e Chintala (2016) afirmam que a função de ativação ReLU é recomendada para o gerador, enquanto a função *Leaky* ReLU é preferida para o discriminador. Os autores ainda sugerem o escalonamento dos dados de entrada para o gerador em uma escala de -1 a 1 utilizando uma função tangente hiperbólica.

Por fim, para a escolha do algoritmo de otimização da função, a literatura mostra que o algoritmo de Adam é mais eficiente na estabilização e geração de dados que o uso do RSMprop (READFORD, METZ e CHINTALA, 2016). Srivastava *et al.* (2018) realizaram um estudo no qual compara diferentes métodos de otimização das GANs e também concluíram que a utilização do Adam resulta em melhores treinamentos. Da mesma forma, Heusel *et al.* (2017) também sugerem o uso do Adam para a otimização. Para o presente trabalho, os parâmetros adotados podem são explicados na seção 4.1.2.

2.6. Conditional Generative Adversarial Networks (cGANs)

As Redes Adversárias Generativas Condicionais, ou seja, as *Conditional Generative Adversarial Networks* (cGANs), podem ser considerada uma extensão melhorada das GANs proposta por Mirza e Osindero (2014). Uma vez que as cGANs usam uma abordagem supervisionada e fornecem resultados mais controláveis (Lan *et al.* 2020), ela é uma ferramenta promissora para gerar imagens a partir de textos, imagens a partir de outra imagem e realizar a tradução de figuras (MIYATO e KOYAMA, 2018). Além disso, Gauthier (2015) afirma que elas são eficientes quando é necessário gerar imagens a partir da descrição falada ou escrita. Então, nas cGANs, as imagens são fornecidas como entrada para as redes e espera-se como saída uma imagem correspondente gerada condicionalmente (ISOLA *et al.*, 2018).

Segundo Mirza e Osindero (2014), nas cGANs, o Discriminador e o Gerador apresentam uma informação auxiliar. Essa informação é necessária para classificar alguns dados, pois eles

podem apresentar rótulos e assim gerar dados baseado em determinada condição. Mirza e Osindero (2014) e Douzas e Bacao (2018) dizem que a cGAN recebe um ruído aleatório *z* e um atributo categórico *c*. O Gerador é modificado recebendo um espaço adicional enquanto o Discriminador é alterado. Em seguida, a equação original das GANs é adaptada e a função objetivo é, ao mesmo tempo, maximizar a perda do Discriminador e minimizar a perda do Gerador, conforme a Equação 3:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data(x)}}[logD(x|y)] + \mathbb{E}_{z\sim p_{z(z),y\sim p(y)}}[\log(1 - D(G(z,y),y))] \quad (3)$$

Douzas e Bacao (2018) afirmam que o treinamento das cGANs é igual ao treinamento das GANs. No entanto, nas cGANs, o Discriminador distingue a amostra gerada pelo Gerador onde a distribuição alvo da amostra gerada é *x* e a variável condicional é representada por *y* (MIYATO e KOYAMA, 2018). Após o treinamento, o Gerador cria dados sintéticos a partir de classes predefinidas pelos dados originais (DOUZAS e BACAO, 2018).

Em relação a validação de modelos computacionais, as cGANs podem ser utilizadas para representar diferentes categorias de paciente, peças, recursos (médicos, enfermeiras, operadores) e representam diferentes características de uma entidade. Nesse sentido, é possível realizar a validação condicional, porém utilizando apenas um treinamento único.

2.7. Testes Estatísticos

As GANs têm o objetivo de suprir a necessidade dos dados terem que seguir um pressuposto estatístico uma vez que elas conseguem se adaptar a qualquer formato de dados (BROWNLEE, 2020). Utiliza-se, portanto, os dados do sistema real ou do sistema simulado para treinar o gerador G e gerar dados sintéticos com a finalidade de enganar o discriminador D. Após o treinamento correto, os dados não utilizados para o treinamento podem ser julgados pelo discriminador. O teste de classificação dos dados é utilizado para essa etapa.

Ainda, é possível determinar uma tolerância pela qual os modelos podem ser considerados válidos. Sargent (2015) conclui que a tolerância é importante, pois os dados podem variar tanto para cima quanto para baixo e, mesmo assim, o modelo continuar válido dentro de um limite de aceitação definido previamente. Assim, pode-se aplicar o Teste de Equivalência para a Diferença de Duas Proporções.

2.7.1. Teste de classificação dos dados

Lopez-Paz e Oquab (2017) dizem que um teste de classificação de dados denominado *Classifier Two-sample Tests* (C2ST) tem como objetivo testar se duas amostras P e Q são

classificadas de forma iguais ou diferentes. Dessa maneira, ao se classificar um conjunto de dados, os autores afirmam que, utilizando o teorema central do limite, a distribuição de classificação segue uma distribuição normal, conforme mostra a Equação 4:

$$f(x) \sim \mathcal{N}\left(\bar{p}, \frac{\bar{p}(1-\bar{p})}{n_{te}}\right) \quad (4)$$

Sendo:

- f(x) é a distribuição de classificação dos dados
- $\bar{p}$ é a média de classificação dos doados
- n_{te} é o número de amostras

Assim, a GAN é treinada com valores reais até se estabilizar. Como discutido na seção 2.5.2, uma GAN efetiva é aquela que consegue enganar o discriminador e quando dados sintéticos são testados por ela, a probabilidade é dada ao acaso, ou seja, 50.0% (BRONWLEE, 2020). Dessa maneira, o ideal é que a distribuição $f(x) \sim \mathcal{N}\left(\frac{1}{2}, \frac{1}{4n_{te}}\right)$, ou seja, $\bar{p} = \frac{1}{2}$. Apesar disso, para o trabalho, será considerado GANs eficientes as quais o discriminador avalie os dados entre 45.0% e 55.0%.

2.7.2. Teste de Equivalência

Os testes de hipóteses são desenvolvidos para fornecer suporte para a hipótese nula (LAKENS, 2017). Meyners (2012) afirma que um teste de hipótese é uma regra de decisão para verificar se a hipótese nula (H_0) é justificável dado um conjunto de dados. Assim, não é possível rejeitar H_0 a menos que haja fortes evidências, ou seja, apenas se a ocorrência dos dados for menor que um nível de significância escolhido. Geralmente, o nível de significância (α) corresponde a 5%.

O teste estatístico de hipótese dá suporte para rejeitar ou não H_0. Caso H_0 seja rejeitada, subtende-se que o seu complemento, ou seja, a hipótese alternativa (H_1) é verdadeira. Por outro lado, se H_0 não for rejeitada, não significa necessariamente que H_1 é falsa. Não rejeitar H_0 pode estar ligado ao fato de que não há diferença significativa entre os dados medidos ou o experimento é insuficiente para captar essa diferença. Resumindo, só é possível provar H_1 rejeitando H_0, mas nunca se pode provar H_0 (MEYNERS, 2012).

Montgomery e Runger (2018) asseguram que nos testes de hipóteses para comparação entre duas proporções populacionais, a hipótese nula (H_0) assume que a diferença entre a proporção de duas populações (p_1 e p_2) são iguais, enquanto a hipótese alternativa (H_1) diz que a diferença entre a proporção de duas populações é diferente (Equação 5), ou seja:

$$H_0: p_1 - p_2 = 0$$
$$H_1: p_1 - p_2 \neq 0 \quad (5)$$

Lakens (2017) conclui que é impossível estatisticamente que a diferença entre as proporções seja exatamente zero. Dessa forma, surgiu-se o teste de equivalência no qual mede se a diferença entre as proporções de duas populações está entre um intervalo de equivalência, ou seja, entre um limite de tolerância. A abordagem utilizada é o "*Two one-sided test*" (TOST), ou seja, dois testes unilaterais criado por Schuirmann (1987).

O intervalo de equivalência é dado por δ, no qual há um valor inferior ($-\delta$) e um valor superior (δ). Assim, a H_0 no teste corresponde a duas hipóteses, onde a diferença entre as proporções é maior que $-\delta$ e a diferença entre as proporções é menor que δ. Por outro lado, a H_1 diz que a diferença entre as proporções está dentro do intervalo de $-\delta$ e δ, conforme mostra a Equação 6 (SCHUIRMANN, 1981):

$$H_{01}: p_1 - p_2 < -\delta \; ou \; H_{02}: p_1 - p_2 > \delta$$
$$H_1: -\delta \leq p_1 - p_2 \leq \delta \quad (6)$$

2.7.2.1. Determinação dos parâmetros do teste

Dado o teste de classificação realizado pelas GANs, tem-se que a função que representa a classificação dos dados de teste é dada pela Equação 7, enquanto a função de classificação dos dados de referência é dada pela Equação 8:

$$p_1 = p_t \quad (7)$$

$$p_2 = p_r \quad (8)$$

O teste de hipótese avalia a diferença entre os valores dos dados que serão testados e os valores de referência. Dessa forma, a diferença (D) entre as médias (p_1 - p_2) é representado na Equação 9:

$$D = p_1 - p_2 \quad (9)$$

Como explicado na seção anterior, a função de classificação dos dados pode ser aproximada pela Equação 4. Assim, o desvio padrão dos dados testados é calculado de acordo com a Equação 10, enquanto o desvio padrão dos dados de referência segue a Equação 11:

$$s_1 = \frac{p_1(1 - p_1)}{n_1} \quad (10)$$

$$s_2 = \frac{p_2(1-p_2)}{n_2} \quad (11)$$

Onde n_1 representa o número de dados na amostra de testes e n_2 o número de dados presente na amostra de referência.

2.7.2.2. Limites de Equivalência

O limite de equivalência (δ) representa o intervalo no qual as diferenças das proporções dos dados são aceitáveis. Fazendo um paralelo aos modelos de validação, o limite de equivalência corresponde a tolerância na qual o modelo de SED pode ser considerado válido. Geralmente δ é escolhido de forma simétrica ou em porcentagens em relação à média das amostras. Porém, se necessário, a escolha pode ser feita de forma assimétrica (MEYNERS, 2012). Para o presente trabalho, as tolerâncias serão consideradas simétricas e devem estar entre 0 e 1, representando um intervalo entre 0.0% e 100.0%.

2.7.2.3. Intervalo de Confiança

O Intervalo de Confiança para um Teste de Equivalência é denotado como (1 - α)*100%. Meyners (2012) afirma que H_0 somente é rejeitada se a intersecção dos dois intervalos de confiança unilaterais cair completamente dentro do intervalo fechado do limite de equivalência [δ_{inf}, δ_{sup}]. Para se obter o Intervalo de Confiança do teste (limite inferior e limite superior), utiliza-se as Equações 12 e 13:

$$L_{inf} = p_1 - p_2 - z_{(1-\alpha)} * \sqrt{s_1 + s_2} \quad (12)$$

$$L_{sup} = p_1 - p_2 + z_{(1-\alpha)} * \sqrt{s_1 + s_2} \quad (13)$$

Nas Equações 12 e 13, o valor de $z_{(1-\alpha)}$ é calculado pela estatística de teste da distribuição normal.

2.7.2.4. Estatística de Teste

Segundo Laster, Johnson e Kotler (2006), o cálculo da estatística de teste para esse teste é semelhante ao teste convencional de proporções. A única diferença é que se deve considerar a tolerância em seu cálculo, conforme mostram as Equações 14 e 15:

$$z_1 = \frac{p_1 - p_2 - \delta}{\sqrt{s_1 + s_2}} \quad (14)$$

$$z_1 = \frac{p_1 - p_2 + \delta}{\sqrt{s_1 + s_2}} \quad (15)$$

2.7.2.5. *P-value*

O cálculo do *p-value* é realizado para os dois testes unilaterais. H_0 é rejeitada se o *p-value* máximo entre os dois testes for maior que o nível de significância (α). Portanto, se o *p-value* é menor que α, conclui-se que as proporções entre as duas amostras são equivalentes (LEWIS, 1999; WALKER e NOWACKI, 2011). O cálculo do *p-value* está descrito na Equação 16:

$$\begin{matrix} H_0: \Delta \leq \delta_{inf} & P_{H_0(z \geq z_1)} \\ H_0: \Delta \geq \delta_{sup} & P_{H_0(z \leq z_2)} \end{matrix} \tag{16}$$

2.7.2.6. Poder do Teste

Montgomery e Runger (2018) dizem que o poder do teste estima a probabilidade de rejeitar H_0 se H_0 for realmente falsa. Portanto, no Teste de Equivalência o poder é a probabilidade de concluir que a diferença da população está dentro do limite de equivalência quando ela realmente está. Walker e Nowacki (2011) e Meyners (2012) dizem que o poder do teste está ligado diretamente ao tamanho da amostra, à diferença escolhida no intervalo de equivalência, ao desvio padrão e ao valor de α. Assim, amostras maiores, diferenças próximas do centro dos dois limites de equivalência e desvio padrão pequeno dão mais poder ao teste. Em relação à α, valores mais altos dão mais poder ao teste. Entretanto, aumentar o valor de α significa aumentar a probabilidade do Erro do Tipo I, afirmando a equivalência quando não existe. O poder do teste é dado pela Equação 17 (CHOW *et al.*, 2018):

$$Poder = 2\Phi\left(\frac{\delta - |p_1 - p_2|}{\sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}} - z_\alpha\right) - 1 \tag{17}$$

2.8. Considerações Finais

Após a apresentação dos conceitos referentes a SED, validação de modelos computacionais, IA, GANs e testes estatísticos, é possível observar que as GANs são uma ferramenta poderosa que pode ser integrada na SED com o objetivo de validar modelos computacionais. Devido à adaptabilidade das GANs a qualquer tipo de dados, cabe ao trabalho desenvolver e testar uma metodologia de validação para modelos considerando uma tolerância nos dados, devido à adaptabilidade dos dados para qualquer tipo de distribuição.

3. MÉTODO DE PESQUISA

Este capítulo tem como objetivo a apresentação do método de pesquisa utilizado na tese. Ele apresenta os conceitos para o entendimento do Experimento e o passo a passo para a realização de uma pesquisa envolvendo o método.

3.1. Classificação da Pesquisa

A pesquisa desenvolvida é classificada de natureza aplicada, explicativa em relação aos seus objetivos, apresenta uma abordagem quantitativa e utiliza o experimento como método (MIGUEL *et al.*, 2014). A Figura 3.1 mostra a classificação da presente pesquisa.

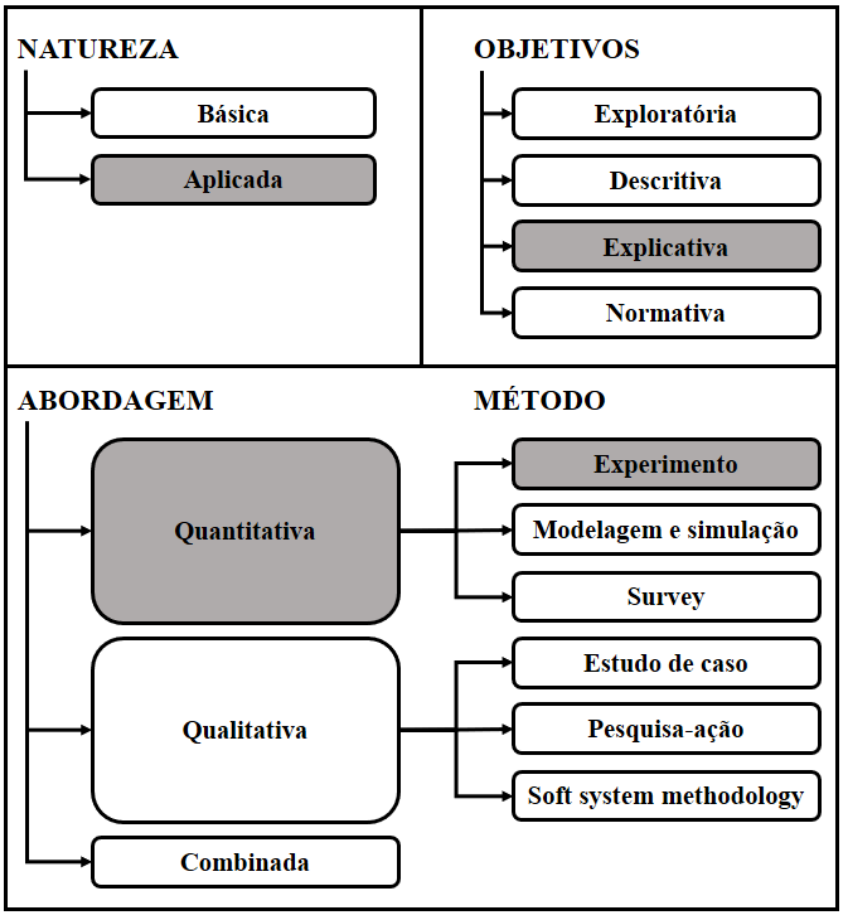

Figura 3.1 - Classificação da pesquisa

Fonte: Adaptado de Miguel *et al.* (2014)

Appolinário (2006) diz que a pesquisa é considerada básica quando ela está ligada ao conhecimento científico e não tem o objetivo comercial. Já a pesquisa de caráter prático (foco do presente trabalho) tem como finalidade desenvolver novos produtos ou processos e aplicar os resultados encontrados. Uma pesquisa explicativa visa ampliar generalizações, definir leis mais amplas, relacionar ou gerar novas hipóteses e estruturar ou definir modelos teóricos

(JUNG, 2004). Além disso, tem como finalidade identificar os fatores que determinam os fenômenos e explica o porquê dos problemas. Geralmente, quando utilizada nas ciências naturais, requer o uso de experimentação, enquanto nas ciências sociais requer o método observacional (GIL, 2008). A presente pesquisa visa desenvolver um método de validação de modelos de SED utilizando uma técnica de IA (GANs) para substituir testes estatísticos que precisam seguir determinados pressupostos.

Para a classificação da pesquisa em relação à abordagem, ela é definida como quantitativa, pois traduz números, opiniões e informações a fim de analisá-los (MIGUEL *et al.*, 2014). Denzin e Lincon (2017) asseguram que a pesquisa quantitativa consegue medir as opiniões, reações, hábitos e atitudes de um sistema por meio de uma mostra que o represente estatisticamente. Assim, deve-se obedecer a um plano pré-estabelecido, com a finalidade de enumerar ou medir os eventos e utilizar a teoria para desenvolver as hipóteses e as variáveis de pesquisa. Em relação ao método de pesquisa, será utilizado o experimento para desenvolver as GANs como ferramenta de validação dos dados obtidos através do modelo simulado.

3.2. Experimento

De acordo com Hair Júnior *et al.* (2005), o experimento é um método em que o pesquisador controla uma causa potencial e observa as mudanças correspondentes nos efeitos supostos. Lazar, Feng e Hochheiser (2017) dizem que o experimento busca generalizações por meio de técnicas de coleta que são realizadas durante a experiência. Além disso, os autores afirmam que o pesquisador é um agente ativo que apresenta controle sobre os experimentos.

O experimento pode ser realizado em qualquer lugar, porém, deve apresentar as seguintes propriedades: manipulação, controle e distribuição aleatória. Assim, o pesquisador deve planejar alterações para manipular pelo menos uma característica dos elementos estudados; criar um grupo para introduzir um ou mais controle sobre a experiência e designar de forma aleatória quais elementos participaram dos grupos experimentais (LAZAR, FENG e HOCHHEISER, 2017).

Jung (2004) diz que o método experimental permite que as variáveis independentes sejam manipuladas e seus efeitos em uma ou mais variáveis dependentes sejam medidos. Quanto mais controle o experimentador obtiver a respeito do sistema, maior a validade interna do experimento (KIDDER, 2004).

De acordo com Hair *et al.* (2005), a pesquisa Experimental é caracterizada quanto ao tipo, no qual ela pode ser experimental, quase experimental ou pré-experimental. A pesquisa experimental possui uma atribuição aleatória aos grupos pertencentes aos testes, onde um grupo

pode ser controlado e o outro não. Dessa maneira, é possível diminuir a probabilidade de explicações alternativas para as conclusões aferidas. Isso não acontece nas pesquisas quase ou pré-experimentais. As pesquisas quase experimentais são indicadas para estudos onde não é possível obter o controle sobre a aleatoriedade dos participantes ou quando elas são realizadas no campo. Nas pesquisas pré-experimentais não há comparação entre dois grupos, uma vez que apenas o grupo que recebe a manipulação é estudado (HERNANDEZ *et al.*, 2014).

3.3. Etapas do experimento

O experimento deve conter cinco etapas: Definição do contexto, Planejamento, Execução, Análise e Interpretação e Apresentação. Para Lazar, Feng e Hochheiser (2017), na primeira fase é estabelecido os termos do problema existente, os objetivos e as metas. No planejamento, o experimento é determinado e a instrumentação é preparada. Na terceira fase, o experimento é de fato executado. Na análise e interpretação, os dados são coletados e analisados com suporte de ferramentas estatísticas. Por fim, na última fase, os dados são apresentados em formato de um relatório.

Para o presente trabalho, a fase de planejamento compreende a criação do algoritmo das GANs para o treinamento de dados e o julgamento do conjunto de dados reais e do modelo simulado. Assim, esta tese visa propor um *framework* para a validação de modelos computacionais de SED. O experimento tem como objetivo testar distribuições teóricas e um modelo que representa um hospital de campanha, onde os dados são controlados.

Na segunda fase, ou seja, na experimentação, as distribuições teóricas são definidas e a métrica de saída para mostrar a eficiência do algoritmo é a quantidade de acertos dos testes realizados. Diferentes tamanhos amostrais são testados na execução do experimento. A análise e interpretação dos dados são baseadas nas curvas construídas a partir dos testes e o relatório com os resultados são apresentados. Todo passo a passo e os seus detalhes podem ser encontrados no Capítulo 4, onde o *framework* proposto é apresentado em detalhes e no Capítulo 5, onde o método proposto é validado.

4. FRAMEWORK DE VALIDAÇÃO

Neste Capítulo são apresentados os passos necessários para a validação de modelos de SED utilizando as GANs e o teste de Equivalência. A partir desse ponto, será utilizado o termo cGANs, pois a mesma metodologia pode ser utilizada tanto para dados condicionais quanto para dados independentes de classes. Um *framework* com o passo a passo para realizar a validação é apresentado, juntamente com a explicação de como realizar o treinamento das cGANs e o modo de conduzir o teste considerando as tolerâncias admitidas para o modelo. Além disso, é apresentado uma escala para a validação dos modelos computacionais

4.1. *Framework* para a validação utilizando cGANs

O método proposto tem como objetivo a validação de modelos de SED, comparando os resultados do sistema real com o sistema simulado utilizando as cGANs. Além disso, o método permite que o tomador de decisão estabeleça uma tolerância considerada aceitável para a comparação dos dados. Ainda, se o modelo não for validado dentro da faixa de tolerância escolhida pela equipe de validação, o algoritmo informa qual a menor faixa de tolerância para considerá-lo válido. A Figura 4.1 mostra o passo a passo para a aplicação do método.

Como mostra a Figura 4.1, o método é dividido em duas fases: a "*Fase de Treinamento*", representada pelos símbolos na cor cinza e a "*Fase de Teste*", onde os símbolos estão na cor branca. Uma vez que se quer comparar os dados reais com dados simulados, o usuário deve escolher qual deles (reais ou simulados) será utilizado na etapa de Treinamento. Recomenda-se o uso do conjunto de dados que contém mais dados na fase de treinamento pois eles mostram resultados mais confiáveis (explicado na Seção 5.1.5). Geralmente, é mais fácil gerar dados a partir da simulação e assim, os dados provenientes do modelo de simulação são empregados nessa etapa. A partir desse ponto, os dados utilizados na primeira etapa são chamados de "*Dados de Treinamento*" (DT) e o conjunto de dados utilizados na Fase de Teste, são chamados de "*Dados de Comparação*" (DC).

De acordo com o método, os DT são importados de uma base do Excel®, lidos e preparados. Logo em seguida, esses dados inseridos servem como base para o treinamento das cGANs, as quais são treinadas até atingir uma condição de parada. Além disso, o Discriminador $D(x)$ julga os dados gerados com a finalidade de verificar se os dados sintéticos são similares aos dados inseridos e assim, garantir que o treinamento das cGANs foram eficientes. Em seguida, os DC também são importados de uma base do Excel®, lidos e preparados. Da mesma forma, o discriminador $D(x)$ julga os dados e o Teste de Equivalência para a Diferença de Duas

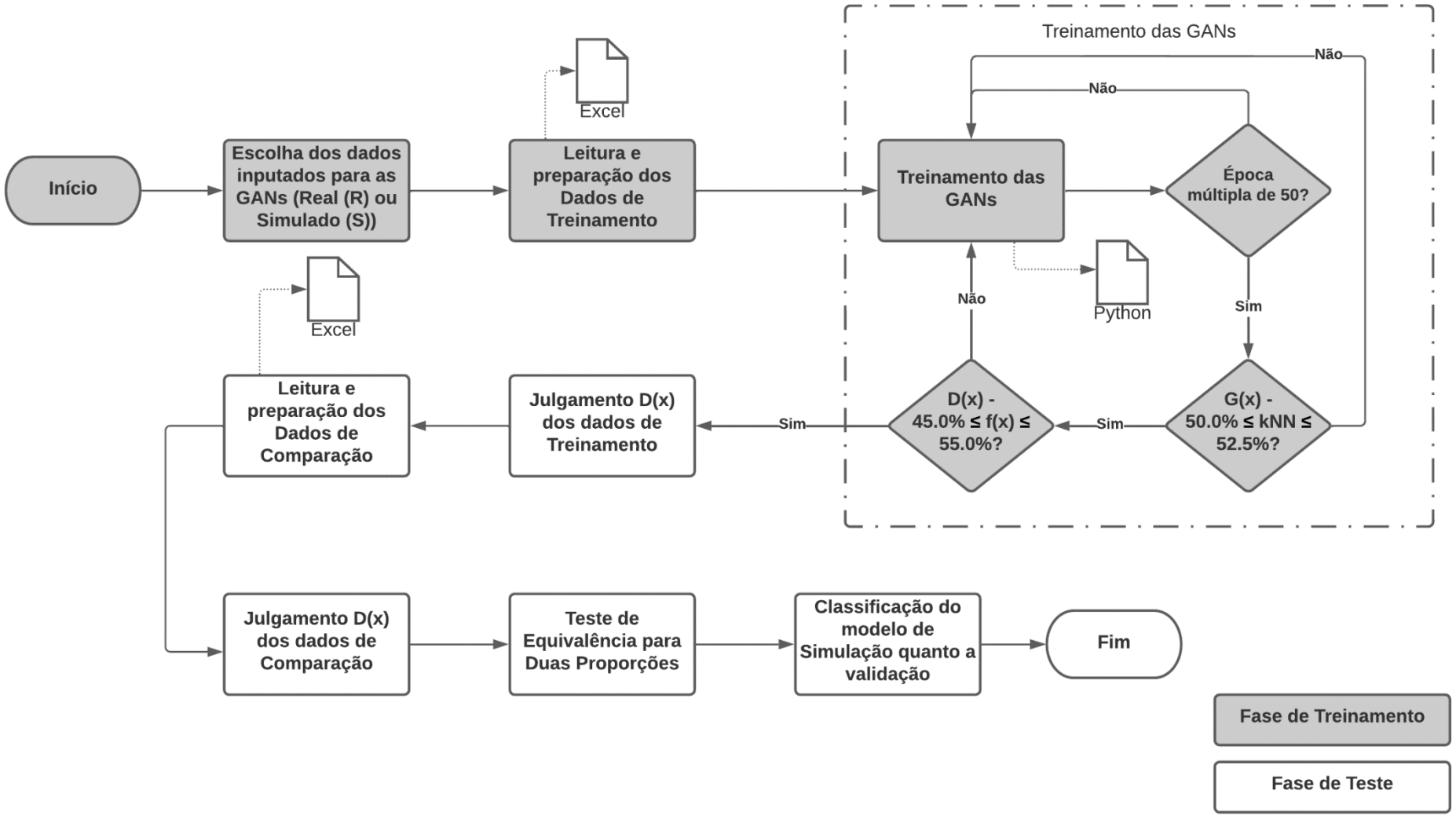

Figura 4.1 - Framework para a validação de modelos de SED utilizando cGANs

Proporções é realizado. Todas as etapas foram programadas em Python® usando as bibliotecas *TensorFlow*, *Keras* e *Scikit-learn*. O código pode ser encontrado no Apêndice B.

4.1.2. Fase de Treinamento

A fase de Treinamento consiste nas etapas em que as cGANs são treinadas e passam a gerar amostras sintéticas tão confiáveis quanto os dados inseridos até atingir as condições definidas. Primeiro, o algoritmo importa os DT do banco de dados de origem, geralmente uma tabela do Excel®. Cada observação de uma entidade específica deve estar na mesma linha, enquanto as colunas representam características individuais das amostras. Se o usuário deseja modelar uma distribuição que apresenta mais de uma característica por entidade, todos os atributos da mesma observação devem ser registrados na mesma linha. Dessa forma, assume-se que os dados inseridos para o treinamento são suficientes e não é necessário substituir valores ausentes, limpar ou avaliar *outliers*. Além disso, o algoritmo não considera dados não numéricos. É possível inserir números inteiros, reais, booleanos e dados categóricos codificados. Além disso, os dados são dimensionados usando o método *Robuster Scale* (dimensionamento robusto) (RAJU *et al*. 2020) pois ele leva a modelos de ML mais eficazes (FENG, 2021). O *Rubuster Scale* é utilizado porque a técnica não restringe os dados gerados em distribuições onde as caudas são limitadas.

Para que as cGANs gerem dados tão iguais quanto aos inseridos para o seu treinamento, alguns parâmetros devem ser definidos. Kurach *et al.* (2018) mostram uma série de parâmetros e valores que foram eficientes para configurar o algoritmo de treinamento das GANs. Assim, os parâmetros descritos no estudo foram utilizados para atingir os resultados mais rapidamente. De acordo com Casier *et al.* (2020), a Análise de Componentes Principais, do inglês *Pincipal Component Analysi*s (PCA), é recomendada para reduzir o espaço de entrada em problemas de DL. A abordagem propõe definir o tamanho do espaço latente e a densidade da camada oculta como um múltiplo do número de dimensões, baseado no PCA e no número dos dados de entrada (*M*v). A Tabela 4.1 apresenta os parâmetros usados para treinar as GANs.

A cada 50 épocas, o algoritmo é avaliado. Como explicado na seção 2.5.2, a avaliação é feita para garantir que as amostras sintéticas correspondam aos dados de treinamento. Nesse sentido, o estudo utiliza os Testes de Classificação de Duas Amostras – *Classifier Two-Sample Test* (C2ST) baseado na técnica do *k-NN* (CAI *et al.*, 2019; DAVID e OQUAB, 2017) para classificar os dados. No *k-NN*, a classe de observação é determinada pelo número de *k* vizinhos mais próximos (COVER e HART, 1967). Assim, o classificador recebe o conjunto de todos os

Tabela 4.1 - Parâmetros de configuração das cGANS

Parâmetros	Valores adotados
Tamanho Espaço Latente	$3M_{v=99\%}$
Número de camadas ocultas	2
Densidade das camadas ocultas	$8M_{v=99\%}$
Função de ativação da camada de entrada e da camada oculta	*Leaky Rectified Linear Unit* (passo = 0.2)
Função de ativação da camada de saída do gerador	Linear para o Robust Scaler
Função de ativação da camada de saída do discriminador	Sigmoide
Tamanho do lote	64
Algoritmo de otimização do gerador	Adam ($\alpha = 0.00002, \beta_1 = 0.5, \beta_2 = 0.999, \varepsilon = 10^{-8}$)
Algoritmo de otimização do discriminador	Adam ($\alpha = 0.0001, \beta_1 = 0.5, \beta_2 = 0.999, \varepsilon = 10^{-8}$)

dados sintéticos e os dados de treinamento e tenta separá-los. Ambos os dados são escalonados e possuem a mesma quantidade na amostra. Segundo David e Oquab (2017), se os dados sintéticos são perfeitamente realistas, o algoritmo *k-NN* classifica cada observação de forma aleatória e a precisão dos dados gerados pelas cGANs (A_c) é de 50.0%. Por outro lado, se os dados sintéticos não forem realistas, o classificador pode separar facilmente as observações, e a (A_c) esperada chega a 100%. No entanto, os dados gerados pelas cGANs podem estar em uma tolerância e assim, é realizado um teste de uma proporção. O resultado do *k-NN* deve estar dentro da faixa de tolerância. Dessa forma, a Acurácia final dos dados gerados $\left(A_{c_{final}}\right)$ é dada pela Equação 18:

$$\mathrm{A}_{C\,final} = \min\left(2 - 2\mathrm{A}_C, 100\%\right) \tag{18}$$

É válido lembrar que os dados sintéticos gerados pelas cGANs devem corresponder a um valor mínimo dos dados inseridos. Para o presente trabalho, considera-se o valor mínimo de 95.0%. Dessa maneira, a classificação dos dados do *k-NN* deve estar entre 50.0% e 52.5%, o que corresponde à uma $A_{c_{final}}$ entre 95.0% (*Ac* = 52.5%) e 100% (*Ac* = 50.0%).

Além disso, as cGANs precisam atingir uma segunda condição para interromper o seu treinamento. De acordo com Brownlee (2020), o Discriminador $D(x)$ avalia aleatoriamente os dados de treinamento. Se o Gerador $G(x)$ gerou dados muito próximos aos dados inseridos, significa que ele conseguiu enganar o Discriminador $D(x)$. Então, $D(x)$ discrimina os dados em torno de 50.0%. No entanto, observa-se que o discriminador não pode atingir 50.0% na

maioria das vezes. Nesse sentido, as GANs são treinadas até que $D(x)$ discrimine os dados entre 45.0% e 55.0%. Se o treinamento não atingir a acurácia mínima do modelo escolhida pela equipe de validação (*p-value* $\leq \alpha$), nem $D(x)$ classificar os dados entre 45.0% e 55.0% na mesma época, o treinamento continua até o número máximo de interação (épocas), que deve ser definido pelo usuário. Se esse número é alcançado sem atingir nenhuma condição, pode-se considerar que os resultados alcançados não foram satisfatórios.

Em suma, na Fase de Treinamento, o modelador deve escolher qual conjunto de dados devem ser inseridos no algoritmo das cGANs, podendo eles serem os Dados Reais (DR) ou os Dados Simulados (DS). As cGANs consideram os dados inseridos no algoritmo como dados reais. Esses dados podem ser os dados do modelo de simulação ou os dados do sistema fixo, porém, as cGANs os considera como dados reais. As cGANs realizam o treinamento a partir dos dados inseridos e, a cada 50 épocas, há uma verificação. A primeira verificação diz respeito ao *k-NN*. Os dados inseridos são comparados com os dados gerados pelo gerador e eles devem estar entre 50.0% e 52.5%. Estar nesse intervalo significa que a acurácia de treinamento que as cGANs conseguiram atingir foi entre 95.0% e 100.0%. Em outras palavras, os dados gerados pelas cGANs são, no mínimo, 95.0% iguais aos que foram inseridos. Se essa condição foi atingida, há também a verificação do Discriminador $D(x)$. Ele também avalia os dados que foram gerados pelas cGANs e esses dados sintéticos devem estar entre 45.0% e 55.0%. Se essa condição for atingida significa que $D(x)$ não sabe distinguir os dados reais (aqueles que foram inseridos) dos dados gerados pelas cGANs (sintéticos). Dessa maneira, tem-se valor de referência no qual o Discriminador julga os DT (que podem ser DR ou DS). Na segunda fase, o Discriminador julga o segundo conjunto de dados que serão comparados e é realizado o TEDDP.

4.1.2. Fase de Teste

A fase de teste começa após as cGANs atingirem o nível mínimo de acurácia determinada pelo modelador $\left(A_{c_{final}}\right)$ e o Discriminador $D(x)$ estar entre 45.0% e 55.0%. Neste ponto, o algoritmo já discriminou os DTs, uma vez que o resultado de $D(x)$ foi utilizado para atingir a segunda condição. Em seguida, os DCs são inseridos no algorítmo e começam a ser preparados. Se na primeira etapa os DRs forem inseridos para o treinamento das cGANs, os DSs são utilizados como DCs. Caso contrário, se os dados fornecidos para as cGANs forem os DSs, deve-se utilizar o DRs na segunda fase. Assim, os dados precisam estar da mesma forma em que os DTs foram inseridos na primeira fase. Então, os dados são redimensionados e

escalonados, levando em consideração os valores máximo e mínimo dos dados treinados. Por fim, o Discriminador $D(x)$ faz o julgamento dos DCs.

Na etapa final, é realizado um teste de Equivalência para a Diferença de Duas Proporções. A etapa consiste em verificar estatisticamente se os DTs e os DCs estão dentro de uma tolerância aceitável, a qual o modelador define previamente. Ou seja, o usuário define a tolerância de acordo com a necessidade de cada projeto. Afirmar que os DCs e os DTs são classificados de forma igual dentro de uma tolerância significa afirmar que os dados estão em uma faixa de precisão aceitável, mesmo assim pode-se considerá-los estatisticamente iguais. Assim como explicado na seção 2.7.2, o TEDDP é realizado considerando a tolerância escolhida.

A tolerância considerada segue a mesma proporção que a calulada para o *k-NN*. Dessa maneira, se o modelador deseja que a tolerância máxima (Tol_{max}) entre os dois conjuntos de dados seja de 5.0%, o algoritmo necessita fazer todos os testes considerando $Tol_{max}/2$, ou seja, a tolerância inserida corresponde à 2.5%. Como o Discriminador $D(x)$ julga os dados em torno de 50.0% e o teste considera a diferença entre o julgamento dos DTs e DCs, essa diferença pode atingir um valor máximo de 50.0%. Isso significa que, se após o treino, o discriminador julgou os dados de treinamento em torno de 50.0%, a diferença máxima é de 50.0% para mais ($D(x)$ considera os dados de comparação equivalente à 100.0%) ou 50.0% para menos ($D(x)$ considera os dados de comparação equivalente à 0.0%).

4.2. Classificação da validação do modelo de Simulação

Balci (2010) afirma que o modelo construído não deve ser definido como um modelo perfeitamente preciso ou totalmente inválido. O autor mostra que é possível construir faixas com níveis de aceitação como: muito forte, forte, satisfatório, marginal, deficiente e insatisfatório. A mesma ideia é compartilhada por Tsioptsias, Tako e Robinson (2016) que dizem que faltam definições concretas para os níveis de validação.

Dessa maneira, caso o modelo não valide considerando a tolerância escolhida, o algoritmo define qual é a mínima diferença para considerá-lo válido. O processo do TDEPP é realizado aumentando a tolerância em 0.1% a cada rodada. Assim, baseado na faixa proposta pelos autores, o modelo pode ser classificado em seis níveis, conforme mostra a Tabela 4.2. A primeira coluna indica a tolerância real dos dados, ou seja, o quanto os DTs podem ser considerados diferente dos DCs de acordo com a escolha do modelador. Já a segunda coluna

indica os valores considerados pelo algoritmo para o cálculo e a classificação da validação baseado na tolerância real.

Tabela 4.2 - Classificação do modelo de Simulação de acordo com a tolerância

Faixa de Tolerância (real)	**Faixa de Tolerância (algoritmo)**	**Classificação**
0.0% - 10.0%	0.0% - 5.0%	Muito forte
10.0% - 20.0%	5.0% - 10.0%	Forte
20.0% - 40.0%	10.0% - 20.0%	Satisfatória
40.0% - 60.0%	20.0% - 30.0%	Marginal
60.0% - 80.0%	30.0% - 40.0%	Deficiente
> 80.0%	> 40.0%	Insatisfatória

5. APLICAÇÃO

Este Capítulo tem como propósito validar o *framework* apresentado no Capítulo 4 por meio de distribuições teóricas contínuas, discretas, dados condicionais e um modelo de simulação. Além disso, é apresentado o poder do teste para essas distribuições. Por fim, três casos de modelos de SED reais são submetidos a validação utilizando as cGANs e o Teste de Equivalência para a Diferença de Duas Proporções.

5.1. Validação do *Framework* e Poder do Teste

Montgomery e Runger (2018) afirmam que o Poder de um teste estatístico estima a probabilidade de rejeitar H_0 se H_0 for realmente falso. Por outro lado, o valor β, denominado erro do Tipo II, é a diferença entre 100% de chance de acertar a resposta correta e o Poder do Teste. Portanto, no TEDDP, ele pode ser considerado como a probabilidade de afirmar que a diferença das duas proporções das amostras está dentro do limite de equivalência quando ela realmente está. O Poder do Teste está vinculado ao tamanho da amostra, ao intervalo de equivalência, ao desvio padrão e ao valor de α (WALKER e NOWACKI, 2011; MEYNERS, 2012). Assim, amostras maiores, diferenças próximas de zero e pequenos desvios-padrão aumentam o poder do teste. Se o teste permitir escolher um valor maior de α, o poder também aumentará. Porém, aumentar o valor de α significa aumentar a probabilidade de Erro Tipo I, afirmando equivalência quando ela não existe.

A fim de validar o método proposto e o Poder do Teste, foram realizados experimentos envolvendo 15 distribuições teóricas. As distribuições estatísticas foram treinadas até atingirem as condições explicadas na seção 4.1.2. Dessa maneira, a acurácia final das cGANs $\left(A_{C_{final}}\right)$ necessita de ser pelo menos 95.0%, sendo atingida em no máximo 10.000 épocas. Ou seja, os dados sintéticos gerados pelas cGANs devem ser, no mínimo 95.0% estatisticamente iguais aos dados inseridos. Quando as distribuições testadas forem uma distribuição condicional, o número máximo de épocas passa a ser 10000 vezes a quantidade de distribuições condicionais. Assim, se há duas distribuições com rótulos de dados condicionais, elas podem ser treinadas em até 20000 épocas. A Figura 2.1Figura 5.1 mostra um fluxograma de como o poder do teste foi realizado, enquanto a Tabela 5.1 mostra as distribuições usadas para o treinamento e suas características.

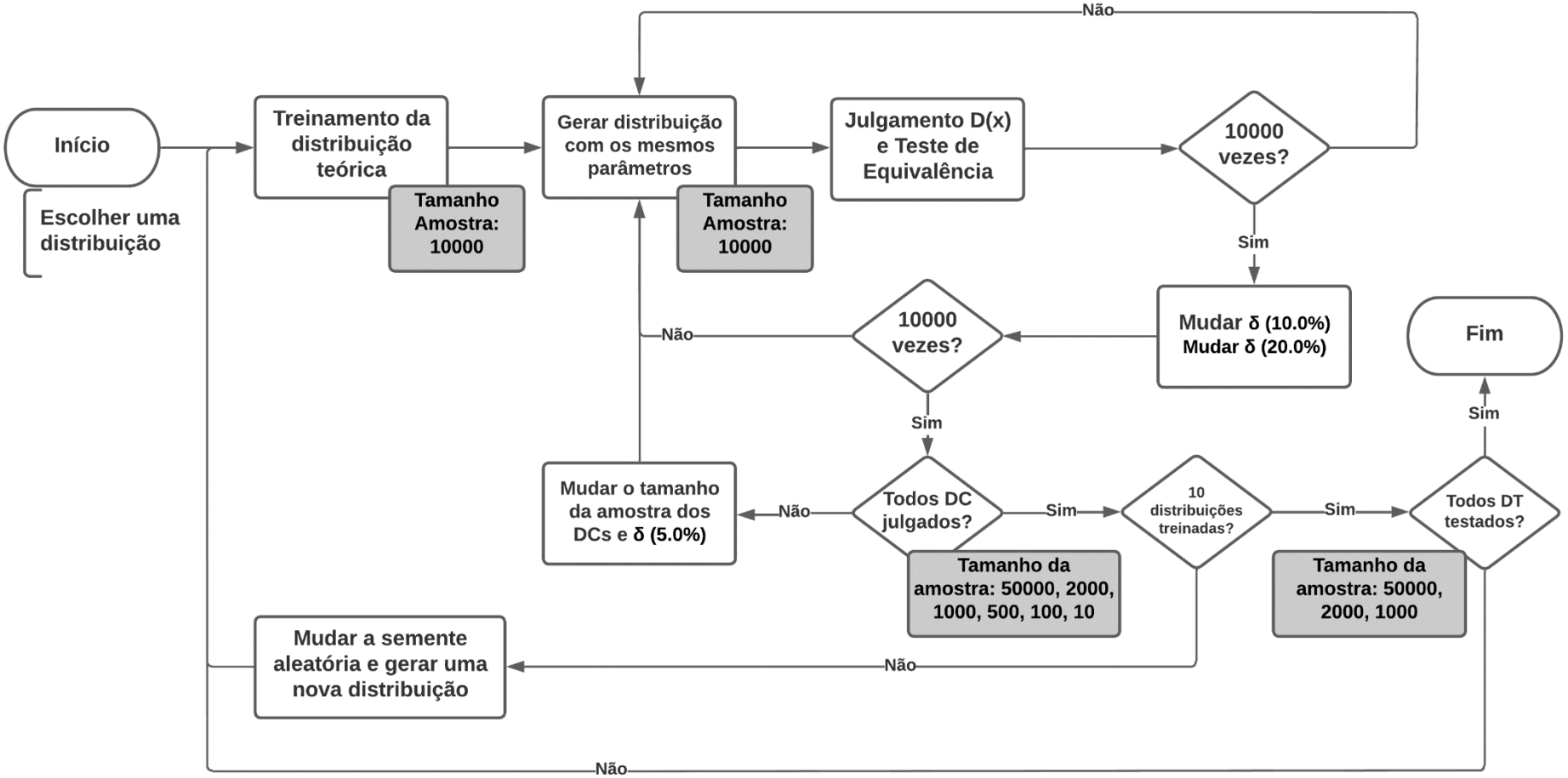

Figura 5.1 - Fluxograma para Validação e Poder do Teste das cGANs

Tabela 5.1 - Distribuições testadas para validar o método

Distribuição	**Parâmetros**	**Correlação**	**Tipo**
Normal	μ: 100; σ: 3	-	Contínua
Exponencial	μ: 1; λ: 0	-	Contínua
Uniforme	$lim_{inferior}$: 10; $lim_{superior}$: 20	-	Contínua
Lognormal	μ: 0; σ: 0.6	-	Contínua
Bimodal	μ_1: 100; σ_1: 10 μ_2: 140; σ_2: 10	-	Contínua
Normal Bivariada	$\mu_{x,y}$: [100, 100] σ_{xy}: [[9.0, 2.4][2.4, 9.0]]	0.8	Contínua
Normal Bivariada	$\mu_{x,y}$: [100, 100] σ_{xy}: [[9.0, -2.4][-2.4, 9.0]]	-0.8	Contínua
Normal Bivariada	$\mu_{x,y}$: [100, 100] σ_{xy}: [[9, 0][0, 9]]	0.0	Contínua
Normal Multivariada	$\mu_{x,y}$: [100, 100, 100] σ_{xy}: [[9.0, -2.7, -2.1] [2.7, 9.0, -1.8][-2.1, -1.8, 9.0]]	x_1x_2: 0.9 x_1x_3: -0.7 x_2x_3: -0.6	Contínua
Normal Multivariada	$\mu_{x,y}$: [100, 100, 100] σ_{xy}: [[9, 0, 0][0, 9, 0][0, 0, 9]]	x_1x_2: 0.0 x_1x_3: 0.0 x_2x_3: 0.0	Contínua
Poisson	λ: 10	-	Discreta
Binomial	p: 100; n: 0.6	-	Discreta
Weibull Beta	α: 10; α: 2.5; β: 4	-	Condicional
Triangular Triangular	mín: 4; mod: 8; máx: 10 mín: 8; mod: 10; máx: 14	-	Condicional
Triagem LOS	Resultados do modelo de simulação	0.32	Simulação

Para a validação do método, primeiro foi necessário definir uma distribuição teórica e seus parâmetros, conforme mostra a Tabela 5.1. Essa primeira distribuição apresentou um tamanho amostral de 10000 para os DT. Após as cGANs serem treinadas, atingir as condições determinadas e o Discriminador $D(x)$ julgar os DTs, uma distribuição com as mesmas características (DC) foi gerada, também com 10000 amostras. Logo em seguida, a nova distribuição (DCs) foi submetida ao julgamento de $D(x)$ e o TEDDP foi executado considerando a classificação dos dois conjuntos de dados. Nesta etapa, foi considerado uma tolerância real (δ) de 0.05 (5.0%). Como ambas as distribuições (DTs e DCs) apresentam os mesmos parâmetros, espera-se que o julgamento dos dados esteja dentro da faixa de tolerância

pré-estabelecida. Após esta etapa, novas distribuições com as mesmas características também foram geradas, porém com diferentes sementes aleatórias, por 10000 vezes.

Por exemplo, foi gerada uma distribuição normal N(100,3) com 10000 dados sendo considerada como DT e inseridos para o treinamento das cGANs. Depois de atingidas as condições de treinamento ($A_{C_{final}} \geq 95.0\%$ e $45.0\% \leq D(x) \leq 55.0\%$), outra distribuição normal N(100,3) com tamanho amostral de 10000 dados também foi gerada (DC), e assim, o teste de Equivalência foi realizado. O processos de gerar uma distribuição N(100,3) com 10000 dados, o conjunto de dados ser submetido ao julgamento de $D(x)$ considerando uma tolerância real de 5.0% e o teste de Equivalência foram repetidos por 10000 vezes. Novamente, o mesmo procedimento foi realizado, considerando então uma tolerância real δ de 0.10 (10.0%) e uma tolerância real δ de 0.20 (20.0%).

Após o teste de Equivalência ocorrer 10000 vezes gerando distribuições iguais às dos dados inseridos com tamanho amostral 10000, a mesma distribuição com os mesmos parâmetros foi gerada com 5000 dados. Isso foi necessário para verificar a influência do tamanho da amostra no teste e no julgamento de $D(x)$. Todos os procedimentos mencionados anteriormente foram realizados e os DCs foram gerados para tamanhos de amostras de 2000, 1000, 500, 100 e 10 dados. Este *loop* foi realizado dez vezes para verificar o Poder do Teste da distribuição treinada considerando os DTs com 10000 dados em seu conjunto. Em seguida, o mesmo ciclo foi realizado, variando o número de dados na Fase de Treinamento, ou seja, a distribuição teórica para o treinamento (DT) foi gerada com 5000, 2000 e 1000 dados. As 15 distribuições citadas na Tabela 5.1 foram submetidas ao teste de Equivalência para determinar o Poder do Teste do método e algoritmo desenvolvido e validar o *framework* proposto no Capítulo 4. Todos os retângulos em cinza na Figura 5.1 indicam uma mudança no tamanho amostral dos dados e um novo ciclo de testes. Todos os testes foram realizados em computação em nuvens, utilizado os serviços da *Amazon®*, *Amazon Web Service* (AWS) que permite testes mais rápidos e em paralelo.

5.1.2. Distribuições de probabilidade contínuas

5.1.2.1. Univariadas

A primeira distribuição testada foi uma distribuição normal N(100,3) com 10000 dados na Fase de Treinamento. Assim, foram geradas 10 distribuições normais com os mesmos parâmetros, mas em sementes aleatórias diferentes. A Figura 5.2 mostra o treinamento de uma das distribuições treinadas na época 1, 50, 150 e 200. Vale lembrar que os dados em verde (real)

são os dados inseridos no algoritmo, no qual as cGANs os considera real e começa a gerar dados sintéticos (laranja) semelhante àqueles fornecidos para o treinamento.

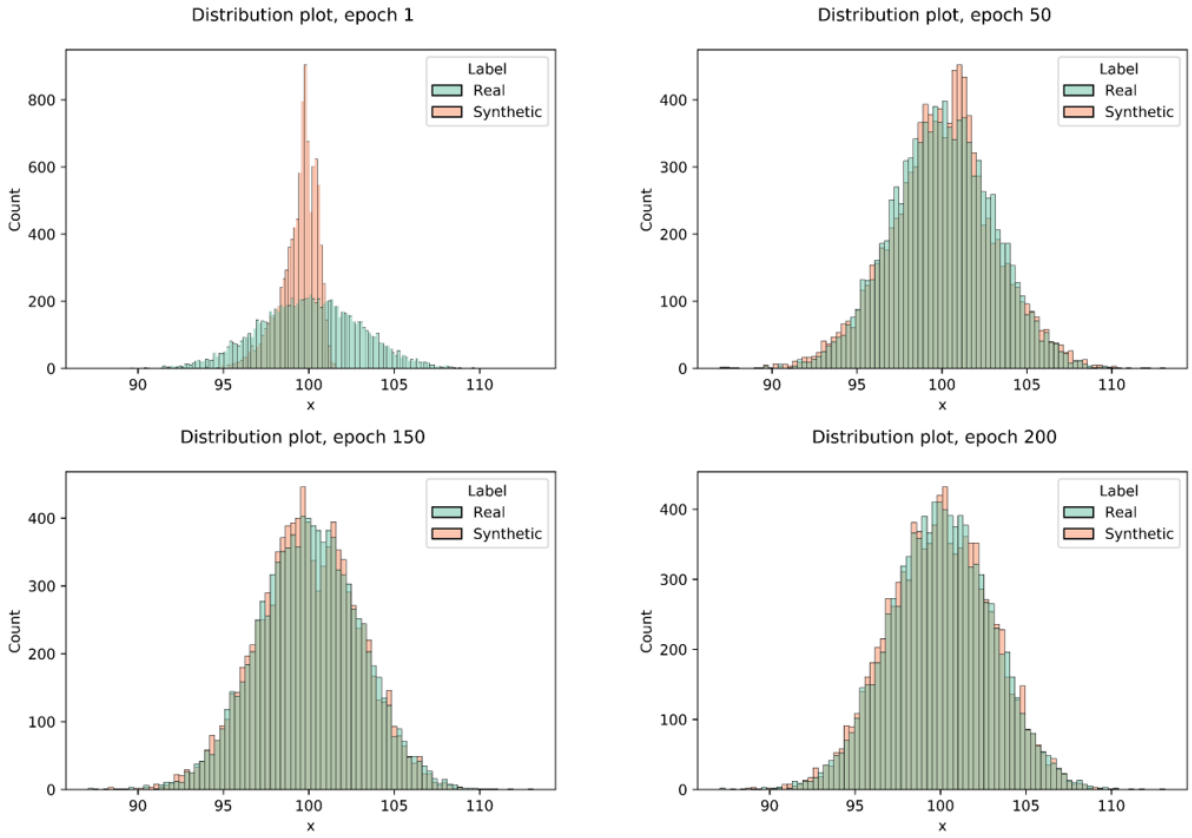

Figura 5.2 - Treinamento da Distribuição Normal

De acordo com os testes, a distribuição normal atinge as condições em torno de 225 épocas, com $A_{C_{final}}$ de 97.93% para 10000 dados na primeira fase. No entanto, se o número de dados for diminuído na Fase de Treinamento, as cGANs necessitam de mais interação para fornecer resultados semelhantes. Por exemplo, se forem inseridos 5000 dados para o treinamento, as cGANs precisariam de cerca de 250 épocas, enquanto 2000 dados precisariam de 795 iterações e 1000 dados atingiriam as condições, na média, em 1195 épocas. A $A_{C_{final}}$ mínima foi atingida no treinamento com 5000 dados (97.46%), enquanto o treinamento com 1000 dados alcançou a $A_{C_{final}}$ máximo (98.55%). Além disso, na média, o Discriminador $D(x)$ julga as distribuições geradas na Fase de Treinamento em torno de 50.24%. O máximo alcançado foi com os DTs de tamanho 1000 (51.88%) e o mínimo foi apresentado por 5000 dados (49.45%). A Tabela 5.2 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a realização do processo descrito na Figura 5.1.

Tabela 5.2 - Resumo do treinamento da distribuição normal – 10 ciclos

		Desv. Pad	Inferior	Superior
DT – 10000 dados				
Época	225.00	181.43	112.55	337.45
Acurácia	97.93	0.93	97.35	98.51
Discriminador	49.63	2.95	47.80	51.46
DT – 5000 dados				
Época	570.00	1138.03	-135.35	1275.35
Acurácia	97.46	1.02	96.83	98.09
Discriminador	49.45	3.04	47.57	51.33
DT – 2000 dados				
Época	795.00	620.24	410.58	1179.42
Acurácia	98.22	1.05	97.57	98.87
Discriminador	49.98	2.42	48.48	51.48
DT – 1000 dados				
Época	1195.00	1341.53	363.53	2026.47
Acurácia	98.55	0.98	97.94	99.16
Discriminador	51.88	3.12	49.94	53.82

A Figura 5.3, a Figura 5.4 e a Figura 5.5 apresentam as curvas do Poder do Teste para os tamanhos de amostra de 10000, 5000, 2000 e 1000 para a Fase de Treinamento considerando a tolerância de 5.0%, 10.0% e 20.0% respectivamente. De acordo com a curva da Figura 5.3, ou seja, quando a tolerância é de 5.0%, são necessárias 10000 amostras tanto para os DCs quanto para os DTs. Ao considerar 5000 dados nos DTs, o Poder do Teste cai para 60.13% considerando um tamanho amostral de 10000 dados nos DCs. Por outro lado, se for levado em consideração 10000 dados no DTs e 5000 no DCs, o Poder do Teste é de 83.44%. O algoritmo não detectou semelhança quando foram testados 1000 dados na fase de treinamento.

Em relação às curvas geradas para as tolerâncias de 10.0%, não há diferença entre os resultados gerados pelas amostras com 10000 e 5000 na Fase de Treinamento. Quando o Discriminador $D(x)$ julgou distribuições normais apresentando 10000 e 5000 amostras nos DCs e considerando uma tolerância de 10.0%, ambos obtiveram 100.0% de sucesso. No entanto, se o tamanho da amostra nos DC diminuir para 1000, a porcentagem de sucesso cai para 77.89%. Para distribuições normais com tamanho de amostra de 500 para DCs, o Poder do Teste gira em torno de 38.66%. O mesmo comportamento é mostrado nas curvas que possuem 2000 e 1000 dados na Fase de Treinamento.

Para o presente estudo, considera-se necessário obter pelo menos 80.00% de sucesso para ter um Poder de Teste eficiente. Dessa maneira, quando for necessário considerar 5.0% de tolerância dos dados, recomenda-se pelo menos um tamanho amostral de 5000 dados na

segunda fase e 10000 amostras na fase de treinamento. Por outro lado, se for considerar uma tolerância de 10.0%, é necessário, pelo menos, 1000 dados para os DCs se o treinamento for feito com 10000 amostras ou, pelo menos, 3000 amostras nos DCs se o treinamento for realizado com 1000 dados.

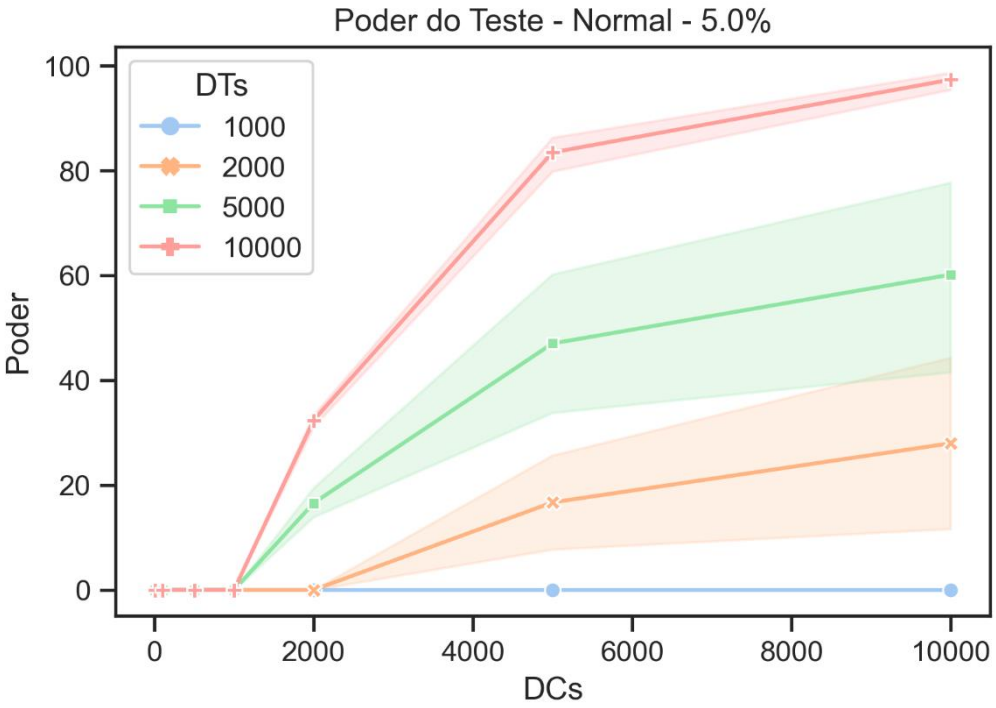

Figura 5.3 - Poder do Teste - Normal - Tolerância 5.0%

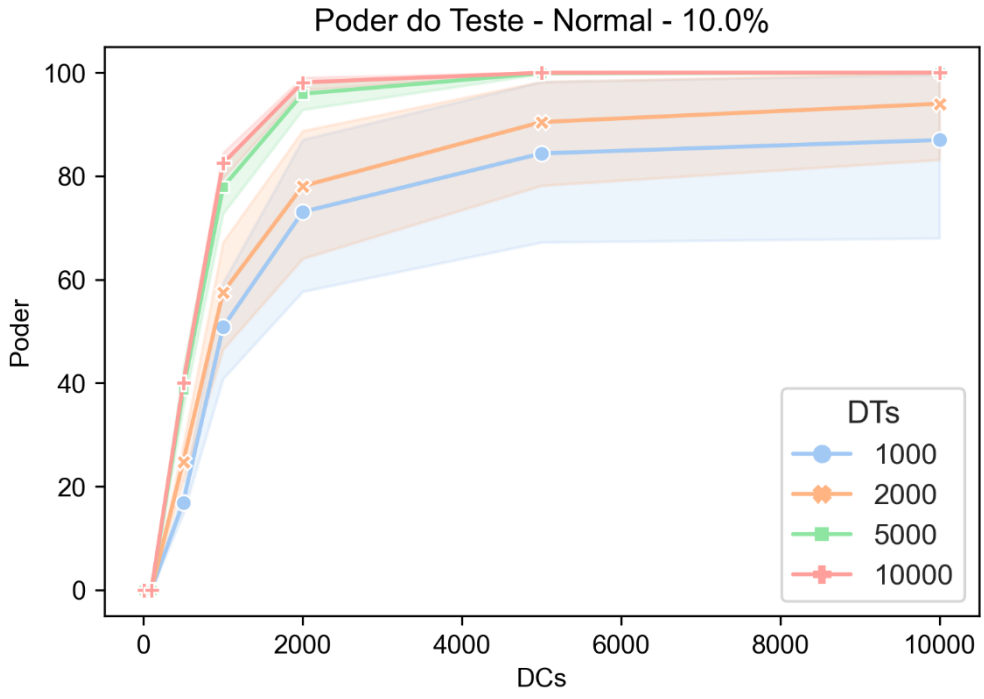

Figura 5.4 - Poder do Teste - Normal - Tolerância 10.0%

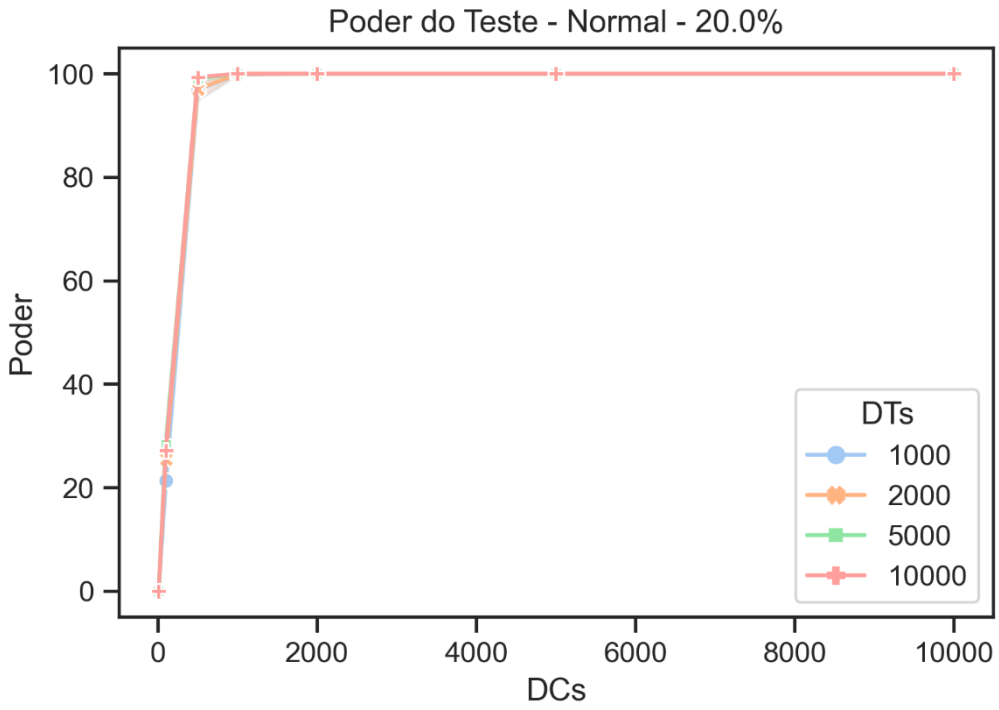

Figura 5.5 - Poder do Teste - Normal - Tolerância 20.0%

Embora amostras menores apresentem bons resultados em ambas as fases, existem algumas desvantagens. Primeiro, o intervalo de confiança foi maior quando se utilizou 1000 dados na Fase de Treinamento ao invés de utilizar 10000 dados. Segundo, quando se compara a curva com a mesma quantidade de dados na Fase de Treinamento, mas variando o número da amostra em DC, amostras menores apresentam um Poder de Teste menor. Nesse sentido, como o intervalo de confiança começa a aumentar, não se pode garantir o Poder do Teste para amostras menores. Por fim, quanto menor o intervalo considerado para a tolerância, mais difícil será o sucesso no Teste de Equivalência. Ao analisar a curva gerada com os dados considerando a tolerância de 20.0%, o Poder do Teste aumenta consideravelmente para todos os DCs, independentemente do número de amostras na Fase de Treinamento. Ao treinar os dados com um tamanho amostral de 1000 na primeira fase, é necessária uma distribuição de tamanho amostral de 500 dados, para atingir um poder de teste de 96.88%. Apesar de aumentar o Poder do Teste ao utilizar tolerâncias maiores no Teste de Equivalência, a comparação se torna mais fraca, uma vez que pode se considerar que os dados DT e DC são iguais, sendo que na realidade eles estão cada vez mais distantes.

O mesmo teste foi realizado para a distribuição Exponencial, a distribuição Uniforme, a distribuição Lognormal e uma distribuição Bimodal, conforme mostrado na Tabela 5.1. A partir deste ponto, todos os resultados são inferidos considerando a tolerância de 10.0%. Apesar das curvas de 5.0% terem sido geradas, na maioria dos resultados, o Poder do Teste atingiu

80.0% quando a primeira fase apresentou um tamanho amostral de 10000 dados e a segunda fase, no mínimo 5000. Quantidades abaixo desses valores tanto para os DTs quanto para os DCs não atingiram os níveis satisfatórios. Além disso, foi considerado, na faixa de validação, uma validação forte para dados que estão em uma diferença de até 10.0% para a tolerância real.

As três primeiras distribuições apresentam resultados semelhantes à distribuição Normal. No entanto, todas elas precisaram de mais épocas para atingir as condições de parada, mesmo na Fase de Treinamento com tamanho amostral de 10000 dados. Considerando 10000 dados na Fase de Treinamento, a distribuição Exponencial necessitou, em média, de 540 epócas para atingir uma $A_{C_{final}}$ de 96.81% e o discriminador julgou os dados em torno de 51.82%. Já a distribuição Uniforme precisou de 365 iterações, enquanto a distribuição Lognormal necessitou de 600 épocas. Mais informações e as curvas do poder do teste para essas distribuições podem ser encontradas no Apêndice C.

Como o Poder do Teste mínimo considerado bom para a distribuição é de 80.0%, recomenda-se usar pelo menos 1000 amostras nos DCs quando as cGANs são treinadas com 10000 dados. Por outro lado, é necessário pelo menos uma amostra de tamanho 5000 (DCs) se o treinamento for executado com 1000 DTs. Apesar desse *trade-off*, o intervalo de confiança para as curvas com menos dados na amostra de treinamento é maior, indicando que amostras maiores devem ser utilizadas nessa etapa, levando em consideração distribuições contínuas.

Por fim, a distribuição Bimodal necessitou de 2010 épocas na fase de treinamento para atingir as condições ao utilizar 10000 dados, enquanto foram necessárias 4065 iterações para uma amostra de 1000 dados. Observa-se que quanto mais dados na fase de treinamento, menor a quantidade de épocas necessárias. Além disso, a distribuição Bimodal apresenta resultados diferentes em termos de tamanho de amostra na fase de treinamento. De acordo com os testes, não é possível julgar os dados considerando uma amostra menor que 500 para a tolerância de 10.0%, uma vez que em todos os testes com esse tamanho da amostra, as distribuições não foram reconhecidas (Poder do Teste = 0.0%). Sugere-se assim, o uso de pelo menos 1500 dados ao treinar cGANs com 10000 amostras e 5000 dados se a Fase de Treinamento tiver 2000 amostras. Ainda, não se recomenda o uso de apenas 1000 dados na primeira fase porque a maior taxa de sucesso para identificar a mesma distribuição foi, em média, de 48.12% julgando 10000 dados. O Apêndice C apresenta as curvas e o resumo do treinamento da distribuição bimodal.

5.1.2.2. Multivariadas

O método também tem como objetivo validar um modelo de SED que apresenta mais de um *output* usando apenas um teste. Dessa maneira, três distribuições Normais Bivariadas

foram treinadas. A primeira apresenta correlação positiva (0.8), a segunda negativa (-0.8) e a terceira não tem correlação (0.0). Além disso, foram treinadas e testadas duas distribuições Multivariadas (três distribuições Normais Multivariadas) conforme a Tabela 5.1. A Figura 5.6 mostra o treinamento da distribuição Normal Bivariada sem correlação para a primeira e a última época, enquanto a Figura 5.7 mostra o treinamento para a distribuição Multivariada com correlação com 5000 dados de treinamento.

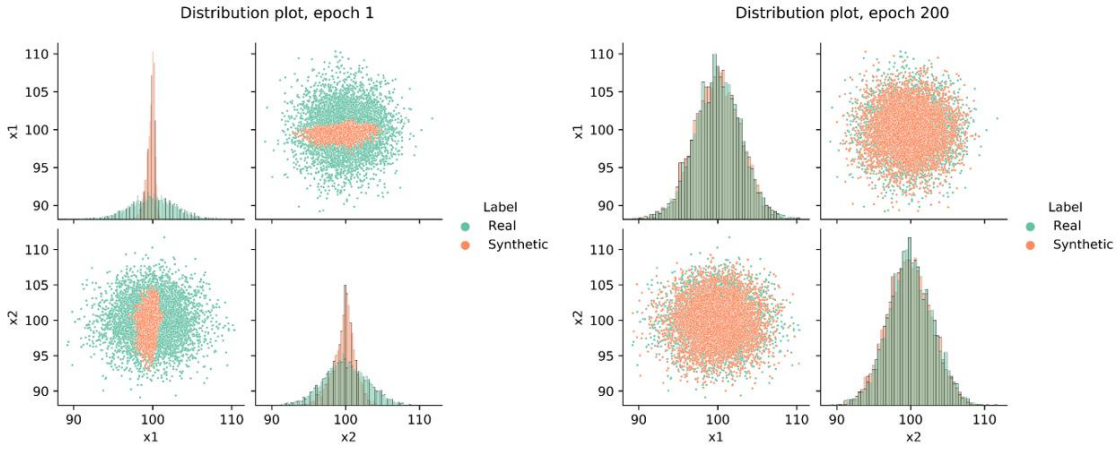

Figura 5.6 - Treinamento da Distribuição Normal Bivariada - sem correlação

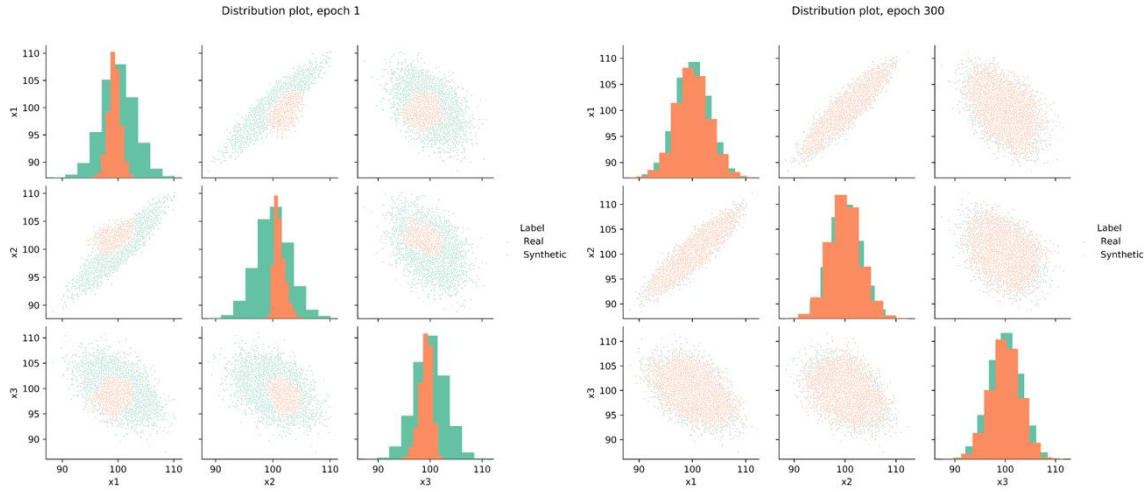

Figura 5.7 - Treinamento da Distribuição Multivariada correlacionada

As três distribuições Bivariadas Normais apresentaram resultados bastante semelhantes considerando o número de épocas para atingir as condições, acurácia e o julgamento de $D(x)$. O número mínimo de épocas necessárias foi para a distribuição com correlação positiva em 10000 dados no treinamento, atingindo as condições em torno de 205 épocas. A distribuição que necessitou de mais iterações para chegar no mesmo resultado foi a distribuição com correlação positiva e 1000 dados na primeira fase, com cerca de 1140 épocas.

Ao comparar as distribuições contínuas onde mais de uma dimensão é testada, observa-se que é necessário obter mais dados tanto na Fase de Treinamento quanto na Fase de Teste. O Poder do Teste para o tamanho da amostra de 1000 (DTs) não atingiu pelo menos 80.00% em nenhum caso. A porcentagem máxima de sucesso foi de 44.79% julgando a distribuição Bivariada com correlação negativa considerando 10000 dados na Fase de Teste (DCs). Pode-se afirmar o mesmo para as distribuições Bivariadas treinadas com tamanho amostral de 2000 (DTs). Tanto as distribuições com correlação positiva quanto a negativa não obtiveram pelo menos 80.00% de julgamentos corretos. Ambos atingiram cerca de 75.00% considerando o tamanho amostral de 10000 na Fase de Teste. Nesse sentido, de acordo com as curvas geradas para essas distribuições, recomenda-se o uso de pelo menos 5.000 dados para treinar as cGANs e pelo menos 2000 para discriminá-los. No entanto, se não houver correlação, apenas 3000 dados são necessários na primeira fase, enquanto cerca de 2000 devem ser fornecidos na segunda fase.

Ao aumentar uma dimensão nas distribuições, ou seja, testar os dados com três distribuições Normais Multivariadas, o número de épocas necessárias para atingir as condições estipuladas também aumenta. Isso indica que, quanto mais dimensões se quer testar, mais tempo de treinamento o algoritmo necessita. O mesmo pode ser observado quanto ao número de dados necessário na primeira fase. Quanto mais dimensões se quer testar de uma só vez, maior a quantidade de dados necessário. Há ainda uma relação entre a correlação das distribuições. Se os dados forem correlacionados, mais épocas são necessárias para atingir as condições de parada.

Os resultados para o julgamento considerando 2000 e 1000 dados na primeira fase mostraram que um baixo tamanho amostral torna o Poder do Teste muito baixo. Para a distribuição Normal Multivariada com correlação, a maior taxa de acerto foi de 30.77% para o tamanho amostral de 2000 nos DTs e 22.33% para o tamanho amostral de 1000. Apesar de apresentar uma taxa de sucesso maior que o primeiro teste de multivariadas, as distribuições sem correlações atingiram 49.45% de sucesso considerando 2000 dados nos DTs e 14.03% considerando 1000 dados. Assim, recomenda-se, em ambos os testes, pelo menos 5000 dados de treinamento e 4000 de DC, ou 10000 na primeira fase e 1500 na segunda para distribuições com correlação e 3000 sem correlação. A Figura 5.8 mostra as curvas para a distribuição Normal Multivariada com correlação, enquanto a Figura 5.9 apresenta as curvas para uma Normal multivariada sem correlação. Vale ressaltar que quanto menos dados inseridos no algoritmo na Fase de Treinamento, maior o intervalo de confiança do Poder do Teste, o que

indica que os resultados não são tão assertivos. Mais informações a respeito do treinamento das distribuições Multivariadas estão apresentadas no Apêndice D.

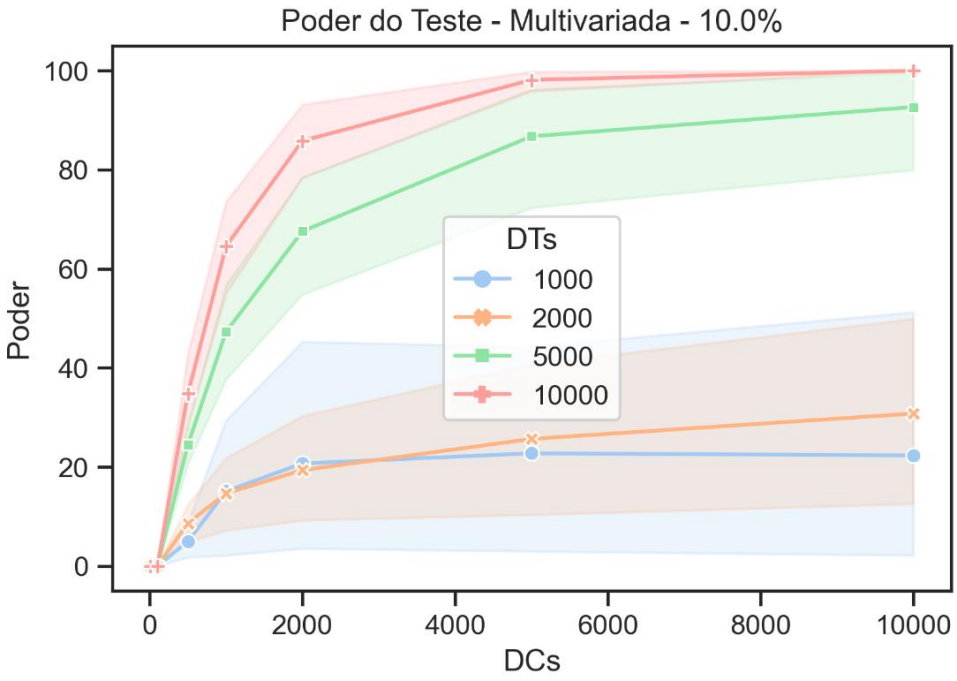

Figura 5.8 - Poder do Teste - Normal Multivariada correlacionada - Tolerância 10.0%

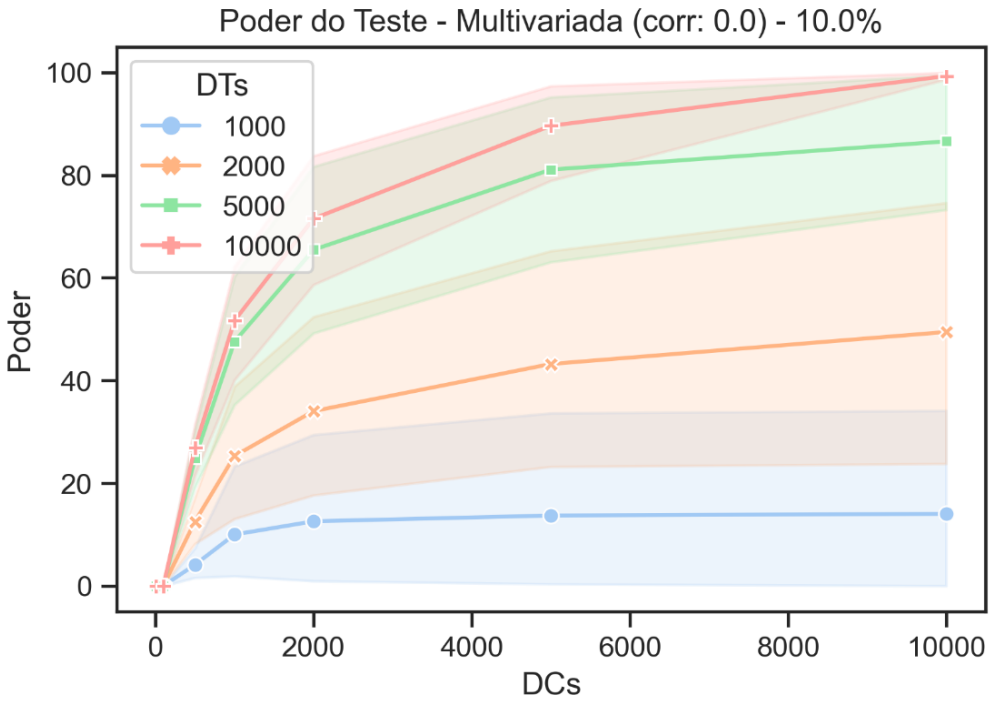

Figura 5.9 - Poder do Teste – Normal Multivariada sem correlação – Tolerância 10.0%

Por fim, os resultados mostram que as distribuições contínuas alcançam bons resultados em poucas interações. No entanto, se for fornecido poucos dados (cerca de 2000) na Fase de Treinamento, o algoritmo precisa de mais épocas para fornecer resultados confiáveis. Além

disso, para o teste com mais de uma variável para cada observação, o algoritmo requer um tamanho de amostra maior para fornecer pelo menos 80.0% do Poder do Teste.

5.1.2. Distribuições de probabilidade discretas

O algoritmo também permite a inserção de amostras de dados discretos. Nesse sentido, foram testadas duas distribuições de probabilidade discretas. A metodologia foi aplicada em uma distribuição de Poisson (λ = 10) e uma distribuição Binomial (p = 0.6, n = 100), conforme descrito na Tabela 5.1. Diferente das distribuições contínuas, as distribuições discretas precisam de mais interações para atingir a $A_{C_{final}}$ mínima de 95.0%. Além disso, em alguns dos testes, as cGANS não conseguiram atingir às condições de julgamento e precisão em 10000 épocas e, neste caso, não entraram nos resultados da análise. Assim, uma nova semente aleatória do Python® foi determinada para essas distribuições até conseguir atingir as condições necessárias. A Figura 5.10 mostra o treinamento dos dados para a distribuição de Poisson considerando quatro épocas e 2000 dados na primeira fase.

A distribuição de Poisson necessitou de 270 épocas para o seu treinamento quando a amostra continha 10000 dados, enquanto para 1000 dados amostrais, foram necessárias, em média, 1365 épocas. Até aqui, todas as distribuições apresentaram comportamentos semelhantes em relação ao seu treinamento. Quanto mais dados amostrais, menos iterações são necessárias. Para a distribuição Binomial, houve uma inversão na quantidade de dados e o número de iterações. Ao treinar a distribuição com 10000 dados, foram necessárias, em média, 490 épocas e para 5000, 315 iterações foram o suficiente.

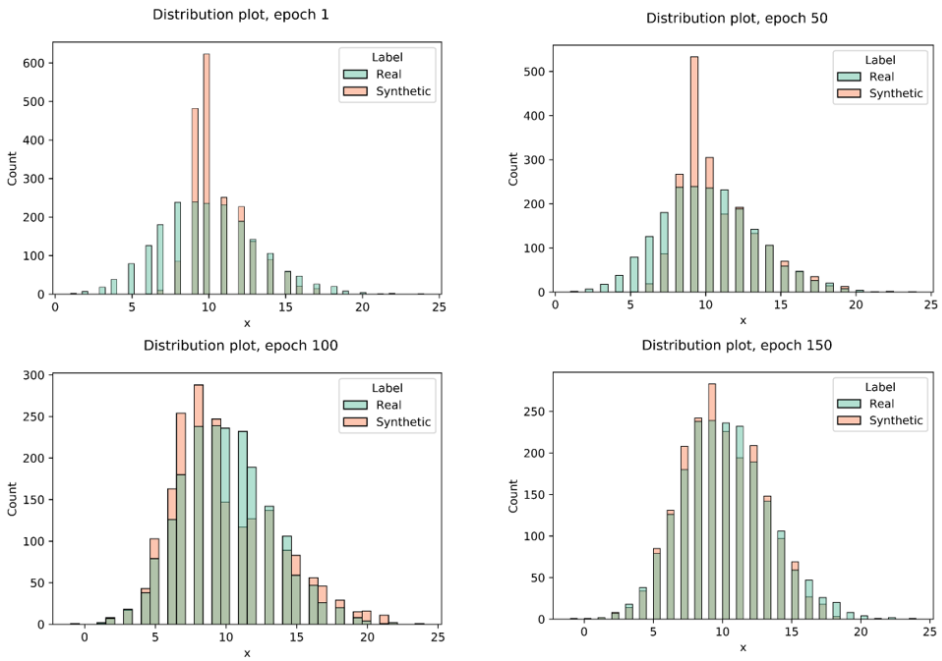

Figura 5.10 - Treinamento da Distribuição de Poisson

Em se tratando do Poder do Teste, ambas as distribuições têm os mesmos resultados se a fase de treinamento tiver um tamanho amostral de 10000 ou 5000 dados. Assim, é necessária apenas uma amostra de 1000 dados para se obter 80.0% do Poder de Teste ao realizar o teste de Equivalência considerando a tolerância de 10.0%. Além disso, a distribuição de Poisson apresentou melhores resultados do que a Binomial para 1000 dados na sua primeira fase. O percentual máximo de acertos para Poisson foi de 94.89%, enquanto o Binomial obteve 65.55%. Vale ressaltar que, mesmo comparando com as distribuições contínuas, houve um aumento significativo no Poder do Teste para as distribuições contínuas com menos dados de treinamento (1000 dados). A curva gerada pelos 2000 dados na primeira fase para o Binomial, é semelhante à curva gerada por um treinamento com 5000 e 10000 amostras. Além disso, os resultados mostraram que os intervalos de confiança para ambas as distribuições são menores do que o apresentado nas distribuições contínuas. Resumindo, embora as cGANs precisem de mais iterações para gerar dados semelhantes para probabilidade discreta, essa quantidade é eficiente para julgá-los. A Figura 5.11 mostra a curva do Poder de Teste para a distribuição de Poisson. Mais detalhes do treinamento das cGANs para as distribuições discretas podem ser encontradas no Apêndice E.

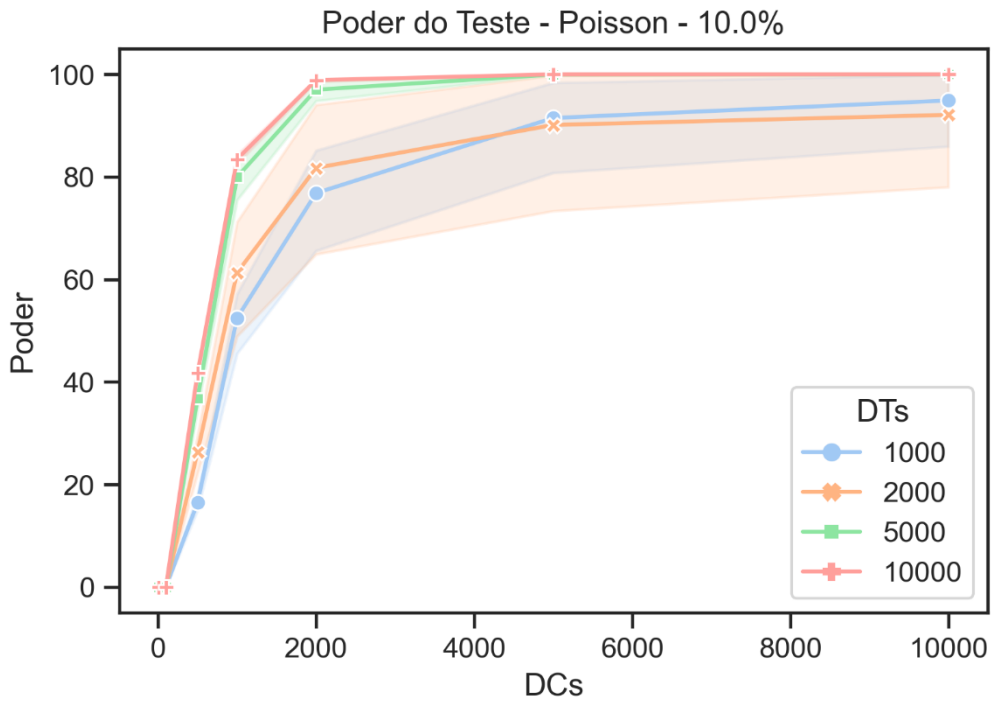

Figura 5.11 - Poder do Teste – Poisson – Tolerância 10.0%

5.1.3. Distribuições contínuas usando dados condicionais

Como descrito na Seção 4.1.2, o algoritmo permite treinar e julgar dados baseado em uma condição. Assim, pode-se realizar a validação de uma métrica que apresenta mais de um

tipo de entidade, como diferentes graus de pacientes dentro de um hospital, utilizando apenas um treinamento e um teste de Equivalência.

Para verificar o comportamento dos dados condicionais, foram realizados dois testes usando cGANs em distribuições de probabilidade contínuas. O primeiro teste foi realizado treinando uma distribuição Weibull (β = 10) e uma distribuição Beta (2.5, 4). O segundo teste julgou duas distribuições Triangulares T(4, 8, 10) e T(8, 10, 14), conforme a Tabela 5.1. Para a identificação dos tipos de dados na Fase de Treinamento, as primeiras distribuições (Weibull e Triangular (4, 8, 10)) receberam um rótulo de valor 1 e as segundas (Beta e Triangular (8, 10, 14)) receberam um rótulo de valor 2. Em ambos os testes, foram fornecidas amostras de tamanho 10000 para cada distribuição, totalizando assim 20000 dados para o treinamento.

Quando os dados são condicionais, as cGANs precisam de mais épocas para atingir as condições de parada de treinamento. Dessa forma, como a amostra do treinamento apresenta o dobro de dados, o número máximo de épocas também foi dobrado, aumentando para, no máximo, 20000 iterações. A Figura 5.12 mostra o treinamento para a distribuição Weibull e Beta.

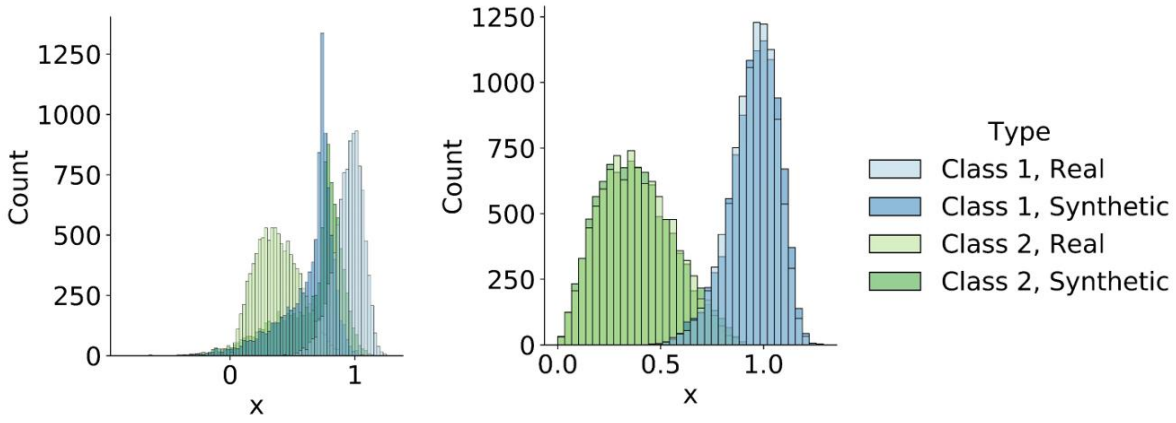

Figura 5.12 - Treinamento de dados condicionais - Weibull e Beta

O número médio de épocas necessárias foi de 2075 para as distribuições Weibull e Beta e 3075 para distribuições triangulares considerando 10000 dados para cada uma. Quando os cGANs treinam a distribuição com 1000 amostras, eles precisam de 2445 e 1780 interações, respectivamente. A $A_{C_{final}}$ máxima atingida pelos conjuntos de dados foi de 98.59% e 97.78% para a Weibull e Beta e Triangulares, respectivamente, considerando 1000 dados no treinamento.

Embora o tempo de treinamento (número de iterações) seja maior quando os dados são condicionais, o discriminador julga mais fácil as mesmas distribuições. Para ambos os testes, o Poder do Teste chega a pelo menos 80.0% ao treinar com 10000, 5000 e 2000 dados para cada

distribuição e a segunda fase precisa de 2000 dados quando eles são treinados com uma tolerância de 5.0%. Ao passar a tolerância para 10.0%, a segunda fase necessita de apenas 500 dados. Para essa tolerância um pouco maior, se o treinamento utiliza 1000 dados, o Poder do Teste atinge 77.09% nas distribuições Beta e Weibull para 500 dados nos DCs e 77.81% para distribuições Triangulares considerando 1000 dados na fase de Teste. Além disso, o intervalo de confiança para o Poder do Teste é menor do que no treinamento utilizando dados que não são condicionais. Ao usar cGANs, o discriminador julga mais fácil o conjunto dos dados. A Figura 5.13 mostra a curva do Poder do Teste para as distribuições Triangulares para a tolerância do teste de 10.0%. Mais detalhes do treinamento dos dados condições estão apresentados no Apêndice F.

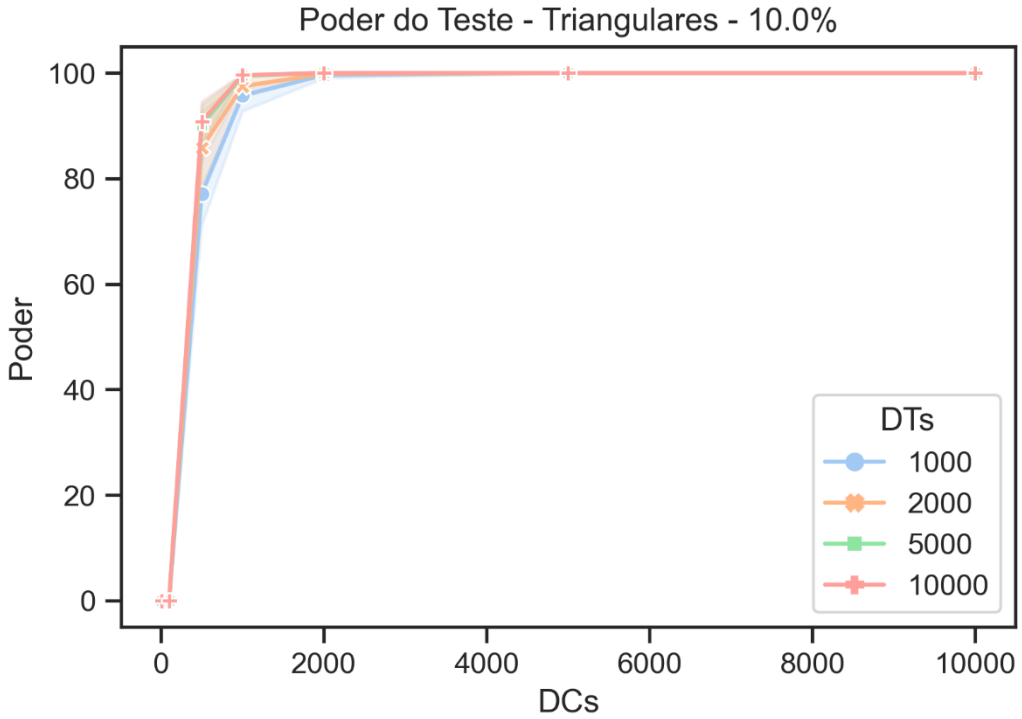

Figura 5.13 - Poder do Teste - Dados Condicionais - Triangulares - Tolerância 10.0%

5.1.4. Modelo de Simulação

Finalmente, a técnica foi aplicada para validar um modelo de SED. O modelo representa um hospital de campanha que atende pacientes com suspeita de COVID-19 em uma cidade de médio porte de Minas Gerais. Os pacientes chegam ao hospital, passam pelo registro e seguem para a triagem. Na triagem, os pacientes podem receber seis classificações: branco, azul, verde, amarelo, laranja e vermelho. Se o paciente for do tipo vermelho, ou seja, o mais crítico, ele passa por uma estabilização e depois é transferido para o hospital principal da cidade. Pacientes classificados nas cores laranja e amarelo são considerados como casos moderados, enquanto os outros são casos leves. Diferente dos pacientes classificados como vermelhos, os outros são

atendidos por um dos quatro médicos disponíveis. Após a consulta, o paciente vai para a medicação e/ou faz um raio-X, ou receber alta direto. Alguns pacientes com classificação moderada necessitam de internação, que pode ser de um a três dias. Se internados, eles são direcionados para uma área isolada e recebem tratamento pelos enfermeiros e médicos responsáveis. A Figura 5.14 apresenta o modelo simulado no software FlexSim®.

Figura 5.14 - Modelo computacional hospital de COVID

Para testar o método, foram considerados como saídas do modelo para a validação o Tempo Porta Triagem (TPT) e o Tempo de Permanência (LOS) dos pacientes que não necessitam de internação. O TPT é o tempo decorrido entre o momento em que o paciente chega ao hospital até o momento em que ele é triado. Já o LOS, é o tempo total de permanência do paciente, desde a sua entrada até a sua alta. Os dados inseridos nas cGANs para o treinamento (DTs) e considerados reais foram obtidos através da replicação do modelo. Em seguida, novas réplicas foram realizadas para obter os dados da segunda fase e considerar os dados da simulação. Ou seja, cerca de 500 réplicas foram realizadas no modelo, onde 100 delas foram destinadas para os dados reais e as outras 400 para os testes e construção das curvas. Isso foi realizado para testar o método não somente em saídas e distribuições teóricas, mas também para modelos reais. O modelo atingiu uma precisão aceitável em torno de 825 épocas considerando 10000 dados na primeira fase, enquanto precisou de 2900 interações para 1000 dados. A $A_{C_{final}}$

máxima foi atingida com 10000 DTs com a precisão em torno de 98.42%. A Figura 5.15 mostra a evolução e o treinamento dos dados para o TPT e LOS com 5000 dados.

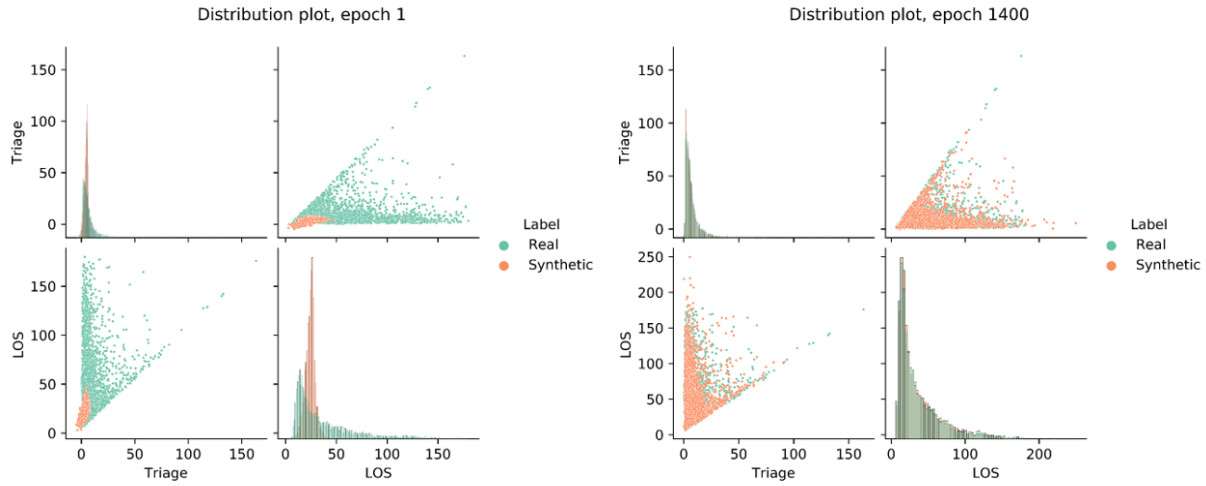

Figura 5.15 - Treinamento do Modelo de Simulação - TPT e LOS

A melhor taxa de acerto do Poder de Teste foi de 98.21%, obtido por um tamanho amostral de 10000 dados na Fase de Treinamento e 10000 nos DCs. O Poder do Teste diminuiu para 91.58% quando foi considerado 5000 dados na segunda fase e 70.92% considerando apenas 2000. Além disso, os resultados não foram bons quando foram inseridos 5000 dados no treinamento. Para essa quantidade de dados na primeira fase, é necessário, pelo menos, uma amostra de 6000 dados para atingir o poder do teste de 80.0%. Além disso, a melhor taxa de sucesso foi de 86.43% considerando 10000 dados nos DCs. Dessa forma, recomenda-se utilizar no treinamento a amostra de tamanho 10000 e 6000 na segunda fase, pois, dessa maneira, o Poder do Teste sobe para cerca de 93.0%. Se for considerado 2000 ou 1000 dados no treinamento, o discriminador julgou corretamente apenas 10.10% e 5.13% para um tamanho de amostra de 10000 nos DCs, mostrando que um tamanho amostral pequeno não consegue capturar o comportamento das curvas das saídas do modelo.

Por outro lado, se a tolerância dos dados for de 20.0%, o Poder do Teste aumenta para 100.0% considerando 10000, 5000 e 2000 dados na Fase de Treinamento e mais de 1000 dados no conjunto de DCs. No entanto, os testes mostraram um poder de 80.96% para um tamanho de amostral de 10000 nos DCs e 1000 nos DTs. Apesar disso, embora seja necessário um tamanho amostral maior para o modelo de SED com mais de uma variável, o método se mostrou eficiente. A Figura 5.16 mostra a curva para o modelo de simulação com tolerância de 10.0%. Outras informações a respeito do treinamento e das curvas estão no Apêndice G.

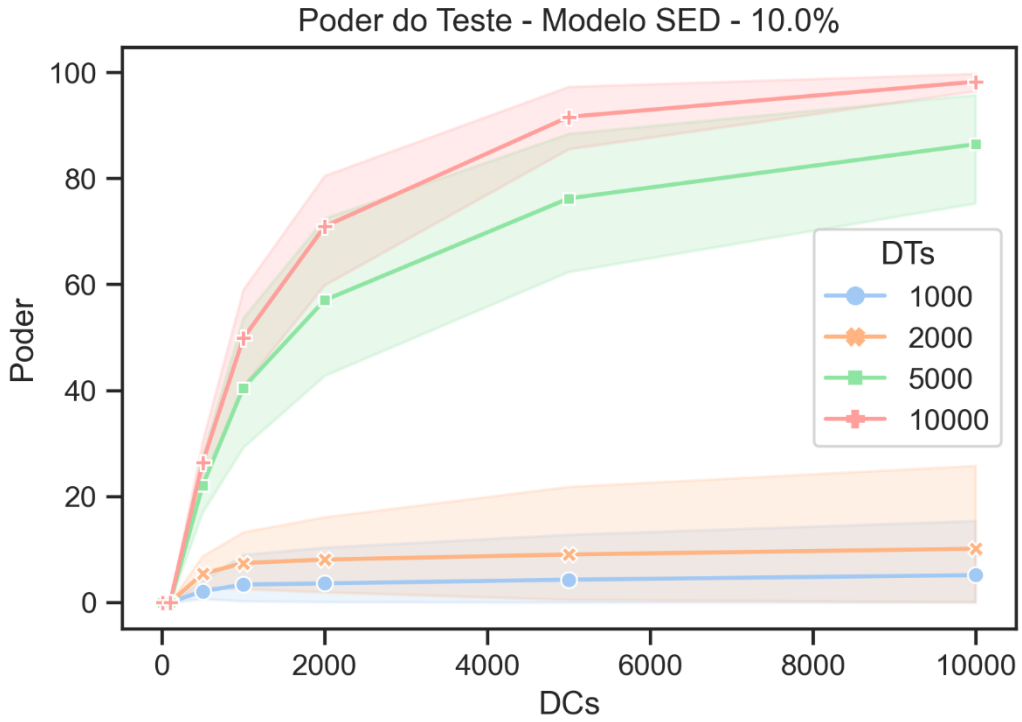

Figura 5.16 - Poder do Teste - Modelo de Simulação - Tolerância 10.0%

5.1.5. Conclusões do Poder do Teste

Os testes mostraram que as cGANs foram eficientes para treinar os dados através de distribuições teóricas contínuas e discretas e também para os dados condicionais. Além disso, as saídas do modelo de SED também mostraram que a abordagem proposta obteve sucesso. Em todos os testes, fica evidente que, quanto mais dados o algoritmo recebe na fase de treinamento, melhor o julgamento dos dados e menor o intervalo de confiança das curvas geradas. Apesar disso, há um *trade-off* entre a $A_{C_{final}}$ e a quantidade de dados inseridos na primeira fase, uma vez que a $A_{C_{final}}$ máxima de todos os testes ocorreu quando as amostras dos DTs eram de 1000 dados. Embora haja esse *trade-off*, todas as cGANs só pararam seu treinamento após atingir a $A_{C_{final}}$= 95.0%. Além disso, o aumento do número de dados para a fase de teste implica em maior tempo computacional, uma vez que mais dados devem ser processados a cada época.

Apesar do trabalho considerar uma $A_{C_{final}}$ = 95.0% para todos os testes, há também a possibilidade de o modelador escolher uma precisão menor, o que implica em menos iterações, pois o algoritmo pode atingir a condição de parada em menos tempo. Por outro lado, considerar uma $A_{C_{final}}$ menor implica em fazer um treinamento onde os dados podem se distanciar dos dados inseridos e não refletir ao conjunto que se quer treinar. Cabe ao modelador e equipe de

validação decidir as estratégias e a $A_{C_{final}}$ na qual o modelo de simulação pode ser considerado aceitável.

A tolerância também é um fator que se deve levar em consideração. Em todos os casos, ao se julgar as distribuições com a tolerância de 5.0%, é necessário um tamanho amostral maior em ambas as fases. Os testes mostraram que é necessário pelo menos 5000 dados na Fase de Teste. Apesar disso, o algoritmo se mostrou eficiente ao julgar os dados em 10.0%, o que mostra que modelos de simulação podem ser validados utilizando essa tolerância, que o método considera uma validação "muito forte".

Ainda pode-se concluir que, se for validar o modelo com mais de uma métrica de uma só vez, também são necessários mais dados na primeira fase. Assim, recomenda-se o uso conjunto de dados que contenha um tamanho amostral maior para a primeira fase e a amostra menor para os DCs. O modelador pode decidir qual dado usar em cada fase, o simulado ou do sistema real. Porém, apesar do crescente avanço da indústria 4.0 e a facilidade da coleta de dados do sistema real, ainda é mais fácil gerar dados no simulador. Dessa maneira, sugere-se o uso de dados simulados para a Fase de Treinamento (maior quantidade) e os dados reais para a Fase de Teste (menor quantidade).

5.2. Validação de modelos reais

Após os testes realizados nas distribuições teóricas e mostrado a eficiência do Poder do Teste no método proposto (Seção 5.1), a metodologia foi aplicada em três objetos de estudo reais. Os modelos construídos são dois modelos daárea de manufatura e um na área da saúde. Para cada um dos modelos, será necessário responder à nove perguntas:

1. Quais os *outputs* de saída do modelo, ou seja, quais as métricas de validação?
2. Quais conjuntos de dados serão inseridos na primeira fase (DTs) e quais serão utilizados na fase de teste (DCs)?
3. Qual o número de iterações necessárias para atingir uma $A_{C_{final}}$ mínima de 95.0%? Após o treinamento, qual a $A_{C_{final}}$ atingida?
4. Qual o $D(x)$ para os DTs?
5. Qual o $D(x)$ para os DCs?
6. Se o modelo não for validado considerando 5.0% de tolerância, qual a tolerância mínima para poder considerá-lo válido?
7. Baseado no resultado na pergunta anterior, qual a faixa de classificação de validação do modelo?

8. Qual o Poder do Teste?
9. Qual a probabilidade (β) de cometer o Erro do Tipo II, ou seja, considerar um modelo válido, sendo que na realidade, não é?

Vale lembrar que o presente trabalho assume que os modelos foram construídos seguindo a metodologia proposta por Montevechi *et al.* (2010), onde já foram definidos os objetivos do sistema, a construção do modelo conceitual e sua validação, a modelagem dos dados de entrada foi feita de forma correta, o modelo foi construído no *software* de simulação e verificado.

5.2.1. Modelo 1 – Linha de montagem

5.2.1.1. Descrição do modelo

O primeiro modelo estudado e simulado representa uma linha de montagem de *karts* utilizando peças Lego®. O processo foi criado em laboratório para fins didáticos. A linha é composta por 5 postos de trabalho, sendo os 4 primeiros divididos entre as tarefas de montagem e o último corresponde à inspeção. Para a reposição de matéria-prima, há um estoque central no qual as peças dos 4 primeiros postos ficam agrupadas em lotes de 8 unidades. Quando necessária a reposição, os operadores se deslocam do posto de trabalho até ao estoque central para a reposição dos *kits*. Além disso, entre cada posto de trabalho há um estoque intermediário, onde o que cada operador produziu fica aguardando para a produção no próximo posto. A linha tem como capacidade total a produção de 40 unidades.

O processo se inicia no Posto 1 com um fluxo empurrado, isto é, a produção ocorre de maneira contínua independente do consumo do processo seguinte. Ocasionalmente há uma quebra de máquina neste posto e, quando isso acontece, o Operador 1 deve parar a produção até que o problema na máquina seja resolvido. Já no Posto 2 o fluxo é puxado, ou seja, o Operador 2 só inicia a montagem quando o Operador 3 puxa a peça produzida no processo anterior. Em seguida o *kart* segue para o posto 3, que produz em lotes de 2 unidades. Neste posto há um *kanban* que tem a função de sinalizar para o Operador 4 quando cada lote está finalizado. Com isso, o Operador 4 se desloca até o estoque intermediário do Posto 3, leva o lote de duas unidades para o seu posto, finalizando a produção dos *karts* em lotes de 4 unidades. A cada lote produzido, o Operador 5 inicia o processo de inspeção, realizando medições, verificando o funcionamento e a qualidade dos carrinhos. Os *karts* aprovados são embalados e conduzidos em lotes de 4 unidades até a área de logística. Se o *kart* for reprovado, o Operador

5 leva o produto até a área de rejeitos. A taxa de retrabalho neste processo é de 10.0%. A Figura 5.17 apresenta a representação do modelo no *software* Flexsim®.

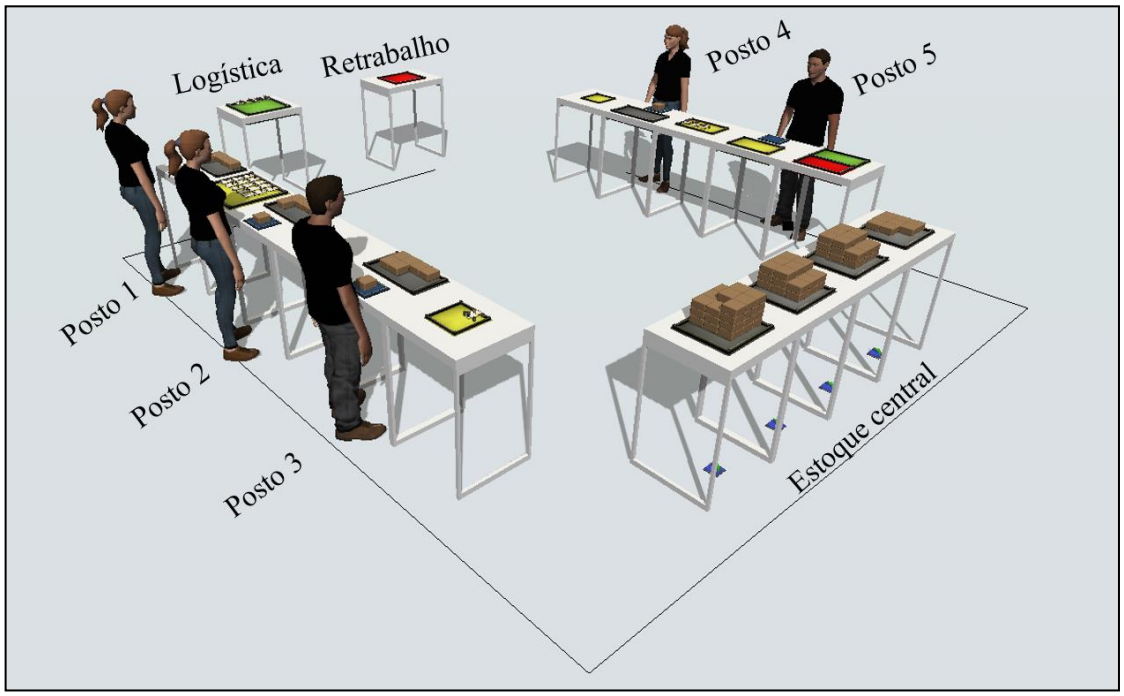

Figura 5.17 - Modelo 1 - Linha de montagem

5.2.1.2. Validação do modelo

A métrica para a validação do modelo é o *lead time* de montagem de cada *kart*. Os dados reais foram coletados através da filmagem do processo, em que cada rodada corresponde a montagem de 40 *karts*. Mais detalhes a respeito da construção do modelo podem ser encontrados em Rocha (2020). Foram coletados os dados de 960 *karts*, através da filmagem do processo, sendo os DRs considerados para a segunda fase, ou seja, eles são os DCs. Como o modelo de simulação permite que mais dados sejam gerados, foram realizadas 250 réplicas (cada réplica produz 40 *karts*) obtendo-se assim, 10000 dados para os DTs.

Após o treinamento das cGANs, foram necessárias 2400 épocas para atingir uma $A_{C_{final}}$ equivalente à 96.20% dos dados inseridos. O Discriminador julgou os DTs com um $D(x)$ igual à 54.05%, enquanto os DCs foram julgados em 50.83%. A Figura 5.18 mostra o treinamento do *lead time* dos *karts* na primeira iteração, duas épocas intermediárias e o resultado final.

Considerando uma tolerância real de 5.0% e realizando o teste de Equivalência, não se pode afirmar estatisticamente que a classificação dos dados reais e simulados são iguais (*p-value*: 0.066). O teste atingiu um Poder de 77.77%, o que implica que a probabilidade de cometer o erro do tipo II é de 22.23%. Ou seja, a probabilidade de afirmar que a diferença entre a classificação dos dois conjuntos de dados estarem dentro do intervalo de tolerância sendo que

eles não estão é de 22.23%. Apesar de não validar o modelo com uma tolerância real de 5.0%, a diferença mínima para considerar que o modelo está validado é de 12.0% (*p-value*: 0.04966). Como explicado na seção 4.2, o algoritmo refaz o teste aumentando a tolerância em 0.1% a cada rodada. Sendo assim, pode-se dizer que a validação do modelo está classificada em uma faixa de aceitação "Forte", conforme proposto na Seção 4.2. O resultado do teste apresentado pelo Python® pode ser encontrado no Apêndice H.

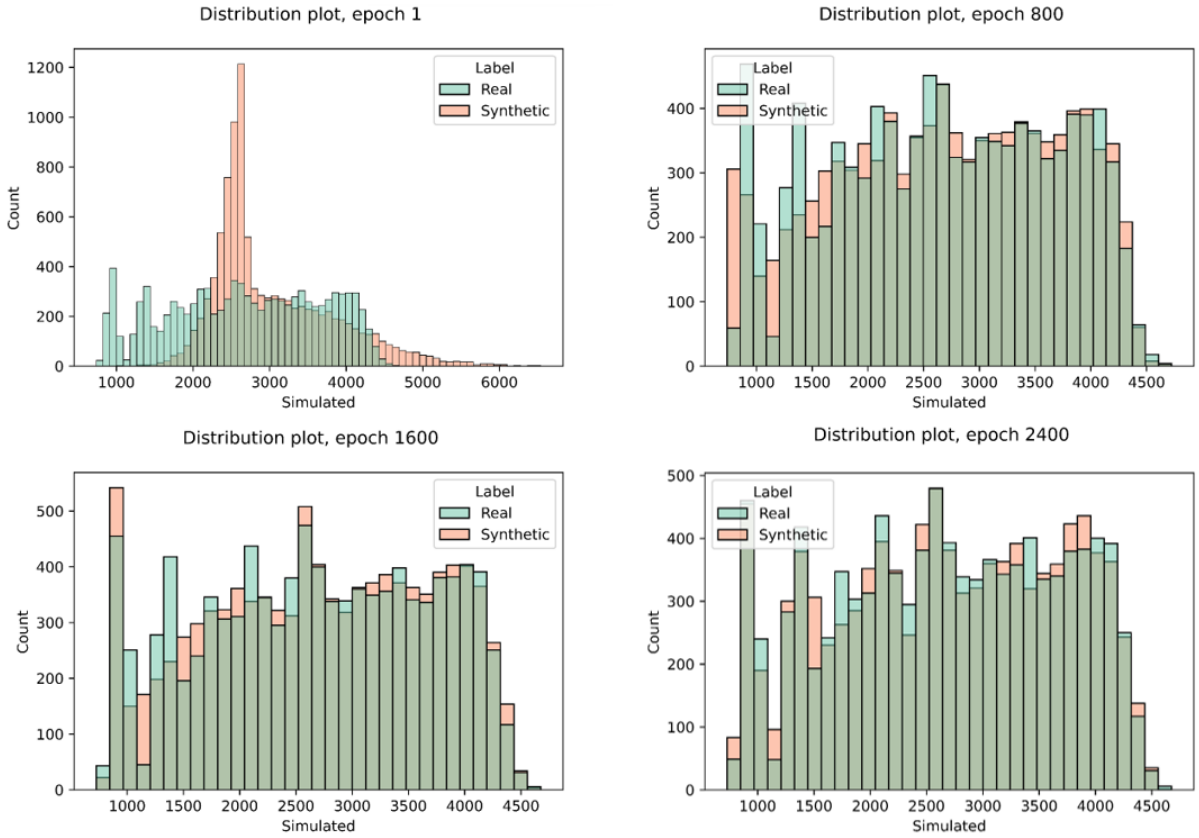

Figura 5.18 - Treinamento do *output* do Modelo 1 - *Lead time*

Vale a pena ressaltar que o método proposto visa comparar os dados levando em consideração a distribuição, a média e o desvio padrão de uma só vez. Dessa maneira, pode-se considerar que a confiança do teste é de 95.0%. Se a validação for realizada com testes já presentes na literatura, é necessário comparar os dados através de suas médias e desvio padrão. Assim, seria necessário realizar dois testes estatísticos, diminuindo a confiança do modelo para 90.25% (0.95*0.95).

5.2.2. Modelo 2 – Confecção de roupas

5.2.2.1. Descrição do modelo

O segundo modelo corresponde a uma confecção de roupas de médio porte que produz roupas de inverno e verão e está classificada no segmento de *fast fashion* (produção rápida de

roupas). O modelo simulado corresponde a uma linha que confecciona roupas de inverno e verão. Para o presente estudo, apenas dados coletados para as roupas de inverno foram utilizados. Apesar de ter trinta estações de trabalho disponíveis, a linha tem nove funcionários trabalhando em dez processos, desde o corte de tecidos até a expedição das roupas.

O processo começa com o Operador 1, que transporta a matéria-prima da Área de Recebimento, desenrola o tecido e realiza o corte (Processo “A”). Após o corte, o tecido segue para o primeiro processo de costura (Processo “B”), realizado pelo Operador 2. Então, o tecido segue para o “Processo C”, onde é executado outra etapa de costura pelo Operador 3. O próximo estágio é o “Processo D”, o qual apresenta dois operadores (Operador 4 e Operador 5) e duas máquinas de costura trabalhando em paralelo. Depois, o produto segue para o "Processo E", no qual é realizado a bainha e mandado para a costura da etiqueta (Processo “F”). Esses processos são realizados pelos Operadores 6 e 7, respectivamente. O último processo de costura é o arremate feito pelo Operador 8. Após o arremate, as roupas são passadas e dobradas pelo Operador 9 no Processo “G”. Por fim, as roupas são enviadas para a área de Expedição em lotes de 50 unidades. Entre cada etapa, existe um estoque intermediário e todos os transportes são realizados manualmente pelos operadores responsáveis pelo processo anterior. O sistema foi modelado no *software* FlexSim® e está representado na Figura 5.19**Erro! Fonte de referência não encontrada.**. Mais detalhes a respeito do sistema podem ser encontrados em Santos *et al.* (2021).

Figura 5.19 - Modelo 2 - Confecção de roupas

5.2.2.2. Validação do modelo

O objetivo do modelo é avaliar os estoques intermediários que estão sendo o gargalo do fluxo. Dessa forma, os dois estoques em que os produtos passam mais tempo aguardando são os estoques entre os processos B e C e entre os processos C e D. Portanto, os Estoques 2 e 3 são as métricas de validação do modelo computacional.

Os dados reais foram coletados através dos registros da própria empresa. Como o modelo de simulação consegue gerar mais dados que os coletados do sistema real, os DSs foram inseridos para o treinamento das cGANs. Foram considerados 15000 dados nos DTs e 6519 para os DCs. A Figura 5.20 mostra o treinamento das duas métricas, onde foram necessárias 1950 épocas para atingir uma $A_{C_{final}}$ de 95.8% dos dados inseridos. O discriminador julgou os DTs com um $D(x)$ de 47.86%, enquanto os DCs foram julgados em 44.09%.

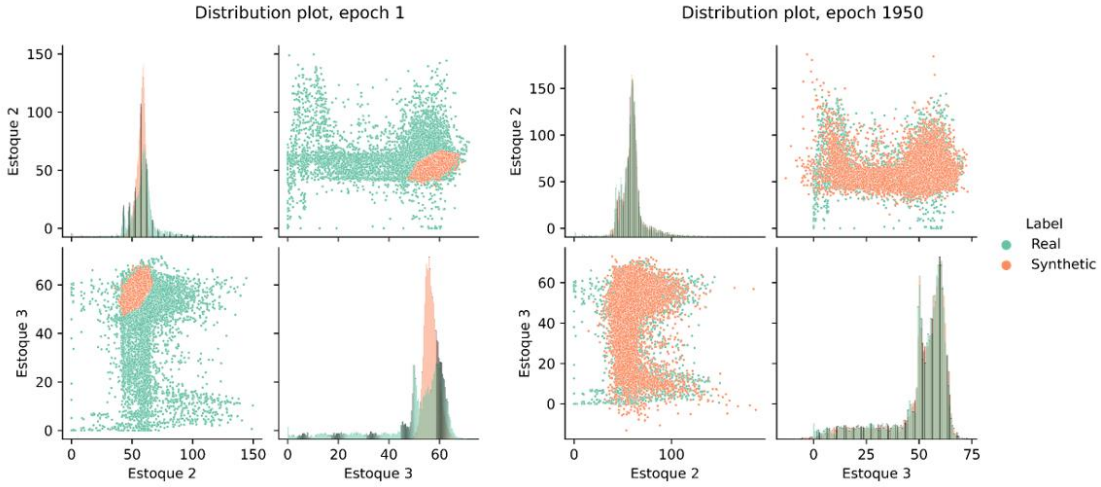

Figura 5.20 - Treinamento de *outputs* do Modelo 2 - Estoques 1 e 2

Ao realizar o teste de Equivalência para uma tolerância real de 5.0% pode-se afirmar que a diferença da classificação dos DTs e dos DCs não está estatisticamente dentro do intervalo estipulado (*p-value*: 0.996). Apesar disso, é necessária uma diferença mínima de 9.98% (*p-value:* 0.04961), o que implica em uma validação considerada "Muito Forte". O Poder do Teste resultou em 0.0%, indicando que a probabilidade de cometer o erro do tipo II é de 100%. Da mesma maneira que o primeiro objeto de estudo, o teste apresenta uma confiança de 95.0%, pois os dados foram classificados baseados na média e desvio padrão ao mesmo tempo. Se as duas métricas fossem validadas através dos testes da literatura, poderia ser realizado um teste *2-sample* T^2 *de Hotteling* que manteria a confiança em 95.0% ao analisar as médias do conjunto dos dois dados, porém não avaliaria o desvio padrão. Caso contrário, poderia ser realizado dois testes para analisar as médias das duas métricas de saída e dois testes para analisar os desvios

padrão dos dois conjuntos de dados. Dessa forma, a confiança do teste cairia para 81.45% (0.95^4).

Ainda, foi realizado o treinamento com as cGANs e o Teste de Equivalência de forma individual para cada um dos Estoques com o mesmo número de dados na Fase de Treinamento e na Fase de Teste para a avaliação das duas métricas de uma só vez. Para o Estoque 2, foram necessárias 950 épocas para atingir uma $A_{C_{final}}$ equivalente à 96.2% dos dados inseridos, enquanto o Estoque 3 atingiu as condições de parada em 1200 épocas com uma $A_{C_{final}}$ de 96.7%. O Discriminador julgou os DTs em 47.67% para o Estoque 2 e 47.55% para o Estoque 3, enquanto os DCs foram julgados, para o Estoque 2 e 3 em 48.37% e 49.53%, respectivamente.

Os dados foram submetidos ao teste de Equivalência e, para o Estoque 2, pode-se afirmar que a diferença na classificação dos DTs e DCs está, estatisticamente, dentro do intervalo de tolerância requisitado (*p-value*: 0.007). O Poder do Teste para o Estoque 2 foi de 100.0%, indicando que a probabilidade de cometer o erro do tipo II é de 0.0%. Para o Estoque 3, não foi possível afirmar que a diferença entre as classificações está dentro da tolerância real de 5.0% (*p-value*: 0.244). Nesse caso, a probabilidade de cometer o erro do tipo II é de 1.93% o que implica no Poder do Teste de 98.07%. Apesar de não ser validado com 5.0% de tolerância real, o algoritmo indica que o diferença mínima necessária é de 6.42%. Dessa maneira, a validação de ambas as métricas de forma individual também estão classificadas na faixa de validação "Muito Forte". Se optar pela validação individual de cada métrica através da abordagem, a confiança é maior que a de dois testes de média e dois de desvio padrão, porém menor que a validação conjunta, caindo para 90.25%. A Tabela 5.3 mostra o resumo da validação do modelo, apresentando o número de épocas necessária, a $A_{C_{final}}$, o número de dados no DT e DC e a faixa de validação. O resultado dos testes apresentados pelo Python® também podem ser encontrados no Apêndice H.

Tabela 5.3 - Resumo da validação Modelo 2 - Confecção de roupas

	Épocas	$A_{C_{final}}$	**DT**	**DC**	**Validado (5.0%)**
Estoque 2 e 3	1950	95.8%	15000	6519	9.98% - Muito forte
Estoque 2	950	96.2%	15000	6519	✓ - Muito forte
Estoque 3	1200	96.7%	15000	6519	6.42% - Muito forte

5.2.3. Modelo 3 – Hospital

5.2.3.1. Descrição do modelo

O último sistema estudado é um hospital de grande porte da cidade de São Paulo (HSP) com atendimento a diversas especialidades. O pronto atendimento atende cerca de 400 pessoas/dia e apresenta uma variedade de fluxos, onde o paciente pode fazer consultas, exames de coleta (sangue, urina), exames de diagnóstico de imagens (raio-X, ultrassom, etc.), cirurgias e internação. Além disso, a equipe é composta por enfermeiros, técnicos de enfermagem, técnicos de raio-X e ultrassom, entre outros, médicos de várias especialidades e recepcionistas que trabalham em turnos.

Os pacientes chegam ao HSP e podem ser considerados de emergência ou pacientes em condições normais. Se o paciente for de emergência, ele é transportado pela enfermeira de emergência para a avaliação médica e, em paralelo, ocorre o seu registro. Caso contrário, se considerados normais, eles retiram o número de uma senha e aguardam para serem triados. Ao ser triado pela enfermeira de triagem, o paciente pode seguir três fluxos: emergência, ortopedia ou o fluxo normal. Na triagem, o paciente é classificado dentre cinco cores: vermelho, laranja, amarelo, verde e azul. Ao ser diagnosticado com um procedimento de ortopedia, o paciente recebe uma avaliação médica e seu registro é feito ao mesmo tempo da avaliação por um acompanhante. Após a avaliação, dependendo da gravidade do problema, ele é submetido a uma cirurgia com a equipe médica, ou apenas um procedimento ortopédico, onde um técnico em ortopedia o realiza. Se o paciente é considerado de emergência após a triagem, ele segue o mesmo caminho de pacientes que já chegam em estado de urgência. Por fim, se o paciente apresenta uma classificação mais branda, ele aguarda o seu registro e o atendimento para a avaliação médica.

Após esse ponto, o fluxo passa a ser comum para os três tipos de pacientes. O paciente realiza até cinco procedimentos que podem ocorrer em paralelo ou quando o recurso e local estiverem liberados para o uso. Esses cinco procedimentos são: espera pelo médico especialista que não está no pronto atendimento ou está em uma outra atividade, estabilização, medicação, espera pela internação ou a própria internação.

Se o paciente espera pelo médico, ele o aguarda e depois passa pela consulta. Se for para estabilização, ele é estabilizado. Se é necessária uma medicação, o técnico de enfermagem a realiza e o paciente pode ainda receber uma segunda medicação, fazer algum tipo de coleta de material para exame, realizar um exame de raio-X, ultrassom, tomografia, eletrocardiograma, eco cardiograma ou realizar e esperar um procedimento externo. Da mesma maneira, se o paciente precisa realizar mais de um desses procedimentos, ele é feito na medida em que os recursos vão sendo liberados. Se ele precisa de internação, ele aguarda a sua liberação e depois é internado. Caso a internação ocorra primeiro que os outros procedimentos

necessários, os exames, coleta, medicação e consulta são realizados no próprio leito. Por fim, o paciente é liberado e sai do sistema. O tempo de internação não é considerado escopo do projeto. A Figura 5.21 apresenta a tela do modelo computacional do HSP construído no *software* FlexSim®.

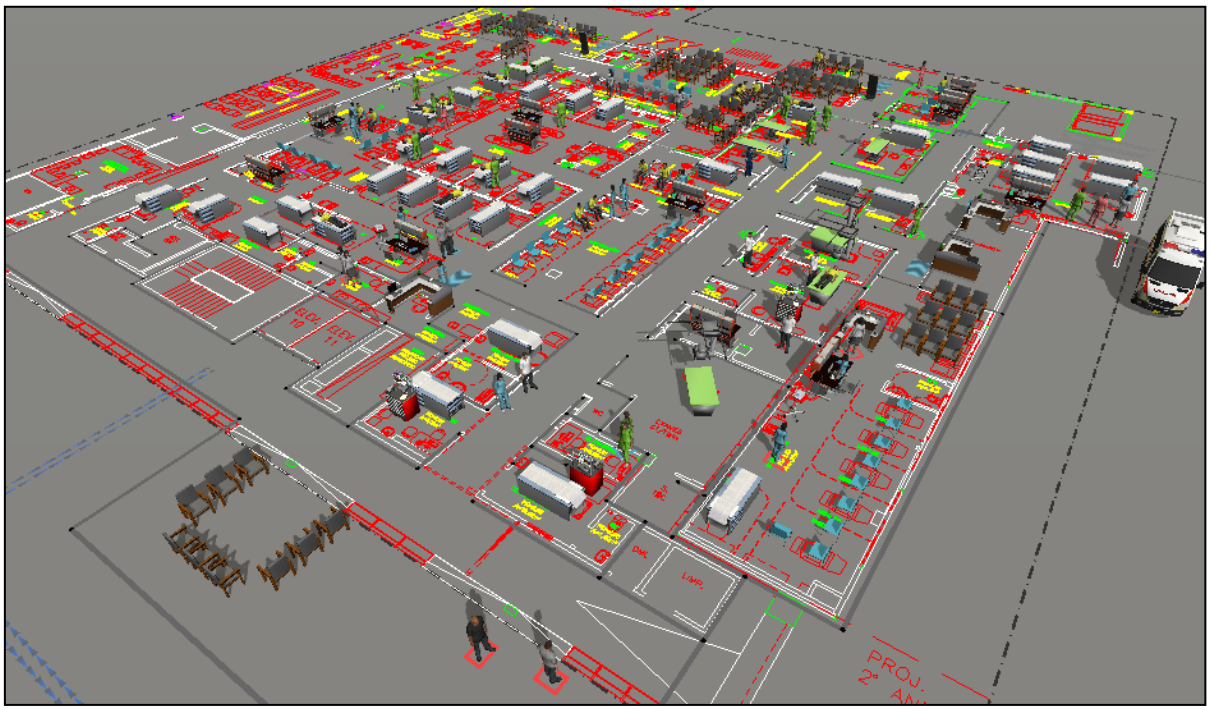

Figura 5.21 - Modelo 3 - Hospital

5.2.3.2. Validação do modelo – Paciente classificação verde

O modelo foi validado de duas maneiras. A primeira delas, consiste em validar através do Tempo Porta Triagem (TPT), Tempo Porta Médico (TPM) e Tempo de Permanência (LOS) dos pacientes de classificação verde no hospital. Ou seja, o modelo foi validado considerando o tempo em que o paciente chega ao hospital até ser triado, o tempo em que o paciente chega ao hospital até ser atendido pelo médico e o tempo que leva desde a chegada até receber alta.

Os dados do sistema real foram coletados através do sistema do próprio hospital que controla os tempos através de registros feitos pela equipe. Para a validação e inserção dos dados nas cGANs, optou-se por utilizar os DS para o treinamento e os DR para a Fase de Teste. Do sistema real, foram coletados dados de 6684 pacientes, enquanto a simulação forneceu 12456 dados. Para atingir as condições finais de parada e treinamento do modelo, foram necessárias 600 épocas, com a $A_{C_{final}}$ equivalente à 96.6% dos dados inseridos. O discriminador das cGANs julgou os DTs em $D(x)$ = 48.67%, enquanto os DCs foram classificados como iguais em 34.96%. A Figura 5.22 mostra o treinamento dos dados para a validação dos pacientes do tipo verde.

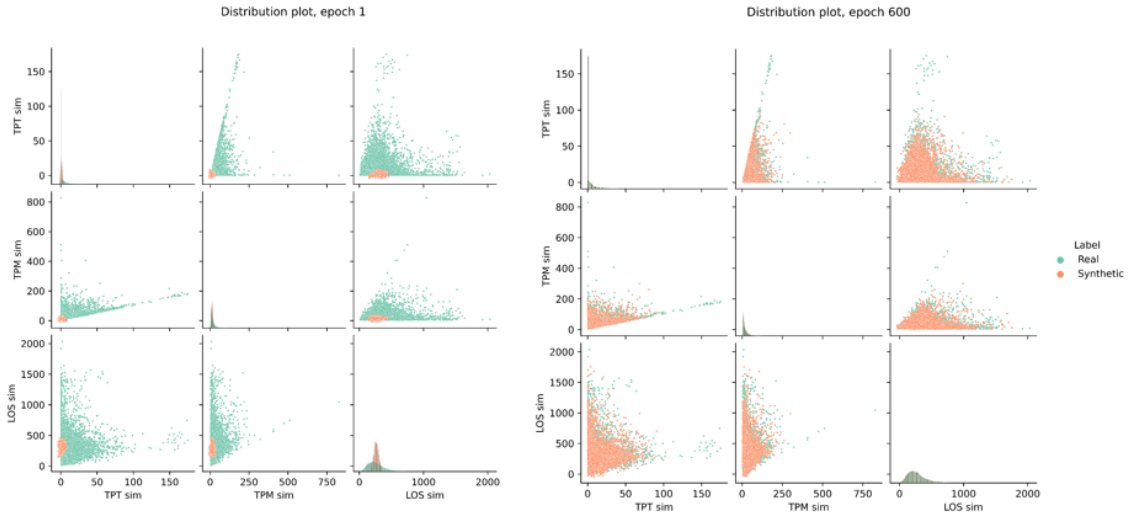

Figura 5.22 - Treinamento de *outputs* do Modelo 3 - TPT, TPM e LOS

De acordo com os resultados apresentados pelo teste de Equivalência e considerando uma tolerância real de 5.0% dos dados, não foi possível afirmar estatisticamente que a diferença da classificação das métricas coletadas do sistema real e do modelo simulado estão dentro do intervalo estabelecido (*p-value*: 1.000). O poder do teste atingido foi de 0.0%, indicando que a probabilidade de cometer o erro do tipo II é de 100.0%. Apesar disso, o modelo pode ser considerado validado se for estimado uma tolerância real mínima de 29.84% (*p-value*: 0.049) entre os conjuntos de dados. Dessa forma, pode-se dizer que a validação do modelo está em uma faixa considerada "Satisfatória".

Da mesma maneira que os outros objetos de estudo, a confiança do modelo é de 95.0% pois apenas é utilizado um teste estatístico envolvendo as médias e os desvios padrão das três métricas de validação. Se a validação for feita de forma individual, a confiança do modelo cai para 73.51% (0.95^6), pois são necessários seis testes estatísticos (três para avaliar as médias e três para avaliar os desvios).

Com o intuito de comparação, foi realizada a validação de cada uma das métricas de forma separada. A mesma quantidade de dados foi imputada em cada um dos treinamentos das três métricas e manteve-se o conjunto de dados tanto para a primeira quanto a segunda fase.

Para o TPT, as cGANs não conseguiram atingir uma $A_{C_{final}}$ mínima de 95.0% considerando 10000 épocas, porém, foi possível atingir uma $A_{C_{final}}$ equivalente à 93.6% em 2300 épocas. O Discriminador julgou os DTs em $D(x)$ = 53.59% e os DCs em 52.72%. Após o teste de Equivalência, pode-se afirmar que a diferença entre a classificação dos dados está estatisticamente dentro da tolerância real estipulada de 5.0% (*p-value*: 0.029), classificando em

uma validação “Muito Forte”. O Poder do Teste é de 100.0%, indicando que a probabilidade β é de 0.0%.

Em relação ao TPM, foram necessárias 300 épocas atingindo uma $A_{C_{final}}$ de 95.7%. Os DTs foram julgados em 49.12%, enquanto os DCs atingiram o valor de 50.18%. De acordo com os resultados, a diferença entre a classificação dos dois conjuntos de dados está, estatisticamente, dentro da tolerância (*p-value*: 0.029). A validação, para essa métrica, é uma validação “Muito Forte”, onde o Poder do Teste é de 99.9%.

Finalmente, para a validação do LOS, foram necessárias 700 épocas com uma $A_{C_{final}}$ de 98.4%. O Discriminador avaliou os dados da primeira fase em 45.69% e os dados da segunda em 38.44%. Os resultados mostram que não é possível afirmar que a diferença da classificação dos DS e os dados reais estão dentro do intervalo de tolerância estipulado (*p-value*: 1.000). A probabilidade de cometer o erro do Tipo II é de 100%, enquanto o Poder do Teste é de 0.0%. Como não se pode validar o LOS de forma individual considerando uma tolerância real de 5.0%, os testes foram realizados e, a tolerância mínima necessária é de 16.96% (*p-value*: 0.4961). Isso implica em uma validação “Forte” para o modelo. Se os três testes forem realizados de forma independentes, a confiança deles cai para 85.74%. A Tabela 5.4 mostra o resumo da validação das três métricas em conjunto e individuais. Além disso, os resultados completos também podem ser encontrados no Apêndice H.

Tabela 5.4 - Resumo da validação Modelo 3 - Paciente verde

	Épocas	$A_{C_{final}}$	**DT**	**DC**	**Validado (5.0%)**
TPT, TPM, LOS	600	96.6%	12456	6684	29.84% - Satisfatório
TPT	2300	93.6%	12456	6684	✓ - Muito forte
TPM	300	95.7%	12456	6684	✓ - Muito forte
LOS	700	98.4%	12456	6684	16.96% - Forte

5.2.3.2. Validação condicional – Tipo de Paciente

A segunda validação do modelo foi através dos LOS dos cinco tipos de pacientes. Conforme mostrado na seção 4.1.2, o algoritmo consegue detectar e treinar os dados se eles apresentam dados condicionais. Como o hospital tem em seus registros a hora em que o paciente chegou (coleta através de uma senha) e o momento da sua alta, os dados para a validação foram coletados levando em consideração cada tipo de paciente.

Os dados inseridos para as cGANs e avaliados para o treinamento foram os dados simulados, sendo um total de 91882. Como a validação é condicional, o paciente do tipo 1 apresentou 1200 dados, o do tipo 2 tem 12456, o do tipo 3 tem uma amostra de tamanho 47787,

enquanto os pacientes do tipo 4 e 5 apresentaram dados de tamanho amostral de 29201 e 1238, respectivamente. Para a segunda fase, o número da amostra foi de 67336 DCs, onde o tipo 1 tem um tamanho amostral de 218, o tipo 2 tem 6684 dados, enquanto os tipos 3, 4 e 5 apresentam uma amostra de tamanho 34692, 24771 e 971, respectivamente. O treinamento necessitou de 250 épocas para atingir uma $A_{C_{final}}$ de 98.4%, julgando os DTs em 51.99% e os DCs em 51.70%. A Figura 5.23 apresenta o treinamento dos dados do LOS para os cinco tipos de pacientes sendo treinados simultaneamente.

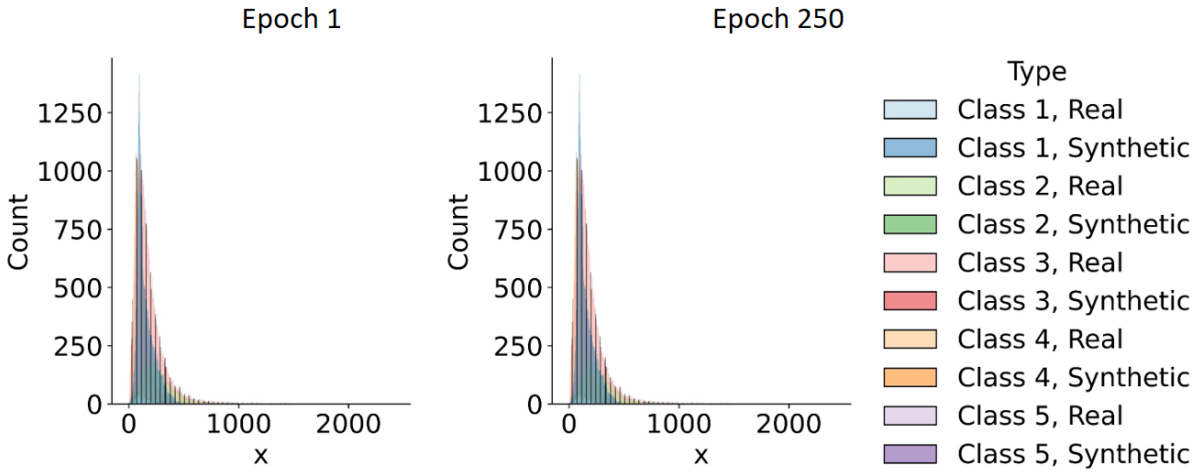

Figura 5.23 - Treinamento de outputs do Modelo 3 - LOS condicional

Para o teste de Equivalência, pode-se afirmar que o modelo foi validado estatisticamente ao considerar 5.0% de tolerância real entre as amostras (*p-value*: o.000), apresentando um Poder de Teste de 100.0%. Dessa maneira, pode-se concluir que o modelo de simulação apresenta uma validação “Muito Forte”.

Além da validação feita simultânea utilizando os atributos para separar os dados dos tipos de pacientes, o LOS de cada um foi validado separadamente. Assim, foram treinados cinco conjuntos de dados. A Tabela 5.5 mostra o resumo do número de Épocas, a $A_{C_{final}}$ dos modelos, a quantidade de dados na Fase de treinamento e Teste e se o modelo foi validado considerando a tolerância real de 5.0%. Ainda, apresenta a classificação da validação de cada um dos tipos de pacientes. Em todos os casos, os DS foram inseridos na fase de Treinamento e os DR foram para a fase de Teste.

Além das informações apresentadas na Tabela 5.4 a respeito do teste, vale ressaltar que o Discriminador julgou os DTs em 54.42%, 45.69%, 54.52%, 47.22% e 45.56% para os Pacientes 1, 2, 3, 4 e 5 respectivamente. Quando os DCs foram submetidos para a classificação dos dados, o discriminador julgou os dados para o paciente 1 em 56.42%, o 2 em 38.44%, o 3

em 55.19%, o 4 em 37.91% e finalmente o conjunto dos dados para o paciente 5 em 58.19%. De acordo com os resultados, pode-se afirmar que o Poder do Teste para a comparação dos dados do paciente 1 é de 92.50% (β = 7.50%), para o paciente do tipo 2 é de 0.0% (β = 100.0%), para o tipo 3 é de 100.0% (β = 0.0%), enquanto o Poder para os pacientes do tipo 4 e 5 são 0.0% (β = 100.0%) e 0.0% (β = 100.0%).

Tabela 5.5 - Resumo da validação Modelo 3 – LOS

	Épocas	$A_{C_{final}}$	**DT**	**DC**	**Validado (5.0%)**
Condicional	250	98.4%	91882	67336	✓ - Muito forte
Paciente 1	550	99.0%	1200	218	16.04% - Forte
Paciente 2	700	98.4%	12456	6684	16.96% - Forte
Paciente 3	250	98.2%	47787	34692	✓ - Muito forte
Paciente 4	650	98.6%	29201	24771	20.04% - Satisfatória
Paciente 5	3300	98.1%	1238	971	32.26% - Satisfatória

Como mostra a Tabela 5.4, o modelo foi válido para os dados condicionais e o LOS para o paciente do tipo 3, considerando uma tolerância de 5.0%. Para o LOS dos pacientes do tipo 1 e 2, a diferença da classificação dos dados não está estatisticamente dentro do intervalo estabelecido. É necessária uma tolerância real mínima de 16.04% para o paciente do tipo 1 (*p-value*: 0.04984) e 16.96% para o paciente do tipo 2 (*p-value*: 0.04961), considerando uma validação "Forte". Já para os pacientes do tipo 4 e 5, é necessário, pelo menos, uma tolerância real de 20.04% *(p-value*: 0.04898) e 32.26% (*p-value*: 0.04965), respectivamente, sendo uma validação "Satisfatória". Vale ressaltar que, quando se utiliza dados condicionais o discriminador consegue julgar os dados mais facilmente, separando uma distribuição da outra. O mesmo comportamento foi mostrado na validação dos pacientes quando utilizado distribuições condicionais e as distribuições validadas isoladamente.

Finalmente, da mesma forma que os outros modelos, ao utilizar o método das cGANs, a confiança do teste é de 95.0%. Se for considerado a validação para o LOS em cada tipo de paciente utilizando as cGANS, a confiança cai para 77.38% (0.95^5). Ainda, se for realizado o teste de média e desvio padrão para cada uma das classificações do paciente, a confiança do modelo cai para 59.87% (0.95^{10}), uma vez que é necessário realizar um teste de média e outro de desvio padrão para cada conjunto de dados, totalizando 10 testes de hipóteses. Vale ainda ressaltar que, ao validar cada tipo individualmente, o conjunto de dados do paciente do tipo 5 necessita de 13.2 vezes mais iterações para conseguir uma validação em uma classificação inferior do que ao julgar todos os dados conjuntamente.

6. CONCLUSÕES

As conclusões da tese estão divididas em quatro seções. A primeira seção mostra a síntese dos resultados e as limitações desta pesquisa. Na sequência são apresentadas as sugestões para os trabalhos futuros e por fim, são realizadas as considerações finais.

6.1. Síntese dos resultados

A tese teve como objetivo propor um método de validação de modelos computacionais de SED para avaliar uma ou mais métricas de saída, aumentando a confiabilidade dos testes e, ao mesmo tempo, considerando uma tolerância para a comparação dos dados simulados com os dados reais. O método proposto foi dividido em duas fases. A primeira, denominada "Fase de Treinamento", tem como objetivo o treinamento dos dados através das cGANs e a segunda, "Fase de Teste", visa discriminar os dados treinados e os dados que se quer comparar. Além disso, na segunda fase, os dois conjuntos de dados são submetidos a um Teste de Equivalência. O teste analisa estatisticamente se a diferença entre o julgamento dos DTs e DCs estão dentro da faixa de tolerância determinada pelo modelador.

As GANs foram utilizadas pois, sendo uma técnica de inteligência artificial, conseguem gerar amostras sintéticas que se adaptam a qualquer conjunto de dados. Mas especificamente, foram utilizadas as cGANs, ou seja, GANs que utilizam dados condicionais, pois assim é possível validar dados de modelos que apresentam diferentes tipos de entidades, como peças, pacientes ou recursos. O algoritmo foi programado utilizando as bibliotecas de IA do Python®. Vale ainda lembrar que todos os testes para a confirmação do método proposto foram feitos através do *Amazon Web Services* (AWS), o que permitiu que os experimentos fossem rodados em computação em nuvem, em processadores mais eficientes e simultâneos.

No geral, na "Fase de Treinamento", o modelador escolhe o conjunto de dados que devem ser inputados no algoritmo, podendo ser os DRs ou os DSs. As cGANs realizam o treinamento e, a cada 50 épocas, verificam através da técnica do *k-NN* se os dados sintéticos gerados pelo Gerador $G(x)$ são, no mínimo, 95.0% iguais aos que foram inseridos. Além disso, o Discriminador $D(x)$ avalia se os dados gerados estão entre 45.0% e 55.0%, pois, se essa condição for atingida, significa que $D(x)$ não sabe distinguir os dados inseridos daqueles gerados pelas cGANs. Na Fase de Teste, o segundo conjunto de dados é submetido ao julgamento do Discriminador e o TEDDP é realizado, considerando ou modelo válido ou não dentro da faixa de tolerância real.

Como os testes estatísticos estão propensos a erros e, para validar o método proposto e também verificar o Poder do Teste, foram realizados experimentos em distribuições contínuas, discretas, condicionais e em um modelo de SED. Testes com diferentes sementes aleatórias para a distribuição e diferentes número de amostras tanto para os DTs quanto para os DCs foram executados, totalizando 10 ciclos para cada um dos experimentos. A partir dos testes, as curvas com o Poder do Teste para a tolerância real de 5.0%, 10.0% e 20.0% foram geradas. Os resultados mostraram que é mais eficiente o uso do conjunto de dados que apresenta uma amostra maior na Fase de Teste enquanto o conjunto de tamanho amostral menor na Fase de Treinamento. Além disso, quando se utiliza os dados de tamanho amostral maior na primeira fase, a confiança do Poder do Teste aumenta, apresentando intervalos de confiança menores. Dessa maneira, apesar do crescente avanço na coleta de dados com a Indústria 4.0, sugere-se o uso dos dados simulados na primeira etapa, pois é mais fácil gerar dados através do simulador do que a coleta de dados reais.

Os resultados ainda mostram a diferença na acurácia dos acertos quando é avaliado uma métrica ou mais em conjunto. Assim, quanto mais métricas são avaliadas de uma só vez, maior deve ser a quantidade de dados inseridos no treinamento das cGANs. Por outro lado, se imputar mais dados confere um resultado mais assertivo na validação, é necessário mais tempo computacional para a realização do treinamento.

Ainda, segundo a literatura, não é possível afirmar se um modelo está totalmente válido ou inválido. Assim, sugere-se classificar a validação em faixas que mostram o quão válido o modelo é. Dessa maneira, foi proposto uma classificação de acordo com a tolerância mínima necessária para se considerar o modelo válido. Ela foi dividia em seis faixas: Muito Forte, Forte, Satisfatória, Marginal, Deficiente e Insatisfatória.

De acordo com os testes realizados, é possível concluir que o método é eficiente quando os dados do modelo real e do modelo simulado estão disponíveis e com, no mínimo 1000 dados. Com a finalidade de aplicar a metodologia em casos reais, três modelos de simulação foram submetidos à validação, sendo dois deles de manufatura e um na área da saúde. O primeiro modelo foi validado com apenas uma métrica de saída, enquanto o segundo apresentou dois outputs que foram avaliados simultaneamente. O terceiro foi dividido em duas validações, sendo a primeira uma validação de três métricas e a segunda uma avaliação condicional de cinco tipos de pacientes. Além disso, para o segundo e o terceiro modelo, houve a validação das métricas individuais. Como os modelos já haviam sido validados pelos métodos apresentados na literatura em outros estudos, assume-se que os testes de validação apresentaram sucesso ao serem realizados.

6.2. Limitações da pesquisa

A pesquisa apresenta algumas limitações que devem ser levadas em consideração. Para o treinamento dos dados, admite-se que eles já foram preparados e que não há dados faltantes ou *outliers*. Isso é necessário pois o algoritmo não consegue detectar esses tipos de falhas, o que pode levar ao treinamento incorreto.

Os parâmetros utilizados para o treinamento das cGANs foram escolhidos de duas formas. Alguns deles foram de acordo com o que a literatura recomenda, enquanto outros foram submetidos à teste. Assim, os parâmetros submetidos a testes apresentam valores que mais se adequaram e fizeram com que as cGANs se estabilizassem mais rápidas no treinamento.

Outro fator limitante na pesquisa é a faixa de tolerância em que os dados podem se encontrar. Para a pesquisa admitiu-se que essa faixa fosse simétrica, ou seja, que a classificação dos dados pode estar deslocada tanto para a direita, quanto para a esquerda. Isso significa que os DCs podem ser 5.0% positivo ou 5.0% negativo em relação aos DTs. Também, em relação à faixa de tolerância, não foi considerada a possibilidade de o intervalo da validação ser diferente dependendo da métrica escolhida. Dessa forma, se mais de uma métrica for avaliada, deve-se utilizar aquela que mais restringe o modelo.

Por fim, os três modelos utilizados para a validação com as cGANs fazem parte de outros estudos de simulação nos quais já foram validados com técnicas tradicionais presentes na literatura. Assim, admite-se que todos eles foram construídos utilizando métodos de validação já consagrados, e que as etapas de modelagem conceitual, coleta de dados, construção do modelo computacional e verificação foram realizadas de forma correta.

6.3. Recomendações para trabalhos futuros

Foram identificadas as seguintes oportunidades e sugestões para trabalhos futuros:

- Recomenda-se o teste, avaliação e otimização dos parâmetros utilizados no algoritmo de treinamento das cGANs que não estão indicados na literatura. Esses parâmetros podem gerar tempo de treinamento menor, ou seja, as condições de parada e estabilização das cGANs podem ser alcançadas mais rápidas e com menos esforço computacional.
- Recomenda-se ainda realizar o Teste de Equivalência onde as tolerâncias podem ser diferentes para cada uma das métricas avaliadas. Em especial, testar o uso assimétrico de tolerâncias, pois sistemas podem requerer que os dados sejam aceitáveis em somente um nível de tolerância.

- Na validação do método proposto e avaliação do Poder do Teste, foi realizado o treinamento de distribuições com, no máximo, 10000 dados. Dessa maneira, sugere-se, para trabalhos futuros, o treinamento das mesmas distribuições aplicadas na tese e em outras para um número maior de dados na primeira e segunda fase. Com esses testes, pode-se chegar a números ótimos tanto para a Fase de Treinamento como para a Fase de Comparação, minimizando o *trade-off* entre tempo computacional e confiança dos resultados devido ao tamanho amostral.
- Testes com *outputs* onde as métricas podem ser contínuas e discretas devem ser realizados, pois os resultados encontrados mostraram que, muitas vezes, as distribuições discretas não conseguiram atingir as condições em 10000 iterações.
- Treinar os dados inseridos nas cGANs assumindo que a $A_{C_{final}}$ mínima dos dados gerados pelo Gerador seja de 90.0% e avaliar os impactos nas curvas de Poder do Teste.
- Gerar um software com a proposta da validação proposta no trabalho;
- Por fim, sugere-se o treinamento dos dados com variações de GANs, como as DCGAN (Deep *Convolutional Generative Adversarial Network*), WGAN (*Wasserstein Generative Adversarial Network*), BigGAN (*Big Generative Adversarial Network*), entre outras e avaliar o comportamento do Discriminador ao julgar os dados.

6.4. Considerações finais

O método proposto mostrou que as cGANs foram eficientes para treinar dados e realizar a comparação de dados do sistema real com dados simulados. Foi possível, através do uso de IA, Testes Estatísticos, computação em nuvem e modelos de simulação mostrar e desenvolver uma técnica que permite a validação de modelos de SED. A metodologia propôs e se mostrou eficiente ao considerar uma margem de erro entre os dois conjuntos de dados comparados e ainda mostrou ser melhor que testes estatísticos convencionais presentes na literatura. Isso é possível pois a avaliação das métricas é feita de forma conjunta, o que aumenta a confiança do teste e diminui a probabilidade de dizer que um modelo é válido sendo que na verdade não é, ou seja, diminuindo a probabilidade de cometer o erro do tipo II que é considerado o mais grave em um processo de validação.

APÊNDICE A – Revisão Sistemática da Literatura

O Apêndice A apresenta na integra os resultados encontrados da Revisão Sistemática da Literatura para a validação computacional de modelos de SED. Parte desses resultados já foram apesentados na seção 1.2 da tese de forma resumida.

A.1. Validação de modelos computacionais

Validade é considerada como o grau em que um modelo representa fielmente sua contraparte do sistema (ZEIGLER e NUTARO, 2016). Sua avaliação é de suma importância para projetos de simulação, visto que, por natureza, o processo de construção de um modelo é propenso a erros, gerados no processo de tradução do sistema real em um modelo conceitual que será, mais tarde, convertido em um modelo computacional (HARREL *et al.*, 2012).

Nesse sentido, processos de verificação e validação (V&V) foram desenvolvidos para minimizar esses erros e fazer modelos úteis para atender às finalidades para as quais foram criados (LEAL *et al.*, 2011). Enquanto a verificação do modelo visa garantir que as lógicas do modelo computacional e sua implementação estejam corretas, a validação do modelo tem como objetivo a comprovação de que um modelo computacional possui uma faixa satisfatória de precisão consistente com o propósito pretendido sobre o domínio da aplicabilidade planejada (SARGENT, GOLDSMAN e YAACUB, 2016).

Ou seja, os processos de validação têm como objetivo a quantificação da precisão do modelo, comparando os resultados da simulação com resultados experimentais ou operacionais no mundo real (THACKER *et al.*, 2004), determinando se o modelo representa adequadamente o sistema para o propósito desejado (LAW, 2015; SARGENT e BALCI, 2017). Com isso, mesmo que não se possa garantir que um modelo seja 100% válido, a validação é parte crucialmente importante da construção de confiança nos resultados derivados de um modelo de simulação, sendo considerada indispensável para estabelecer a credibilidade dos modelos desenvolvidos (KIESLING, LÜTHI e KHAYARI, 2005; CHWIF, MUNIZ e SHIMADA, 2008; OLSEN e RAUNAK, 2015).

Apesar da reconhecida importância da etapa de V&V, Brade e Lehmann (2002) destacam que, infelizmente, a maior parte do processo é incompleto e fragmentário. Ainda segundo os autores, por muitas vezes os resultados de V&V não são bem estruturados, nem bem documentados, deixando lacunas e falta de evidências de como foram realizados. Para que esse problema possa ser resolvido, a literatura apresenta uma vasta lista de técnicas para se utilizar nessa etapa. Porém, a aplicação das técnicas de V&V ao longo do ciclo de vida de um

projeto de simulação pode ser demorada e dispendiosa (CHWIF, MUNIZ E SHIMADA, 2008; SARGENT, 2013). Na prática, sob pressão de tempo para concluir um estudo de simulação, a V&V e a sua documentação são sacrificadas primeiro (BALCI, 1994) e, como consequência, todas as etapas seguintes podem não responder à realidade (BANKS e CHWIF, 2011). Esse problema pode se tornar cada vez mais grave, na medida em que modelos de simulação têm se tornando mais complexos (HOLLMANN, CRISTIÁ e FRYDMAN, 2014).

Segundo Sargent (2013), uma das dificuldades para as atividades de validação se deve ao fato de que infelizmente não há um conjunto de testes específicos que possam ser facilmente aplicados para determinar a "correção" de um modelo. Além disso, não existe algoritmo para determinar quais técnicas ou procedimentos usar (SARGENT e BALCI, 2017). De acordo com Tsioptsias, Tako e Robinson (2016), a falta de validação perfeita ou única nos modelos leva aos modeladores a realizar um trade-off de validade. Não existe um padrão de quais variáveis do modelo selecionar para o processo de V&V, uma vez que, segundo Raunak e Olsen (2014), cada elemento da simulação é um candidato a validação.

A.1.1. Revisão Sistemática da Literatura

Com o objetivo de verificar o estado da arte da validação computacional em projetos de SED, foi realizada uma Revisão Sistemática da Literatura (RSL). A RSL é um método que tem como objetivo integrar a pesquisa empírica para criar generalizações a respeito de um determinado assunto (BIOLCHINI *et al.*, 2007). Ela não deve ser interpretada como uma simples revisão da literatura, pois oferece apoio da pesquisa para a seleção, avaliação e análise dos estudos escolhidos (COOK *et al.*, 1997). Biolchini *et al.* (2007), Kitchenhan e Charters (2007) e Denyer e Tranfield (2008) afirmam que a RSL utiliza a literatura já desenvolvida com o intuito de responder perguntas através de três etapas definidas: planejamento, execução e descrição dos resultados. Dessa forma, essas perguntas devem ser respondidas através da seleção de estudos que se adequem a critérios de elegibilidade pré-estabelecido (MOHER *et al.*, 2015). As análises dos resultados obtidos pela RSL podem ser qualitativas ou quantitativas, contribuindo para a resposta de alguma pergunta definida na fase de planejamento (COOK *et al.*, 1997; TRANFIELD, DENYER e SMART, 2003).

Além disso, quando se realiza uma RSL, todas as etapas envolvidas no método de pesquisa devem ser reproduzidas com o menor viés possível, pois assim é possível realizá-la novamente, gerando um resultando próximo ao original (DENYER e TRANFIELD, 2008). Apesar de apresentar grande vantagem em relação à revisão de literatura convencional, Hammersley (2001), Learmonth e Harding (2006) e Morrel (2008) afirmam que a RSL não tem

efeitos significativos para gestão e ciências sociais, pois as evidências são baseadas em características locais e familiaridades que podem afetar diretamente nos resultados encontrados. Diante disso, Sousa Júnior *et al.* (2019) afirmam que a RSL é apropriada para estudos de Engenharia de Produção, uma vez que os resultados encontrados nos estudos presentes na literatura podem responder a perguntas propostas baseado em determinado assunto que estão relacionadas a problemas globais.

A RSL deve ser reproduzida levando em consideração seis características: deve ser explícita, transparente, metódica, objetiva, padronizada, estruturada e reproduzível (BOOTH, PAPAIOANNOU e SUTTON, 2012). Diante disso, o trabalho seguiu os seguintes passos: (i) planejamento; (ii) busca/triagem; (iii) análise/síntese e apresentação dos resultados.

A.1.1.1. Planejamento

Para a primeira fase da RSL, foi definido um painel para analisar e verificar como a etapa de validação computacional está consolidada na literatura. Para isso, uma busca exploratória foi realizada nas bases de dados *Scopus*, *Science Direct* e *Web of Science*. Optou-se por essas bases pois são consideradas as maiores (ALRABGHI e TIWARIA, 2015) e que possuem uma alta taxa de citação (FRANCESCHINI; MAISANO; MASTROGIACOMO, 2014). O grupo para a pesquisa foi formado por cinco experientes pesquisadores na área de simulação, sendo dois deles professores que atuam na área há pelo menos 10 anos, 1 aluno de pós-doutorado e dois estudantes de doutorado. Uma pesquisa relacionando a validação computacional ("*computer model validation OR validation OR V&V*"), simulação (*AND* "*simulation*") e artigos teóricos (*AND* "*Literature review*" *OR* "*State of the art*" *OR* "*Overview*") foi realizada nas bases mencionadas a fim de encontrar artigos que consolidem as teorias já encontradas.

Como resultado, encontrou-se dois artigos relacionados ao tema. Banks e Chwif (2011) apresentam uma série de instruções e guias a respeito de como conduzir um projeto de simulação. Além disso, eles destacam a etapa de validação e verificação como uma importante fase. Tsiopptsias, Tako e Robinson (2016) fizeram uma revisão da literatura abordando as técnicas de validação existentes e os tipos de testes usados para avaliar um modelo computacional. Apesar de encontrar pesquisas a respeito da validação computacional, há uma gama de dicas, métodos e guias de como conduzir essa etapa de forma fragmentada na literatura. Assim, a proposta em desenvolver a RSL consiste em discutir a teoria já apresentada, mostrando os principais pontos de vista em comum e quais se divergem. Além disso, definir quais etapas devem ser seguidas ao se conduzir a validação de um modelo computacional e mostrar quais

gaps ainda estão presentes na literatura. Para atingir aos objetivos propostos, foram formuladas as perguntas de pesquisas (PP) para a extração dos dados. Dessa forma, foram elaboradas 11 PPs para investigar a maturidade e como a validação computacional é desenvolvida:

- PP1: Existe um passo a passo a ser seguido em um processo de validação?
- PP2: Quais técnicas de validação são apresentadas na literatura?
- PP3: Dentro as técnicas apresentadas, há uma divisão em categorias?
- PP4: Quais os parâmetros do modelo são utilizados para a validação?
- PP5: Como os dados de entrada afetam na validação do modelo?
- PP6: Até que ponto um modelo pode ser considerado válido?
- PP7: Existe uma tolerância para que modelos computacionais sejam validados?
- PP8: Se existe uma tolerância, qual é a tolerância ideal?
- PP9: Quais técnicas estatísticas devem ser utilizadas para a validação computacional?
- PP10: Quando utilizadas técnicas estatísticas, qual a relação dos erros do tipo I, II e III com a simulação?
- P11: Há um meio de quantificar e documentar a etapa de validação computacional?

A.1.1.2. Busca/Triagem

A pesquisa pelos artigos foi realizada no dia 15 de janeiro de 2020, nas bases de dados *Scopus, Science Direct* e *Web of Science*. Como mencionando anteriormente, escolheu-se essas bases por serem as bases que englobam o maior número de artigos. Três grupos de termos foram definidos para a busca e estão apresentados na Tabela A.1.

Tabela A.1- Grupos de termos para a busca nas bases de dados

Simulação	Validação	Teoria
Discrete Event Simulation	computer model validation model valid* validation and verification verification and validation V&V	state of art framework overview conduct process history literature review model

O primeiro grupo restringe a pesquisa apenas a SED. O segundo evidencia que os artigos devem apresentar, em seu escopo, como a validação computacional foi inserida no projeto. Por fim, o terceiro grupo faz referência aos artigos que expõem a parte teórica da validação computacional. As palavras de busca utilizadas poderiam estar contidas no título, resumo ou

palavras-chaves. Foi utilizado o conector booleano *OR* para palavras que estão entre um mesmo grupo e o conector *AND* para palavras que estão em grupos diferentes. Não houve restrição de data para a pesquisa dos artigos.

Foram encontrados 191 artigos na base de dados *Scopus,* 7 artigos na base de dados *Web of Science,* 74 na base *Science Direct* e 25 artigos relacionados a outras fontes, totalizando 297 artigos para análise. Foram detectados 56 artigos duplicados. Sendo assim, 241 artigos passaram para a etapa de leitura de título e resumo.

Nessa etapa, foi utilizada uma revisão por pares. A revisão por pares é feita através de dois especialistas no assunto que leem o título e resumo e decidem, separados, se o artigo deve seguir para a análise e leitura total ou é eliminado, a fim de evitar discordâncias (OCA *et al.*, 2015). Recomenda-se utilizar uma revisão por pares, pois os riscos de perder informações ficam em torno de 0% e 1%. Quando as revisões são feitas individualmente, o número de informações faltantes sobe para 8% (EDWARDS *et al.*, 2002). Os critérios para inclusão dos artigos foram: (i) usar o inglês ou português como escrita; (ii) apresentar teoria a respeito da validação de modelos computacionais; (iii) ser um artigo publicado em um periódico ou conferência e (iv) estar disponível para o download completo. Os artigos foram excluídos quando: (i) não estão aplicados a simulação a eventos discretos e (ii) quando apresentam evidências que utilizaram apenas a aplicação da validação sem uma explicação teórica do seu uso.

Para medir a concordância dos experts em relação a triagem dos artigos, foi realizado um teste de concordância de atributos. O *Kappa de Cohen* é medido (LANDIS e KOCH, 1977), sendo que valores próximos a 1 indicam concordância perfeita e, valores próximos à zero sugerem que não existe acordo e pode ser representado como se a concordância fosse realizada ao acaso (WATSON e PETRIE, 2010). Dessa forma, Landis e Koch (1977), Watson e Petrie (2010) e McHugh (2012) classificam o teste em seis categorias: “pobre” ($k < 0{,}00$); “fraco” ($0.00 \leq k \leq 0.20$); “razoável” ($0.21 \leq k \leq 0.40$); “moderado” ($0.41 \leq k \leq 0.60$); “considerável” ($0.61 \leq k \leq 0.80$) e “quase perfeito” ($k > 0.80$).

Os artigos foram divididos em subgrupos de tamanho 20 e disponibilizados para análise um a um, sucessivamente. Somente após a avaliação de um subgrupo pelos pares e a concordância dos grupos ser testada (*Kappa* acima de 0.60) o próximo subgrupo seria liberado. Se o teste apresentasse valores inferiores a 0.60, os especialistas deveriam identificar a causa das discordâncias e rever o grupo de artigos. O nível de concordância se manteve adequado durante todo o processo de avaliação, apresentando valores de *Kappa* abaixo de 0.60 somente para dois subgrupos. Como houve divergências nesses dois casos, a equipe se reuniu para discutir a respeito dos artigos e alinhar os pontos que se divergiram. Apesar desses dois grupos

apresentarem uma discordância, pode-se afirmar que o nível de concordância dos experts estava alinhado.

Por fim, realizou-se um teste de análise de concordância geral. Dessa forma, dentre os artigos analisados atingiu-se um grau de concordância de 93.4% (89.5% - 96.2%), sendo estatisticamente significativo (nível de confiança de 95.0%; *p-values* = 0.000). O *Kappa de Cohen*, para a concordância geral, foi de 0.812, indicando uma concordância quase perfeita entre os avaliadores.

Após a etapa de rastreamento e a identificação dos artigos que apresentaram os critérios de inclusão e exclusão, 190 artigos foram excluídos, passando 51 (21.1%) para a fase de leitura integral. Na fase de leitura integral, 22 artigos foram excluídos, sendo que 3 não foram encontrados disponíveis em sua versão total, 5 não apresentava aplicação da SED, 12 não responderam a nenhuma das PPs definidas na etapa de planejamento e 2 eram parte de um capítulo de livro. Dessa maneira, foram selecionados 29 artigos para a etapa de síntese qualitativa e análise dos dados. A Figura A.1 apresenta como foi conduzida a RSL.

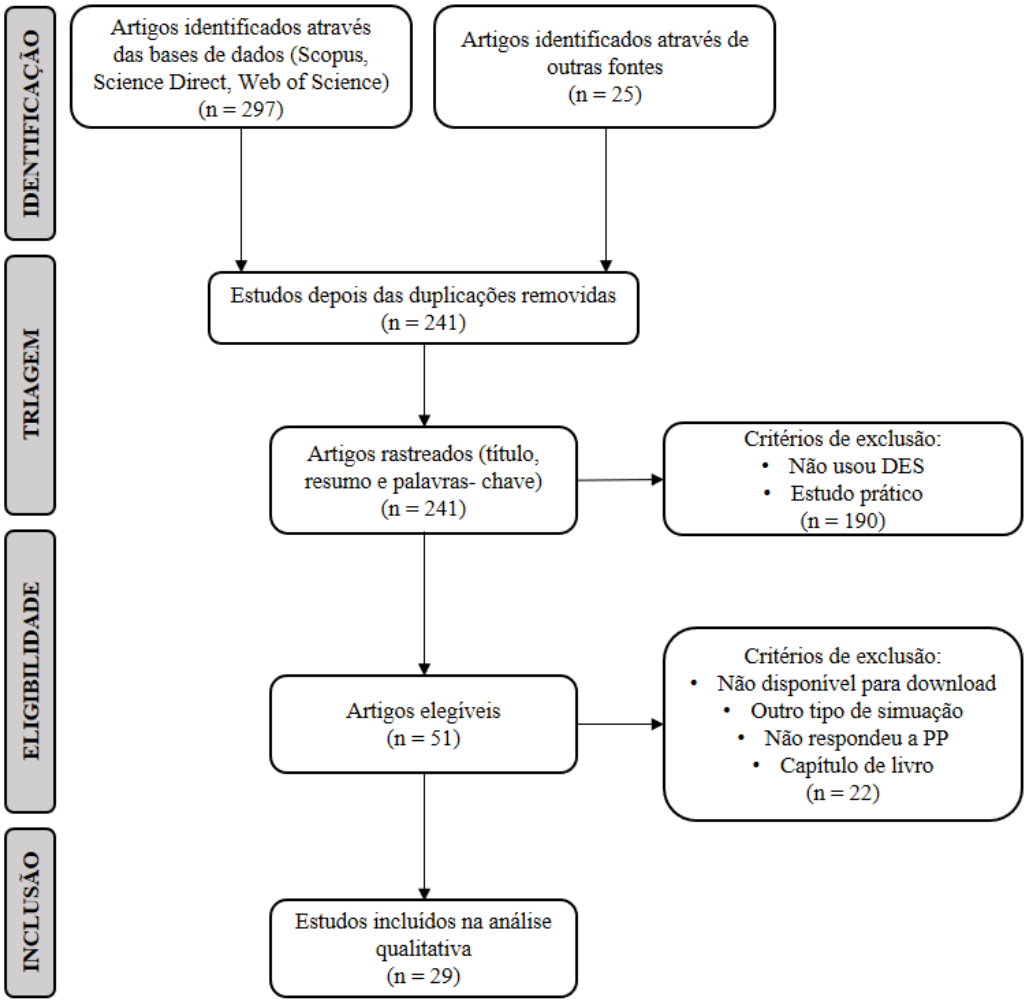

Figura A.1 - Fluxo de execução da RSL

A.1.1.3. Análises/síntese e apresentação dos resultados

Na fase de análise e síntese dos resultados, optou-se por utilizar o Microsoft Excel® para compilar os resultados encontrados. Assim, as informações a respeito de cada PP foram extraídas e armazenadas. Para a tabulação dos dados, foi criada uma tabela mostrando quais PPs foram respondidas, quais os autores que mais se dedicam a pesquisas em relação a validação computacional e a evolução ao longo dos anos. Adicionalmente, gráficos e tabelas foram incorporados ao estudo para facilitar a visualização dos dados reunidos. A apresentação dos resultados e as discussões a respeito da RSL estão disponíveis na seção A.2.2.

A.2.2. Resultados

A primeira análise dos dados compreende numa revisão bibliométrica a respeitos dos artigos encontrados na literatura. O primeiro registro, de acordo com a RSL, que apresenta teoria a respeito da validação computacional é apresentado no início dos anos 1980 (BALCI e SARGENT, 1981), apresentando uma metodologia para análise de custo e risco em validação estatística de modelos computacionais. De fato, Sargent e Balci (2017) afirmam que a validação computacional, antes dos anos 1970s, era pouco difundida. Churchman, Ackoff e Arnoff (1957) asseguravam que a simulação deveria ser feita em seis etapas, sendo que a quarta etapa compreendia em testar o modelo e a as soluções derivadas. Essa etapa que os autores descrevem, seria basicamente realizar a validação do modelo. Apesar dessa semelhança com o processo de validação atual, o primeiro trabalho que de fato apresentou algo que se assemelha com a validação é o de Fishman e Kiviar (1968) que discutiam estatísticas da simulação (SARGENT e BALCI, 2017). Com o surgimento do computador e as tecnologias que eles começaram a permitir, os estudos começaram a ser mais relevantes a partir dos anos de 1970s e 1980s (SARGENT e BALCI, 2017). Isso pode ser observado na Figura A.2 e comprovado, com uma tendência de crescimento estatisticamente significativa (*p-value* = 0.02).

Dentre os 29 artigos analisados, 46 autores pesquisam sobre a validação computacional. Os autores que mais se destacam são: Sargent (autor de 9 artigos), Balci (autor de 6 artigos); Chwif e Robinson (autores de 3 artigos), seguidos por Goldsman, Olsen, Raunak e Yaacub, com dois artigos cada. Quanto ao tipo de publicação, 62.1% (18) dos artigos foram publicações de periódicos, enquanto os outros 37.9% (11) foram artigos encontrados em conferências. Dentre os *Journals*, destacam se o *Journal of Simulation* (4), seguidos pelo *Simulation* (3) e *European Journal of Operational Research* (2). Em relação aos artigos encontrados em conferências, a maioria (7) estão presentes na *Winter Simulation Conference* (WSC). Também

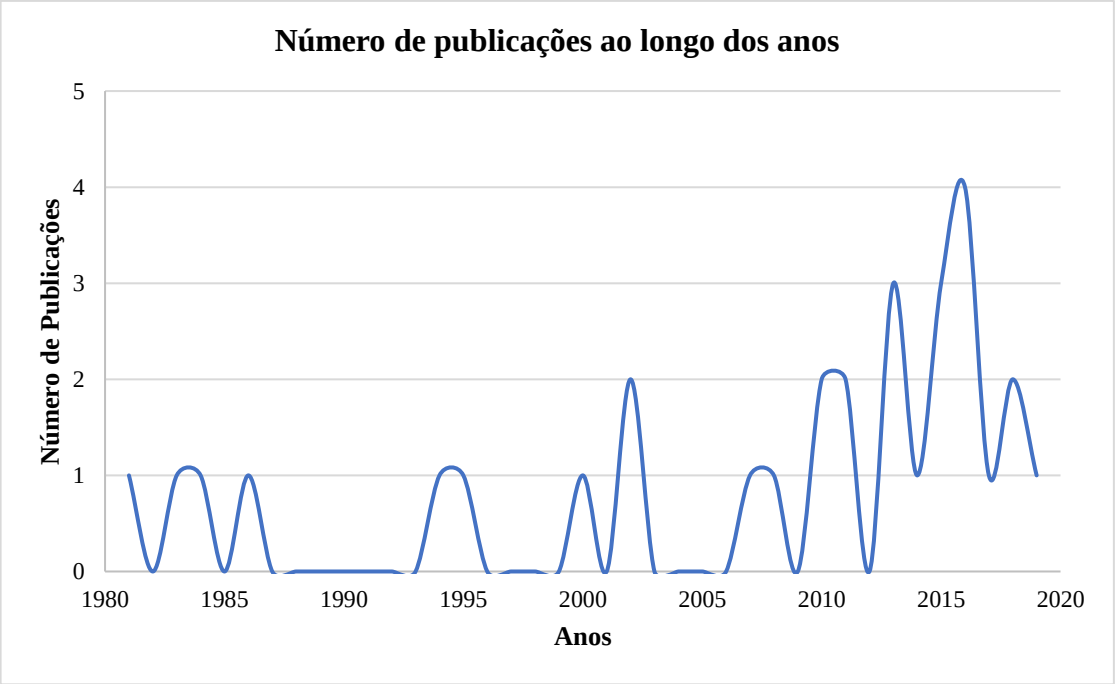

Figura A.2 - Número de publicações ao longo dos anos

foi medido o número de PPs respondidas em cada artigo. A PP mais respondida foi a PP2 (24), seguidas pelas PP1 (22) e PP5 (16). As menos respondidas são as PP8 (4) e PP7 (8), respectivamente. Os artigos mais completos, ou seja, aqueles que responderam mais perguntas são os estudos de Sargent (2013) e Sargent, Goldsman e Yaacub (2016). A Tabela 2.2 resume os resultados encontrados.

Dentre os 29 artigos analisados, 46 autores pesquisam sobre a validação computacional. Os autores que mais se destacam são: Sargent (autor de 9 artigos), Balci (autor de 6 artigos); Chwif e Robinson (autores de 3 artigos), seguidos por Goldsman, Olsen, Raunak e Yaacub, com dois artigos cada. Quanto ao tipo de publicação, 62,1% (18) dos artigos foram publicações de periódicos, enquanto os outros 37,9% (11) foram artigos encontrados em conferências. Dentre os *Journals*, destacam se o *Journal of Simulation* (4), seguidos pelo *Simulation* (3) e *European Journal of Operational Research* (2). Em relação aos artigos encontrados em conferências, a maioria (7) estão presentes na *Winter Simulation Conference* (WSC). Também foi medido o número de PPs respondidas em cada artigo. A PP mais respondida foi a PP2 (24), seguidas pelas PP1 (22) e PP5 (16). As menos respondidas são as PP8 (4) e PP7 (8), respectivamente. Os artigos mais completos, ou seja, aqueles que responderam mais perguntas são os estudos de Sargent (2013) e Sargent, Goldsman e Yaacub (2016). A Tabela A.2 resume os resultados encontrados.

Tabela A.2 - Resumo dos resultados encontrados

Autores	Ano	Fonte	Pergunta de Pesquisa										
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Akpan e Shanker	2019	Periódico		✓			✓						
Balci	1994	Conferência	✓				✓						✓
Balci	2010	Periódico	✓	✓	✓	✓	✓	✓			✓	✓	
Balci e Sargent	1981	Periódico	✓				✓		✓		✓	✓	✓
Balci e Sargent	1982	Periódico	✓				✓	✓				✓	✓
Balci e Sargent	1984	Periódico	✓	✓	✓			✓			✓	✓	
Banks e Chwif	2011	Periódico	✓	✓	✓		✓	✓		✓	✓		
Bayarri *et al.*	2007	Periódico	✓	✓		✓	✓	✓				✓	
Chwif *et al.*	2008	Periódico	✓	✓			✓		✓				
Foures *et al.*	2013	Conferência	✓	✓		✓			✓				
Kapoor e Shah	2016	Periódico	✓	✓	✓	✓	✓						
Kleijnen	1995	Periódico	✓			✓							
Leal *et al.*	2011	Periódico	✓	✓	✓	✓							
Olsen e Raunak	2015	Conferência				✓					✓	✓	
Pisaniello *et al.*	2018	Conferência	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Popovics *et al.*	2016	Periódico	✓	✓			✓		✓		✓	✓	
Raunak e Olsen	2014	Conferência		✓	✓		✓				✓	✓	
Robinson	2002	Periódico	✓	✓	✓	✓	✓	✓			✓		
Robinson e Brooks	2010	Periódico		✓				✓					✓
Sadoun	2000	Periódico		✓	✓	✓	✓						
Sargent	1986	Conferência	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Sargent	2013	Periódico	✓	✓		✓			✓		✓		
Sargent	2015	Periódico		✓	✓	✓							✓
Sargent e Balci	2017	Conferência		✓		✓		✓	✓	✓	✓	✓	
Sargent *et al.*	2015	Conferência	✓	✓									
Sargent *et al.*	2016	Conferência	✓	✓	✓	✓	✓	✓					✓
Tsioptsias *et al.*	2016	Conferência	✓	✓	✓								
Wang	2013	Conferência	✓	✓									
Yin e McKay	2018	Periódico	✓	✓				✓				✓	
Total			22	24	13	15	16	12	8	4	12	12	7

A.2.2.1. Metodologia para a condução da validação computacional

A PP1 visa identificar nos artigos metodologias ou passo a passo de condução de um projeto de validação. Observou-se que 23 estudos mostram ou pelo menos indicam como a validação computacional deve ser conduzida. Apesar do presente estudo focar na validação do modelo computacional, Robinson (2002), Balci (2010) e Leal *et al.* (2011) afirmam que a validação é de grande importância em toda a etapa da simulação. Para a validação computacional, ela deve ocorrer de forma cíclica (POPOVICS, PFEIFFER e MONOSTORI,

2016), contínua (SARGENT, 1986; KLEIJNEN, 1995; BALCI, 2010; WANG, 2013) e iterativa (TSIOPTSIAS; TAKO e ROBINSON, 2016) ao longo do desenvolvimento do modelo (FOURES; ALBERT e NKETSA, 2013). Parte do modelo deve ser construído, verificado e validado antes de prosseguir. Banks e Chwif (2011) sugerem que os modelos devem ser construídos dos detalhes mais simples para os mais complexos, facilitando assim a validação e evitando retrabalho (reduz o tempo de modelagem) quando os modelos se tornam complexos e os erros são mais difíceis de serem encontrados. Além disso, quando os erros são detectados mais cedo assegura-se uma melhor qualidade ao projeto (BALCI, 2010; FOURES; ALBERT e NKETSA, 2013). De fato, várias versões do modelo são realizadas antes de se chegar a versão final, uma vez que problemas podem aparecer ao longo de sua construção (SARGENT, 2013).

Dois ou três grupos de pessoas devem estar envolvidos na etapa de validação computacional: o time de modeladores, o usuário final do modelo e/o um time independente, sendo que os dois primeiros são obrigatórios. Quando a validação ocorre pelo time de modeladores, o próprio time decide a validade do modelo através de avaliações e testes que são realizados quando o modelo é construído, denominado de auto validação. Por outro lado, quando o modelo é validado pelo usuário final (co-validação), o usuário deve apresentar uma sinergia com a equipe de modeladores e determina o quão satisfatório o modelo é em cada etapa de sua construção (SARGENT, 2013; KAPOOR e SHAH, 2016; YIN e MCKAY, 2018).

Apesar do próprio usuário do modelo ou do time de modeladores poderem realizar a validação, indica-se a validação independente (SARGENT, 2013; KAPOOR e SHAH, 2016; SARGENT e BALCI, 2017; YIN e MCKAY, 2018). Um terceiro membro, que não faz parte da equipe e que também não é o cliente, realiza, de forma independente a validação. Esse time independente deve ter o conhecimento e o propósito da construção do modelo. A validação independente apresenta uma probabilidade muito maior de outras pessoas aceitarem o modelo válido e os resultados estarem corretos, evitando assim o erro do tipo II (erro β), relacionado a PP10. Recomenda-se a utilização desse tipo de validação quando se deseja a aceitação pública do modelo e quando o problema apresenta uma situação de risco com alto custo envolvido (SARGENT, 2013). Balci (2010) apresenta uma série de “regras de ouro” que devem ser seguidas na condução da validação e diz que a validação independente é mais significativa que os outros tipos. Assim, esse tipo de validação permite que os resultados finais não estejam enviesados. Outra maneira de evitar o viés na validação é a utilização de mais de uma pessoa, método e/ou técnica (BANKS e CHWIF, 2011).

Por fim, Yin e Mckay (2018) afirmam que existe um quarto tipo de validação. Essa validação é chamada de validação de *scoring*, onde um modelo pontual é comparado com um modelo simulado e assim é determinado a sua validade.

A validação pode ser caracterizada sob outro ponto de vista, sendo ela para sistemas reais (podem ser medidos) e sistemas hipotéticos (não podem ser medidos). No primeiro caso, a validação é baseada na comparação do modelo construído no computador e as medições realizadas no sistema físico. Nessa abordagem, deve-se explorar, de forma mais completa possível, as saídas dos modelos reais com os modelos simulados. Nos sistemas hipotéticos, o sistema modelado é criado apenas para um *design* e geralmente não é possível obter um satisfatório grau de confiança do modelo (SADOUN, 2000; SARGENT, 2013).

A literatura apresenta alguns *frameworks* e dicas para conduzir um projeto de validação (BALCI e SARGENT, 1983; BALCI, 1994; BAYARRI *et al.*, 2007; CHWIF; MUNIZ; SHIMADA, 2008; SARGENT, 2013; WANG 2013). Além disso, com a RSL, foi possível observar 10 etapas pelas quais os artigos sugerem que um projeto de validação deve passar:

1) **Planejar:** é necessário desenvolver e pensar em quais etapas e momentos a validação deverão ser inseridos no projeto, pois assim facilita na construção do projeto e o modelador sabe exatamente para que ponto ele deve seguir.
2) **Determinar o contexto de uso do projeto:** deve-se determinar o contexto em que o modelo será utilizado definindo todas as perguntas pelas quais ele irá responder. O modelo deverá ser validado para todas os critérios de avaliação. Além disso, a definição e contextualização do problema afeta de forma direta na credibilidade do modelo.
3) **Determinar a equipe de validação:** como explicado anteriormente, a validação pode ser realizada através do modelador, do usuário final do modelo, ou até mesmo por terceiros que não tem relação direta com os dois membros anteriores. Balci (1985) afirma que o ideal é apresentar um time que possui conhecimento suficiente do sistema real, modeladores, analistas de simulação e pessoas com ampla experiência no campo da simulação.
4) **Definir um intervalo de precisão para a validação associados com a incerteza:** Bayarri *et al.* (2007) dizem que é necessário, primeiramente, entender as incertezas associadas ao modelo, onde é necessário fazer uma lista com os insumos importantes e adotar incertezas a cada um deles. Além disso, os parâmetros mais importantes devem ser escolhidos para estruturar o processo de validação (relacionado diretamente com as PP4 e PP5). As variáveis de saída também apresentam um intervalo de aceitação (PP8),

pois o comportamento do modelo e os resultados simulados nunca são exatamente iguais ao mundo real (POPOVICS; PFEIFFER e MONOSTRI, 2016).

5) **Validar por mais de uma técnica e determinar as condições experimentais:** As condições do experimento devem ser definidas nas etapas iniciais do modelo e de preferência em comum acordo de toda a equipe envolvida no projeto. As condições do experimento devem produzir dados essenciais para a validação, além de definir o intervalo em que o modelo é aceito. Recomenda-se a utilização de mais de uma técnica de validação para que se possa ter credibilidade ao final da validação. Ainda, deve-se definir quais técnicas existentes na literatura (PP2) são aplicáveis para o projeto. Banks e Chwif (2011) afirmam que o modelo pode ser validado através da sua estabilidade, onde ele é rodado por um tempo e se torna estável a partir de determinado ponto ou a partir de sua instabilidade, como por exemplo o aumento da taxa de chegada de uma fila que não tem capacidade de suportá-la. Os testes devem ser realizados sempre que possível de forma objetiva, usando de estratégias e testes estatísticos, pois eles apresentam um apelo maior na validação do modelo (PP9 e PP10).
6) **Realizar primeiramente os testes mais rigorosos e definir quais procedimentos seguir:** os testes devem ser feitos ao longo de todo o processo de concepção do modelo computacional e para o projeto de simulação. Testes mais rigorosos devem ser feitos primeiros, pois quando os modelos se tornam longos e complexos é difícil de se realizar testes completos. Além disso, alguns critérios de avaliação podem falhar para alguns modelos e passar para outros. Assim, recomenda-se que os envolvidos na validação definam quais procedimentos serão seguidos na validação.
7) **Comparar dados reais com os dados gerados pelo modelo:** sempre que possível, ou seja, no caso da construção e simulação de modelos existentes, recomenda-se que pelo menos um teste seja feito comparando os dados gerados pelo sistema real, com os dados gerados pelo modelo simulado. Ainda, se possível, realizar os testes estatísticos para gerar maior credibilidade ao modelo.
8) **Revisão periódica do modelo:** se o modelo for utilizado por um longo período de tempo para auxílio na tomada de decisões, é necessário revisá-lo de tempos em tempos para que a sua credibilidade seja assegurada.
9) **Validação contínua e cíclica:** como mencionado anteriormente, o modelo deve ser validado passo a passo e continuamente no seu processo de construção para evitar futuros retrabalhos.

10) **Documentação:** todo o processo de validação do modelo computacional deve ser registrado a fim de mostrar a toda a equipe quais as premissas foram utilizadas e em quais condições o modelo é válido.

A Tabela A.3 apresenta o resumo de todas as etapas descritas anteriormente e quais artigos recomenda cada uma delas.

Tabela A.3 - Resumo das etapas de um projeto de validação

Artigo	Etapas									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Balci (1994)					✓				✓	
Balci (2010)		✓			✓	✓			✓	
Balci e Sargent (1981)			✓	✓						
Balci e Sargent (1983)				✓	✓		✓			
Balci e Sargent (1984)		✓		✓						
Banks e Chwif (2011)	✓				✓	✓			✓	✓
Bayarri *et al.* (2007)		✓		✓	✓	✓			✓	
Chwif; Muniz e Shimada (2008)							✓			
Foures; Albert e Nketsa (2013)					✓				✓	
Kapoor e Shah (2016)			✓							
Kleijnen (1995)									✓	
Leal *et al*. (2011)									✓	
Popovics; Pfeiffer e Monostori (2016)		✓	✓	✓				✓	✓	
Robinson (2002)									✓	
Robinson e Brooks (2010)			✓							
Sadoun (2000)							✓			
Sargent (1986)									✓	
Sargent (2013)		✓		✓	✓		✓	✓	✓	✓
Sargent; Goldsman e Yaacub (2016)			✓	✓	✓					
Tsioptsias; Tako e Robinson (2016)									✓	
Wang (2013)	✓				✓				✓	✓
Yin e McKay (2018)			✓							

2.2.2.2. Técnicas de validação e classificação

As PP2 e PP3 estão ligadas à questão das técnicas de validação que são descritas na literatura. Sargent e Balci (2017) afirmam que existem registro de mais de 75 técnicas. De fato, esse número foi comprovado na RSL, onde 78 técnicas foram identificadas. Os mesmos autores ainda asseguram que até os anos de 1967, a validação era dividida em três grandes frentes: racionalismo, empirismo e economia positiva. A partir dos anos 1970 é que as técnicas utilizadas atualmente começaram a surgir. Isso se deve ao fato do avanço computacional que

permitiu que gráficos, testes estatísticos e análise dos dados pudessem ser realizados mais rápidos e com mais precisão.

Dentre as técnicas que se destacam na literatura, tem-se: análise de sensibilidade; animação; análise de gráficos (histogramas, *boxplots*, gráficos de dispersão); testes estatísticos (testes t, intervalos de confiança, análise de média, desvio padrão e variância); validação face a face e teste de *Turing*. As três primeiras técnicas foram citadas em 11 artigos dos 24 que responderam a essa pergunta, enquanto as últimas três foram citadas por 10. Ainda destacam as técnicas de análise de condições extremas (7 artigos); análise de regressão; comparação do modelo simulado com modelos anteriores e comparação dos dados históricos com os dados simulados (mencionados em 6 artigos).

Um breve resumo de cada uma das técnicas mais relevantes é apresentado a seguir:

a) Análise sensitiva: é necessário realizar uma mudança nos valores das variáveis e parâmetros de entrada a fim de observar o efeito no comportamento das saídas do modelo (BALCI, 1994; SARGENT, 2013). Recomenda-se utilizar esse tipo de técnica para modelos hipotéticos, pois ainda não se tem dados reais para realizar comparações (KLEIJNEN, 1995).
b) Animação: o modelo é comparado apenas visualmente através de imagens gráficas que vão se formando ao longo da simulação (BALCI, 1994; KLEIJNEN, 1995). Além de envolver o cliente na etapa de validação, a animação, quando realizada em 3D, facilita a detecção de erros no modelo e na correspondência da simulação com o modelo real e os dados gerados (AKPAN e SHANKER, 2019).
c) Análise de gráficos: podem ser utilizados para comparar os resultados obtidos do modelo simulado com o sistema real. É considerado uma técnica subjetiva, pois vai de acordo com o entendimento do cliente e do modelador. Apesar disso, eles são práticos e podem ser utilizados em conjunto com outras técnicas de validação como o teste de *Turing* e a validação face a face (BALCI, 1994; SARGENT, 2013).
d) Testes estatísticos: os testes estatísticos são considerados os mais efetivos na validação de modelos computacionais pois, segundo Popovics, Pfeiffer e Monostori (2016), eles evitam a subjetividade que muitas técnicas apresentam. Devem ser utilizados para comparar diferença de médias, variâncias e distribuições de diferentes variáveis de saídas do modelo simulado com saídas do sistema real (BALCI, 1994; SARGENT, GOLDSMAN e YAACUB, 2016). Assim, os testes estatísticos são indicados para comparação de modelos reais e não para modelos hipotéticos que não apresentam dados históricos.

e) Validação face a face: há entrevistas com os experts e futuros usuários do modelo de simulação para comparar se o modelo comporta conforme o sistema real (SARGENT, 1986; BALCI, 1994; TSIOPTSIAS, TAKO e ROBINSON, 2016).
f) Teste de *Turing*: de acordo com Balci (1994), dois conjuntos de dados de saídas dos modelos (sistema real e simulado) são apresentados a um especialista para que esse possa identificá-los. Se o especialista não conseguir obter diferença entre os dados, é um indicativo de que o modelo pode ser validado.
g) Teste de condições extremas: o modelo é testado em condições extremas de cargas de trabalho e as saídas devem funcionar em qualquer combinação possível de dados. Apesar disso, quando se aumenta o fluxo do modelo, é esperado que o comportamento não seja o normal (BALCI, 1994; SARGENT, 2013).
h) Análise de regressão: usada para comprovar que pequenas alterações no modelo não criam efeitos significativos no resultado. Geralmente é utilizada modificando o conjunto de dados que já foram utilizados anteriormente (BALCI, 1994).
i) Comparação do modelo simulado com modelos anteriores: o resultado do modelo é comparado com modelos anteriores que já foram validados (SARGENT, 2013).
j) Comparação dos dados históricos com os dados simulados: são utilizados para a construção do modelo (entradas) e depois para a comparação das saídas (SARGENT, 2013). Geralmente são utilizados em conjunto com os testes estatísticos.

Na literatura é possível encontrar algumas divisões (em categorias) para as técnicas de validação. Não existe um consenso entre essa divisão, porém, muitos autores fazem a divisão entre técnicas subjetivas e objetivas (BALCI e SARGENT, 1981; BALCI e SARGENT, 1984; LEAL *et al.*, 2011; SARGENT, 2013; KAPPOR e SHAH, 2016; SARGENT, GOLDSMAN e YAACUB, 2016). As técnicas objetivas são preferidas, uma vez que utiliza de métodos matemáticos ou estatísticas para comparar modelos simulados com sistemas observáveis ou não. Assim, dados reais são requeridos para que procedimentos estatísticos possam ser realizados e determinar a validade do modelo computacional. Segundo Balci (1998) e Wang (2013), as técnicas objetivas fornecem evidências únicas, mas a sua aplicação requer experiência e conhecimento mais aprofundado pelos modeladores.

As técnicas subjetivas são baseadas no julgamento dos tomadores de decisão que conhecem o modelo que está sendo simulado, assim, dificilmente elas garantem os resultados obtidos nos experimentos da simulação (LEAL *et al.*, 2011). Essas técnicas são consideradas como única possibilidade quando nenhum método objetivo é possível de ser aplicado (WANG,

2013). Apesar de ambas as categorias apresentarem vantagens e desvantagens, recomenda-se a utilização das duas para a validação mais completa.

Outras divisões ainda podem ser encontradas na literatura. Paisaniello Jr. *et al.* (2018) dividem as técnicas em três categorias: qualitativa, quantitativa informal e quantitativa formal. A primeira, o modelador procura saber se o modelo simulado comporta de maneira qualitativa com o mundo real e não se baseia em indicadores reais. A segunda categoria compara os resultados do modelo simulado como o modelo real. Apesar disso, os dados coletados do sistema real são pequenas amostras obtidas através de entrevistas com especialistas do processo em um determinado momento. A categoria quantitativa formal contempla de técnicas estatísticas para validar todos os indicadores possíveis do sistema.

Por fim, Balci (1994) divide as técnicas em seis avaliações de credibilidade: informal, estática, dinâmica, simbólica, restritiva e formal. As informais são aquelas subjetivas. As técnicas estáticas estão ligadas à avaliação do modelo em relação ao seu código e não requer a sua execução. Para as técnicas dinâmicas, é necessário a execução do modelo para avaliar as suas saídas. As técnicas simbólicas se preocupam em avaliar o comportamento do modelo diante a sua execução. As restritivas são empregadas para avaliar a exatidão do modelo, enquanto as formais são as mais efetivas com provas matemáticas. O nível matemático e a complexidade da aplicação de cada uma crescem a partir da informal para a formal.

A.2.2.3. Parâmetros de validação e dados de entrada

A validação do modelo computacional só acontece de forma correta se os dados de entrada do modelo também forem confiáveis e refletirem a realidade do sistema real. Diante disso, as PP4 e PP5 visam responder quais parâmetros de validação são importantes e devem ser levados em consideração no modelo, como os dados de entrada se comportam e se relacionam com a validação computacional e como o modelo é considerado válido.

O modelo deve ser validado o quanto possível (BANKS e CHWIF, 2011). Apesar disso, autores como Sargent (2015), Sargent, Goldsman e Yaacub (2016) afirmam que não é possível identificar quais parâmetros devem ser validados, mas que o modelo deve ser desenvolvido com uma finalidade determinada e a sua validade deve ser definida com relação a esse propósito. Os autores ainda dizem que se o modelo pretende responder à várias questões, a sua validade deve ser em relação a cada uma delas. Além disso, quando o modelo é construído para responder a mais de uma pergunta simultaneamente, deve-se fazer uma comparação multivariada das saídas, para incorporar as correlações existentes entre elas (BALCI, 1994). Isso se dá ao fato de que um modelo pode ser válido para um conjunto de variáveis e condições experimentais e

inválido em outro (SARGENT, 2013). Assim, pode-se cometer o erro de validar um modelo baseado em apenas uma métrica e assumir premissas que não são verdades em relação ao todo.

Apesar da literatura não apresentar métricas consolidadas que devem ser validadas em todos os modelos computacionais, Raunak e Olsen (2014) propõem a divisão dos elementos validáveis em três aspectos: recursos, características solicitadas e fluxos de trabalho. Os recursos são taxa de utilização, disponibilidade e propriedades individuais de cada um deles; as características solicitadas são consideradas as chegadas das entidades no início ou em qualquer etapa intermediária do processo; os fluxos de trabalho estão ligados ao tempo de serviço, tempo de espera e características das filas. Apesar de não dividir dentro dessas categorias, autores como Sadoun (2000), Chwif, Muniz e Shimada (2008), Leal *et al.* (2011), Kapoor e Shah (2016), Popovics, Pfeiffer e Monostori (2016), Pisaniello Jr. *et al.* (2018) citam essas métricas como elementos validáveis utilizados em seus trabalhos. Se a escolha das métricas são variáveis aleatórias, indica-se a utilização de média e variância para a sua validação (SARGENT, 2015; SARGENT, GOLDSMAN e YAACUB, 2015; SARGENT, GOLDSMAN e YAACUB, 2016).

Para que o modelo seja realmente válido e reflita a realidade, os dados de entrada devem estar consistentes com o modelo simulado. As variáveis de entrada devem ser identificadas de acordo com a importância e influência no estado do sistema em relação ao escopo do projeto (BALCI, 1994). Uma maneira de identificar quais variáveis são importantes para o sistema é utilizar a análise de sensibilidade em cada uma delas (KLEIJNEN, 1995; ROBINSON e BROOKS, 2010). Assim, técnicas de validação também devem ser aplicada nos dados de entrada para que todos os seus inputs sejam validados (BALCI, 1994; BALCI, 2010). Apesar disso, Sargent (2013) e Sargent e Balci (2017) afirmam que a obtenção de dados de entrada que reflitam a realidade pode ser cara, difícil e demorada, sendo as principais razões para que a validação e a precisão do modelo falhem.

Diante disso, Chwif, Muniz e Shimada (2008) certificam que há três situações em que os dados de entrada afetam a validação do modelo, sendo elas: quando não há dados do sistema real disponível, existem dados apenas sobre a saída do sistema real e quando existem dados de entrada e saída. Essas situações aumentam a qualidade e a validade do modelo da primeira para a última. Os autores ainda dizem que as saídas devem apresentar uma relação linear com as entradas e, essas entradas devem ser independentes uma das outras.

A.2.2.4. Limites e precisão da validação

Nenhum modelo computacional é considerado 100% válido. De fato, Banks e Chwif (2011) afirmam que é possível invalidar um modelo de simulação, mas é impossível de validá-

lo na sua totalidade. Assim, as PP6, PP7 e PP8 buscam responder até que ponto um modelo é considerado válido e se existe uma tolerância para essa faixa de precisão.

Balci e Sargent (1981) e Balci (2010) dizem que a validade e a faixa de aceitação deve ser de acordo com o propósito em que o modelo é construído. Desta forma, Sargent, Goldsman e Yaacub (2015) e Sargent, Goldsman e Yaacub (2016) garantem que a precisão exigida em um modelo de simulação é especificada pelo intervalo que corresponde à diferença entre a variável de saída do sistema real e a variável de saída do sistema simulado, onde há um limite superior e um limite inferior.

Esse intervalo de confiança, também chamado de precisão deve ser definido pelo cliente ou usuário do modelo. Sargent, Goldsman e Yaacub (2016) indicam que a precisão deve ser definida antes do modelo ser construído ou em um estágio bem inicial. Quanto menor for o intervalo mais significativos são as informações que eles fornecem (BALCI, 1994; TSIOPTSIAS, TAKO e ROBINSON, 2016). Apesar disso, diminuir o intervalo de confiança gera um *trade-off* entre a validade do modelo e o seu custo. Quanto maior a confiança do modelo, mais caro ele se torna (SARGENT, 2013). Robinson (2002) assegura que outra forma de aumentar a confiança do modelo é utilizar mais de uma técnica de validação. Se alguma das técnicas utilizadas falharem, significa que o modelo é menos preciso. No entanto, o resultado da validação não é definido como uma variável binária e pode ser representado dentro de uma faixa de confiança de 0 a 100% (BALCI, 2003; LEAL *et al*., 2011; OLSEN e RAUNAK, 2015).

As zonas de tolerâncias podem ser definidas para os parâmetros escolhidos para a validação, sendo que eles podem ser diferentes de acordo com a sua relevância para o modelo (POPOVICS, PFEIFFER, e MONOSTORI, 2016). Apesar disso, Sadoun (2000) e Bayarri et al. (2007) afirmam que determinar a tolerância de cada métrica não é uma tarefa fácil. Existem muitas incertezas nos parâmetros de entrada do modelo que surgem com base em dados, opiniões de especialistas ou por problemas anteriores. Além disso, se as execuções são caras, poucos dados do sistema real podem estar disponíveis para a comparação com o sistema simulado. Esses dados do sistema real podem ser limitados e ruidosos fazendo com que o modelo computacional se afaste do sistema real. A fim de minimizar esse problema, Sargent (2015) criou um teste estatístico onde a tolerância do modelo simulado e real é levado em consideração.

A.2.2.5. Procedimentos estatísticos na validação computacional

Como mencionado anteriormente, existe uma alta variedade de técnicas para determinar se o modelo é válido ou não. Muitas dessas técnicas são subjetivas e vai da experiência do

modelador. Sargent (2013) sugere que as técnicas de intervalos de confiança e testes estatísticos sejam preferíveis na validação do modelo, pois elas fornecem decisões objetivas. Contudo, as vezes não é possível utilizá-las devido a violações nos pressupostos estatísticos ou devido a pequenas quantidades de dados. Dessa maneira, as PP9 e PP10 visam mostrar como os testes estatísticos se relacionam com a validação e como o erro envolvido nesses testes podem levar a um falso positivo ou um falso negativo.

Muitas abordagens de validação são feitas de acordo com a métrica escolhida e utilizando estatísticas univariadas, ou seja, utiliza-se testes paramétricos (*1 sample-t, 2 sample-t*, teste-F) ou testes não paramétricos (*1 sample Wilcoxon*, teste *Mann-Whitney*), dependendo da disponibilidade dos dados. Sargent (2013) afirma em seu estudo que, se as variáveis de interesse de validação forem aleatórias, os testes utilizados são para avaliar se a média ou variância das variáveis de saída do modelo simulado são iguais a do sistema real. Leal *et al.* (2011) propõem um fluxograma onde um passo a passo pode ser seguido para a comparação de intervalos de confiança através dos testes mencionados anteriormente e, de acordo com as propriedades dos dados analisados.

Apesar da literatura apresentar vários testes estatísticos de comparação de média e variância, conforme mostrado no estudo de Sargent e Balci (2017), Sargent (2015) indica que esses testes usam apenas um único ponto ao invés de um intervalo, ou consideram intervalos de forma indireta. Assim, os modeladores assumem que o sistema que está sendo modelado é um sistema observável e que os dados necessários para a validação podem ser obtidos e coletados. Dessa maneira, o autor criou um teste onde uma faixa de precisão ou tolerância é levada em consideração nos intervalos de confiança, podendo ser utilizado em três situações: (i) o tamanho de amostra é fixo e a faixa aceitável de precisão do modelo e o risco do construtor do modelo são especificados; (ii) o tamanho de amostra é fixo, porém a faixa aceitável de precisão é especificada por um limite inferior e superior e o risco do usuário do modelo é dado por uma tolerância nesse limite; (iii) o risco do criador do modelo e o risco do usuário são especificados com o limite superior e inferior, juntamente com a tolerância, e depois o tamanho de amostra é determinado. Sargent (2015) criou esse teste por acreditar que o modelo de simulação nunca é validado na sua totalidade e há faixas de tolerância em que o modelo pode estar contido e ainda ser considerado válido.

Outras técnicas univariadas ainda são encontradas na literatura para a validação do modelo computacional, como é o caso de regressão acompanhado de uma análise de sensibilidade (KLEIJNEN, 1995) e *Design of Experiment* (SARGENT, 2013).

Por fim, Balci (1994) e Sargent, Goldsman e Yaacub (2016) alegam que, se existe correlação entre as saídas do modelo que precisam ser validadas, as técnicas multivariadas devem ser utilizadas. Assim, intervalos de confiança simultâneos são construídos e mostram como as variáveis se comportam como um todo no modelo. Sargent e Balci (2017), em uma análise da validação nos últimos anos mostram que técnicas multivariadas como intervalos de confiança simultâneos, *teste 2 sample* T^2 *de Hotteling* e MANOVA podem ser encontrados na literatura. Apesar disso, o mais utilizado é o teste de Hotteling T^2 que, segundo Balci e Sargent (1983), testa a equivalência das médias de duas populações normais variadas. Nesse caso, se houver correlação entre as saídas do modelo, deve-se testar a sua independência, normalidade multivariada e igualdade das matrizes de variância-covariância (BALCI e SARGENT, 2013).

Balci e Sargent (1984) fazem uma comparação entre as técnicas univariadas e as multivariadas. Os autores dizem que, ao utilizar a primeira, é possível especificar um nível de confiança para cada uma das métricas escolhidas e controlar os comprimentos dos intervalos de precisão, dependendo da sua importância para o modelo. O mesmo não pode ser dito para as técnicas multivariadas, uma vez que o seu intervalo de confiança é comum para todas as saídas. Além disso, os autores mencionam que as técnicas estatísticas univariadas permitem um número de amostra diferente para cada saída, enquanto as multivariadas geralmente não. Para se selecionar uma técnica deve-se levar em consideração as premissas definidas, o tamanho do intervalo de confiança, a igualdade e/ou desigualdade no nível de confiança de cada métrica escolhida e o tamanho das amostras (BALCI e SARGENT, 1984). Assim, há um trade-off para a escolha do tamanho do intervalo de confiança, pois eles são afetados pelos níveis de confiança, variância das variáveis e pelo tamanho de amostra (SARGENT, 2013).

Independentemente da técnica estatística escolhida para verificar a validade do modelo, pode-se cometer três tipos de erros ao realizar o teste (BALCI e SARGENT, 1981; BALCI, 1994; BANKS e CHWIF, 2011): (i) Erro do tipo I; (ii) Erro do tipo II; (iii) Erro do tipo III. O erro (i) acontece quando um modelo válido é rejeitado; o (ii) acontece quando um modelo inválido é aceito e o erro (iii) compreende em solucionar um problema errado. Existe uma probabilidade associada a cada tipo de erro. Assim, o Erro do tipo I (probabilidade α) é chamado de risco do construtor do modelo e o erro do tipo II (probabilidade β) é chamado de risco do usuário do modelo (BALCI e SARGENT, 1981; BALCI, 1994; SARGENT e BALCI, 2017).

Quando se comete erro (i), o custo do modelo começa a crescer, uma vez que o modelo pode ser totalmente desperdiçado (se nunca for utilizado) ou aumentado de forma desnecessária (se o modelador continuar a construí-lo). Para o erro (ii), ações desastrosas podem acontecer, pois tomadas de decisões baseadas no modelo inválido podem levar ao colapso do modelo real

(BALCI e SARGENT, 1981; BALCI, 1994). Assim, o erro do tipo II é considerado o mais grave dentro de um projeto de simulação, pois as tomadas de decisões serão todas baseadas em um modelo que não representa a realidade. Balci e Sargent (1984) e Kleijnen (1995) garantem que é possível diminuir a probabilidade α e β aumentando o número de amostras tanto do modelo real como simulado. Apesar disso, os autores afirmam que há um custo associado a esse aumento, além de ser mais fácil o aumento da amostra no modelo simulado. Por fim, Balci e Sargent (1983) sugerem a criação de gráficos mostrando a relação entre o risco do modelo, a precisão aceitável da validação, o tamanho da amostra e os custos envolvidos na coleta de dados.

A.2.2.6. Documentação e quantificação da validação

A última questão lida com a fase final da validação de um modelo computacional. Dessa maneira, é necessário determinar o quanto o modelo é considerado válido e como foi realizado todo o processo de validação através da sua documentação.

Balci (2010) afirma que o modelo construído não deve ser definido como um modelo perfeitamente preciso ou totalmente inválido. O autor mostra que é possível construir faixas com níveis de aceitação como: excelente, muito bom, satisfatório, marginal, deficiente e insatisfatório. A mesma ideia é compartilhada por Tsioptsias, Tako e Robinson (2016) que dizem que faltam definições concretas para os níveis de validação.

Nesse sentido, Raunak e Olsen (2014) propõe uma metodologia para quantificar a validação de um modelo computacional. A primeira etapa visa identificar os aspectos e elementos do modelo de simulação que podem ser validados, sendo eles: recursos, fluxo de trabalho e características solicitadas. Esses elementos candidatos a validação pertencem a um conjunto denominado $\{va_k = \{et_1^k, et_2^k, ..., et_i^k\}\}$. Assim, et_i^k é um tipo de elemento validável em *k* aspectos. Os *k* aspectos variam de 1 a 3 e representam recursos, características solicitadas e fluxo de trabalho. Para um modelo de simulação, os modeladores podem definir cada tipo de elemento do aspecto va_k que devem ser validados. Assim, et_i^k representa os conjuntos de elementos que exigem validação, onde *i* é a quantidade de elementos que devem ser validados em cada aspecto. Se alguns dos aspectos não for escolhido para validação, como por exemplo, recursos, o seu et_i^k será vazio.

Como alguns ele elementos validáveis do modelo são mais importantes que outros, deve-se apresentar pesos diferentes (r_{ij}^k) ao serem validados. Deste modo, o modelador pode escolher uma escala numérica para determinar os pesos relativos de cada elemento considerado validável. Após essa identificação, deve-se determinar quais elementos são dados de entrada e

quais são informações observáveis de saída. Em seguida, deve-se determinar quais técnicas de validação $\left(v_{ijl}^{k}\right)$ podem ser aplicadas a cada um dos elementos e definir um peso relativo (w_l) a cada uma dessas técnicas. *l* corresponde ao número de técnicas de validação aplicáveis para o elemento *j* do tipo de elemento *i*. Em seu trabalho, Raunak e Olsen (2014) propõem um peso relativo para algumas das técnicas de validação. Ao realizar essa etapa, o modelador deve aplicar a todos os elementos as técnicas escolhidas e, se o modelo for válido para a técnica, atribuir o valor 1, caso contrário, o valor 0. Quando uma técnica de validação apresenta sucesso no elemento validável, aumenta-se a confiabilidade do modelo (OLSEN e RAUNAK, 2015). O último passo é calcular a cobertura de validação (*cv*), conforme mostra a Equação 18:

$$cv = \frac{\sum_{ijk} r_{ij}^{k} \frac{1}{\sum_{l} w_l} \sum_{l} w_l \left(v_{ijl}^{k}\right)}{\sum_{ijk} r_{ij}^{k}} \tag{18}$$

Raunak e Olsen (2014) define a cobertura de validação como a porcentagem das técnicas de validação aplicadas de forma ponderada com sucesso em cada elemento que também foi ponderado, variando de 0 a 1. Ainda afirmam que se ela atende ao limite pré-estabelecido do projeto, uma validação suficiente foi realizada no modelo de simulação. Dessa maneira, esse processo pode definir faixas de valores numéricos para os termos nominais definido por Balci (2010).

Para finalizar, Olsen e Raunak (2015) declaram que existe uma necessidade imediata de melhorar a documentação das atividades de validação realizadas nos modelos de simulação. Nesse sentido, relatórios devem ser redigidos com a finalidade de documentar as técnicas utilizadas, como foram empregadas e as estatísticas importantes (ROBINSON e BROOKS, 2010). Balci (2010) mostra que a documentação é importante pois ela pode ser revista na manutenção do modelo. Sargent (2013) sugere que a documentação deva incluir uma tabela sumarizando o que foi feito, quais elementos e técnica utilizados, o porquê de utilizar cada técnica, os resultados, conclusões e a confiança do resultado.

A.2.4. Desafios e direções futuras na validação computacional

Apesar de não ser um assunto relativamente novo, a validação ainda sofre muito em um projeto de simulação. Devido a agilidade dos projetos e ao custo que vai aumentando com mais aplicação de técnicas, essa etapa é uma das primeiras a serem sacrificadas (BALCI, 1994). Ainda, com o avanço de recursos computacionais, os modelos acabam se tornando cada vez mais complexos e muitas variáveis se tornam objetivos de resposta. Assim, é necessário dar

uma atenção especial a essa etapa, visto que ela é essencial para o sucesso do projeto. Baseado na RSL desenvolvida e nas lacunas encontradas, é possível citar quatro desafios encontrados: metodologia de validação estruturada; determinação nos limites de validação; escolha das melhores técnicas que se encaixam em um projeto e documentação da validação.

Muitos modeladores não sabem como realizar a validação de um modelo computacional ou realizam de forma parcial ou errônea. Além disso, os guias disponíveis na literatura não englobam todas as etapas encontradas na RSL. Portanto, o estudo resumiu os principais tópicos apresentados pelos autores e, para futuros projetos, recomenda-se utilizar e seguir todas as etapas descritas na seção A.2.2.1. Dessa forma, o modelador deve dar uma atenção especial para a validação cíclica do modelo, para que futuros retrabalhos possam ser evitados e, consequentemente, o modelo se torne mais rápido, confiável e mais barato como um todo.

Determinar os limites de validação também é um ponto chave que deve ser levado em consideração ao estruturar a validação computacional. É um consenso na literatura que cada métrica e variável de resposta do modelo deve ser validada. Apesar disso, há algumas medidas que o modelador deve levar em consideração. A primeira delas, que está ligada diretamente aos passos já citados, é em relação ao tamanho do intervalo de confiança. Esse intervalo pode ser igual para todas as métricas, ou cada uma pode ter um tamanho diferente, dependendo da sua importância para a resposta do problema. Se o intervalo de confiança for igual para todas as métricas e houver correlação em pelos menos duas das saídas, o uso de técnicas multivariadas apresenta um modelo mais relevante. Caso contrário, o uso de estatística univariadas são recomendadas.

Com a definição do limite de validação do modelo, que deve ser realizada logo no início do projeto, pode-se aplicar as técnicas de validação. Como observado, existem muitas técnicas na literatura e que podem ser utilizadas de acordo com o tipo de modelo construído ou de acordo com a sua funcionalidade. Sargent, Goldsman e Yaacoub (2016) recomendam o uso de técnicas estatísticas quando se tem um sistema observável e que é possível coletar dados desse sistema. O modelador deve levar em consideração o *trade-off* entre aumentar o número de amostras coletas do sistema real e o custo envolvido na realização do projeto. Recomenda-se também, se houver correlação entre as variáveis de saída e a precisão for a mesma para todas elas, o uso de estatísticas multivariadas como o teste T^2 de Hotteling, para que o comportamento do modelo possa ser testado simultaneamente.

Além das técnicas estatísticas que são as mais confiáveis para a comparação do sistema real com o modelo simulado, sugere-se o uso de outras técnicas objetivas que tem um peso maior para a validação, tais como: análise sensitiva, testes de condições extremas, análise de

regressão e comparação de dados históricos com o simulado. Técnicas subjetivas também podem ser aplicadas com a finalidade de aumentar a confiabilidade do modelo, como o teste de *Turing* e a análise de gráficos. Quando o modelo construído é de um sistema não observável, ou seja, quando não é possível obter os dados históricos (geralmente por ser uma projeção) recomenda-se somente a utilização de técnicas subjetivas, que vai de acordo com a experiência do modelador, sendo elas a validação face a face, animação gráfica e comparação com modelos semelhantes. Apesar da sugestão dessas últimas técnicas serem para sistemas não observáveis, também é possível utilizá-las nos outros sistemas, mas não são tão confiáveis como as técnicas objetivas.

Outro ponto que passa a ser crucial na validação é evitar que os erro do tipo I, II e III aconteçam. Se o erro do tipo II acontecer, passa a comprometer os resultados, pois ele considerado o mais grave para a simulação, uma vez que diz respeito em aceitar um modelo que na verdade está inválido. Assim, consequências drásticas podem ocorrer quando os tomadores de decisão aceitam um modelo com esse tipo de erro.

Por fim, é essencial que o processo de validação seja documentado, mostrando todas as etapas que foram seguidas, quais técnicas utilizadas, quais dessas técnicas obtiveram sucesso e quais não. Recomenda-se o cálculo da cobertura de validação, método proposto por Raunak e Olsen (2014), a fim de se obter, um valor e uma faixa em que o modelo pode ser considerado válido, sem se comportar como uma variável binária.

A.2.5. Conclusões da RSL

Essa seção teve como objetivo sintetizar os principais pontos de vista dos autores a respeito da validação computacional de modelos de SED. Dicas e regras são discutidas na literatura, como as apresentadas por Balci (2010), Banks e Chwif (2011) e Sargent (2013). Apesar disso, não se encontra na literatura uma revisão contendo os pontos de vista dos autores, mostrando os benefícios, passo a passo e as falhas contidas nessa etapa. Para isso, foi realizado uma RSL, onde onze perguntas de pesquisa em relação à condução, técnicas, parâmetros, limites de aceitação, técnicas estatísticas e documentação a respeito da validação foram respondidas. Não foi delimitado o tempo de busca dos artigos, visto que o tema é pouco discutido e, com o passar dos anos há uma evolução no seu estudo.

A primeira conclusão retirada é que a etapa de validação em um modelo de simulação, muitas vezes é sacrificada pelos modeladores, visto que pode consumir um tempo muito grande e assim, tornar o modelo mais caro. Sacrificando essa etapa, o modelo pode responder de forma errada as respostas do problema. Ainda é possível afirmar que a validação pode ser considerada

relativamente subjetiva em um projeto de simulação. Cada projeto é específico para determinada área e desenvolvido para responder a uma ou mais perguntas que podem variar. Assim, é difícil afirmar, previamente, quais parâmetros devem ser validados, quais técnicas são importantes e qual o tamanho da precisão de cada intervalo de acordo com o tipo de projeto. Apesar disso, foi possível, com o estudo, mostrar diretrizes de como conduzir um processo de validação e tornar o modelo o mais próximo da realidade possível. Em relação as perguntas propostas, pode-se concluir:

- Muitos autores discutem etapas pelo qual o modelador deve passar ao validar um modelo computacional, apesar disso, não existe um passo a passo consolidado, ou um fluxograma mostrando como deve se proceder. Foram identificadas 10 etapas, que podem ser seguidas.
- Muitas técnicas de validação são encontradas na literatura. Cabe ao modelador decidir quais as melhores para se aplicar no projeto. Quanto mais técnicas utilizadas, mais confiável o modelo se torna. Apesar disso, há o trade-off, já mencionando, em relação ao custo e tempo de desenvolvimento do modelo. Ainda, segundo Sargent, Goldsman e Yacoub (2016), é preferível o uso de técnicas objetivas quando se tem um modelo observável, visto que estas técnicas apresentam mais confiança para o modelo. Sugere-se, para a validação mais completa do sistema, o uso de mais de uma técnica, como análise sensitiva, teste de *Turing*, validação face a face, comparação gráfica e animação.
- Os dados de entrada do modelo são dados chave para o sucesso do modelo de validação. Antes de serem inseridos no modelo, o modelador também deve validá-los. Autores como Raunak e Olsen (2014) recomendam a divisão dos elementos validáveis em três aspectos: recursos, características solicitadas e fluxos de trabalho. Como mencionado anteriormente, cabe a cada modelador, de forma subjetiva, julgar quais parâmetros são necessários para a validação. Apesar disso, cada métrica que responde a uma pergunta do problema, deve ser validada.
- O tamanho do intervalo de precisão do modelo também é uma escolha subjetiva que deve ser discutida no início do projeto entre cliente e modelador. Cabe a essa equipe decidir se cada métrica deve apresentar o seu tamanho (testes univariados são indicados) ou um tamanho geral para todas elas (testes multivariados).
- Segundo a literatura, testes estatísticos são sempre preferíveis em relação as outras técnicas de validação, pois são mais confiáveis. Quando há correlação entre as saídas

das variáveis, recomenda-se o uso de testes multivariados. Ainda, é necessário ficar atento aos erros que podem ocorrer, mais especificamente, o erro do tipo II.

- O cálculo da cobertura de validação é algo que deve ser feito, para mostrar que o modelo apresenta uma faixa de validação e quais técnicas aplicadas obtiveram sucesso e quais fracassaram (RAUNAK e OLSEN, 2014). Outra medida que pode ser utilizada é uma faixa nominal, apresentada por Balci (2010): excelente, muito bom, satisfatório, marginal, deficiente e insatisfatório.
- A documentação do processo de validação computacional é um procedimento falho. Autores e modeladores deixam de especificar o que foi feito em cada modelo. Essa documentação deve ser realizada para que, se no futuro for necessário, possa ser revisado e sirva de comparação para outros modelos.

APÊNDICE B – Código de treinamento das GANs e TEDDP

Esse código permite a geração de modelos de entrada por meio de GANs. Os modelos exportados são compatíveis com o software FlexSim®.

1. Importa os módulos necessários para execução;
2. Declara a classe "*Data*", que inclui: Leitura de dados, Geração de dados teóricos, Preparação dos dados (limpeza, scale...), Guarda dados descritivos: tamanho amostral, número de variáveis para os dados de Treinamento;
3. Declara a classe "GAN", que inclui: Criação dos componentes da GAN, Treinamento, Avaliação do ajuste, plotar gráficos, Geração de dados sintéticos;
4. Declara classe "*Power*", que inclui: Cálculos de poder, tamanho amostral e acurácia máxima recomendada;
5. Declara funções "*Utils*": Funções para cálculo, conversão e obtenção de acurácia com o usuário Classificador *k-NN*, entre outros;
6. Função "*Main*": executa o pipeline desejado
7. Declara a classe "*Comapred Data*", que inclui: Leitura de dados, Geração de dados teóricos, Preparação dos dados (limpeza, scale...), Guarda dados descritivos: tamanho amostral, número de variáveis para os dados de Comparação;
8. Declara a classe "*Equivalence Test*", que inclui: Teste de Equivalência, Geração da tabela com o resumo dos resultados, Gráfico do Teste de Equivalência e Poder do Teste e Faixa de validação.

B.1. Importar Módulos

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.preprocessing import MinMaxScaler, RobustScaler
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
import tensorflow as tf
from tensorflow.keras.layers import Dense, LeakyReLU, Dropout, Input, Concatenate
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras import optimizers, losses, metrics
from tensorflow.keras.utils import plot_model
from numpy.random import choice
from sklearn.utils import shuffle
from scipy.stats import norm
from scipy.optimize import fmin
from prettytable import PrettyTable
import warnings
import math
import os
import imageio
```

```
import warnings
from collections import namedtuple
from IPython.display import clear_output
import xlsxwriter
from pathlib import Path
import time
from scipy import stats
import statsmodels.api as sm
from statsmodels.graphics.gofplots import qqplot_2samples
```

B.2. Classe *Data*

```
class Data():

    """
    This class comprises methods to read, generate and preparate real data.
    """
    def __init__(self):
        self.original_data = None

    # Read data from excel file
    def read_excel(self, excel_file_name = None, sample_size = None):
        df = pd.read_excel(excel_file_name)
        # If user wants to sample from the excel data
        if sample_size is not None and sample_size < df.shape[0]:
            self.original_data = df.sample(sample_size)
        # Else, we get and shuffle all data
        else:
            self.original_data = df.sample(frac=1)
        self._dataset_name = excel_file_name

    # Prepare data
    def prepare_data(self, scaler_estimator):
        # Get copy of original data
        self.real_data = self.original_data.copy()
        # Force columns to be string
        self.real_data = self.real_data.rename(columns=lambda x: str(x))
        # Dataframe may not contain a column named as "_"
        if '_' in self.real_data.columns:
            self.data.real_data.rename(columns={'_':'_*'}, inplace=True)
            print('\033[93m' + "Illegal column name: '_'. Name changed to '_*'" +
'\033[0m')
        # Remove object columns whose names don't start with "_"
        columns_to_drop = self.real_data.select_dtypes(include=['object']).columns
        columns_to_drop =
np.array(columns_to_drop[~columns_to_drop.str.startswith('_')])
        self.real_data.drop(columns=columns_to_drop, axis = 1, inplace=True)
        if len(columns_to_drop) > 0:
            print('\033[93m' + "The following columns were excluded for containing
categorical data:" + '\033[0m')
            for column in columns_to_drop:
                print(column)
            print('\033[93m' + "If these columns should be considered as data
classes, please start their names with '_'." + '\033[0m')
            print("")
        # Add at least one class column
        if len(self.real_data.columns[self.real_data.columns.str.startswith('_')])
== 0:
            self.real_data['_'] = 0
        # Handle missing data
        ## Report missing data
        n_dropped_observations = np.sum(self.real_data.isna().sum(axis=1) > 0)
        if n_dropped_observations > 0:
            print('\033[93m' + "Missing data by variable (%):")

print(round(100*self.real_data.isna().sum(axis=0)/self.real_data.shape[0],1))
            print("")
```

```
            print("Total dropped observations:")
            print(np.sum(self.real_data.isna().sum(axis=1) > 0))
            print("" + '\033[0m')
        ## Drop missing data
        self.real_data = self.real_data.dropna()
        # Get original class columns
        self._original_class_columns =
np.sort(self.real_data.columns[self.real_data.columns.str.startswith('_')])
        # Get original class data types
        self._original_class_dtypes = [self.real_data[col].dtype.kind for col in
self._original_class_columns]
        # Record unique values (for undumming)
        string_class_columns = [col for col, kind in
zip(self._original_class_columns, self._original_class_dtypes) if kind == 'O']

        unique_class_values = np.empty(len(string_class_columns),
dtype='str').tolist()
        for idx, col in enumerate(string_class_columns):
            unique_class_values[idx] =
np.sort(self.real_data[col].unique()).tolist()
        self._unique_class_values = unique_class_values
        # Describe real data
        print("Data summary:")
        print(self.real_data.describe())
        sns.pairplot(self.real_data)
        plt.show()
        # Get dummies
        self.real_data = pd.get_dummies(self.real_data,
prefix_sep='__',drop_first=True)
        #Reorganize data according to data type (classes, whose columns start with
"_", values)
        column_order =
np.concatenate((self.real_data.columns[self.real_data.columns.str.startswith('_')],

self.real_data.columns[~self.real_data.columns.str.startswith('_')]))
        self.real_data = self.real_data.loc[:,column_order]
        # Identify int columns (this identification will enable rounding them after
generating)
        int_columns = self.real_data.select_dtypes(include=['integer']).columns
        self._int_columns = [col for col in int_columns if not col.startswith("_")]
        # Scale data
        self.scaled_data = self.real_data.copy()
        self.scaled_data =
pd.DataFrame(scaler_estimator.fit_transform(self.scaled_data), columns =
self.real_data.columns)
        self._scaler = scaler_estimator
        # Record data characteristics
        self._sample_size = self.real_data.shape[0]
        self._columns = self.real_data.columns
        self._class_columns = self._columns[self._columns.str.startswith('_')]
        self._value_columns = self._columns[~self._columns.str.startswith('_')]
        self._scaled_labels = self.scaled_data.loc[:, self._class_columns]
        self._scaled_values = self.scaled_data.loc[:, self._value_columns]
        self._nvariables = len(self._value_columns)
        # PCA
        from sklearn.decomposition import PCA
        pca = PCA(n_components=self._nvariables)
        pca.fit(self.scaled_data.loc[:, self._value_columns])
        for n, ratio in enumerate(pca.explained_variance_ratio_):
            if sum(pca.explained_variance_ratio_[0:(n+1)]) >= 0.99:
                self._ncomponents = n + 1
                break
```

B.3. Classe GAN

```
class GAN():
```

```
    """
    This class comprises methods to build, train, evaluate, and export GANs.
    As well as methods to generate synthetic data and to plot them.

    """

    def __init__(self, data):

        # Assign data as attribute
        self.data = data
        # Create folder to save results
        self._output_path = f'{Path().absolute()}\\{time.strftime("%Y%m%d-
%H%M%S")}_{self.data._dataset_name}'
        os.mkdir(self._output_path)
        # Define latent space size
        # Rule recommended by this study: 3 * number of principal components that
explain >99% of data variation
        self._latent_space_size = 3*data._ncomponents
        # Define generator and discriminator networks
        # Rule recommended by this study:
        # density = 8 * number of principal components that explain >99% of data
variation
        self.generator = self._define_generator(density=8*data._ncomponents)
        self.discriminator =
self._define_discriminator(density=8*data._ncomponents)
        # Define GAN
        self.GAN = self._define_GAN(self.generator, self.discriminator)
        # Define standard noise
        # This noise will be used to generate synthetic data and, with them,
evaluate the generator during training
        # This is important to avoid the multiple testing problem
        self.standard_noise = self.generate_noise(sample_size=data._sample_size)
        # Create epoch counter attribute
        self.epoch = 0

    def _build_base_ann(
        self,
        name,
        input_shape,
        density,
        hidden_layers,
        output_shape,
        output_activation,
        ):

        # Define leaky relu
        lrelu = lambda x: tf.keras.activations.relu(x, alpha=0.2)
        # Define the input layer
        ## First input layer component: data classes
        input_class = Input(shape=(len(self.data._class_columns),), name =
"input_classes")
        ## Second input layer component: values
        input_values = Input(shape=(input_shape,), name = ("input_latent" if name
== "generator" else "input_data"))
        ## Concatenate input components
        inputs = Concatenate(axis=1, name='inputs')([input_class, input_values])
        # Create base network
        base = Dense(density, activation=lrelu, name = "input_layer")(inputs)
        for hidden_layer in range(hidden_layers):
            x = Dense(density, activation=lrelu, name =
f"hidden_layer_{hidden_layer+1}")(base)
            base = x
        if name == "generator":
            outputs = Dense(output_shape, activation='linear', name =
"output_layer")(base)
        else:
            outputs = Dense(output_shape, activation='sigmoid', name =
"output_layer")(base)
```

```
        model = Model(inputs=[input_class, input_values], outputs=outputs,
name=name)
        return model

    def _define_discriminator(
        self,
        density,
        hidden_layers = 2,
        alpha = 0.0001,
        beta = 0.5,
        beta2 = 0.999,
        epsilon = 1e-8
        ):

        # Build discriminator
        discriminator =
self._build_base_ann(name='discriminator',input_shape=self.data._nvariables,
density=density, hidden_layers=hidden_layers, output_shape=1,
                                    output_activation="sigmoid")
        # Define discriminator optimizer
        optimizer = optimizers.Adam(learning_rate=alpha, beta_1=beta, beta_2=beta2,
amsgrad=False, epsilon = epsilon)
        # Define binary accuracy
        binaryAccuracy = tf.keras.metrics.BinaryAccuracy(
            name='binary_accuracy', dtype=None, threshold=0.45)
        # Compiling discriminator
        discriminator.compile(loss="binary_crossentropy", optimizer=optimizer,
metrics = [binaryAccuracy])
        discriminator.trainable = False
        # Print discriminator summary and save plotted model
        print("")
        print("="*100)
        print(discriminator.summary())
        plot_model(discriminator, to_file=self._output_path + "\\" +
'_discriminator.png', show_shapes=True, show_layer_names=True, expand_nested=True)
        return discriminator

    def _define_generator(
        self,
        density,
        hidden_layers = 2
        ):

        # Define output layer activation function according to chosen scaler
        if isinstance(self.data._scaler, MinMaxScaler):
            output_activation = "sigmoid"
        elif isinstance(self.data._scaler, RobustScaler):
            output_activation = "linear"
        else:
            output_activation = "linear"
            print("Error: for data scaling, you must choose either MinMaxScaler or
RobustScaler.")
        # Build generator
        generator =
self._build_base_ann(name='generator',input_shape=self._latent_space_size,
density=density, hidden_layers=hidden_layers, output_shape=self.data._nvariables,
                                    output_activation=output_activation)

        # Print generator summary and save plotted model
        print("")
        print("="*100)
        print(generator.summary())
        plot_model(generator, to_file=self._output_path + "\\" + '_generator.png',
show_shapes=True, show_layer_names=True, expand_nested=True)
        return generator

    def _define_GAN(
```

```
        self,
        generator,
        discriminator,
        alpha = 0.000025,
        beta = 0.5,
        beta2 = 0.999,
        epsilon = 1e-8
        ):

        # Define GAN/generator optimizer
        optimizer = optimizers.Adam(learning_rate=alpha, beta_1=beta, beta_2=beta2,
amsgrad=False, epsilon = epsilon)
        # Define binary accuracy
        binaryAccuracy = tf.keras.metrics.BinaryAccuracy(
            name='binary_accuracy', dtype=None, threshold=0.45)
        # Combine and compile GAN
        gen_class, gen_noise = self.generator.input
        gen_output = generator.output
        gan_output = discriminator([gen_class, gen_output])
        GAN = Model([gen_class, gen_noise], gan_output, name = "GAN")
        GAN.compile(loss='binary_crossentropy', optimizer=optimizer, metrics =
[binaryAccuracy])
        # Save plotted model
        plot_model(GAN, to_file=self._output_path + "\\" + '_GAN.png',
show_shapes=True, show_layer_names=True, expand_nested=False)
        return GAN

    def generate_noise(self, sample_size):
        # Generate noise (generator input)
        return tf.random.normal(shape=[sample_size, self._latent_space_size])

    def generate_synthetic_sample(self, sample_size, labels, scaled = False, noise
= None, pos_processing = False):
        # This method generates synthetic data using the generator
        generator = self.generator
        scaler = self.data._scaler
        int_cols = self.data._int_columns
        # Generate noise if necessary
        if (noise is None):
            noise = tf.random.normal(shape=[sample_size, self._latent_space_size])
        else:
            noise = noise[0:sample_size]
        # Generate synthetic sample
        synthetic_sample = np.concatenate((labels, generator([labels,
noise]).numpy()), axis=1)
        if pos_processing or not scaled:
        # Unscale data
            synthetic_sample = scaler.inverse_transform(synthetic_sample)
        # If it is necessary to round data
        if (pos_processing):
            synthetic_sample = pd.DataFrame(synthetic_sample, columns =
self.data._columns)
            synthetic_sample.loc[:,int_cols] =
np.round(synthetic_sample.loc[:,int_cols],0)
            # AQUI UNSCALED DATA
        # Scale data
        if pos_processing and scaled:
            synthetic_sample = scaler.transform(synthetic_sample)
            # AQUI SCALED DATA
        # Output must be numpy
        if isinstance(synthetic_sample, pd.DataFrame):
            synthetic_sample = synthetic_sample.to_numpy()
        return synthetic_sample[:,:len(self.data._class_columns)],
synthetic_sample[:,-len(self.data._value_columns):]

    def train(self, target_accuracy,
              label_real = 0.9, label_synthetic = 0.0, batch_size = 64,
              epochs = 10000, alfa = 0.05, disp = True):
```

```
        # Define batch size
        batch_size = define_batch(batch_size, self.data._sample_size)
        # Get training data and create batches
        X = self.data.scaled_data.loc[:,~self.data._columns.str.startswith("_")]
        label = self.data.scaled_data.loc[:,self.data._columns.str.startswith("_")]
        dataset = tf.data.Dataset.from_tensor_slices((X, label))
        dataset = dataset.batch(batch_size, drop_remainder=True).prefetch(1)
        # Grab the separate components
        generator = self.generator
        discriminator = self.discriminator

        D_losses = []
        G_losses = []
        D_accuracies = []
        G_accuracies = []
        Real_prob_Ev = []
        Synt_prob_Ev = []
        pvalue = 1
        dif_prob = 1
        Real_disc = 1

        # Attribute to record epochs with saved graphs
        self._epochs = []
        # Track model accuracy
        self._best_accuracy = 0
        # For every epoch until p-value > alfa
        while (self.epoch < epochs and (pvalue > alfa or Real_disc<=0.45 or
Real_disc>=0.55)):
            # Update epoch attribute
            self.epoch += 1
            # Print epoch
            if (self.epoch == 1 or self.epoch%50 == 0):
                print(f"Currently on Epoch {self.epoch}")
            # For every batch in the dataset
            for batch in dataset:
                # Get batch data
                X_batch, label_batch = batch
                # Cast it to float 32
                X_batch = tf.dtypes.cast(X_batch, tf.float32)
                label_batch = tf.dtypes.cast(label_batch,tf.float32)

                ####################################
                ## TRAINING THE DISCRIMINATOR ######
                ####################################

                # Generate synthetic sample based on noise input
                _, X_synthetic =
self.generate_synthetic_sample(sample_size=batch_size, labels=label_batch,
scaled=True)
                # Concatenate synthetic sample against the real dataframe
                X_synthetic_real = tf.concat([X_synthetic, X_batch], axis=0)
                # Define class labels (synthetic classes follow the same classes as
real batch)
                label_synthetic_real = tf.concat([label_batch, label_batch],
axis=0)
                # Define labels (real or synthetic) for synthetic and real samples
to train the discriminator (according label smoothing parameter)
                # We want the discriminator to distinguish real and synthetic
samples
                y_discriminator = tf.constant([[label_synthetic]] * batch_size +
[[label_real]] * batch_size)
                # Train the discriminator on this batch
                discriminator.trainable = True
                D_loss, D_accuracy =
discriminator.train_on_batch([label_synthetic_real, X_synthetic_real],
y_discriminator)
```

```
                ###################################
                ## TRAINING THE GENERATOR     ######
                ###################################

                # Create noise (latent variables)
                noise = self.generate_noise(sample_size=batch_size)
                # Define labels to train the generator (according to label
smoothing parameter)
                # We want the discriminator to believe that the samples are real
                y_generator = tf.constant([[label_real]] * batch_size)
                # Train the generator on this batch
                # For stability, now the discriminator will not be trained
                discriminator.trainable = False
                G_loss, G_accuracy = self.GAN.train_on_batch([label_batch, noise],
y_generator)
            Real_prob = sum(self.discriminator.predict([self.data._scaled_labels,
self.data._scaled_values]) > (label_real+label_synthetic)/2)/self.data._sample_size
            _Real_prob_Ev = Real_prob
            Real_prob_Ev.append(_Real_prob_Ev)
            labels_pred, sint_pred =
self.generate_synthetic_sample(sample_size=self.data._sample_size,
noise=self.standard_noise, scaled=True, pos_processing=True,
labels=self.data._scaled_labels)
            Synt_prob = sum(self.discriminator.predict([labels_pred, sint_pred]) >
(label_real+label_synthetic)/2)/self.data._sample_size
            _Synt_prob_Ev = Synt_prob
            Synt_prob_Ev.append(_Synt_prob_Ev)
            # Check lack-of-fit using CT2T_knn
            # For the first epoch, every 50 epochs or last epoch
            if (self.epoch == 1 or self.epoch%50 == 0 or self.epoch == epochs):
                # Record discriminator and generator losses and accuracies
                D_losses.append(D_loss)
                D_accuracies.append(D_accuracy)
                G_losses.append(G_loss)
                G_accuracies.append(G_accuracy)
                # Print results?
                if (disp == True):
                    print(f"D_loss = {round(D_loss,3)}. G_loss =
{round(G_loss,0)}.")
                    print(f"D_accuracy = {round(D_accuracy,3)}. G_accuracy =
{round(G_accuracy,3)}.")
                # Evaluate generator according to user defined target accuracy
                _, pvalue, _ =
self.evaluate_generator(target_accuracy=[model_to_CT2T_accuracy(target_accuracy)],
                                                        alfa=alfa)
                dif_prob = abs(Real_prob - Synt_prob)
                print(f"The difference is {round(dif_prob[0]*100,2)}%.")
                Real_disc =
sum(self.discriminator.predict([self.data._scaled_labels,
self.data._scaled_values]) > (label_real+label_synthetic)/2)/self.data._sample_size
                print(f"The discriminator for tested data is
{round(Real_disc[0]*100,2)}%.")
                # Plot losses and pairplot
                self.plot_data(D_losses=D_losses, G_losses=G_losses, pairplot=True,
losses=False) ##Losses
                # Append this epoch: this vector will be used to create a movie
with epochs' exported images
                self._epochs.append(self.epoch)
                # If current model presents the best performance until now
                if (self._accuracy >= self._best_accuracy):
                    # Save model weights
                    generator.save_weights(self._output_path + "\\checkpoint")
                    self._best_accuracy = self._accuracy
        # Load weights from the best epoch
        self.generator.load_weights(self._output_path + "\\checkpoint")
        # Evaluate generator for each possible accuracy class
        _ = self.evaluate_generator(target_accuracy=reference_accuracies.keys(),
```

```
                                                                    alfa=alfa,
describe=True)
        figure = plt.figure()
        plt.plot(Real_prob_Ev, label = "R. prob", zorder=50, alpha=0.7)
        plt.plot(Synt_prob_Ev, label = "S. prob", zorder=40, alpha=0.7)
        plt.xlabel("Epoch")
        plt.ylabel("Probability")
        plt.legend(loc = 1)
        figure.suptitle(f"GAN probability", fontsize=12)
        plt.savefig(self._output_path + "\\" + f"Final probability", dpi=1200)
        plt.close()
        # Plot final results
        self.plot_data(D_losses=D_losses, G_losses=G_losses, pairplot=True,
losses=False)
        print("="*100)
        # If epoch < max number of epochs, the model achieved target accuracy
earlier
        if (self.epoch < epochs):
            print("Target model accuracy achieved.")
        else:
            print("Maximum training time reached.")
        # Return generator
        return generator

        plt.plot(Real_prob, label = "Real_prob")
        plt.legend()

    def evaluate_generator(self, target_accuracy, alfa, describe=False):
        generator = self.generator
        sample_size = self.data._sample_size
        standard_noise = self.standard_noise
        latent_space_size = self._latent_space_size
        scaled_data = self.data.scaled_data
        unscaled_data = self.data.real_data
        labels = self.data._scaled_labels
        Result = namedtuple('evaluation',['accuracy', 'pvalue', 'accuracy_class'])

        # Generate scaled synthetic sample based on standard noise input
        labels, synthetic_sample =
self.generate_synthetic_sample(sample_size=sample_size, labels=labels, scaled=True,
noise=standard_noise, pos_processing=True)
        # Concatenate labels (classes) and other columns
        synthetic_sample = np.concatenate((labels, synthetic_sample), axis=1)
        # Create concatenated test samples
        # Test data = classes + other columns
        test_data = np.concatenate((synthetic_sample, scaled_data))
        # Test label = real or synthetic (0 or 1)
        test_label = np.concatenate((np.zeros(sample_size), np.ones(sample_size)))
        # CT2T_knn is designed to separate synthetic and real samples
        # If synthetic and real samples come from the same distribution, test
accuracy should remain near chance-level
        accuracy, pvalue, target_accuracy = CT2T_knn(test_data, test_label,
target_accuracy, alfa)
        self._accuracy = round(100*CT2T_to_model_accuracy(accuracy),1)
        print(f"P-value = {round(pvalue,3)}. CT2T accuracy =
{round(100*accuracy,1)}%.")
        print(f"Input model accuracy = {self._accuracy}%.")

        if (describe):
            # Generate unscaled synthetic sample based on noise input
            labels, synthetic_sample =
self.generate_synthetic_sample(sample_size=sample_size, labels=labels,
scaled=False, noise=standard_noise, pos_processing=True)
            # Concatenate labels (classes) and other columns
            synthetic_sample = np.concatenate((labels, synthetic_sample), axis=1)
            # Create concatenated test samples
            # Test data = classes + other columns
            test_data = np.concatenate((synthetic_sample, unscaled_data))
```

```
            # Test label = real or synthetic (0 or 1)
            test_label = np.concatenate((np.zeros(sample_size),
np.ones(sample_size)))
            # Print data statistics
            print("Real and synthetic data description:")
            with pd.option_context('float_format', '{:,.2f}'.format):
                df = pd.DataFrame(test_data, columns=self.data._columns)
                df['label'] = test_label
                description = df.groupby('label').describe().transpose()
                description.columns = ['Synthetic', 'Real']
                description['Difference (%)'] = (description['Synthetic'] /
description['Real'] - 1)*100
                print(description)
        return Result(accuracy, pvalue, target_accuracy)

    def plot_data(self, D_losses, G_losses, pairplot=True, losses=True):
        sample_size = self.data._sample_size
        generator = self.generator
        epoch = self.epoch
        standard_noise = self.standard_noise
        labels = self.data._scaled_labels
        sns.set_palette("Set2")

        # If user has chosen to plot the pairplot
        if (pairplot):
            # Generate unscaled synthetic sample based on noise input
            _,synthetic_sample =
self.generate_synthetic_sample(sample_size=sample_size, labels=labels,
scaled=False, noise=standard_noise, pos_processing = True)
            # Concatenate labels (classes) and other columns
            synthetic_sample = np.concatenate((labels, synthetic_sample), axis=1)
            # Sample real data
            sample_df = self.data.real_data.sample(sample_size)
            # Cast synthetic sample to pandas dataframe
            dfS = pd.DataFrame(synthetic_sample, columns = self.data._columns)
            # Create label column = Synthetic
            dfS["Label"] = "Synthetic"
            # Define real dataframe
            dfR = sample_df
            # Create label column = Real
            dfR["Label"] = "Real"
            # Concatenate dataframes
            dfF = pd.concat([dfR, dfS])
            # If just one variable, graph will be a histogram
            if (self.data._nvariables == 1):
                fig, ax = plt.subplots()

sns.histplot(data=dfF.drop(columns=self.data._class_columns),x=self.data._value_col
umns[0],hue='Label',fill =True)
                fig.suptitle(f"Distribution plot, epoch {epoch}", fontsize=12)
                plt.savefig(self._output_path + "\\" + f"Epoch {epoch} -
distributions", dpi=1200)
                plt.close()

            # Else, a pairplot
            else:
                p = sns.pairplot(dfF.drop(columns=self.data._class_columns), hue =
"Label", diag_kind='hist', plot_kws = {"s":3})
                p.fig.subplots_adjust(top=0.9)
                p.fig.suptitle(f"Distribution plot, epoch {epoch}", fontsize=12)
                plt.savefig(self._output_path + "\\" + f"Epoch {epoch} -
distributions", dpi=1200)
                plt.close()

        # If user has chosen to plot losses
        if (losses):
            fig = plt.figure()
            plt.plot(D_losses, label = "D. loss", zorder=50, alpha=0.7)
```

```
            plt.plot(G_losses, label = "G. loss", zorder=40, alpha=0.7)
            plt.xlabel("Epoch")
            plt.ylabel("Loss")
            plt.legend(loc = 1)
            fig.suptitle(f"GAN losses, epoch {epoch}", fontsize=12)
            plt.savefig(self._output_path + "\\" + f"Epoch {epoch} - losses",
dpi=1200)
            plt.close()
```

B.4. Classe *Power*

```
class Power():

    """
    Power calculations for one proportion test
    One-sided power
        H0: p = p0
        H1: p < p0
    p = true population proportion = real_effect/2 + 0.5
    p0 = hypothesized proportion = equivalence limit = tolerable_effect/2 + 0.5
    p1 = comparison proportion = 0.5
    effect = % of unrealistic synthetic observations
    Source: Minitab, Methods and formulas for Power and Sample Size for 1
Proportion
    """

    def __init__(self, data):
        pass

    def power(p0, p1, alfa, n):

        """
        Power calculation for one proportion test
        """

        numerator = p0 - p1 - norm.ppf(1-alfa)*math.sqrt(p0*(1-p0)/n)
        denominator = math.sqrt(p1*(1-p1)/n)
        power = norm.cdf(numerator/denominator)
        return power

    def powerDifEffect(p0, p1, alfa, n, target_power):

        """
        Modified power calculation
        To enable estimating the smallest p0 that can be achieve with the desired
power
        """

        numerator = p0 - p1 - norm.ppf(1-alfa)*math.sqrt(p0*(1-p0)/n)
        denominator = math.sqrt(p1*(1-p1)/n)
        power = norm.cdf(numerator/denominator)
        return abs(power - target_power)

    def powerDifN(n, p0, p1, alfa, target_power):

        """
        Modified power calculation
        To enable estimating the smallest sample size that can lead to the desired
power
        """

        numerator = p0 - p1 - norm.ppf(1-alfa)*math.sqrt(p0*(1-p0)/n)
        denominator = math.sqrt(p1*(1-p1)/n)
        power = norm.cdf(numerator/denominator)
        return abs(power - target_power)
```

```
    def max_model_accuracy(n, alfa = 0.05, target_power = 0.8):

        """
        Estimate the smallest p0 that can be identified by the CT2T test
        And so the maximum input model accuracy which can be proved by the test
with the desired power
        """

        res = fmin(Power.powerDifEffect, x0=0.5, args=(0.5, alfa, n, target_power),
disp=0)
        accuracy = CT2T_to_model_accuracy(res[0])
        return accuracy
```

B.5. *Utils*

```
def CT2T_knn(test_data, labels, target_accuracies, alfa = 0.05):

    Result = namedtuple('CT2T_results',
                        ['accuracy', 'pvalue', 'target_accuracy'])
    test_data, labels = shuffle(test_data, labels, random_state=100)

    knn = KNeighborsClassifier(n_neighbors =
int(round(math.sqrt(test_data.shape[0]),0)))

    base = round(math.sqrt(test_data.shape[0]),0)
    mult = np.arange(1, 6)*base

    def round_up_to_odd(f):
        return np.ceil(f) // 2 * 2 + 1

    mult = np.array([round_up_to_odd(x) for x in mult])

    parameters = {'n_neighbors':mult.astype(int)}
    clf = GridSearchCV(knn, parameters, cv=5)
    clf.fit(test_data, labels)

    accuracy = clf.best_score_

    for target_accuracy in target_accuracies:
        test_statistic = (accuracy - target_accuracy)/math.sqrt(accuracy*(1-
accuracy)/(test_data.shape[0]*2))
        pvalue = norm.cdf(test_statistic, 0, 1)
        if (pvalue <= alfa):
            return Result(accuracy=accuracy, pvalue=pvalue,
target_accuracy=target_accuracy)
    else:
        return Result(accuracy=accuracy, pvalue=pvalue,
target_accuracy=target_accuracy)

def define_batch(desired_batch, sample_size):
    total_size = sample_size*2
    mod = total_size%desired_batch
    if (mod == 0):
        return desired_batch
    else:
        if (mod <= desired_batch/2):
            desired_batch += mod/4
        else:
            desired_batch -= (desired_batch-mod)/4
        return int(desired_batch)

def define_target_accuracy(sample_size):
    max_target_accuracy = round(100*Power.max_model_accuracy(n=sample_size,
target_power=0.8),2)
```

```
    print("Please choose the input model target accuracy. The maximum recommended
value is {}%.".format(max_target_accuracy))
    print("The input model accuracy is the percentage of the generated data that
follows the true data distribution.")
    print("The following reference values can be used to define the target
accuracy.")
    for reference_accuracy in reference_accuracies.values():
        print(reference_accuracy)
    target_accuracy = float(input("Input model target accuracy (%):
".format(max_target_accuracy)))
    return target_accuracy

def CT2T_to_model_accuracy(CT2T_accuracy):
    input_model_accuracy = 1 - 2*max(0, CT2T_accuracy - 0.5)
    return input_model_accuracy

def model_to_CT2T_accuracy(input_model_accuracy):
    CT2T_accuracy = 0.5 + (1 - input_model_accuracy)/2
    return CT2T_accuracy

def create_video(path, epochs, fps = 60):
    images = []
    for epoch in epochs:
        images.append(imageio.imread(path + "\\" + f"Epoch {epoch} -
distributions.png"))
    imageio.mimsave(path + "\\" + "movie.mp4", images, fps=fps)
```

B.6. *Main*

```
reference_accuracies = {
    0.55: 'A - Very strong (>90%)',
    0.60: 'B - Strong (80% - 90%)',
    0.70: 'C - Relatively strong (60% - 80%)',
    0.80: 'D - Moderate (40% - 60%)',
    0.90: 'E - Weak (20% - 40%)',
    1.00: 'F - Negligible (0% - 20%)'
}

def main(distribution, target_accuracy = None, seed = 0):

    #Define random seed
    tf.random.set_seed(seed)
    np.random.seed(seed)

    #Generate or import data
    data = Data()
    data.read_excel(distribution, sample_size = None)

    #Preparate data (remove missing, scale...)
    print("="*100)
    print("Descriptive statistics")
    print("="*100)
    data.prepare_data(RobustScaler())

    #Get user target accuracy
    print("="*100)
    print("Accuracy definition")
    print("="*100)
    if target_accuracy is None:
        target_accuracy = define_target_accuracy(2*data._sample_size)/100
    else:
        target_accuracy /= 100

    #Crate GAN
    print("="*100)
    print("GAN architecture")
```

```
    print("="*100)
    myGAN = GAN(data)

    #Train GAN
    print("="*100)
    print("GAN training started!")
    print("="*100)
    myGAN.train(target_accuracy=target_accuracy, disp=False, epochs=10000,
batch_size = 64)

    # Report results
    print("="*100)
    print("Training has been completed")
    print("="*100)

    #Save input model as txt and excel
    myGAN.save_input_model()

    #Create training video
    create_video(myGAN._output_path, myGAN._epochs, fps=1)

    return myGAN

myGAN = main("Excel_File_Name_Training_Data.xlsx", target_accuracy=95)
```

B.7. Classe *Compared Data*

```
class compared_Data():

    """
    This class comprises methods to read, generate and preparate compared data.
    """

    def __init__(self):
        self.original_data = None

    # Read data from excel file
    def read_excel(self, excel_file_name = None, sample_size = None):
        df = pd.read_excel(excel_file_name)
        # If user wants to sample from the excel data
        if sample_size is not None and sample_size < df.shape[0]:
            self.original_data = df.sample(sample_size)
        # Else, we get and shuffle all data
        else:
            self.original_data = df.sample(frac=1)
        self._dataset_name = excel_file_name

    # Prepare data
    def prepare_data(self):
        # Get copy of original data
        self.compared_data = self.original_data.copy()
        # Force columns to be string
        self.compared_data = self.compared_data.rename(columns=lambda x: str(x))
        # Dataframe may not contain a column named as "_"
        if '_' in self.compared_data.columns:
            self.data.compared_data.rename(columns={'_':'_*'}, inplace=True)
            print('\033[93m' + "Illegal column name: '_'. Name changed to '_*'" +
'\033[0m')
        # Remove object columns whose names don't start with "_"
        columns_to_drop =
self.compared_data.select_dtypes(include=['object']).columns
        columns_to_drop =
np.array(columns_to_drop[~columns_to_drop.str.startswith('_')])
        self.compared_data.drop(columns=columns_to_drop, axis = 1, inplace=True)
        if len(columns_to_drop) > 0:
```

```
            print('\033[93m' + "The following columns were excluded for containing
categorical data:" + '\033[0m')
            for column in columns_to_drop:
                print(column)
            print('\033[93m' + "If these columns should be considered as data
classes, please start their names with '_'." + '\033[0m')
            print("")

        # Add at least one class column
        if
len(self.compared_data.columns[self.compared_data.columns.str.startswith('_')]) ==
0:
            self.compared_data['_'] = 0

        # Handle missing data
         ## Report missing data
        n_dropped_observations = np.sum(self.compared_data.isna().sum(axis=1) > 0)
        if n_dropped_observations > 0:
            print('\033[93m' + "Missing data by variable (%):")
print(round(100*self.compared_data.isna().sum(axis=0)/self.real_data.shape[0],1))
            print("")
            print("Total dropped observations:")
            print(np.sum(self.compared_data.isna().sum(axis=1) > 0))
            print("" + '\033[0m')

        ## Drop missing data
        self.compared_data = self.compared_data.dropna()
        # Get original class columns
        self._original_class_columns =
np.sort(self.compared_data.columns[self.compared_data.columns.str.startswith('_')])
        # Get original class data types
        self._original_class_dtypes = [self.compared_data[col].dtype.kind for col
in self._original_class_columns]

        # Record unique values (for undumming)
        string_class_columns = [col for col, kind in
zip(self._original_class_columns, self._original_class_dtypes) if kind == 'O']
        unique_class_values = np.empty(len(string_class_columns),
dtype='str').tolist()
        for idx, col in enumerate(string_class_columns):
            unique_class_values[idx] =
np.sort(self.compared_data[col].unique()).tolist()
        self._unique_class_values = unique_class_values
        # Describe real data
        print("Data summary:")
        print(self.compared_data.describe())
        sns.pairplot(self.compared_data)
        plt.show()
        # Get dummies
        self.compared_data = pd.get_dummies(self.compared_data,
prefix_sep='__',drop_first=True)
        #Reorganize data according to data type (classes, whose columns start with
"_", values)
        column_order =
np.concatenate((self.compared_data.columns[self.compared_data.columns.str.startswit
h('_')],
self.compared_data.columns[~self.compared_data.columns.str.startswith('_')]))
        self.compared_data = self.compared_data.loc[:,column_order]
        # Identify int columns (this identification will enable rounding them after
generating)
        int_columns = self.compared_data.select_dtypes(include=['integer']).columns
        self._int_columns = [col for col in int_columns if not col.startswith("_")]
        # Scaler compared data
        max_t = myGAN.data.real_data.to_numpy().max(axis=0)
        min_t = myGAN.data.real_data.to_numpy().min(axis=0)
        max_s = myGAN.data.scaled_data.to_numpy().max(axis=0)
        min_s = myGAN.data.scaled_data.to_numpy().min(axis=0)
```

```
        self.scaled_compared_data = (min_s*(self.compared_data - max_t) +
max_s*(min_t - self.compared_data))/(min_t-max_t)
        self.scaled_compared_data = self.scaled_compared_data.replace(np.nan, 0)
        # Record data characteristics
        self._compared_sample_size = self.compared_data.shape[0]
        self._columns = self.compared_data.columns
        self._class_columns = self._columns[self._columns.str.startswith('_')]
        self._value_columns = self._columns[~self._columns.str.startswith('_')]
        self._scaled_compared_labels = self.scaled_compared_data.loc[:,
self._class_columns]
        self._scaled_compared_values = self.scaled_compared_data.loc[:,
self._value_columns]
        self._nvariables = len(self._value_columns)
```

B.8. Classe Teste de Equivalência

```
class Equivalence_test():

    """
    Equivalence Test for two proportion
    H0: p1 - p2 ≤ dif or H0: p1 - p2 ≥ dif
    H1: -dif < p1 - p2 < dif
    """
    def __init__(self, compared_Data):
        self.scaled_compared_labels = compared_Data._scaled_compared_labels
        self.scaled_compared_values = compared_Data._scaled_compared_values
        self.compared_sample_size = len(compared_Data._scaled_compared_values)

    def test(dif,alfa):
        trained_data = sum(myGAN.discriminator.predict([myGAN.data._scaled_labels,
myGAN.data._scaled_values]) > 0.45)/myGAN.data._sample_size
        compared_data =
sum(myGAN.discriminator.predict([myTest.scaled_compared_labels,
myTest.scaled_compared_values]) > 0.45)/myTest.compared_sample_size
        print("="*100)
        print(f"The discriminator for tested data is
{round(trained_data[0]*100,2)}%.")
        print(f"The discriminator for compared data is
{round(compared_data[0]*100,2)}%.")
        print("="*100)

        p1 = trained_data
        p2 = compared_data
        n1 = myGAN.data._sample_size
        n2 = myTest.compared_sample_size
        dif = dif/2

        ##standard deviation
        std_t = p1*(1-p1)/n1
        std_c = p2*(1-p2)/n2

        #confidence interval#
        zalfa = norm.ppf(1-alfa)
        LB = p1 - p2 - zalfa*((std_t+std_c)**0.5)
        UB = p1 - p2 + zalfa*((std_t+std_c)**0.5)

        ##Z-test
        z1 = (p1-p2+dif)/((std_t+std_c)**0.5)
        z2 = (p1-p2-dif)/((std_t + std_c)**0.5)

        ## p-values##
        p_right = 1 - norm.cdf(z1)
        p_left = norm.cdf(z2)
        p_max = round(float(max(p_right,p_left)[0]),5)

        ## power ##
```

```
        power = 2*norm.cdf((dif-abs(p1-p2))/((std_t+std_c)**0.5)-norm.ppf(alfa))-1
        if power >= 0:
            power=power
        else:
            power =0

        # print("Two-Proportion Equivalence Test")
        print()
        print("Descriptive Statistics")

        l = [["Trained Data", n1, round(float((p1)[0]),5),
round(float((std_t)[0]),7)], ["Compared Data", n2, round(float((p2)[0]),5),
round(float((std_c)[0]),7)]]

        table1 = PrettyTable(['Variable', 'N', 'Mean', "StdDev"])
        for rec in l:
            table1.add_row(rec)
        print(table1)
        print()
        print("Proportion (Trained Data) - Proportion (Compared Data)")

        r = [[round(float(p1-p2),7), [round(float(LB[0]),7),round(float(UB[0]),7)],
[round(-dif,7),round(dif,7)]]]

        table2 = PrettyTable(['Difference', '95% CI for Equivalence', "Equivalence
Interval"])
        for rec in r:
            table2.add_row(rec)
        print(table2)

        if p_max >= alfa:
            print("CI is not within the equivalence interval. Cannot claim
equivalence")
        else:
            print("CI is within the equivalence interval. Can claim equivalence")

        print()
        print("Test")
        print("Null hypothesis:       ","p1 - p2 ≤", round(-dif,7), "or p1 - p2 ≥",
round(dif,7))
        print("Alternative hypothesis:", round(-dif,7), "< p1 - p2 <",
round(dif,7))
        print("or")
        print("Null hypothesis:       ","p1 ≤", round(float((p2-dif)[0]),7), "or p1
≥", round(float((p2+dif)[0]),7))
        print("Alternative hypothesis:", round(float((p2-dif)[0]),7), "< p1 <",
round(float((p2+dif)[0]),7))
        print("α level:",alfa)

        s = [["p1 - p2 ≤ lower limit", round(float(z2[0]),2),
round(float(p_right[0]),5)], ["p1 - p2 ≥ upper limit", round(float(z1[0]),2),
round(float(p_left[0]),5)]]

        table3 = PrettyTable(['Null Hypothesis', 'Zα', "P-value"])
        for rec in s:
            table3.add_row(rec)
        print(table3)
        if p_max >= alfa:
            print("The greater of two p-values is", p_max,". Cannot claim the
equivalence.")
        else:
            print("The greater of two p-values is", p_max,". Can claim
equivalence.")
        print()
        warnings.filterwarnings("ignore")
        fig, ax = plt.subplots()
        x = 0
        y = 0.1
```

```
        ax.plot(x,y)
        ax.hlines(y=y, xmin=LB, xmax=UB, color = "blue")
        plt.axvline(x=LB, ymin=0.45, ymax=0.55, color =  "blue")
        plt.axvline(x=UB, ymin=0.45, ymax=0.55, color = "blue")
        plt.axvline(x=-dif, color = "red", linestyle = "--" )
        plt.axvline(x=dif, color = "red", linestyle = "--" )
        plt.axvline(x=0, color = "black", linestyle = ":")
        ax.axes.get_yaxis().set_visible(False)
        fig.suptitle("Equivalence Test: Trained - Compared", fontsize = 14)
        ax.text(-dif+0.001, 0.1049, 'Lower Limit', fontsize = 12)
        ax.text(dif-0.025, 0.1049, 'Upper Limit', fontsize = 12)
        ax.text(LB, 0.09875, '95% CI for Equivalence', fontsize = 11)
        plt.show()
        print()
        print("Power")

        if power != 0:
            u = [[round(float(p1-p2),7), round(float(power[0]),7)]]
        else:
            u = [[round(float(p1-p2),7), power]]
        table4 = PrettyTable(['Difference', 'Power'])

        for rec in u:
            table4.add_row(rec)

        print(table4)
        print()
        p2_test = p2
        Power_Ev = []
        Dif_Ev = []
        p2 = (p1 - 3*dif)
        while (p2 <= p1+3*dif+0.005):
            power_test = 2*norm.cdf((dif-abs(p1-p2))/((std_t+std_c)**0.5)-
norm.ppf(alfa))-1
            if power_test >= 0:
                power_test=power_test*100
            else:
                power_test = 0

            _Power_Ev = power_test
            Power_Ev.append(_Power_Ev)
            _Dif_Ev = p1-p2
            Dif_Ev.append(_Dif_Ev)
            p2 = p2+0.005

        p2 = compared_data
        power_figure, ax = plt.subplots()
        x = Dif_Ev
        y = Power_Ev
        ax.plot(x,y)
        ax.set_xlabel('Difference')
        ax.set_ylabel('Power [%]')
        ax.set_title(f"Power Equivalence Test, nt={n1}, nc={n2}", fontsize = 14)
        plt.plot((p1-p2),power*100,'ro', color = 'black')
        plt.show()
        print()
        dif_real = round(float(dif),1)

        if p_max > alfa:
            print(f"We cannot claim the Equivalence considering {dif*2*100}% of
real tolerance.")
            dif = dif
            while (p_max > alfa):
                ##Z-test
                z1 = (p1-p2+dif)/((std_t+std_c)**0.5)
                z2 = (p1-p2-dif)/((std_t + std_c)**0.5)
                ## p-values##
                p_right = 1 - norm.cdf(z1)
```

```
                p_left = norm.cdf(z2)
                p_max = round(float(max(p_right,p_left)[0]),5)
                dif_real = round(float(dif),2)
                dif = dif + 0.0001
            print(f"The minimun real tolerance to claim the Equivalence is
{dif_real*100}% (p-value = {p_max}).")

        if (dif_real >= 0.0 and dif_real <= 0.05):
            validity_accuracy = "a Very Strong"
        elif (dif_real > 0.05 and dif_real <= 0.10):
            validity_accuracy = "a Strong"
        elif (dif_real > 0.10 and dif_real <= 0.20):
            validity_accuracy = "a Satisfying"
        elif (dif_real > 0.20 and dif_real <= 0.30):
            validity_accuracy = "a Marginal"
        elif (dif_real > 0.30 and dif_real <= 0.40):
            validity_accuracy = "a Deficient"
        elif (dif_real > 0.40):
            validity_accuracy = "an Unsatisfying"
        print(f"The Equivalence test corresponds {validity_accuracy} validation.")
```

B.9. Teste

```
def test(distribution, dif, alfa, seed = 0):

    #Define random seed
    tf.random.set_seed(seed)
    np.random.seed(seed)

    #Generate or import data
    data_compared = compared_Data()
    data_compared.read_excel(distribution, sample_size = None)

    #Preparate data (remove missing, scale...)
    print("="*100)
    print("Descriptive statistics")
    print("="*100)
    data_compared.prepare_data()

    #Crate Test
    print("="*100)
    print("Equivalence Test")
    print("="*100)

    #Create proportion
    myTest = Equivalence_test(data_compared)

    Equivalence_test.test(dif,alfa)

    return myTest

myTest = test("Excel_File_Name_Compared_Data.xlsx ", dif=0.05, alfa=0.05)
```

APÊNDICE C – Distribuições contínuas univariadas

O Apêndice C apresenta o resumo do treinamento dos 10 ciclos e as curvas de teste (tolerância 5.0%, 10.0% e 20.0%) para as distribuições contínuas univariadas: Exponencial, Uniforme, Lognormal e Bimodal.

C.1. Exponencial

A Tabela C.1 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Exponencial. As Figuras C.1, C.2 e C.3 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela C.1 - Resumo do treinamento da distribuição Exponencial - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	540.00	325.58	338.21	741.79
Acurácia	96.81	0.81	96.31	97.31
Discriminador	51.82	2.86	50.04	53.59
DT – 5000 dados				
Época	2145.00	2391.01	663.07	3626.93
Acurácia	97.49	1.32	96.67	98.31
Discriminador	50.17	3.57	47.95	52.38
DT – 2000 dados				
Época	1800.00	1730.29	727.58	2872.42
Acurácia	98.11	0.90	97.55	98.67
Discriminador	51.36	2.55	49.77	52.94
DT – 1000 dados				
Época	1805.00	1047.87	1155.54	2454.46
Acurácia	98.40	0.57	98.04	98.76
Discriminador	50.08	2.94	48.26	51.90

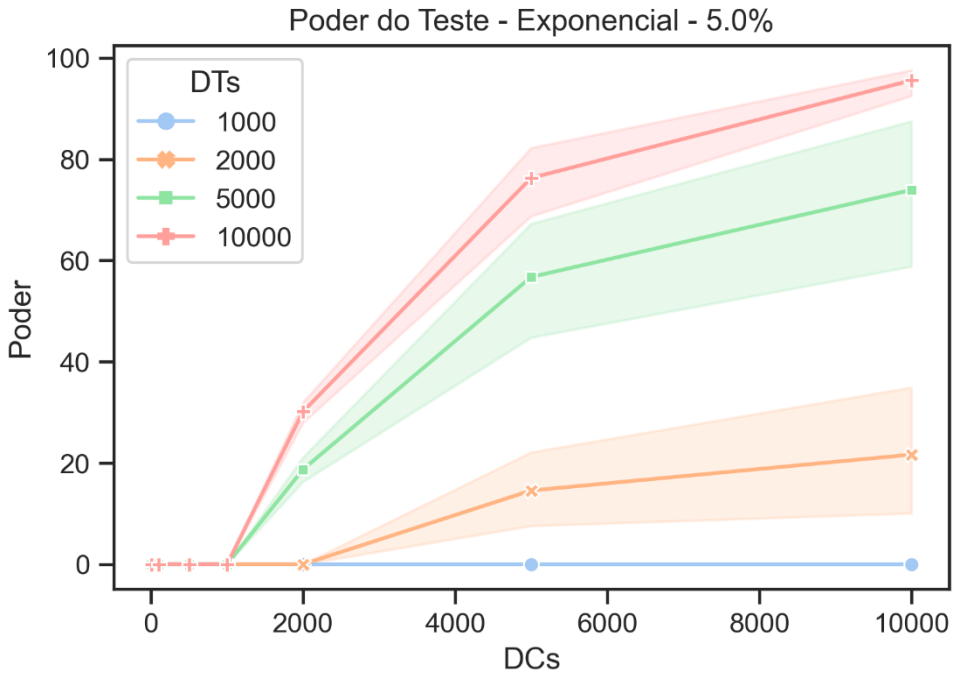

Figura C.1- Poder do Teste - Exponencial - Tolerância 5.0%

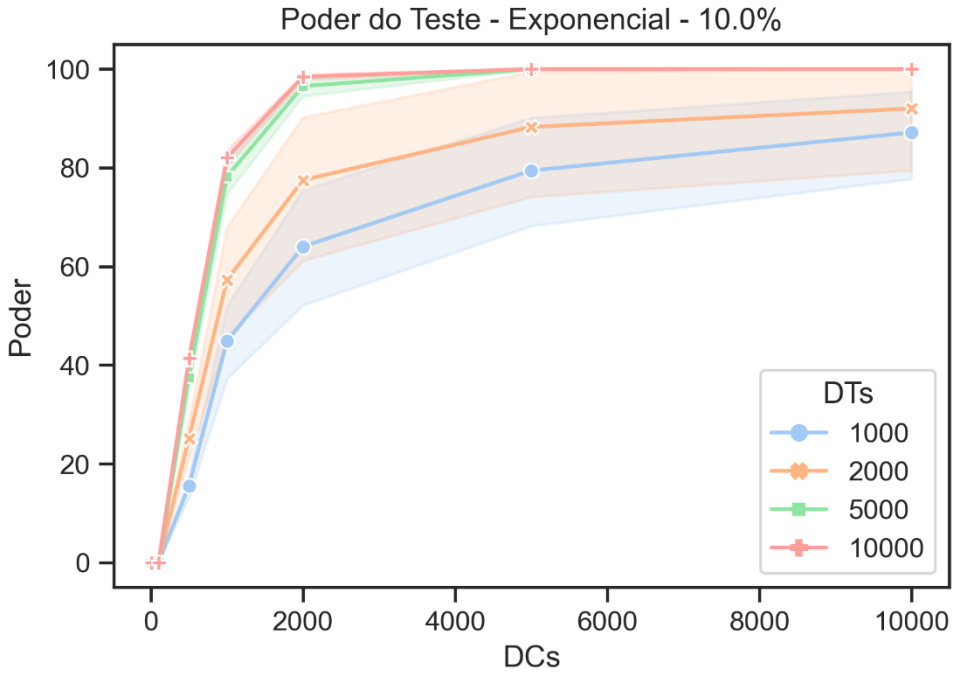

Figura C.2 - Poder do Teste - Exponencial - Tolerância 10.0%

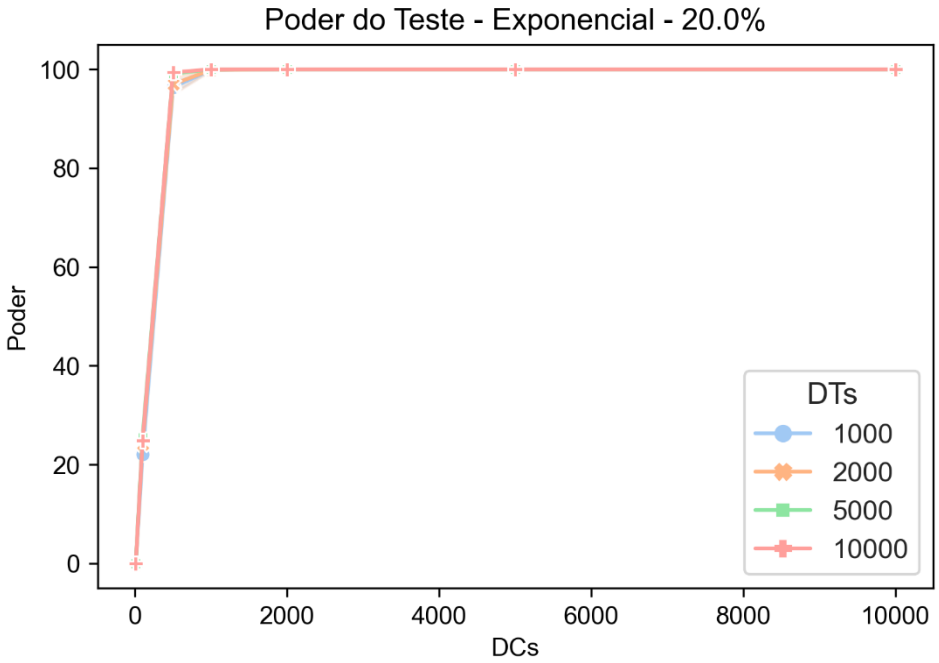

Figura C.3 - Poder do Teste - Exponencial - Tolerância 20.0%

C.2. Uniforme

A Tabela C.2 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Uniforme. As Figuras C.4, C.5 e C.6 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela C.2 - Resumo do treinamento da distribuição Uniforme - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
		DT – 10000 dados		
Época	365.00	163.38	263.74	466.26
Acurácia	96.71	0.70	96.28	97.14
Discriminador	49.94	2.61	48.32	51.56
		DT – 5000 dados		
Época	815.00	321.50	615.74	1014.26
Acurácia	97.25	1.01	96.62	97.88
Discriminador	51.45	2.60	49.84	53.06
		DT – 2000 dados		
Época	1460.00	777.39	952.12	1967.88
Acurácia	98.12	0.96	97.49	98.75
Discriminador	49.73	2.27	48.25	51.21
		DT – 1000 dados		
Época	1805.00	1047.87	1155.54	2454.46
Acurácia	98.40	0.57	98.04	98.76
Discriminador	50.08	2.94	48.26	51.90

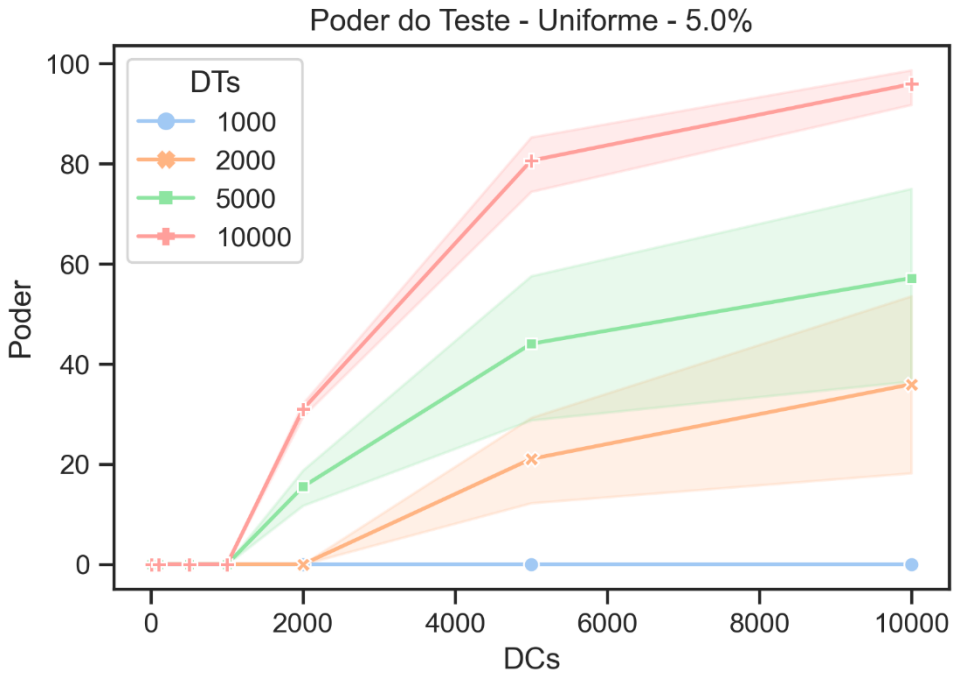

Figura C.4 - Poder do Teste - Uniforme - Tolerância 5.0%

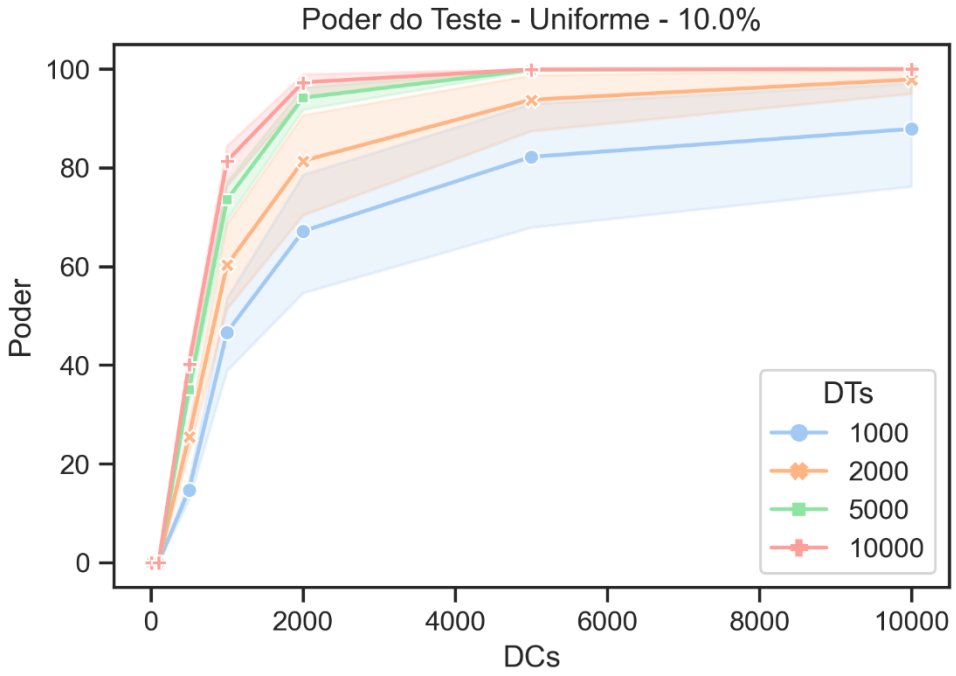

Figura C.5 - Poder do Teste - Uniforme - Tolerância 10.0%

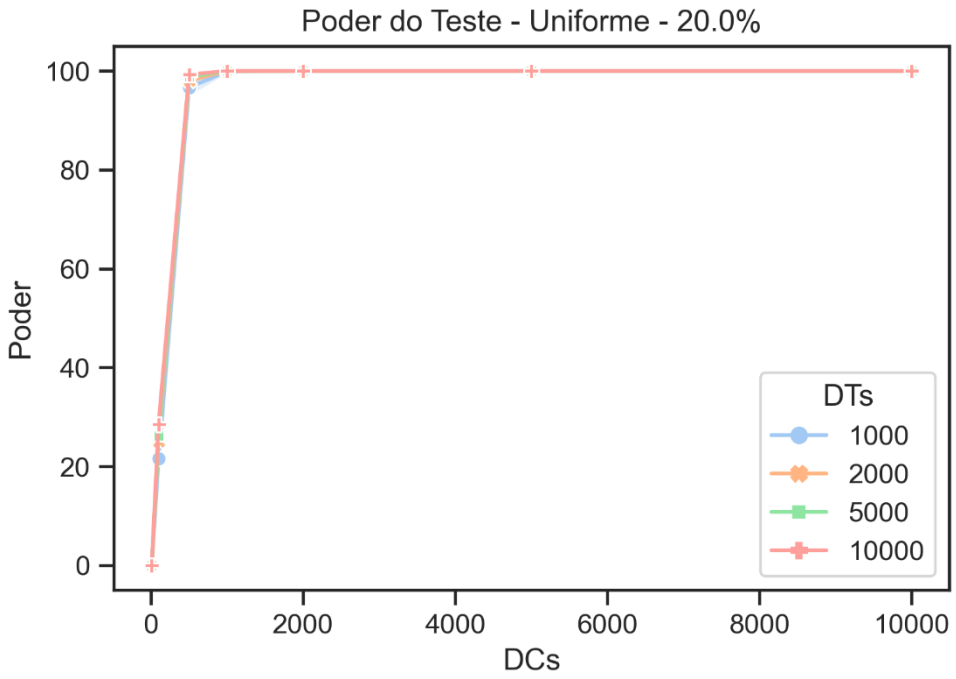

Figura C.6 - Poder do Teste - Uniforme - Tolerância 20.0%

C.3. Lognormal

A Tabela C.3 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Lognormal. As Figuras C.7, C.8 e C.9 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela C.3 - Resumo do treinamento da distribuição Lognormal - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	600.00	567.16	248.48	951.52
Acurácia	97.25	0.98	96.64	97.86
Discriminador	50.03	2.73	48.34	51.72
DT – 5000 dados				
Época	705.00	500.25	394.95	1015.05
Acurácia	97.37	1.22	96.62	98.12
Discriminador	50.59	3.30	48.55	52.64
DT – 2000 dados				
Época	1165.00	445.38	888.96	1441.04
Acurácia	97.32	0.45	97.04	97.60
Discriminador	50.84	2.85	49.07	52.60
DT – 1000 dados				
Época	2280.00	1764.34	1186.47	3373.53
Acurácia	98.58	0.81	98.08	99.08
Discriminador	50.18	2.29	48.76	51.60

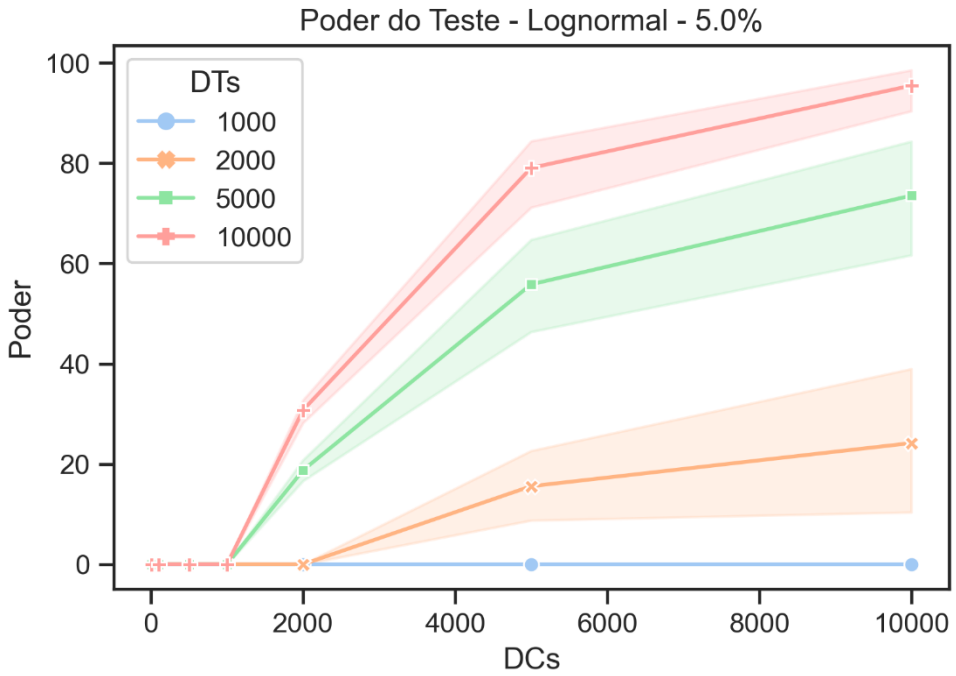

Figura C.7 - Poder do Teste - Lognormal - Tolerância 5.0%

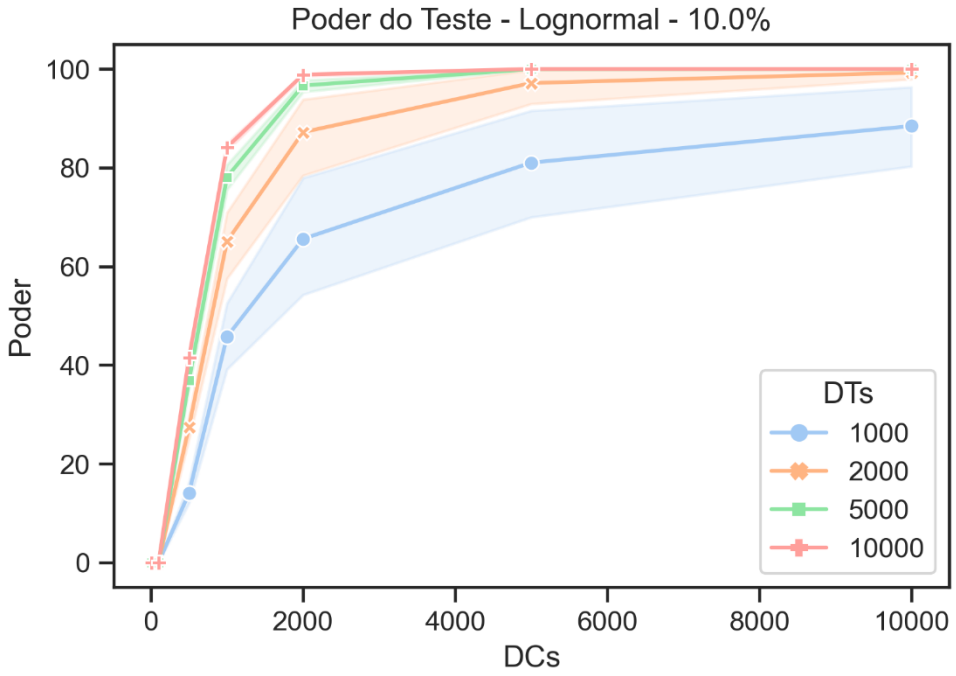

Figura C.8 - Poder do Teste - Lognormal - Tolerância 10.0%

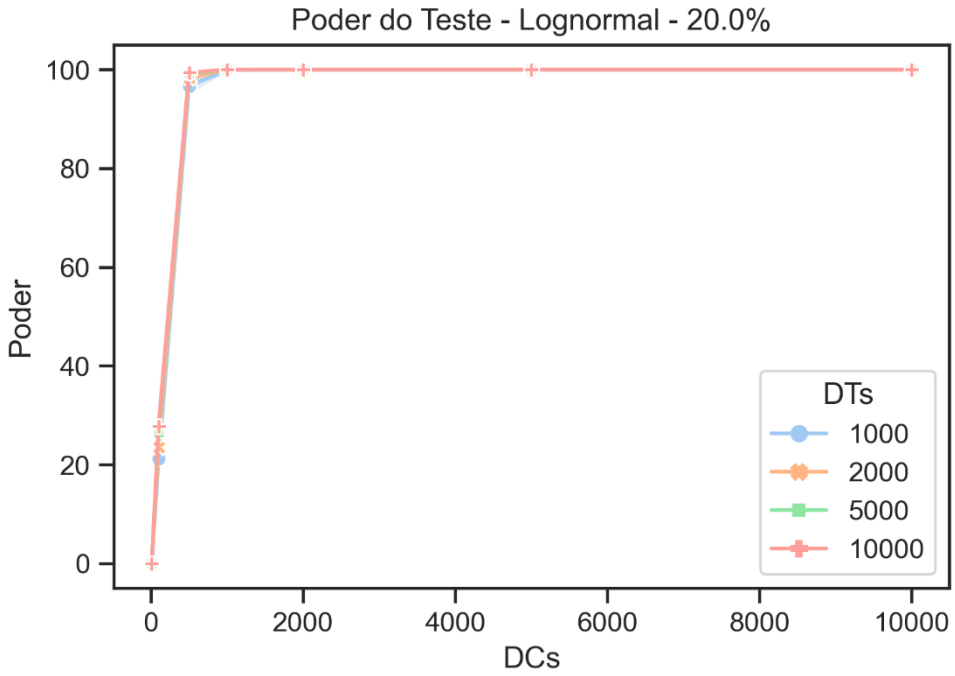

Figura C.9 - Poder do Teste - Lognormal - Tolerância 20.0%

C.4. Bimodal

A Tabela C.4 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Bimodal. As Figuras C.10, C.11 e C.12 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela C.4 - Resumo do treinamento da distribuição Bimodal - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	2010.00	2441.86	496.55	3523.45
Acurácia	96.94	0.83	96.43	97.45
Discriminador	49.02	1.95	47.81	50.22
DT – 5000 dados				
Época	1375.00	1988.05	142.81	2607.19
Acurácia	97.66	0.97	97.06	98.26
Discriminador	52.00	2.38	50.53	53.48
DT – 2000 dados				
Época	4145.00	2994.85	2288.81	6001.19
Acurácia	98.98	0.86	98.45	99.51
Discriminador	49.69	2.42	48.19	51.19
DT – 1000 dados				
Época	4065.00	1751.04	2979.71	5150.29
Acurácia	99.66	0.44	99.39	99.93
Discriminador	49.22	2.94	47.40	51.04

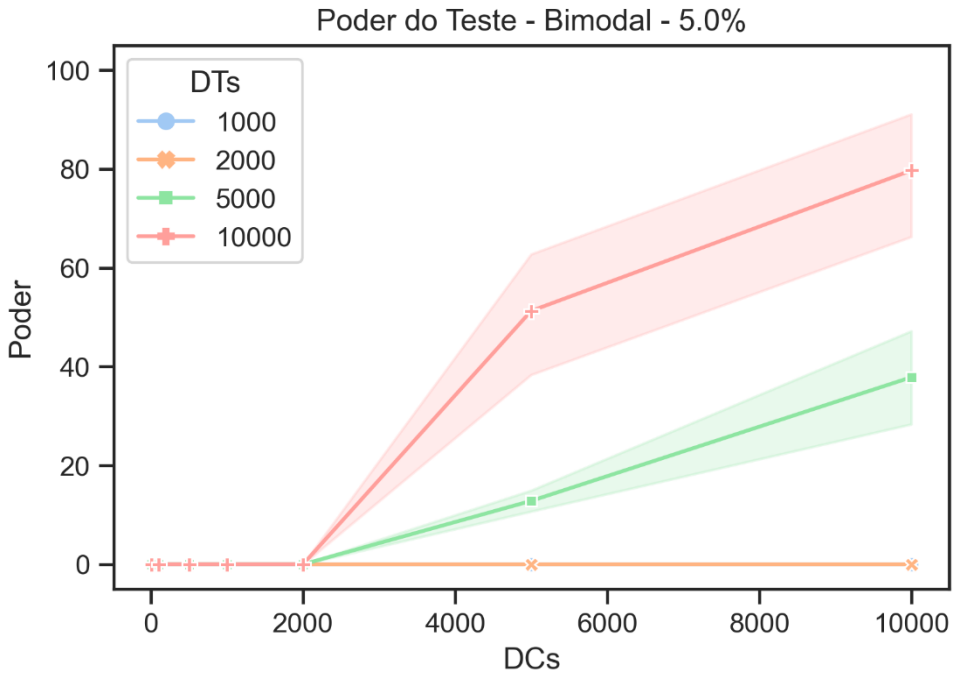

Figura C.10 - Poder do Teste - Bimodal - Tolerância 5.0%

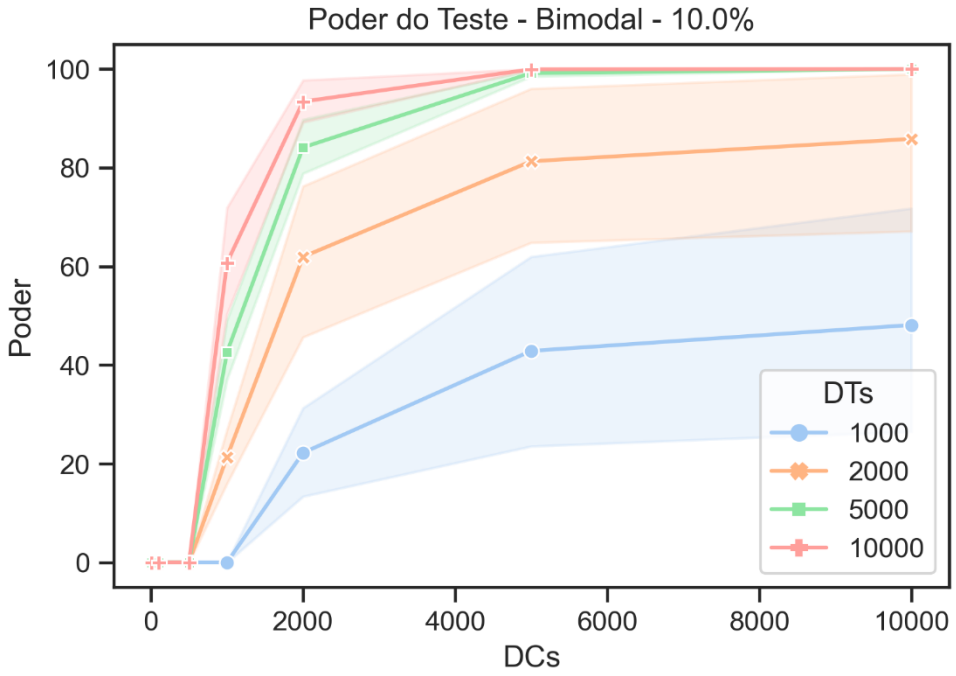

Figura C.11 - Poder do Teste - Bimodal - Tolerância 10.0%

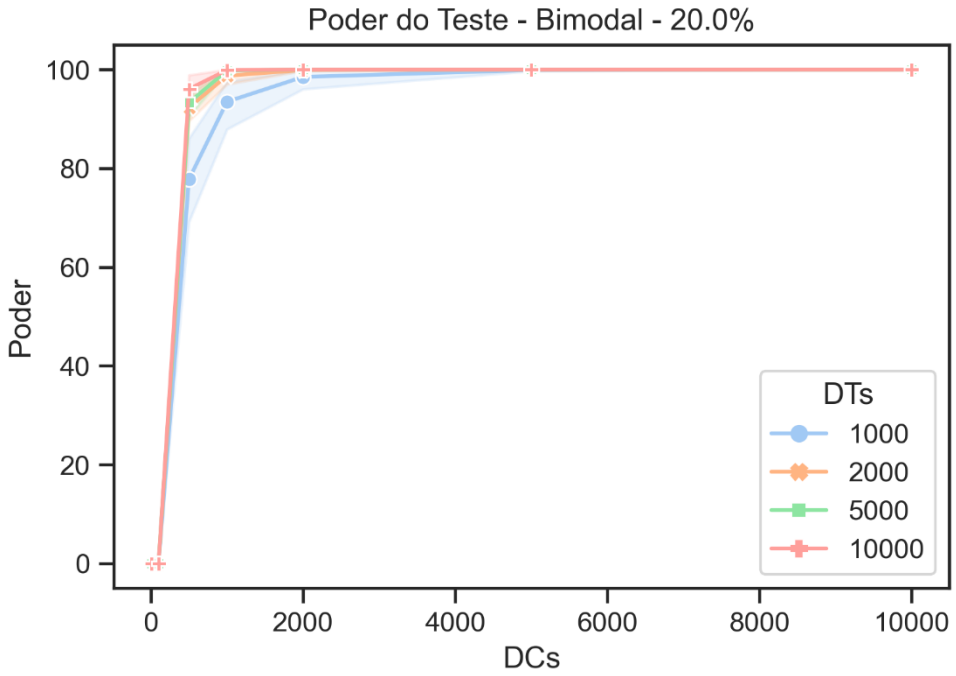

Figura C.12 - Poder do Teste - Bimodal - Tolerância 20.0%

APÊNDICE D – Distribuições contínuas multivariadas

O Apêndice D apresenta o resumo do treinamento dos 10 ciclos e as curvas de teste (tolerância 5.0%, 10.0% e 20.0%) para as distribuições contínuas multivariadas: Normal Bivariada normal, com correlação positiva, negativa e nula e Normal Multivariada (três distribuições normais) com e sem correlação.

D.1. Normal Bivariada – correlação positiva

A Tabela D.1 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Normal Bivariada com correlação positiva. As Figuras D.1, D.2 e D.3 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela D.1 - Resumo do treinamento da distribuição Normal Bivariada (pos) - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	205.00	146.15	114.41	295.59
Acurácia	96.70	1.06	96.04	97.36
Discriminador	50.33	3.39	48.23	52.43
DT – 5000 dados				
Época	315.00	88.35	260.24	369.76
Acurácia	97.05	0.59	96.68	97.42
Discriminador	51.07	3.10	49.15	52.99
DT – 2000 dados				
Época	490.00	139.04	403.82	576.18
Acurácia	97.85	1.02	97.22	98.48
Discriminador	51.48	2.34	50.02	52.93
DT – 1000 dados				
Época	990.00	358.86	767.58	1212.42
Acurácia	98.71	0.76	98.24	99.18
Discriminador	50.57	2.90	48.78	52.36

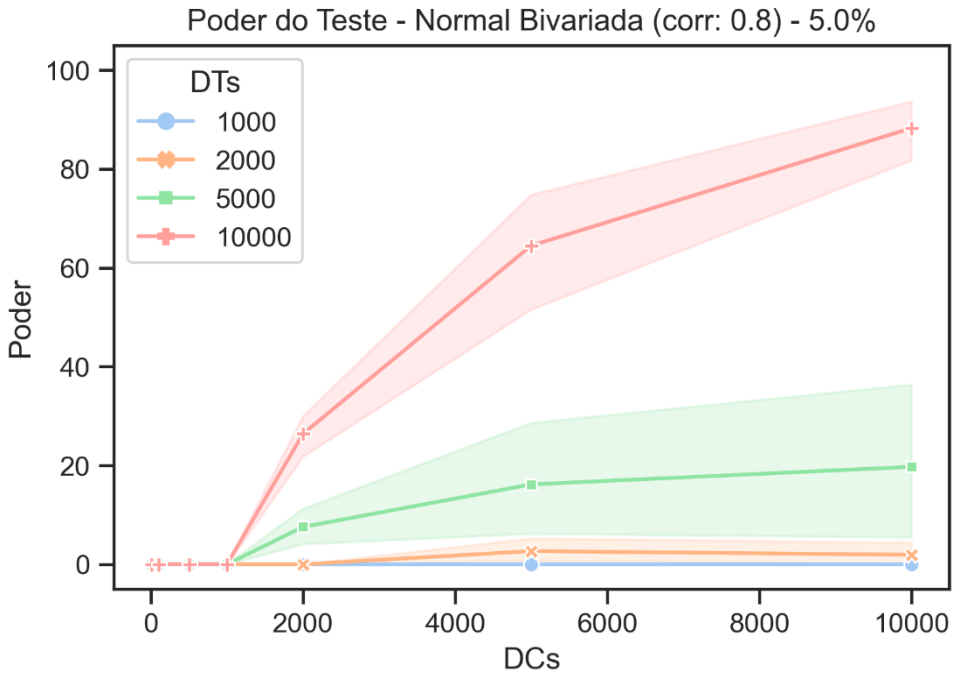

Figura D.1 - Poder do Teste - Normal Bivariada (corr positiva) - Tolerância 5.0%

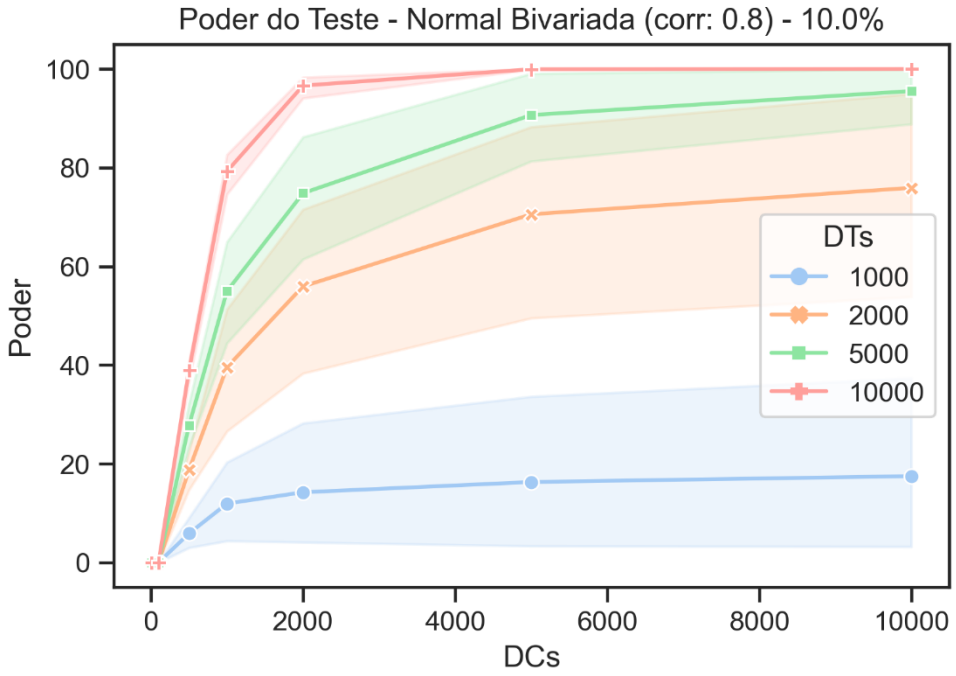

Figura D.2 - Poder do Teste - Normal Bivariada (corr positiva) - Tolerância 10.0%

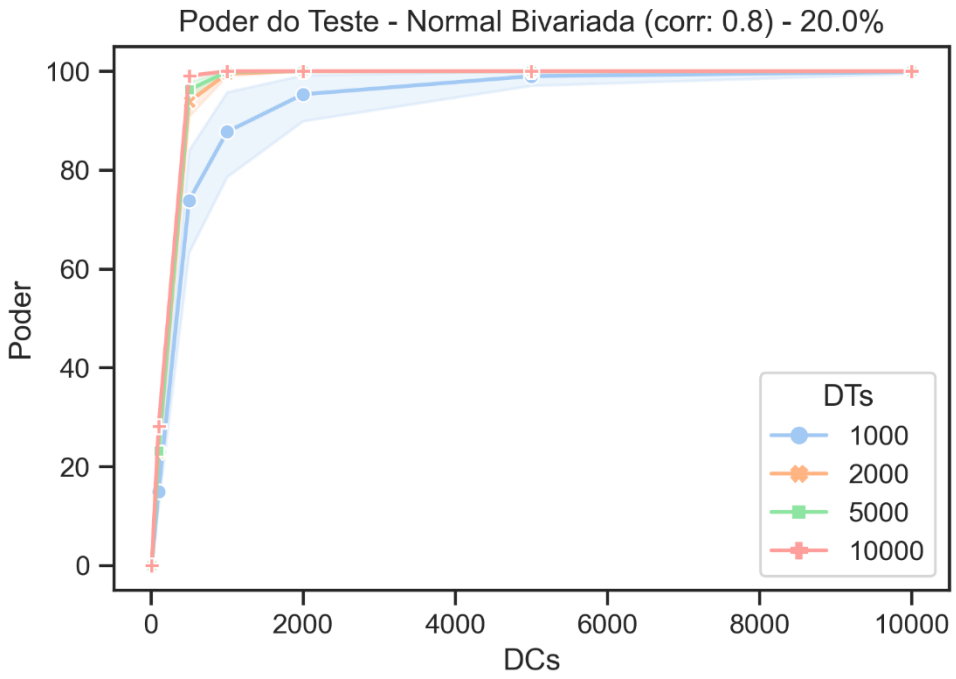

Figura D.3 - Poder do Teste - Normal Bivariada (corr positiva) - Tolerância 20.0%

D.2. Bivariada – correlação negativa

A Tabela D.2 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Normal Bivariada com correlação negativa. As Figuras D.4, D.5 e D.6 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela D.2 - Resumo do treinamento da distribuição Normal Bivariada (neg) - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	250.00	91.29	193.42	306.58
Acurácia	96.90	1.18	96.17	97.63
Discriminador	49.83	2.28	48.42	51.24
DT – 5000 dados				
Época	335.00	102.88	271.24	398.76
Acurácia	97.62	0.64	97.22	98.02
Discriminador	50.21	3.27	48.19	52.24
DT – 2000 dados				
Época	560.00	223.36	421.56	698.44
Acurácia	98.17	0.60	97.80	98.54
Discriminador	51.67	2.72	49.98	53.35
DT – 1000 dados				
Época	1140.00	651.84	736.00	1544.00
Acurácia	99.10	0.82	98.59	99.61
Discriminador	49.93	2.64	48.29	51.57

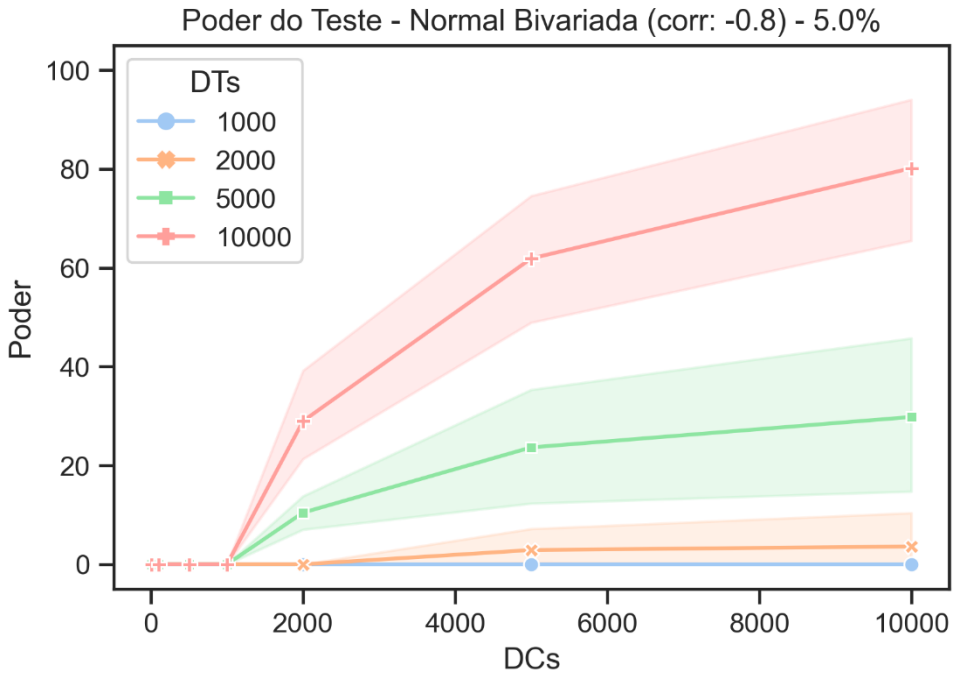

Figura D.4 - Poder do Teste - Normal Bivariada (corr negativa) - Tolerância 5.0%

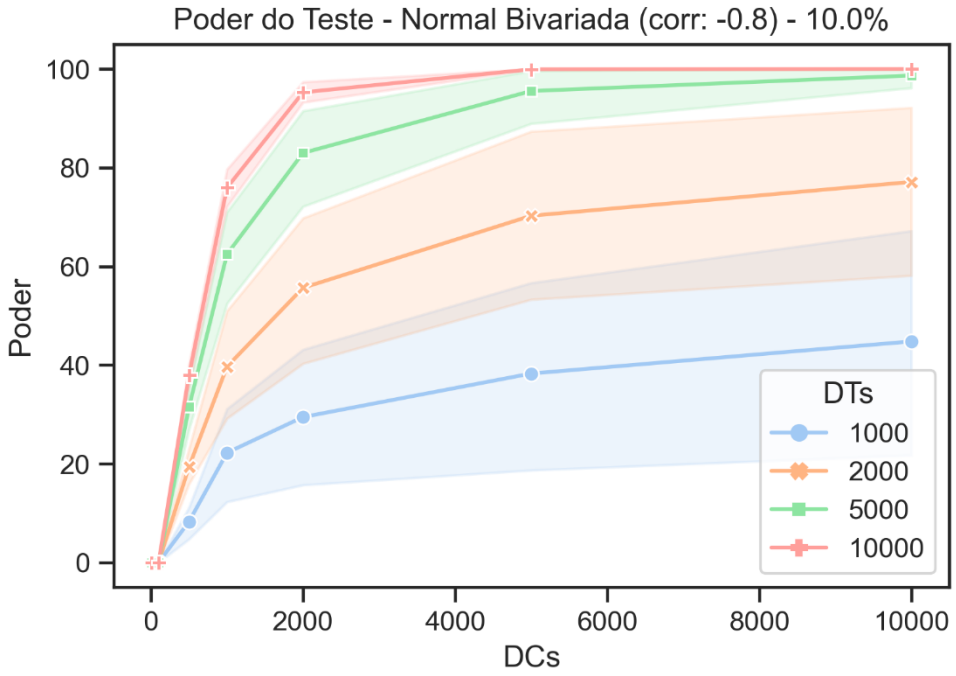

Figura D.5 - Poder do Teste - Normal Bivariada (corr negativa) - Tolerância 10.0%

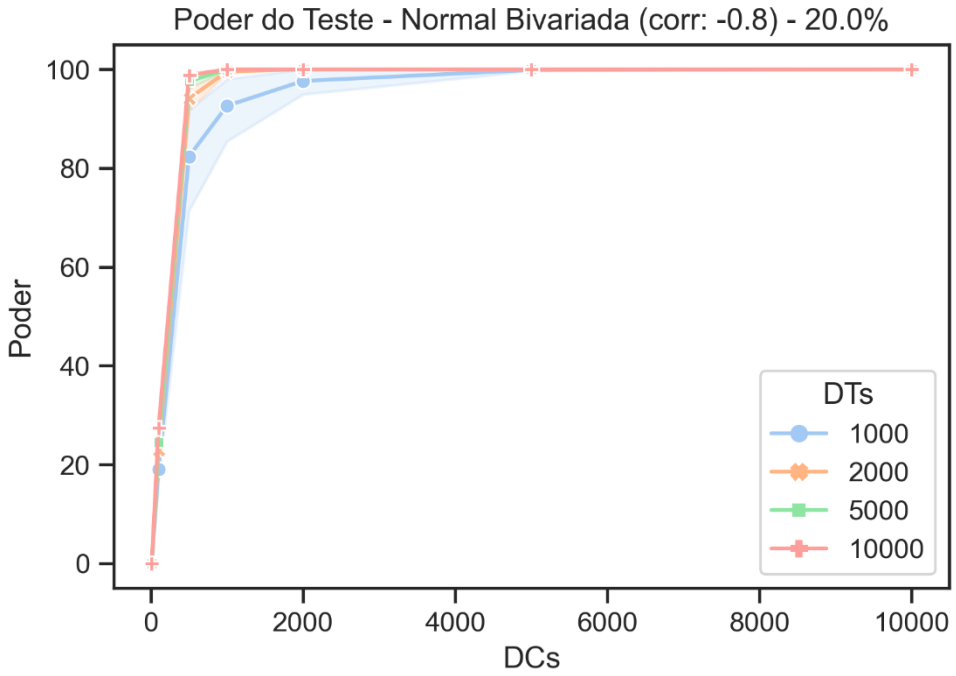

Figura D.6 - Poder do Teste - Normal Bivariada (corr negativa) - Tolerância 20.0%

D.3. Normal Bivariada – correlação nula

A Tabela D.3 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Normal Bivariada sem correlação. As Figuras D.7, D.8 e D.9 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela D.3 - Resumo do treinamento da distribuição Normal Bivariada (nula) - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	215.00	85.15	162.23	267.77
Acurácia	97.49	1.26	96.71	98.27
Discriminador	49.34	2.43	47.83	50.85
DT – 5000 dados				
Época	260.00	69.92	216.66	303.34
Acurácia	97.65	1.05	97.00	98.30
Discriminador	50.94	3.24	48.94	52.95
DT – 2000 dados				
Época	370.00	63.25	330.80	409.20
Acurácia	97.75	0.81	97.25	98.25
Discriminador	49.80	3.02	47.92	51.67
DT – 1000 dados				
Época	895.00	679.64	473.76	1316.24
Acurácia	98.68	0.96	98.09	99.27
Discriminador	51.17	2.71	49.49	52.85

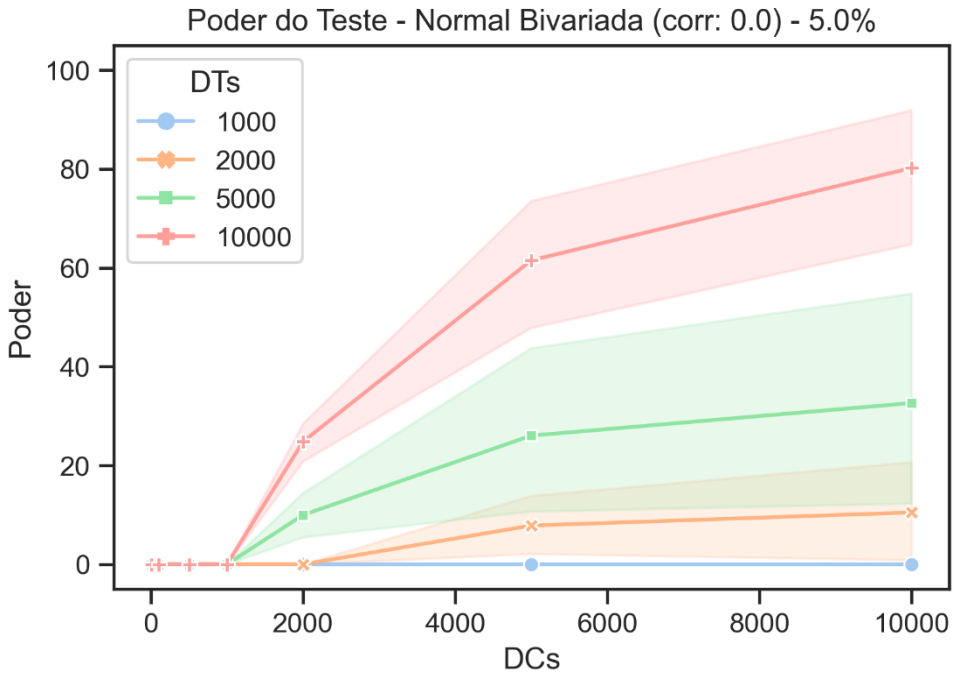

Figura D.7 - Poder do Teste - Normal Bivariada (corr nula) - Tolerância 5.0%

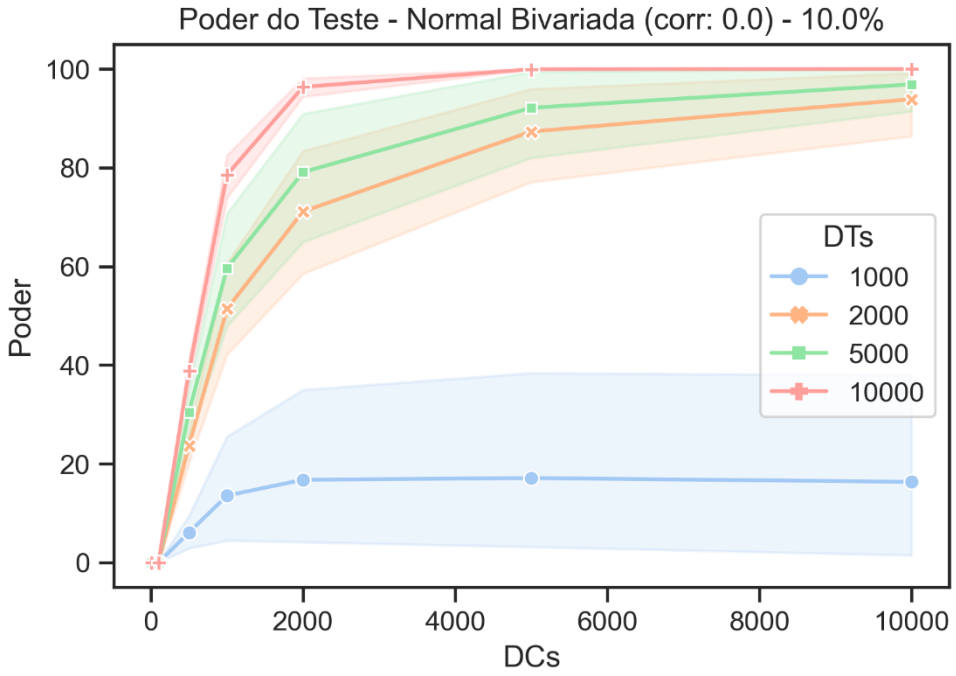

Figura D.8 - Poder do Teste - Normal Bivariada (corr nula) - Tolerância 10.0%

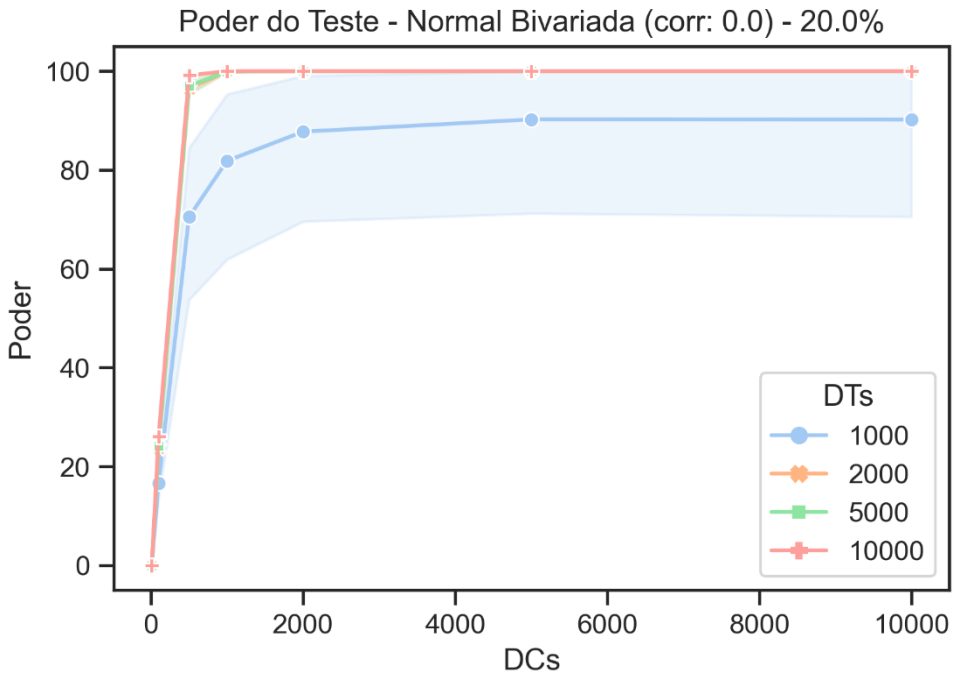

Figura D.0.9 - Poder do Teste - Normal Bivariada (corr nula) - Tolerância 20.0%

D.4. Normal Multivariada correlacionada

A Tabela D.4 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Normal Multivariada correlacionada. As Figuras D.10, D.11 e D.12 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela D.4 - Resumo do treinamento da distribuição Normal Multivariada correlacionada – 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	245.00	76.19	197.78	292.22
Acurácia	97.45	1.08	96.78	98.12
Discriminador	51.20	2.91	49.40	53.00
DT – 5000 dados				
Época	320.00	63.25	280.80	359.20
Acurácia	97.50	1.04	96.85	98.15
Discriminador	48.84	2.58	47.24	50.44
DT – 2000 dados				
Época	560.00	309.84	367.96	752.04
Acurácia	98.30	0.85	97.77	98.83
Discriminador	51.95	2.18	50.59	53.30
DT – 1000 dados				
Época	655.00	221.67	517.61	792.39
Acurácia	98.33	0.52	98.01	98.65
Discriminador	49.51	3.14	47.56	51.46

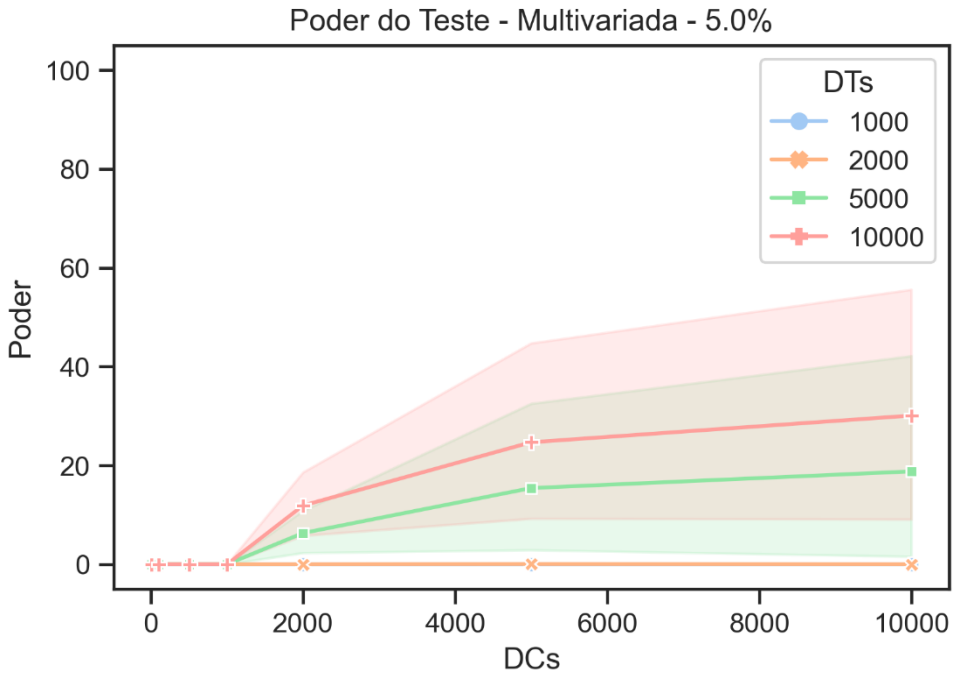

Figura D.10 - Poder do Teste - Normal Multivariada correlacionada - Tolerância 5.0%

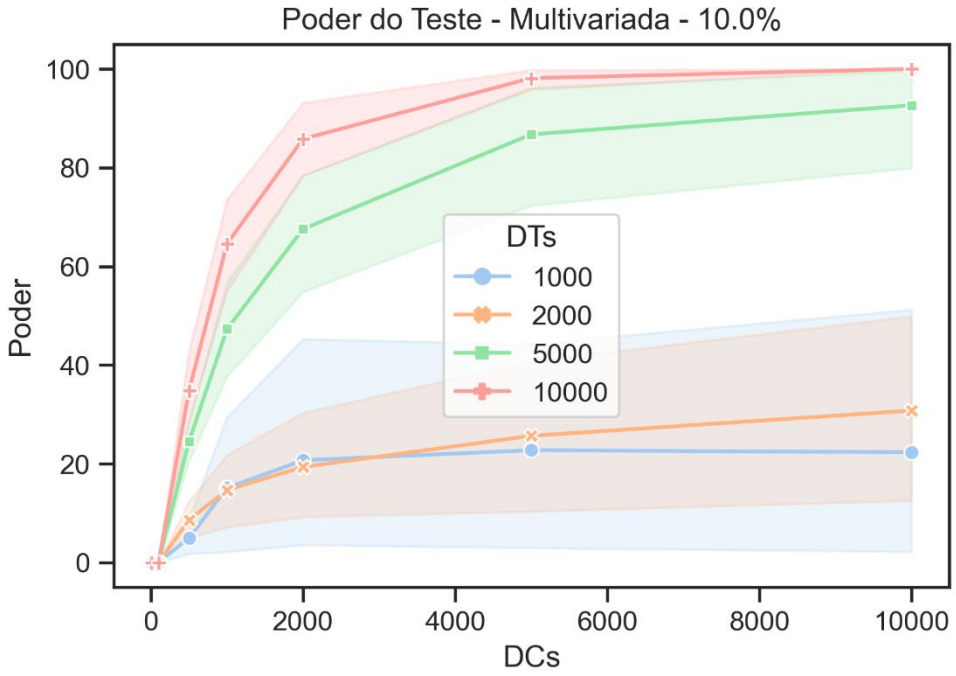

Figura D.11 - Poder do Teste - Normal Multivariada correlacionada - Tolerância 10.0%

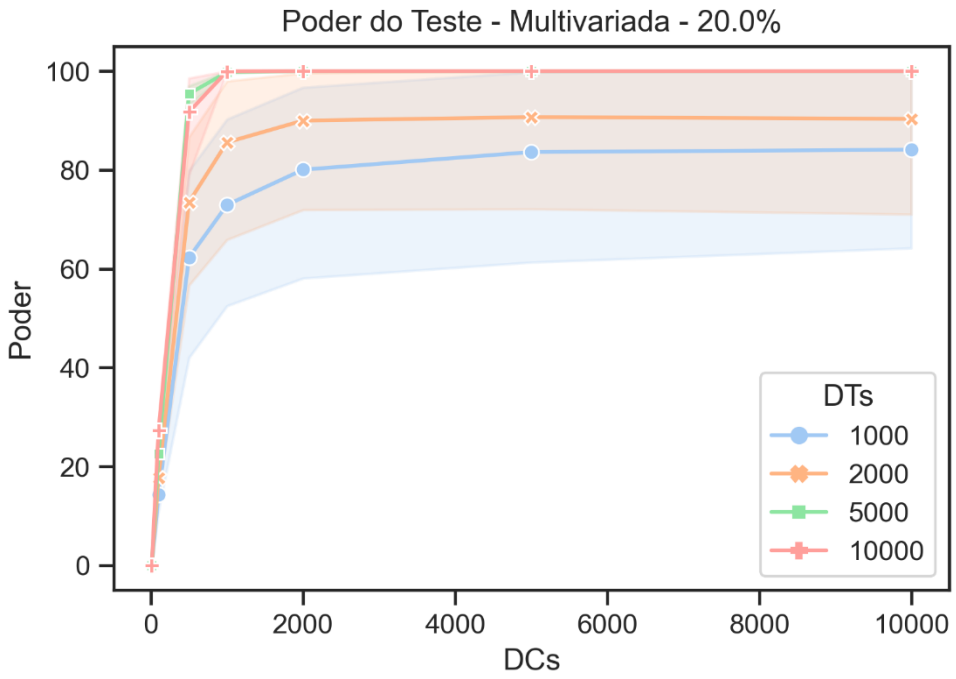

Figura D.12 - Poder do Teste - Normal Multivariada correlacionada - Tolerância 20.0%

D.5. Normal Multivariada não correlacionada

A Tabela D.5 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Multivariada não correlacionada. As Figuras D.13, D.14 e D.15 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela D.5 - Resumo do treinamento da distribuição Normal Multivariada não correlacionada - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	225.00	92.04	167.95	282.05
Acurácia	97.62	1.30	96.81	98.43
Discriminador	51.20	2.20	49.84	52.56
DT – 5000 dados				
Época	230.00	75.28	183.34	276.66
Acurácia	97.02	0.54	96.68	97.36
Discriminador	49.74	2.57	48.15	51.34
DT – 2000 dados				
Época	325.00	63.46	285.66	364.34
Acurácia	98.17	1.05	97.52	98.82
Discriminador	49.26	2.36	47.80	50.72
DT – 1000 dados				
Época	550.00	217.31	415.31	684.69
Acurácia	98.77	0.74	98.31	99.23
Discriminador	51.01	2.39	49.53	52.49

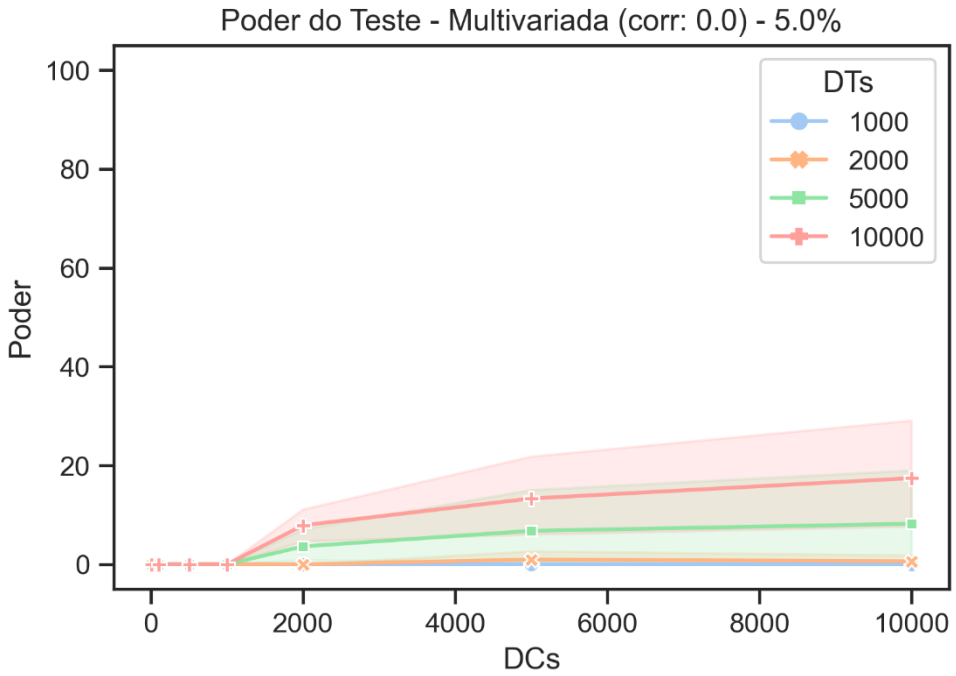

Figura D.13 - Poder do Teste - Normal Multivariada não correlacionada - Tolerância 5.0%

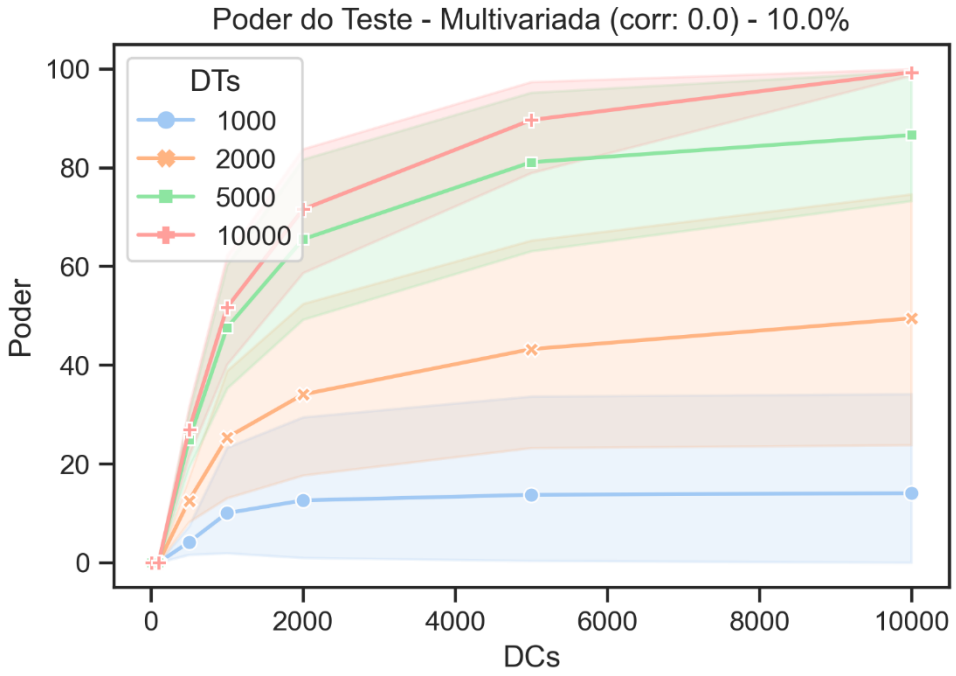

Figura D.14 - Poder do Teste - Normal Multivariada não correlacionada - Tolerância 10.0%

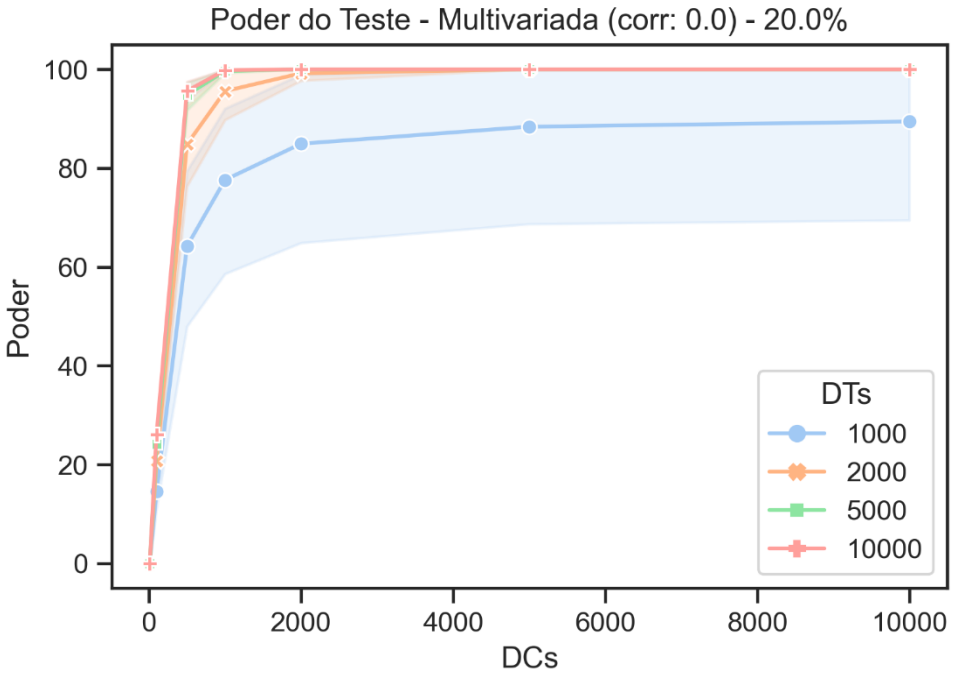

Figura D.15 - Poder do Teste - Normal Multivariada não correlacionada - Tolerância 20.0%

APÊNDICE E – Distribuições discretas

O Apêndice E apresenta o resumo do treinamento dos 10 ciclos e as curvas de teste (tolerância 5.0%, 10.0% e 20.0%) para as distribuições discretas: Poisson e Binomial.

E.1. Poisson

A Tabela E.1 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Poisson. As Figuras E.1, E.2 e E.3 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela E.1 - Resumo do treinamento da distribuição Poisson - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	270.00	181.35	157.60	382.40
Acurácia	96.80	1.34	95.97	97.63
Discriminador	49.54	2.79	47.81	51.27
DT – 5000 dados				
Época	465.00	307.36	274.50	655.50
Acurácia	97.26	0.76	96.79	97.73
Discriminador	49.73	2.44	48.22	51.25
DT – 2000 dados				
Época	585.00	325.79	383.08	786.92
Acurácia	97.79	0.94	97.20	98.38
Discriminador	50.95	3.28	48.91	52.98
DT – 1000 dados				
Época	1365.00	873.07	823.88	1906.12
Acurácia	98.21	0.63	97.82	98.60
Discriminador	49.27	3.57	47.06	51.48

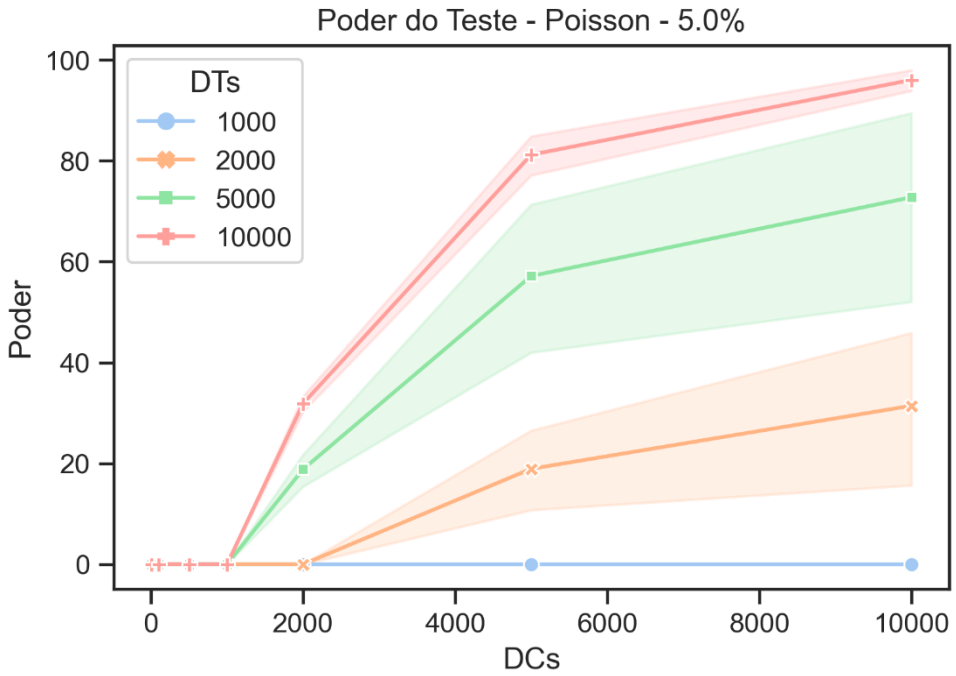

Figura E.1 - Poder do Teste - Poisson - Tolerância 5.0%

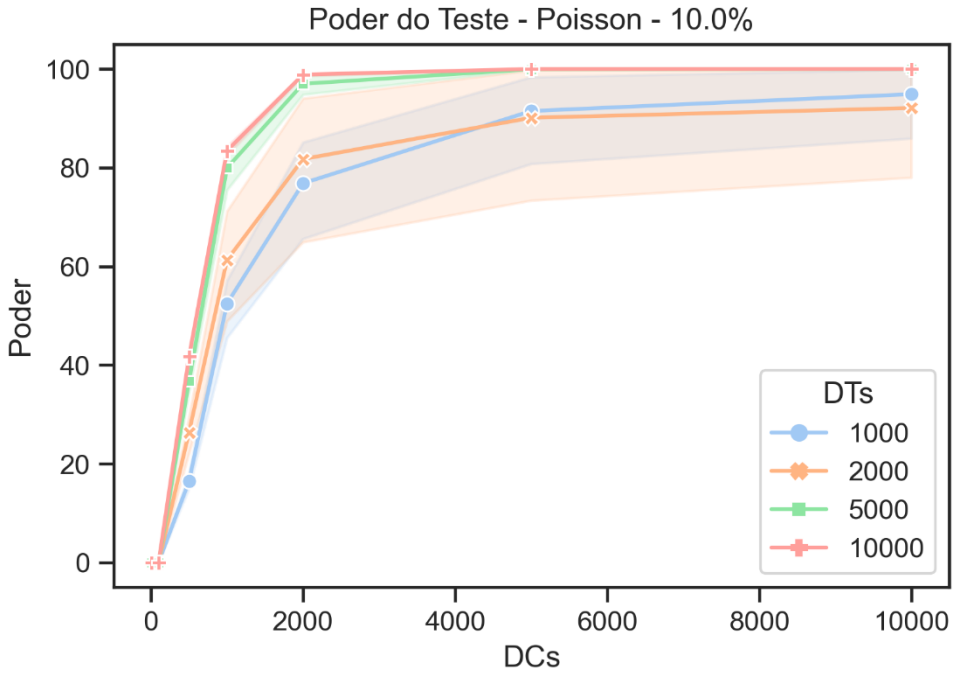

Figura E.2 - Poder do Teste - Poisson - Tolerância 10.0%

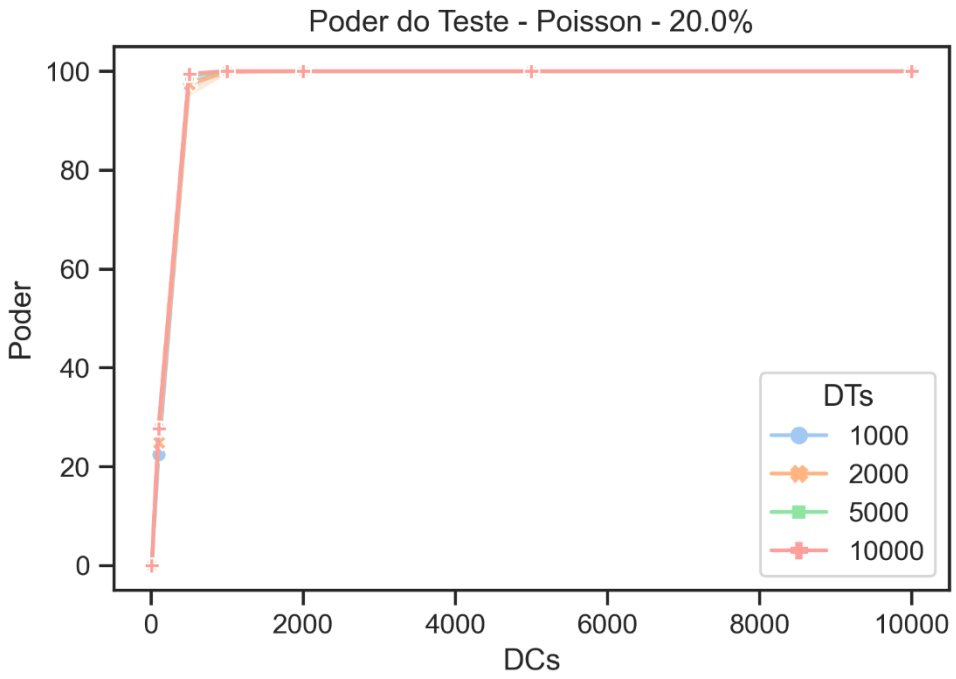

Figura E.3 - Poder do Teste - Poisson - Tolerância 20.0%

E.2. Binomial

A Tabela E.2 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para a distribuição Binomial. As Figuras E.4, E.5 e E.6 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela E.2 - Resumo do treinamento da distribuição Binomial - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	490.00	641.09	92.65	887.35
Acurácia	96.50	0.52	96.18	96.82
Discriminador	49.15	3.31	47.10	51.20
DT – 5000 dados				
Época	315.00	226.14	174.84	455.16
Acurácia	97.30	0.85	96.77	97.83
Discriminador	51.30	2.41	49.81	52.79
DT – 2000 dados				
Época	1070.00	1197.27	327.94	1812.06
Acurácia	98.02	0.82	97.51	98.53
Discriminador	50.03	2.13	48.71	51.35
DT – 1000 dados				
Época	1430.00	1278.93	637.32	2222.68
Acurácia	98.72	0.88	98.17	99.27
Discriminador	49.69	3.28	47.66	51.72

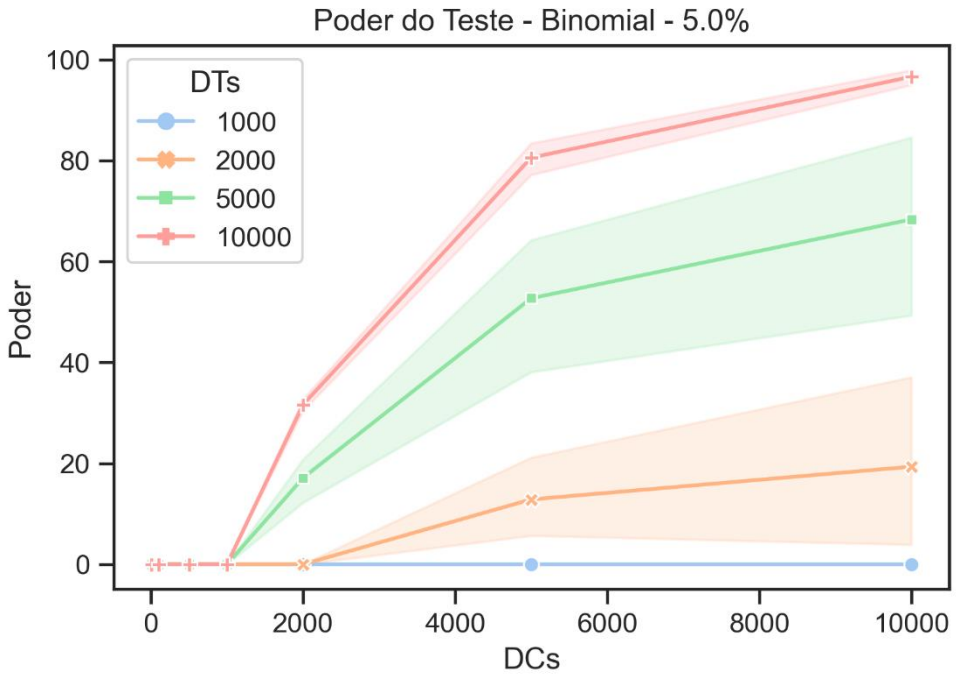

Figura E.4 - Poder do Teste - Binomial - Tolerância 5.0%

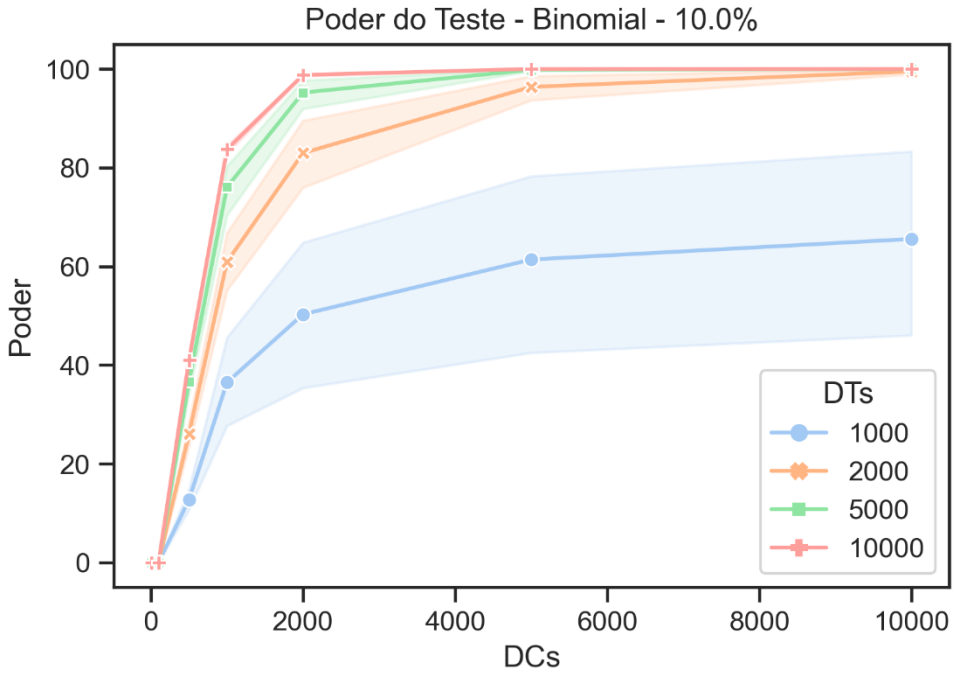

Figura E.5 - Poder do Teste - Binomial - Tolerância 10.0%

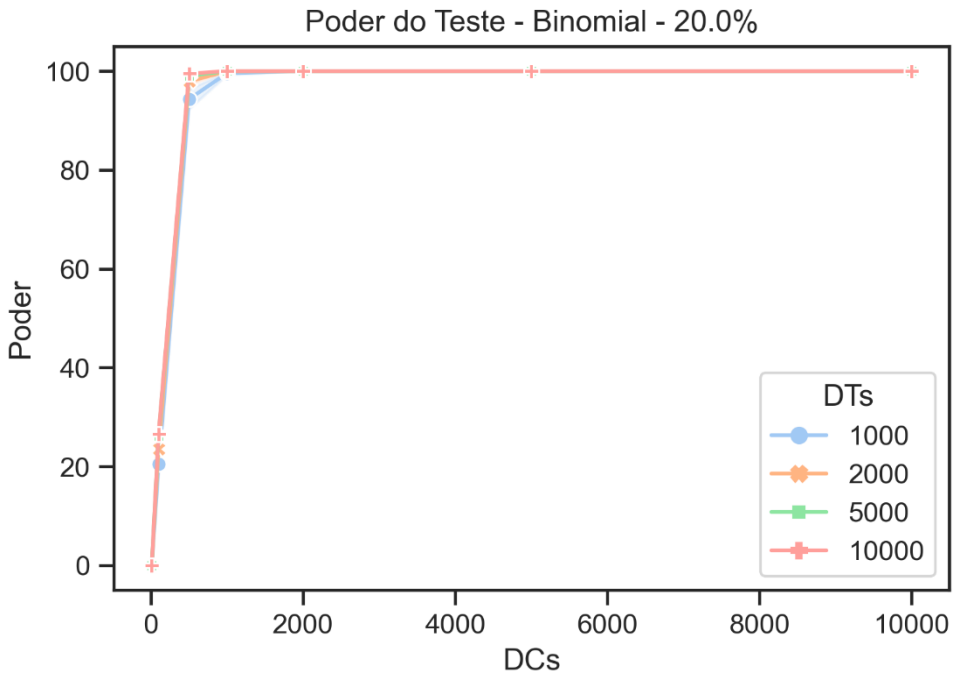

Figura E.6 - Poder do Teste - Binomial - Tolerância 20.0%

APÊNDICE F – Distribuições com dados condicionais

O Apêndice F apresenta o resumo do treinamento dos 10 ciclos e as curvas de teste (tolerância 5.0%, 10.0% e 20.0%) para as distribuições que possuem dados condicionais: Weibul e Beta e as duas distribuições Triangulares.

F.1. Weibul e Beta

A Tabela F.1 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para as distribuições Weibul e Beta com os dados condicionais. As Figuras F.1, F.2 e F.3 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela F.1 - Resumo do treinamento das distribuições Weibul e Beta com dados condicionais - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	2075.00	1776.90	973.69	3176.31
Acurácia	96.71	1.17	95.99	97.43
Discriminador	48.77	2.39	47.28	50.25
DT – 5000 dados				
Época	2520.00	4342.57	-171.50	5211.50
Acurácia	97.12	0.65	96.72	97.52
Discriminador	48.72	2.27	47.31	50.12
DT – 2000 dados				
Época	3490.00	4159.45	912.00	6068.00
Acurácia	97.23	0.84	96.71	97.75
Discriminador	48.45	3.01	46.59	50.32
DT – 1000 dados				
Época	2445.00	4421.88	-295.66	5185.66
Acurácia	98.59	1.12	97.89	99.29
Discriminador	49.99	1.70	48.94	51.04

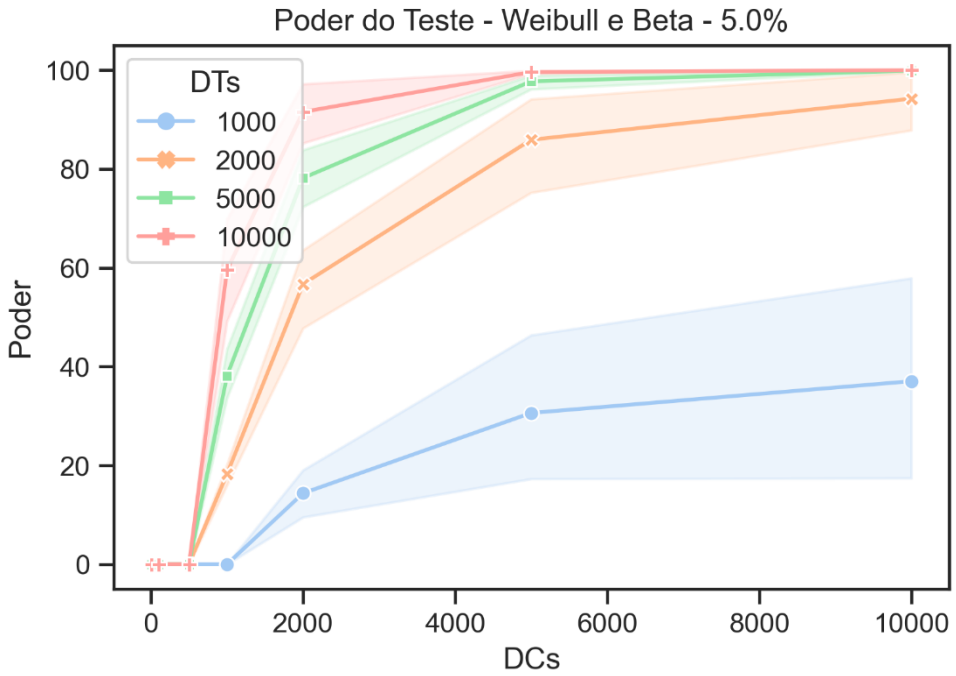

Figura F.1 - Poder do Teste - Weibul e Beta - Tolerância 5.0%

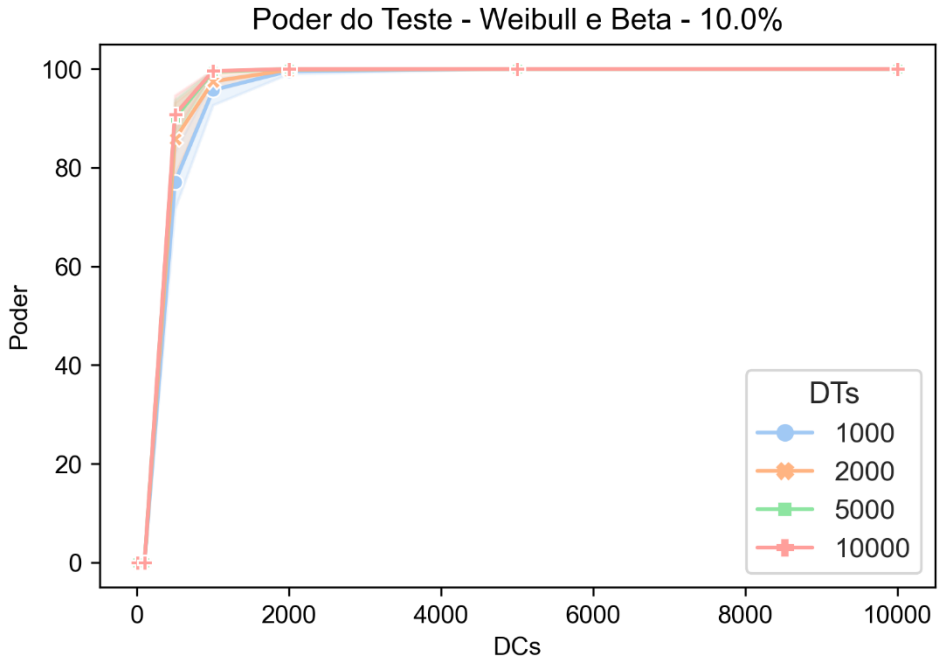

Figura F.2 - Poder do Teste - Weibul e Beta - Tolerância 10.0%

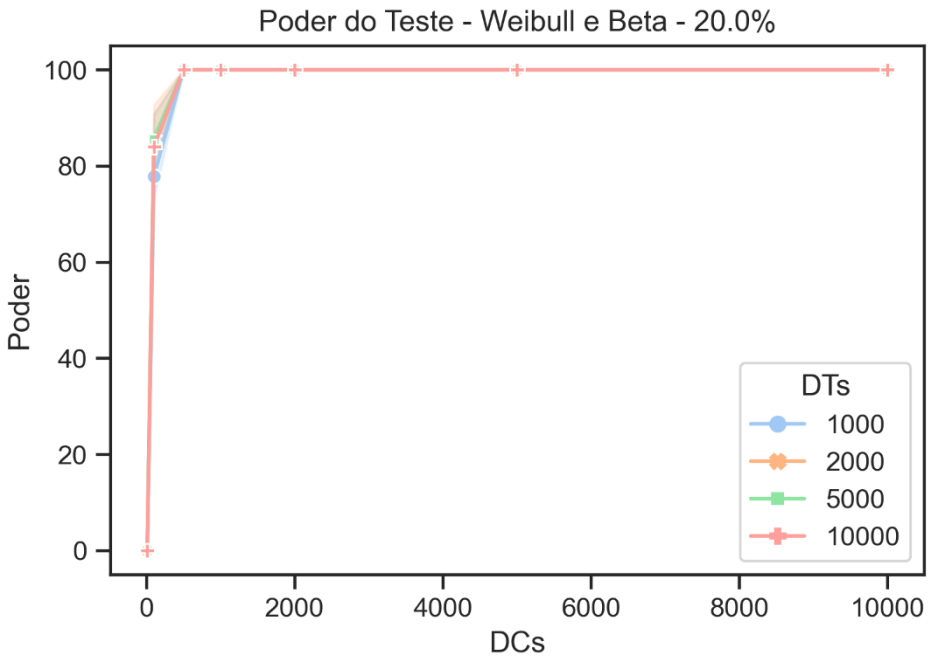

Figura F.3 - Poder do Teste - Weibul e Beta - Tolerância 20.0%

F.2. Triangular

A Tabela F.2 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para as distribuições Triangulares com os dados condicionais. As Figuras F.4, F.5 e F.6 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela F.2 - Resumo do treinamento das distribuições Triangulares com dados condicionais - 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	3170.00	2426.84	1665.86	4674.14
Acurácia	96.53	0.99	95.92	97.14
Discriminador	48.82	1.47	47.91	49.73
DT – 5000 dados				
Época	3865.00	4469.72	1094.69	6635.31
Acurácia	96.54	0.47	96.25	96.83
Discriminador	49.62	3.83	47.25	52.00
DT – 2000 dados				
Época	4110.00	4595.58	1261.68	6958.32
Acurácia	97.35	0.92	96.78	97.92
Discriminador	50.10	3.24	48.09	52.11
DT – 1000 dados				
Época	1780.00	1702.32	724.91	2835.09
Acurácia	97.78	0.54	97.45	98.11
Discriminador	50.56	2.90	48.76	52.36

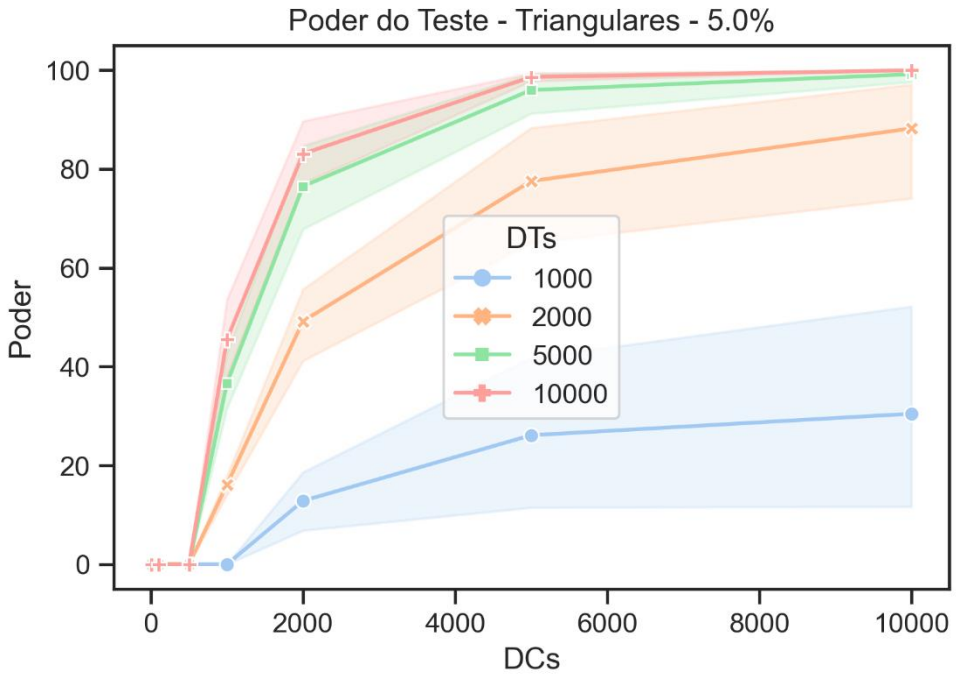

Figura F.4 - Poder do Teste - Triangulares - Tolerância 5.0%

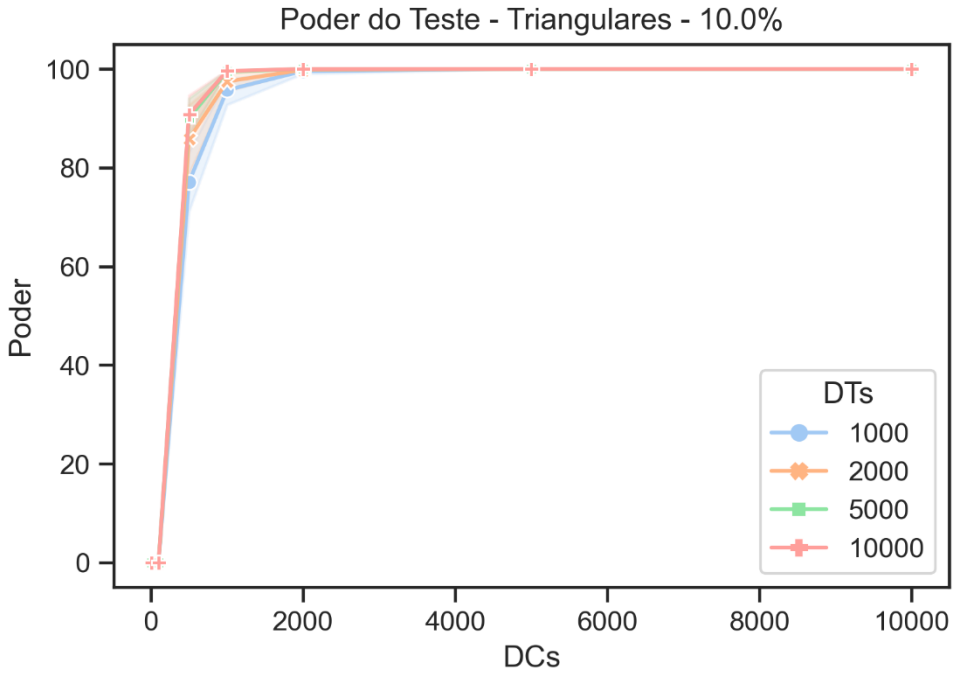

Figura F.5 - Poder do Teste - Triangulares - Tolerância 10.0%

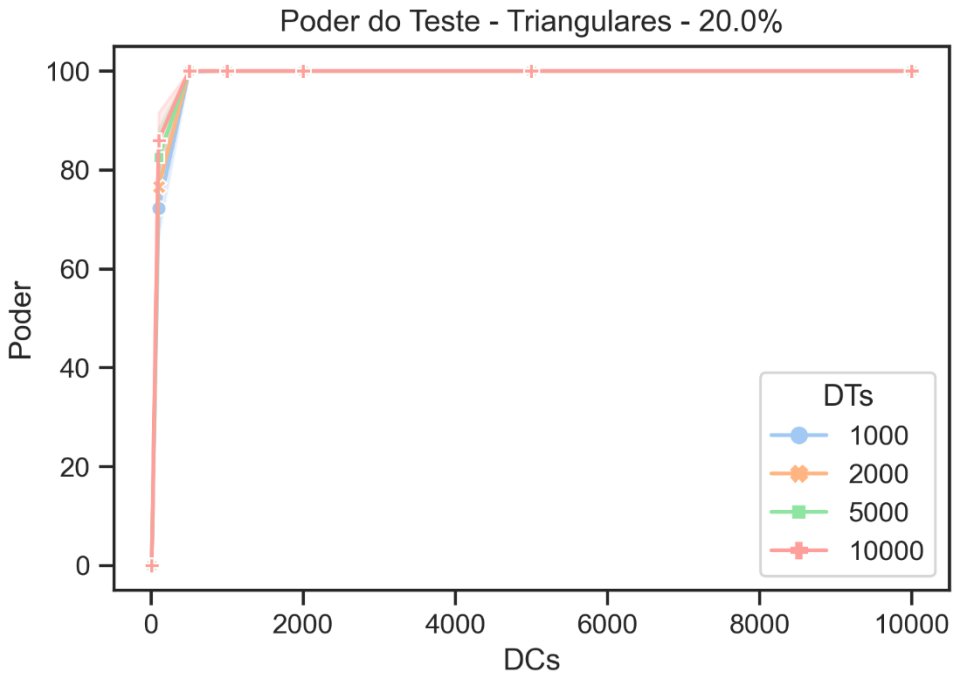

Figura F.6 - Poder do Teste - Triangulares - Tolerância 20.0%

APÊNDICE G – Modelo de Simulação

O Apêndice G apresenta o resumo do treinamento dos 10 ciclos e as curvas de teste (tolerância 5.0%, 10.0% e 20.0%) para o modelo de simulação do hospital que tem como objetivo atender pacientes com suspeita de COVID-19.

A Tabela G.1 resume as condições de parada para o treinamento das cGANs com a média, desvio padrão e o intervalo de confiança (limite superior e limite inferior) para o TPT e LOS. As Figuras G.1, G.2 e G.3 representam as curvas do Poder do Teste para tolerância de 5.0%, 10.0% e 20.0%, respectivamente.

Tabela G.1 - Resumo do treinamento para o modelo de Simulação (TPT e LOS) – 10 ciclos

		Desv. Pad	**Inferior**	**Superior**
DT – 10000 dados				
Época	825.00	473.32	531.64	1118.36
Acurácia	97.00	0.80	96.50	97.50
Discriminador	50.38	2.78	48.66	52.11
DT – 5000 dados				
Época	810.00	397.07	563.90	1056.10
Acurácia	97.63	1.04	96.98	98.28
Discriminador	48.53	2.50	46.98	50.07
DT – 2000 dados				
Época	1840.00	1092.86	1162.65	2517.35
Acurácia	97.50	0.61	97.12	97.88
Discriminador	48.11	2.19	46.76	49.46
DT – 1000 dados				
Época	2900.00	1717.56	1835.47	3964.53
Acurácia	98.42	0.89	97.87	98.97
Discriminador	49.89	3.53	47.70	52.08

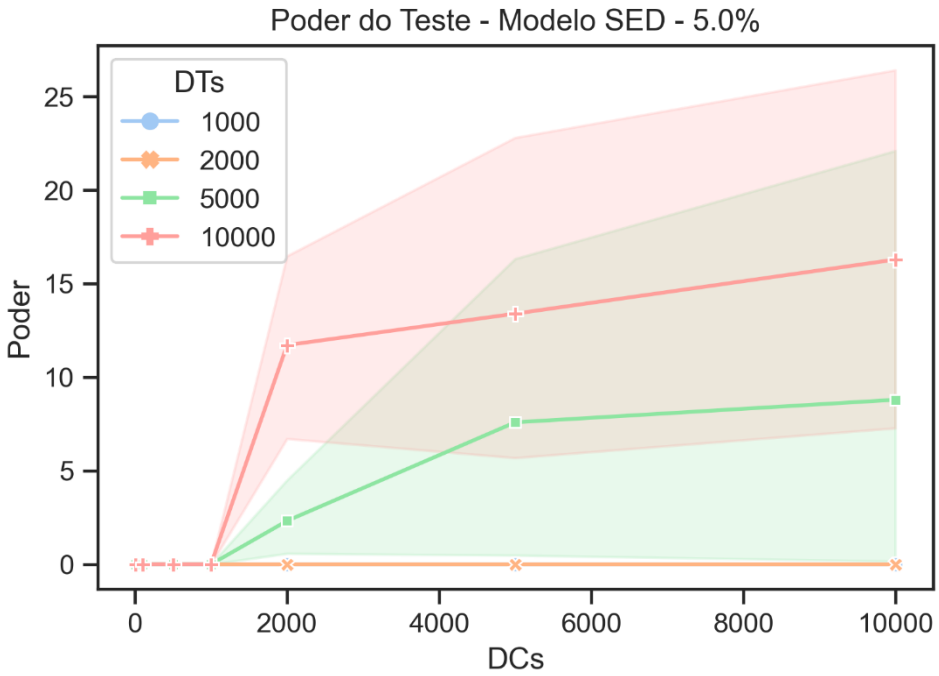

Figura G.1 - Poder do Teste – Modelos SED - Tolerância 5.0%

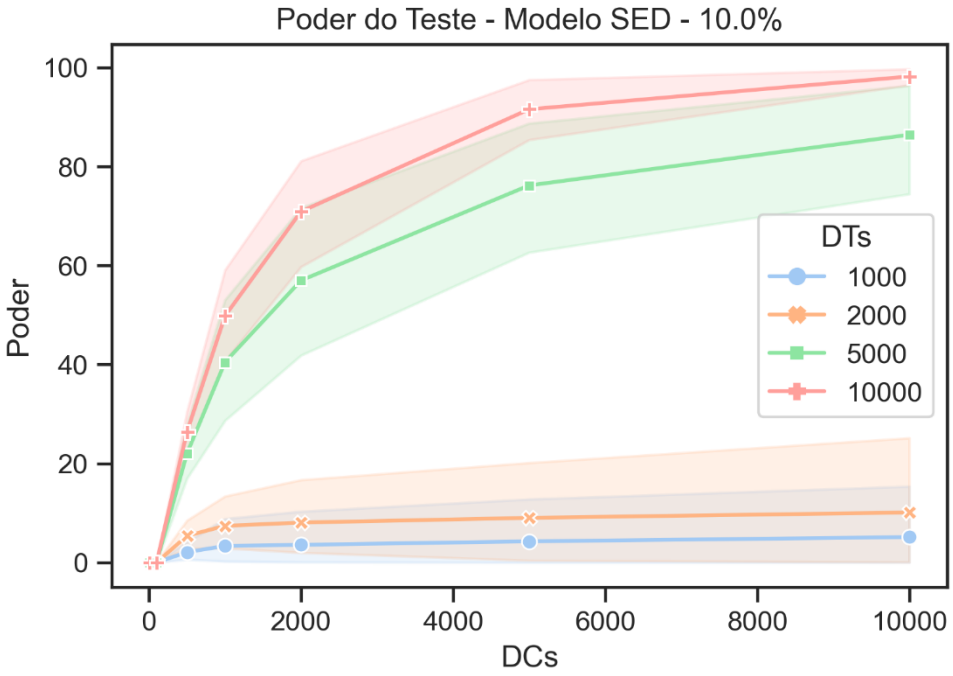

Figura G.2 - Poder do Teste – Modelos SED - Tolerância 10.0%

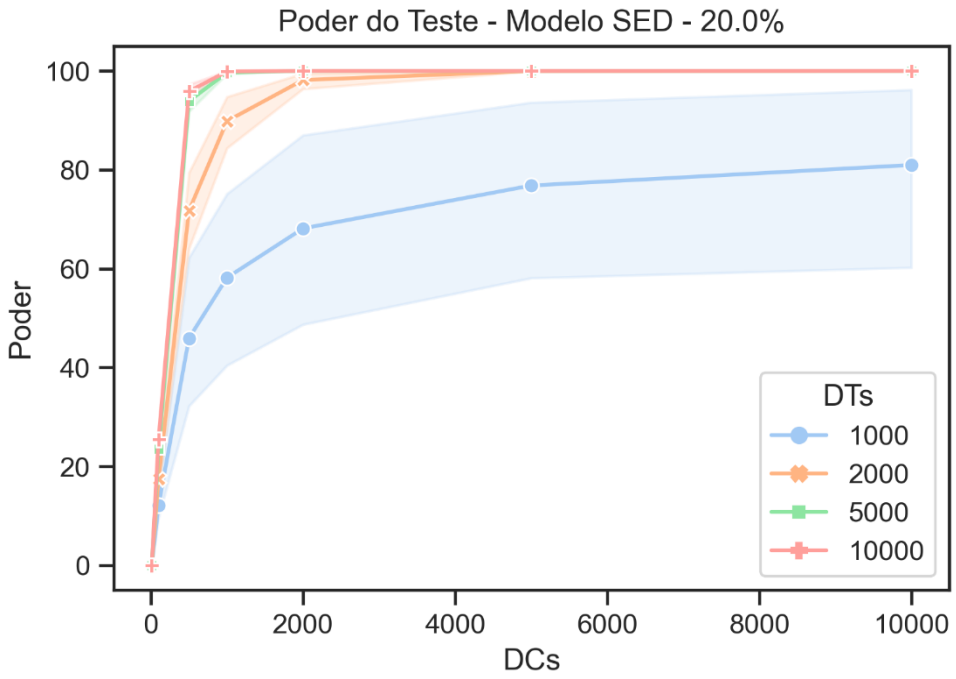

Figura G.3 - Poder do Teste – Modelos SED - Tolerância 20.0%

APÊNDICE H – Testes de Equivalência dos modelos

O Apêndice H apresenta os TEDDP para os modelos reais, sendo eles o modelo da Linha de Montagem, o modelo da Confecção de Roupas e o modelo do Hospital.

H.1. Modelo 1 – Linha de Montagem

A seção mostra a validação para o Modelo 1. A Figura H.1 mostra o TEDDP para os dados do modelo da linha de montagem, considerando como saída o *lead time*.

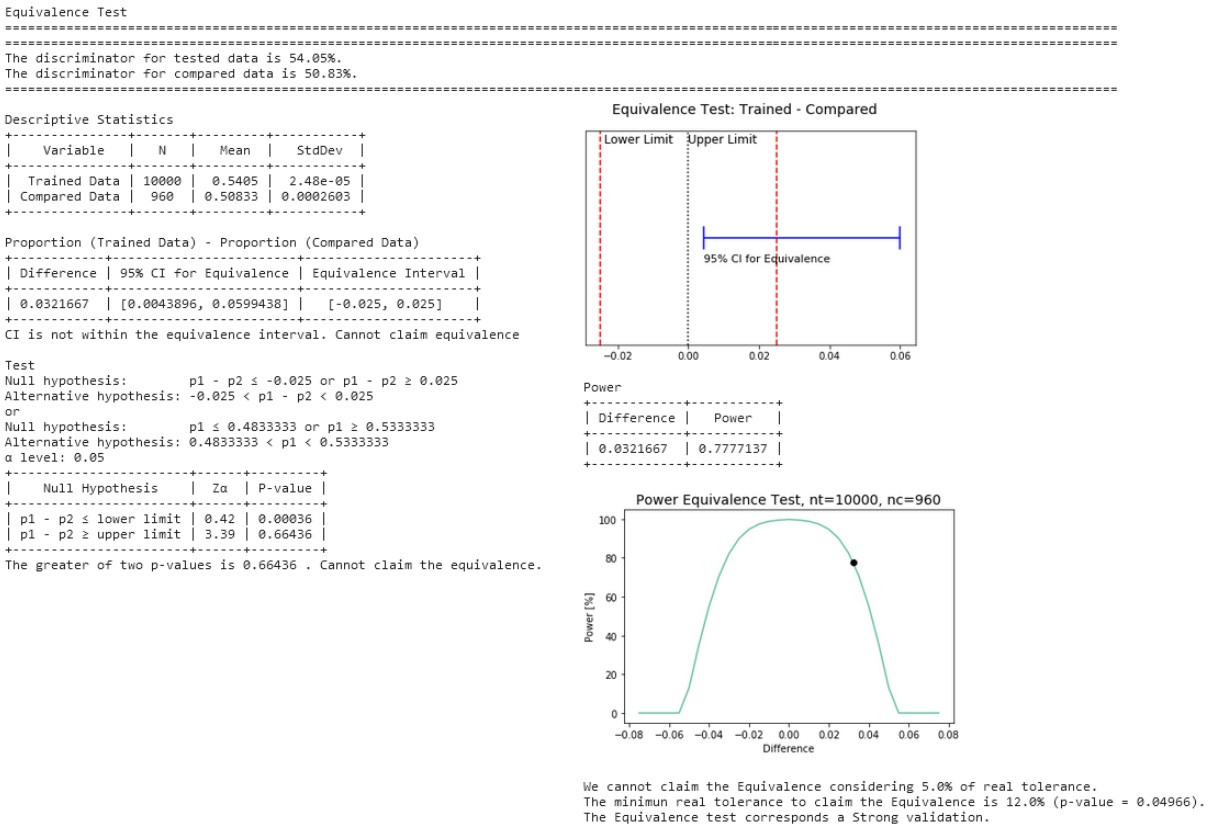

Figura H.1 - Teste Equivalência – Modelo 1 – *Lead Time*

H.2. Modelo 2 – Confecção de roupas

A seção mostra a validação para o Modelo 2, considerando as saídas múltiplas e individuais.

H.2.1. Modelo 2 – Validação Estoque 2 e Estoque 3

A Figura H.2 mostra o TEDDP para os dados do modelo da confecção de roupas, considerando como saída o tempo em que as peças ficam no Estoque 2 e Estoque 3.

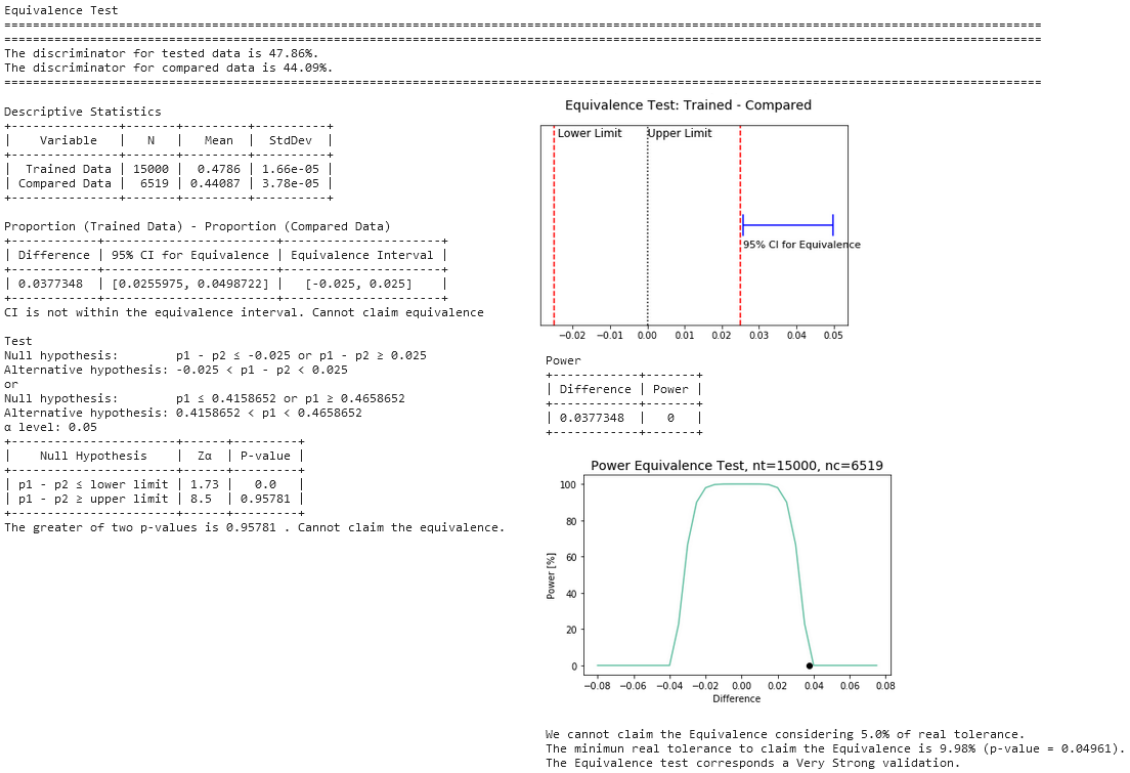

Figura H.2 - Teste Equivalência – Modelo 2 – Estoque 2 e Estoque 3

H.2.2. Modelo 2 – Validação Estoque 2

A Figura H.3 mostra o TEDDP para os dados do modelo da confecção de roupas, considerando como saída o tempo em que as peças ficam no Estoque 2.

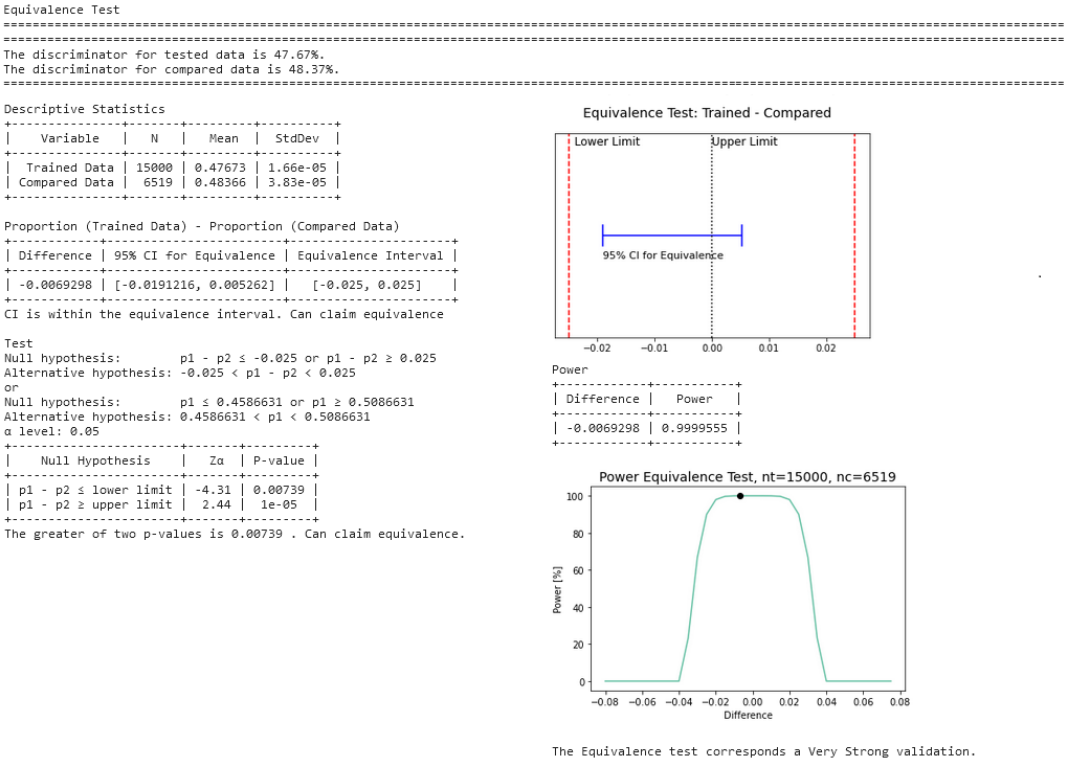

Figura H.3 - Teste Equivalência – Modelo 2 – Estoque 2

H.2.3. Modelo 2 – Validação Estoque 3

A Figura H.4 mostra o TEDDP para os dados do modelo da confecção de roupas, considerando como saída o tempo em que as peças ficam no Estoque 3.

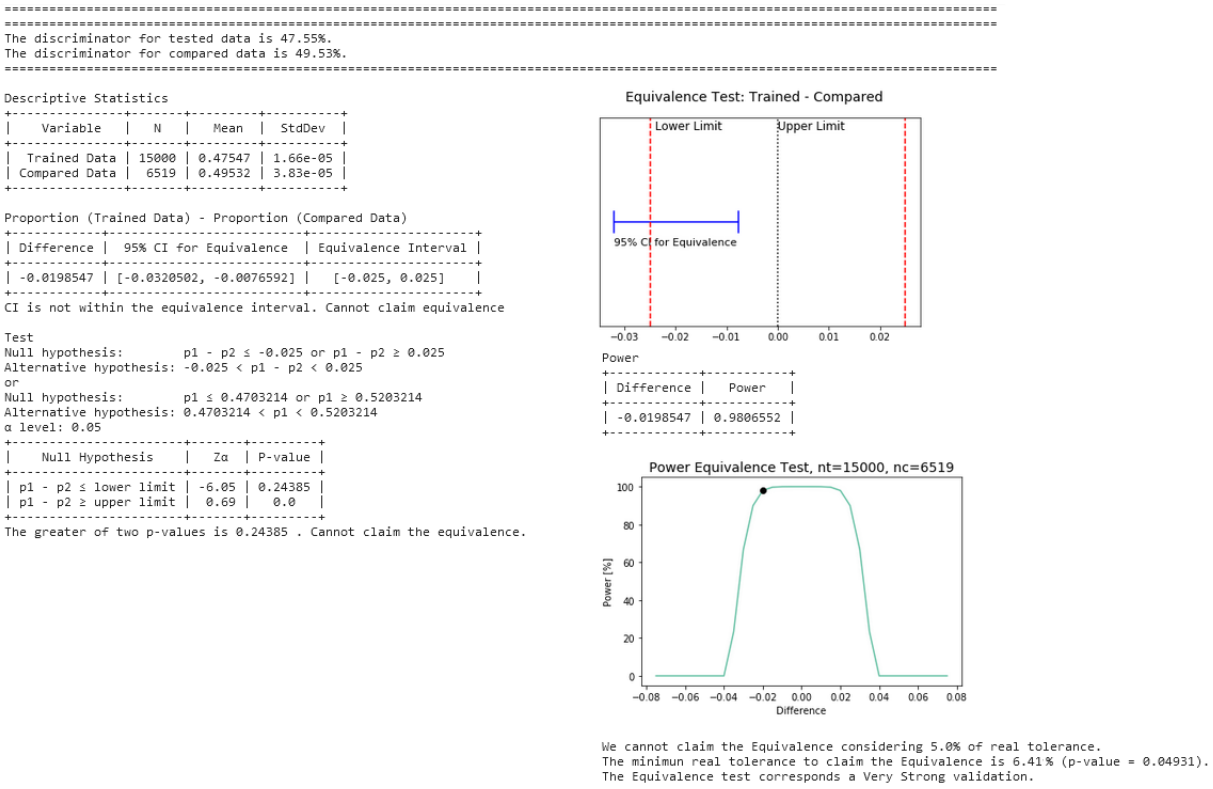

Figura H.4 - Teste Equivalência – Modelo 2 – Estoque 3

H.3. Modelo 3 – Hospital

A seção mostra a validação para o Modelo 3, considerando a validação do modelo para o paciente do tipo 2, onde foram validadas o TPT, TPM e LOS de forma conjunta e individuais. Além disso, a seção mosta a validação do LOS para todos os 5 tipos de pacientes de forma condicional e individual.

H.3.1. Modelo 3 – Validação Paciente Tipo 2

H.3.1.1. Modelo 3 – Validação Paciente Tipo 2 – TPT, TPM, LOS

A Figura H.5 mostra o TEDDP para os dados do modelo do hospital para o paciente do tipo 2, considerando como saída o TPT, TPM e LOS.

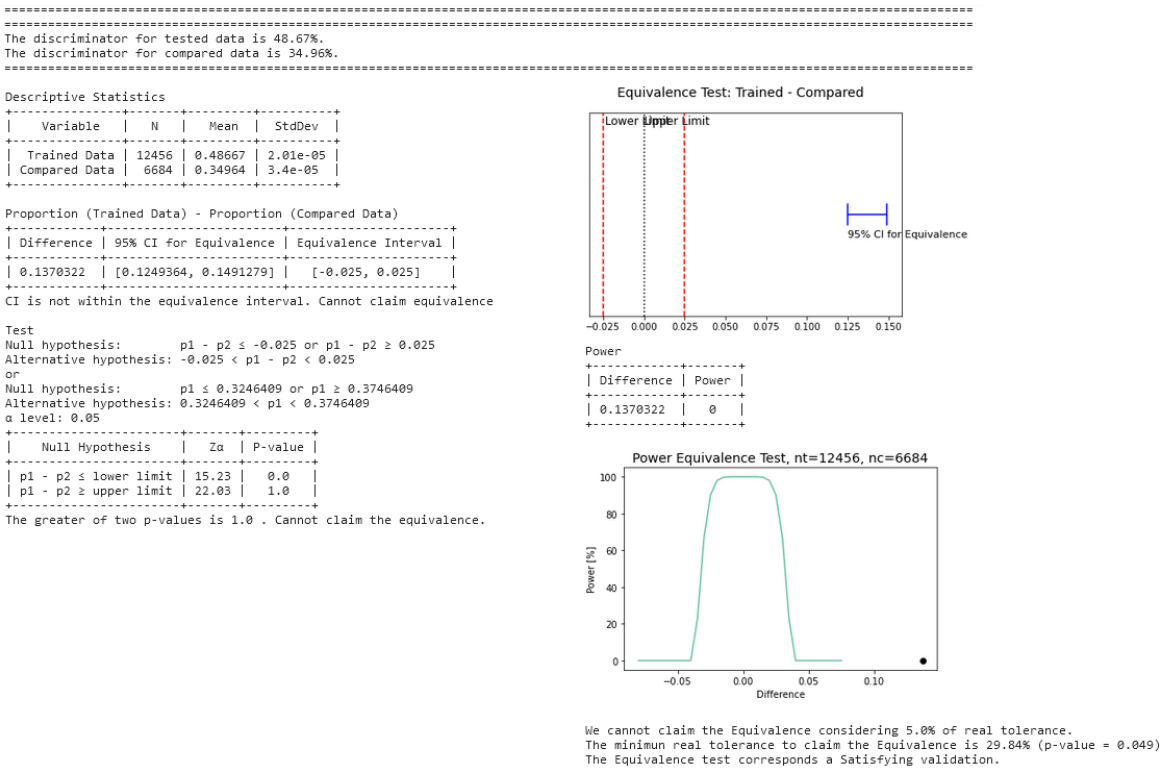

Figura H.5 - Teste Equivalência – Modelo 3 – TPT, TPM, LOS

H.3.1.2. Modelo 3 – Validação Paciente Tipo 2 – TPT

A Figura H.6 mostra o TEDDP para os dados do modelo do hospital para o paciente do tipo 2, considerando como saída o TPT.

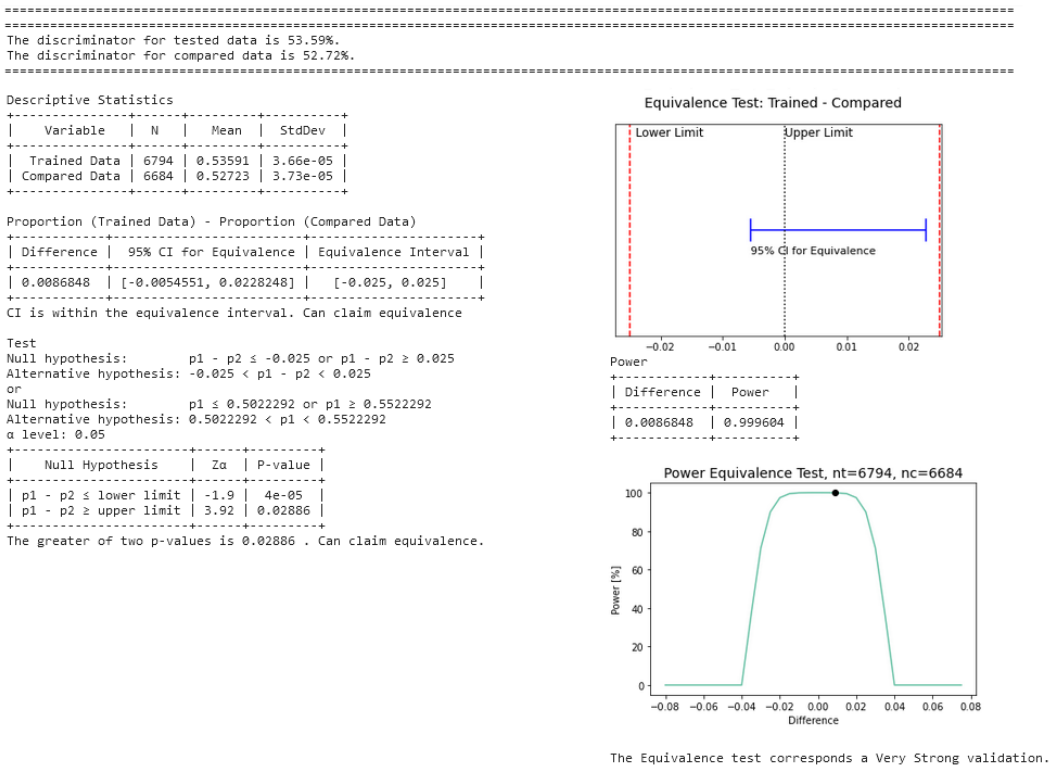

Figura H.6 - Teste Equivalência – Modelo 3 – TPT

H.3.1.3. Modelo 3 – Validação Paciente Tipo 2 – TPM

A Figura H.7 mostra o TEDDP para os dados do modelo do hospital para o paciente do tipo 2, considerando como saída o TPM.

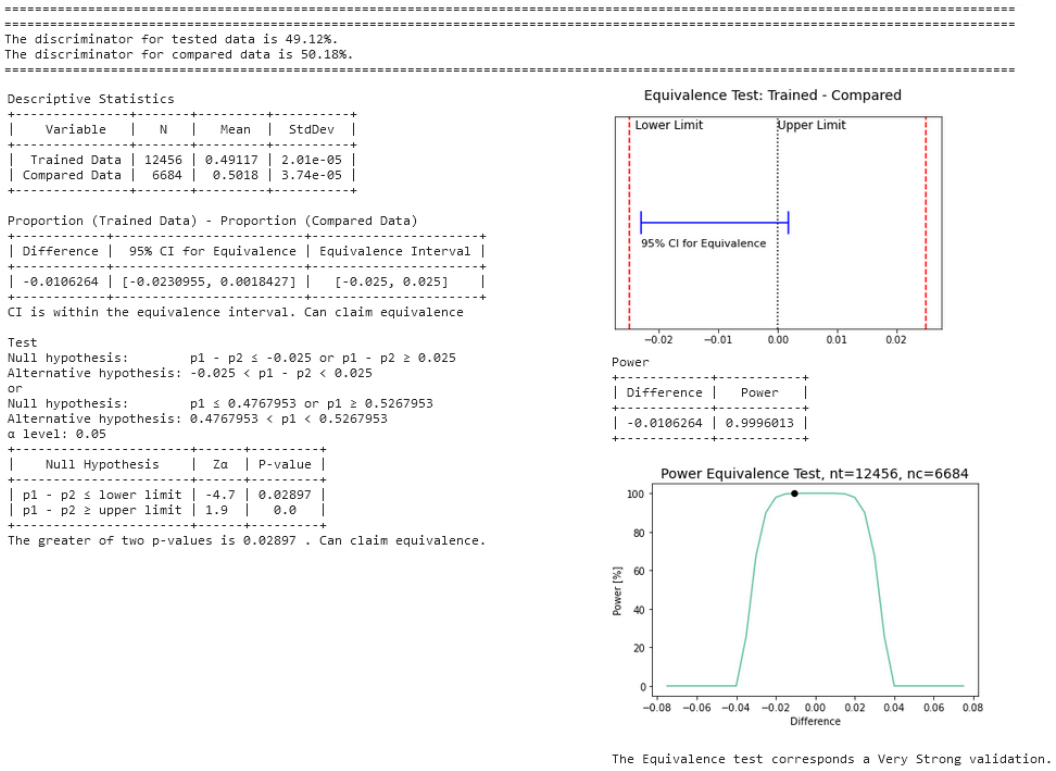

Figura H.7 - Teste Equivalência – Modelo 3 – TPM

H.3.2. Modelo 3 – Validação LOS

H.3.2.1. Modelo 3 – Validação LOS – Condicional

A Figura H.8 mostra o TEDDP para os dados do modelo do hospital para o LOS dos pacientes do tipo 1, 2, 3, 4 e 5, considerando os dados condicionais.

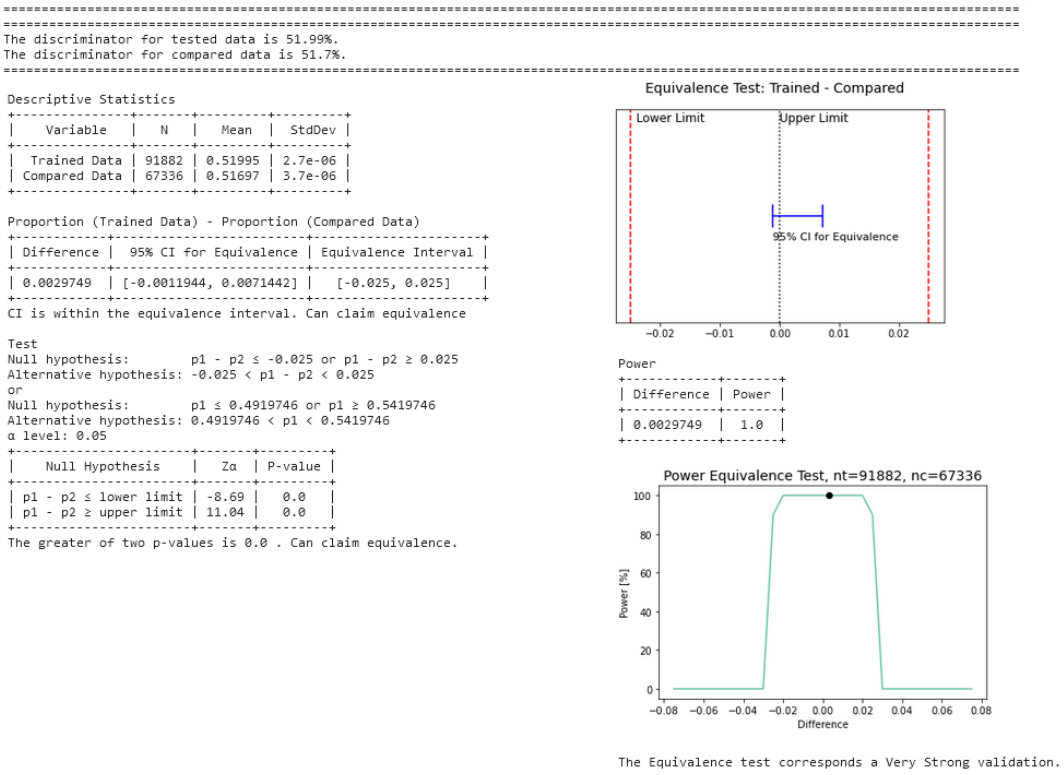

Figura H.8 - Teste Equivalência – Modelo 3 – LOS – Condicional

H.3.2.2. Modelo 3 – Validação LOS – Paciente 1

A Figura H.9 mostra o TEDDP para os dados do modelo do hospital para o LOS do paciente do tipo 1.

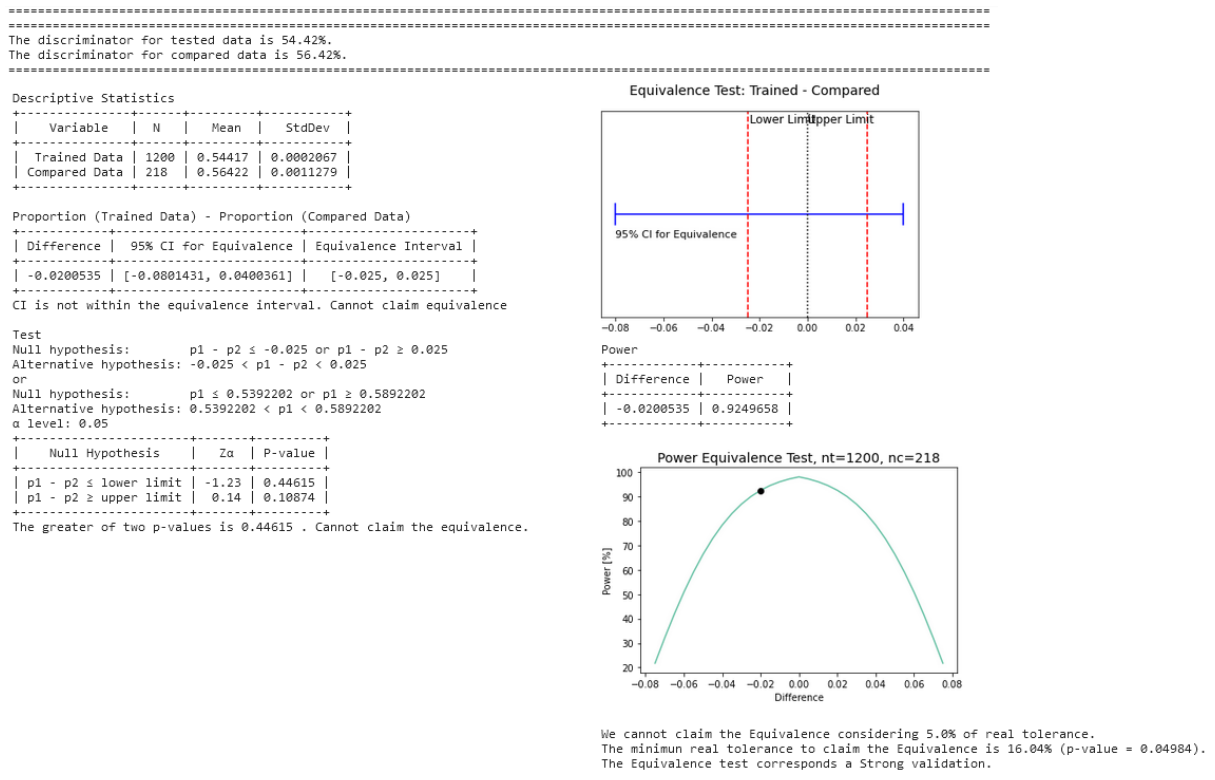

Figura H.9 - Teste Equivalência – Modelo 3 – LOS – Paciente 1

H.3.2.3. Modelo 3 – Validação LOS – Paciente 2

A Figura H.10 mostra o TEDDP para os dados do modelo do hospital para o LOS do paciente do tipo 2.

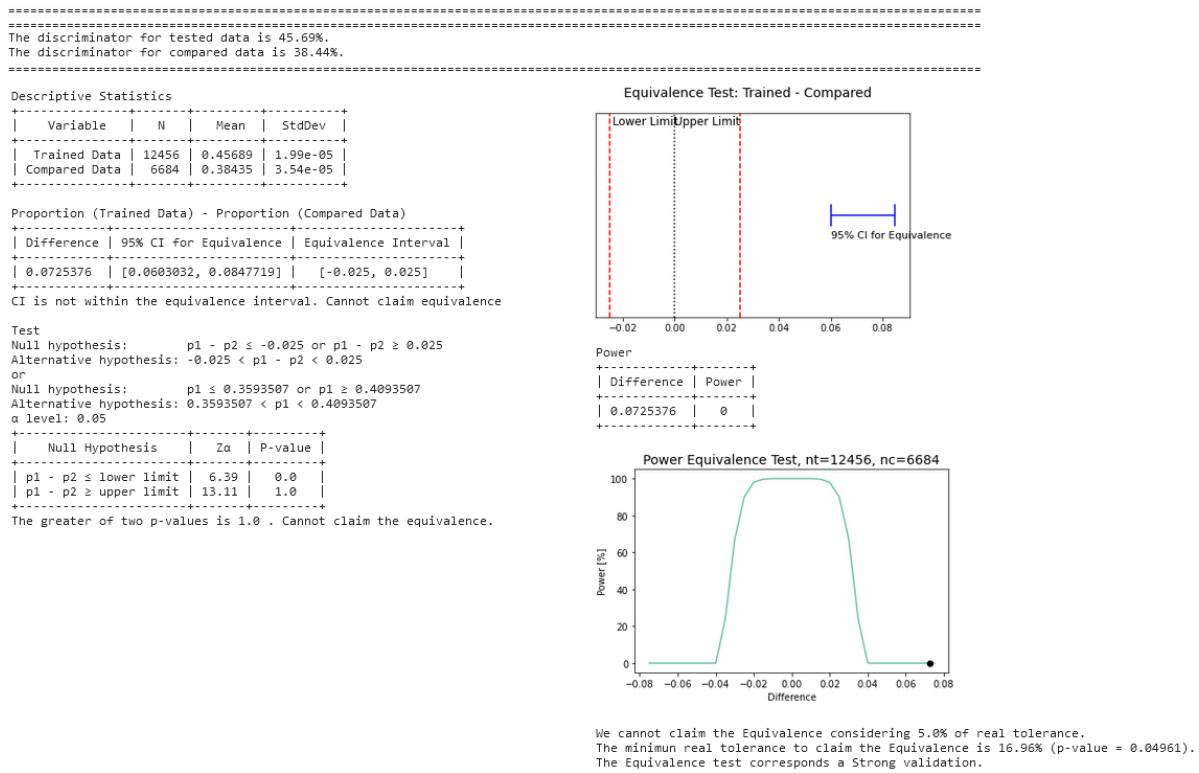

Figura H.10 - Teste Equivalência – Modelo 3 – LOS – Paciente 2

H.3.2.4. Modelo 3 – Validação LOS – Paciente 3

A Figura H.11 mostra o TEDDP para os dados do modelo do hospital para o LOS do paciente do tipo 3.

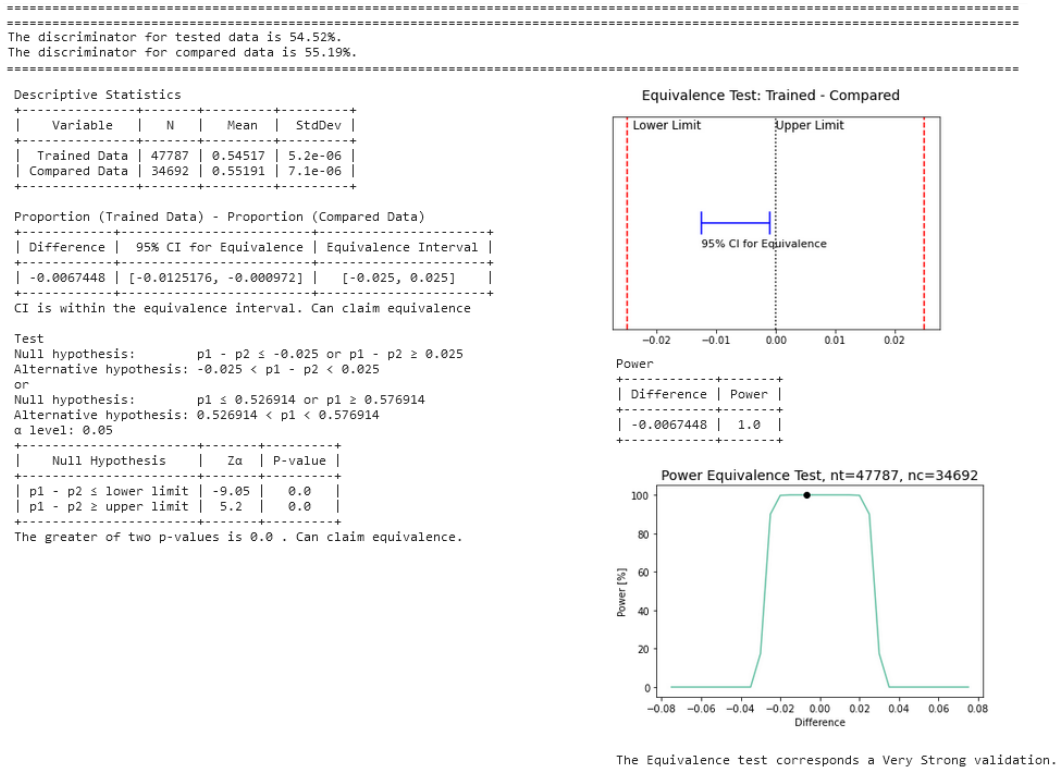

Figura H.11 - Teste Equivalência – Modelo 3 – LOS – Paciente 3

H.3.2.5. Modelo 3 – Validação LOS – Paciente 4

A Figura H.12 mostra o TEDDP para os dados do modelo do hospital para o LOS do paciente do tipo 4.

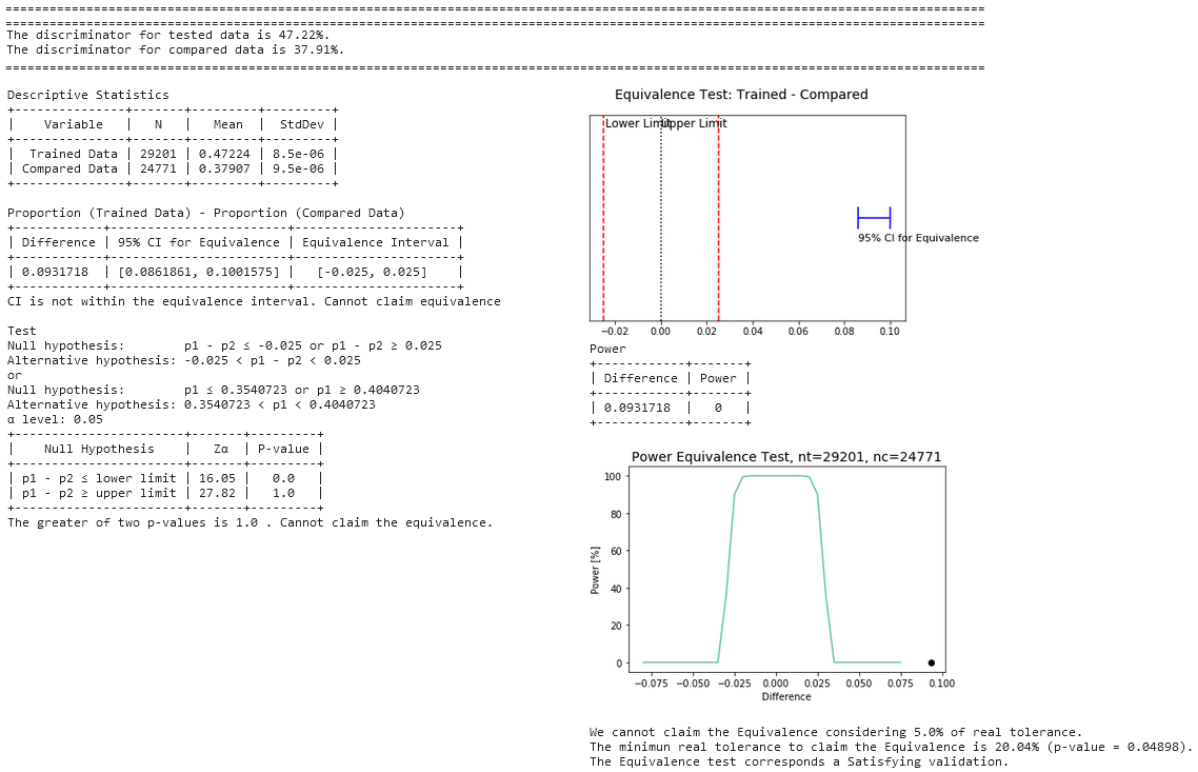

Figura H.12 - Teste Equivalência – Modelo 3 – LOS – Paciente 4

H.3.2.6. Modelo 3 – Validação LOS – Paciente 5

A Figura H.13 mostra o TEDDP para os dados do modelo do hospital para o LOS do paciente do tipo 5.

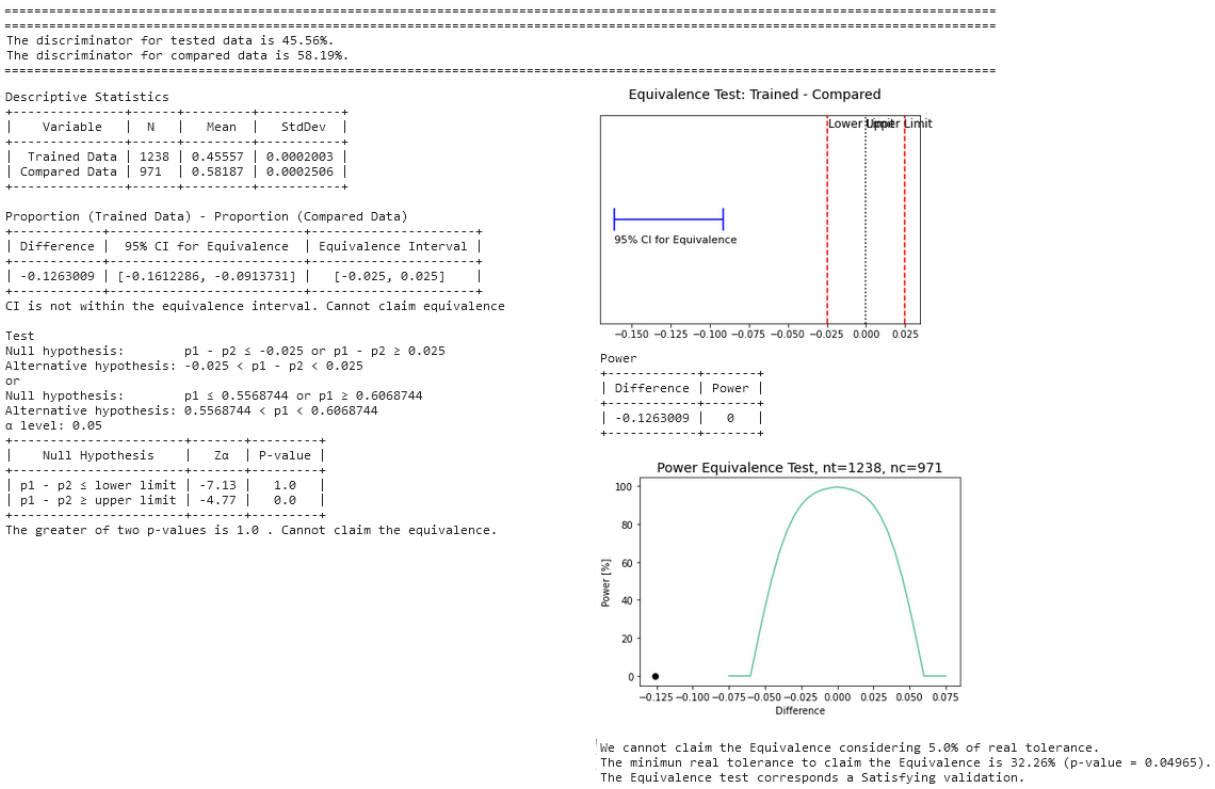

Figura H.13 - Teste Equivalência – Modelo 3 – LOS – Paciente 5

APÊNDICE I – Publicações

Publicações em Revistas:

CAMPOS, A. T.; GABRIEL, G. T.; SANTOS, C. H. dos; TORRES, A. F.; MONTEVECHI, J. A. B. Integrating Computer Simulation and the Normalized Normal Constraint method to plan a temporary hospital for COVID-19 patients. **Journal of the Operational Research Society**, 2022 – JCR: 2.861.

AMARAL, J. V. S. dos; SANTOS, C. H. dos; GABRIEL, G. T.; CARVALHO, R. M. de; MONTEVECHI, J. A. B. Metamodeling-Based Simulation Optimization in Manufacturing Problems: A Comparative Study. **International Journal of Advanced Manufacturing Technology**, v. 120, p. 5205-5224, 2022 – JCR: 3.226.

CAMPOS, A. T.; SANTOS, C. H. dos; GABRIEL, G. T.; MONTEVECHI, J. A. B. Safety assessment for temporary hospitals during the COVID-19 pandemic: a simulation approach. **Safety Science**, v. 147, p. 105642, 2022 – JCR: 4.877.

GABRIEL, G. T.; CAMPOS, A. T.; LEAL, F.; MONTEVECHI, J. A. B. Good practices and deficiencies in conceptual modelling: A systematic literature review. **Journal of Simulation**, v. 16, n. 1, p. 84-100, 2022 – JCR: 2.205.

SANTOS, C. H. dos; GABRIEL, G. T.; AMARAL, J. V. S. dos; MONTEVECHI, J. A. B.; QUEIROZ, J. A. Decision-making in a fast fashion company in the Industry 4.0 era: a Digital Twin proposal to support operational planning. **International Journal of Advanced Manufacturing Technology**, v. 116, p. 1653-1666, 2021 – JCR: 3.226.

GABRIEL, G. T.; CAMPOS, A. T.; MAGACHO, A. de L.; SEGISMONDI, L. C.; VILELA, F. F.; QUEIROZ, J. A.; MONTEVECHI, J. A. B. Lean thinking by integrating with discrete event simulation and design of experiments: an emergency department expansion. **PeerJ Computer Science**, v. 6, p. e284, 2020 – JCR: 1.392.

SAUDE, L. M. S.; GABRIEL, G. T.; BALESTRASSI, P. P. Forecasting of buffalo milk in a Brazilian diary using the ARIMA model. **Buffalo Bulletin**, v. 39, p. 201-213, 2020 – JCR: 0.172.

SANTOS, C. H. dos; GABRIEL, G. T.; CAMPOS, A. T.; QUEIROZ, J. A.; MONTEVECHI, J. A. B. Application of the Discrete Event Simulation integrated with Lean Concepts for implementing improvements in a military field hospital. **Pesquisa Naval** (SDM), 2020.

Publicações em Congressos Internacionais:

MONTEVECHI, J. A. B; CAMPOS, A. T.; GABRIEL, G. T.; SANTOS, C. H. dos . Input Data Modeling: An Approach Using Generative Adversarial Networks. In: Winter Simulation Conference, **Proceedings...** Phoenix, AZ, USA, p. 1-12, 2021.

MONTEVECHI, J. A. B.; SANTOS, C. H. dos; GABRIEL, G. T.; OLIVEIRA, M. L. M de; QUEIROZ, J. A.; LEAL, F. A Method Proposal for Conducting Simulation Projects in Industry 4.0: A Cyber-Physical System in an Aeronautical Industry. In: Winter Simulation Conference, **Proceedings...** Orlando, FL, USA, p. 2731-2742, 2020.

Submissões:

GABRIEL, G. T.; CAMPOS, A. T.; SANTOS, C. H. dos; MONTEVECHI, J. A. B.; LEAL, F. **Validation of Discrete Event Simulation Models: an approach based on Generative Adversarial Networks**, Applied Intelligence – The International Journal of Research on Intelligent Systems for Real Life Complex Problems, 2022 – JCR: 5.086 – Submetido.

Aceito para publicação:

GABRIEL, G. T.; CAMPOS, A. T.; SANTOS, C. H. dos; MONTEVECHI, J. A. B.; LEAL, F. Using Generative Adversarial Networks to Validate Discrete Event. In: Winter Simulation Conference, **Proceedings...** Singapura, 2022.

Prêmios ganhos:

Melhor Artigo SBPO – 2021:

GABRIEL, G. T.; CAMPOS, A. T.; MONTEVECHI, J. A. B.; LEAL, F. Generative Adversarial Networks to validate data in Discrete Event Simulation. In: LIII Simpósio Brasileiro de Pesquisa Operacional - LIII SBPO, 2021, João Pessoa. LIII Simpósio Brasileiro de Pesquisa Operacional - LIII SBPO, 2021.

Primeiro Lugar da Sétima Annual SHS Student Simulation Competition (2021):

GABRIEL, G. T.; AMARAL, J. V. S. do; COUTINHO, G. S.; SANTOS, R. H.; QUEIROZ, J. A. Competição mundial de Lean e Simulação a Eventos Discretos na área Hospitalar.

Primeiro Lugar da Quinta Annual SHS Student Simulation Competition (2019):

GABRIEL, G. T.; VILELA, F. F.; MAGACHO, A. de L.; SEGISMONDI, L. C.; QUEIROZ, J. A. Competição mundial de Lean e Simulação a Eventos Discretos na área Hospitalar.

Melhor Artigo SPOLM – 2019:

SANTOS, C. H.; GABRIEL, G. T.; CAMPOS, A. T.; QUEIROZ, J. A. de; MONTEVECHI, J. A. B. Application of the discrete event simulation integrated with lean concepts for implementing improvements in a military field hospital. Simpósio de Pesquisa Operacional e Logística da Marinha, Rio de Janeiro, 2019.

REFERÊNCIAS BIBLIOGRÁFICAS

ABOURIZK, S. Role of Simulation in Construction Engineering and Management. **Journal of Construction Engineering and Management**, v. 136, n. 10, p. 1140-1153, 2010.

AGNESE, J.; HERRERA, J.; TAO, H.; ZHU, X. A survey and taxonomy of adversarial neural networks for text-to-image synthesis. **Wires Data Mining and Knowledge Discovery**, v. 10, p. 1-26, 2020.

AHMED, H. M.; SCOBLE, M. J.; DUNBAR, W. S. A comparison between Offset Herringbone and El Teniente underground cave mining extraction layouts using a discrete event simulation technique, **International Journal of Mining, Reclamation and Environment**, v. 30, n. 2, p. 71-91, 2016.

AKPAN, I. J.; SHANKER, M. A comparative evaluation of the effectiveness of virtual reality, 3D visualization and 2D visual interactive simulation: an exploratory meta-analysis. **Simulation-Transactions of the Society for Modeling and Simulation International**, v. 95, n. 2, p. 145-170, 2019.

ALPAYDIN, E. **Introduction to Machine Learning**. 4 ed. Adaptive Computation and Machine Learning series, 2020.

ALRABGHI, A.; TIWARI, A. A review of simulation-based optimisation in maintenance operations. In: 15th International Conference on Computer Modelling and Simulation, **Proceedings...** Cambridge, Inglaterra, p. 353-358, 2013.

ALRABGHI, A.; TIWARI, A. State of the art in simulation-based optimisation for maintenance systems. **Computers & Industrial Engineering**, v. 82, p. 167–182, 2015.

ANGRA, S.; AHUJA, S. Machine learning and its application: a review. In International Conference on Big Data Analytics and computational Intelligence, **Proceedings...** Andhra Pradesh, Índia, p. 57-60, 2017.

APPOLINÁRIO, F. **Metodologia da ciência - filosofia e prática da pesquisa**. São Paulo: Editora Pioneira Thomson Learning, 2006.

ATIEH, A. M.; KAYLANI, H.; ALMUHTADY, A.; AL-TAMIMI, O. A value stream mapping and simulation hybrid approach: application to glass industry. **The International Journal of Advanced Manufacturing Technology**, v. 84, p. 1573-1586, 2016.

AVCI, O.; ABDELJABER, O.; KIRANYAZ, S.; HUSSEIN, M.; GABBOUJ, M.; INMAN, D. J. A review of vibration-based damage detection in civil structures: From traditional methods to Machine Learning and Deep Learning applications. **Mechanical Systems and Signal Processing**, v. 147, p. 1-45, 2021.

BALCI, O. Validation, verification, and testing techniques throughout the life cycle of a simulation study. Annals of Operations Research, v. 53, p. 121-173. 1994.

BALCI, O. Golden Rules of Verification, Validation, Testing, and Certification of Modeling and Simulation Applications. **SCS M&S Magazine**, v. 1, n. 4, p. 1-7, 2010.

BALCI, O. How to successfully conduct large-scale modeling and simulation projects. In: Winter Simulation Conference, **Proceedings...** Phoenix, AZ, USA, p. 176-182, 2011.

BALCI, O.; SARGENT, R. G. A Methodology for Cost-Risk Analysis in the Statistical Validation of Simulation Models. **Communications of the ACM**, v. 24, n. 4, p. 190-197, 1981.

BALCI, O.; SARGENT, R. G. Validation of multivariate response models using Hotelling's two-sample T^2 test. **Simulation**, v. 39, n. 6, p. 185-192, 1982.

BALCI, O.; SARGENT, R. G. Validation of Simulation Models via Simultaneous Confidence Intervals. **American Journal of Mathematical and Management Sciences**, v. 4, p. 375-406, 1984.

BANKS, J.; CHWIF, L. Warnings about simulation. **Journal of Simulation**, v. 5, n. 4, p. 279-291, 2011.

BANKS, J.; CARSON II, J. S.; NELSON, B. L.; NICOL, D. M. **Discrete-Event System Simulation**. 5. ed. New Jersey: Pearson Prentice Hall, 2010.

BATEMAN, R. E.; BOWDEN, R. O.; GOGG, T. J.; HARREL, C. R.; MOTT, J. R. A.; MONTEVECHI, J. A. B. **Simulação de sistemas: aprimorando processos de logística, serviços e manufatura**. 1. ed. Rio de Janeiro: Elsevier, 2013.

BAYARRI, M. J.; PAULO, R.; BERGER, J. O.; SACKS, J.; CAFEO, J. A.; CAVENDISH, J.; LIN, C. H.; TU, J. A framework for validation of computer models. **Technometrics**, v. 49, n. 2, p. 138-154, 2007.

BIOLCHINI, J. C. de A.; MIAN, P. G.; NATALI, A. C. C.; CONTE, T. U.; TRAVASSOS, G. H. Scientific research ontology to support systematic review in software engineering. **Advanced Engineering Informatics**, v. 21, n. 2, p. 133–151, 2007.

BOOTH, A.; PAPAIOANNOU, D.; SUTTON, A. **Systematic Approaches to a Successful Literature Review**. 2 ed. SAGE Publications, 2012.

BORJI, A. Pros and cons of GAN evaluation measures. **Computer Vision and Image Understanding**, v.179, p. 41-65, 2019.

BOTÍN, J. A.; CAMPBELL, A. N.; GUZMÁN, R. A discrete-event simulation tool for real-time management of pre-production development fleets in a block-caving project. **International Journal of Mining, Reclamation and Environment**, v. 29, v.5, p. 347-356, 2015.

BRADE, D.; LEHMANN, A. Model Verification and Validation. In: Ince A.N. (eds) Modeling and Simulation Environment for Satellite and Terrestrial Communications Networks. **The Kluwer International Series in Engineering and Computer Science**, v. 645. Springer, Boston, MA, 2002.

BROWNLEE, J. **Generative Adversarial Networks with Python: Deep Learning Generative Models for Image Synthesis and Image**. 1 ed. Machine Learning Mastery, 2020.

CAI, H.; GOGGIN; B., JIANG; Q. Two-sample test based on classification probability. **Statistical Analysis and Data Mining**, v.13, n.1, p. 5-13, 2019.

CASIER, B.; MITEVA, T.; CAPRON, N.; SISOURAT, N. Using principal component analysis for neural network high-dimensional potential energy surface. **The Journal of Chemical Physics**, v. 152, p. 1-10, 2020.

CHANDRAYAN, P. **Supervised Machine Learning Using Linear Regression**: Part1. Medium, 2019.

CHOW, S.C.; SHAO, J.; WANG, H.; LOKHNYGINA, Y. **Sample Size Calculations in Clinical Research.** 3 ed. Taylor & Francis Group: London, 2018.

CHURCHMAN, C. W.; ACKOFF, R. L.; ARNOFF, E. L. **Introduction to Operations Research**. 1 ed. John Wiley & Sons Inc: New York, 1957.

CHWIF, L.; MEDINA, A. C. **Modelagem e Simulação de Eventos Discretos: Teoria e Aplicações**. 4. ed. São Paulo: Elsevier, 2015.

CHWIF, L.; MUNIZ, P. S.; SHIMADA, L. M. A prescriptive technique for V&V of simulation models when no real-life data are available: results from a real-life project. **Journal of Simulation**, v. 2, n. 2, p. 81-89, 2008.

COOK, D. J.; GREENGOLD, N. L.; ELLRODT, A. G.; WEINGARTEN, S. R. The Relation between Systematic Reviews and Practice Guidelines. **Annals of Internal Medicine**, v. 127, n. 3, p. 210–216, 1997.

CORTES, E.; RABELO, L.; SARMIENTO, A. T.; GUTIERREZ, E. Design of Distributed Discrete-Event Simulation Systems Using Deep Belief Networks. **Information**, v. 11, n. 10, p. 1-27, 2020.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n.1, p.21-27, 1967.

CUI, S.; TSENG, H.; PAKELA, J.; HAKEN, R. K. T.; NAQA, I. Introduction to machine and deep learning for medical physicists. **The International Journal of Medical Physics Research and Practice**, v. 47, n. 5, p. 127-147, 2020.

DAVID, L.-P.; OQUAB, M. Revisiting Classifier Two-sample Tests. In: International Conference on Learning Representations, **Proceedings…** Toulon, França, 1-15, 2017.

DE LA FLUENTE, R.; ERAZO, I.; SMITH, R. L. Enabling intelligence processes in simulation utilizing the tensorflow deep learning resources. In: Winter Simulation Conference, **Proceedings...** Gothenburg, Suécia, p.1108-1119, 2018.

DENTON, E. L.; CHINTALA, S.; FERGUS, R. Deep generative image models using a laplacian pyramid of adversarial networks. **Advances in Neural Information Processing Systems (NIPS)**, p. 1486-1494, 2015.

DENYER, D.; TRANFIELD, D.; AKEN, J. E. V. Developing Design Propositions through Research Synthesis. **Organization Studies**, v. 29, n. 3, p. 393-413.

DENZIN, N. K.; LINCOLN, Y. S. **The Sage Handbook of Qualitative Research**. 5 ed. Thousand Oaks: Sage, 2017.

DOLLING, O. R.; VARAS, E. A. Artificial neural networks for streamflow prediction. **Journal of Hydraulic Research**, v. 40, n. 5, p. 547-554, 2002.

DONG, Y.; CHBAT, N. W.; GUPTA, A.; HADZIKADIC, M.; GAJIC, O. Systems modeling and simulation applications for critical care medicine. **Annals of Intensive Care**, v. 12, n. 18, p. 1-10, 2012.

DOUZAS, G.; BACAO, F. Effective data generation for imbalanced learning using conditional generative adversarial networks. **Expert Systems with Applications**, v. 91, p. 464-471, 2018.

DUTTON, D. M.; CONROY, G. V. A review of machine learning. **The Knowledge Engineering Review**, v. 12, n. 4, p. 341-367.

EDWARDS, P.; CLARKE, M.; DIGUISEPPI, C.; PRATAP, S.; ROBERTS, I.; WENTZ, R. Identification of randomized controlled trials in systematic reviews: Accuracy and reliability of screening records. **Statistics in Medicine**, v. 21, n. 11, p. 1635-1640, 2002.

FISHMAN, G. S.; KIVIAT, P. J. The statistics of discrete-event simulation. **Simulation**, v. 10, n. 4, p. 185-195, 1968.

FENG, R. Improving uncertainty analysis in well log classification by machine learning with a scaling algorithm. **Journal of Petroleum Science and Engineering**, v. 196, p. 107995, 2021.

FERREIRA, W. de P.; ARMELLINI, F.; SANTA-EULALIA, L. A. de. Simulation in industry 4.0: a state-of-the-art review. **Computers & Industrial Engineering**, v. 149, p. 1-17, 2020.

FOURES, D.; ALBERT, V.; NKETSA, A. Simulation validation using the compatibility between simulation model and experimental frame. In: Summer Computer Simulation Conference, **Proceedings...** Toronto, ON, Canada, p. 326-332, 2013.

FRANCESCHINI, F.; MAISANO, D.; MASTROGIACOMO, L. Scientific journal publishers and omitted citations in bibliometric databases: Any relationship? **Journal of Informetrics**, v. 8, p. 751-765, 2014.

GARANI, G.; ADAM, G. K. Qualitative Modelling at the Design of Concrete Manufacturing Machinery. **International Journal of Computers and Applications**, v. 30, n. 4, p. 325–330, 2008.

GAUTIER, J. Conditional generative adversarial nets for convolutional face generation. **Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition**, Winter semester 2014, 2014.

GHAHRAMANI, Z. Probabilistic machine learning and artificial intelligence. **Nature**, v. 521, p. 452-459, 2015.

GIL, A. C. **Métodos e Técnicas de Pesquisa Social**. 6 ed. São Paulo: Editora Atlas, 2008.

GOODFELLOW, I. NIPS 2016 Tutorial: Generative Adversarial Nets. **Advances in Neural Information Processing Systems (NIPS)**, p. 1-57, 2016.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning (Adaptive Computation and Machine Learning series)**. 1 ed. The MIT Press, 2016.

GOODFELLOW, I. J.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative Adversarial Nets. **Advances in Neural Information Processing Systems (NIPS)**, p. 1-9, 2014.

HAIR JÚNIOR, F. J.; BABIN, B.; MONEY, A. H.; SAMOUEL, P. **Fundamentos de métodos de pesquisa em administração**. 1 ed. São Paulo: Bookman, 2005.

HAMMERSLEY, M. On 'Systematic' Reviews of Research Literatures: a 'narrative' response to Evans & Benefield. **British Educational Research Journal**, v. 27, n. 5, p. 543-554.

HARRELL, C.; GHOSH, B. K.; BOWDEN, R. **Simulation using ProModel**. 3. ed. New York: McGraw-Hill Education, 2012.

HAWKINS, D. M. The problem of overfitting. **Journal of Chemical Information and Computer Science**, v. 44, n. 1, p. 1-12, 2004.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. 2 ed. Prentice Hall, 1998.

HERNANDEZ, J. M. C.; BASSO, K.; MOLL, M. B. Pesquisa Experimental em Marketing. **Revista Brasileira de Marketing**, v. 13, n. 2, p. 98-117, 2014.

HEUSEL, M.; RAMSAUER, H.; UNTERTHINER, T.; NESSLER, B.; HOCHREITER S. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. **Advances in Neural Information Processing Systems (NIPS)**, p. 6626-6637, 2017.

HILLIER, F. S.; LIEBERMAN, G. J. **Introduction to Operations Research with Student Access Card**. 9. ed. New York: McGraw-Hill Science, 2010.

HOLLMANN, D. A.; CRISTIA, M.; FRYDMAN, C. A family of simulation criteria to guide DEVS models validation rigorously, systematically and semi-automatically. **Simulation Modelling Practice and Theory**, v. 49, p. 1-26, 2014.

ISOLA, P.; ZHU, J-Y.; ZHOU, T.; EFROS, A. A. Image-to-Image translation with Conditional Adversarial Networks. In: IEEE Conference on Computer Vision and Pattern Recognition **Proceedings...** Honolulu, HI, USA, p. 1125-1134, 2017.

JAHANGIRIAN, M.; ELDABI, T.; NASEER, A.; STERGIOULAS, L. K.; YOUNG, T. Simulation in manufacturing and business: A review. **European Journal of Operational Research**, v. 203, p. 1–13, 2010.

JIN, W. Research on machine learning and its algorithms and development. **Journal of Physics: Conference Series**, v. 1544, p. 1-5, 2020.

JU, F.; LEE, H. K.; OSAROGIAGBON, R. U.; YU, X.; FARIS, N.; LI, J. Computer modeling of lung cancer diagnosis-to-treatment process. **Translational Lung Cancer Research**, v. 4, n. 4, p. 404-414, 2015.

JUNG, C. F. **Metodologia para pesquisa e desenvolvimento: aplicada a novas tecnologias, produtos e processos.** 1 ed. Rio de Janeiro: Editora Axcel Books, 2004.

KAPOOR, R.; SHAH, B. J. Simulation model for closed loop repairable parts inventory system with service level performance measures. **International Journal of Services and Operations Management**, v. 23, n. 1, p. 18-42, 2016.

KARIM, M. M.; DAGLI, C. H.; Qin, R. Modeling and Simulation of a Robotic Bridge Inspection System. Procedia Computer Science, V. 168 p. 177-185, 2020.

KARRAS, T.; LAINE, S.; AILA, T. A style-based generator architecture for generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition, **Proceedings...** Long Beach, CA, USA, p. 4396-4405, 2019.

KIDDER, L. H. **Métodos de pesquisa nas relações sociais**. Volume 1: delineamentos de pesquisa. 4 ed. São Paulo: Editora Pedagógica e Universitária Ltda., 2004.

KIESLING, T.; LUTHI, J; KHAYARI, R. E. (2005) Bias in parallel and distributed simulation systems. In: Winter Simulation Conference, **Proceedings...** Orlando, FL, USA, p. 384-393, 2005.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature**

Reviews in Software Engineering. Disponível em: https://edisciplinas.usp.br/pluginfile.php/4108896/mod_resource/content/2/slrPCS5012_highlighted.pdf. 2007.

KLEIJNEN, J. P. C. Statistical validation of simulation models. **European Journal of Operational Research**, n. 87, p. 21-34, 1995.

KLEIJNEN, J. P. C. **Sensitivity Analysis of Simulation Models**, Discussion Paper, Tilbug University. Tilburg, 2009.

KOTSIANTIS, S. B. Supervised Machine Learning: A Review of Classification Techniques. **Informatica**, v. 31, p. 249-268, 2007.

KURACH, K.; LUCIC, M.; ZHAI, X.; MICHALSKI, M.; GELLY, S. The GAN Landscape: Losses, Architectures, Regularization, and Normalization. In: Proceedings of the Reproducibility in Machine Learning Workshop, **Proceedings...** p. 1-16, 2018.

LAKENS, D. Equivalence Tests: A Practical Primer for t Tests, Correlations, and Meta-Analyses. **Social Psychological and Personality Science**, v. 8, n. 4, p. 355-362, 2017.

LAN, L.; LEI, Y.; ZEYANG, Z.; ZHIWEI, F.; WEILING, Z.; NIANYIN, Z.; YIDONG, C.; XIAOBO, Z. Generative Adversarial Networks and Its Applications in Biomedical Informatics. **Frontiers in Public Health**, v. 8, p. 1-14, 2020.

LANDIS, J. R.; KOCH, G. G. The Measurement of Observer Agreement for Categorical Data. **Biometrics**, v. 33, n. 1, p. 159–174, 1977.

LASTER, L. L.; JOHNSON, M. F.; KOTLER, M. L. Non-inferiority trials: the 'at least as good as' criterion with dichotomous data. **Statistcs in Medicine**, v. 25, n. 7, p. 1115-1130, 2006.

LAURINDO, Q. M. G.; PEIXOTO, T. A.; RANGEL, J. J. A. Communication mechanism of the discrete event simulation and the mechanical project softwares for manufacturing systems. **Journal of Computational Design and Engineering**, v. 6, p. 70-80, 2019.

LAW, A. **Simulation modeling and analysis**. Boston: McGraw-Hill Education, 2015.

LAZAR, J.; FENG, J. H.; HOCHHEISER, H. **Research Methods in Human Computer Interaction**. 2 ed. New York: John Wiley & Sons Inc, 2017.

LEARMONTH, M.; HARDING, N. Evidence-based management: the very idea. **Public Administration**, v. 84, n. 2, p. 245-266, 2006.

LEAL, F.; COSTA, R. F. S.; MONTEVECHI, J. A. B.; DE ALMEIDA, D. A.; MARINS, F. A. S. A practical guide for operational validation of discrete simulation models. **Pesquisa Operacional**, v. 31, n. 1, p. 57-77, 2011.

LECUN, Y., BENGIO, Y.; HINTON, G. Deep Learning. **Nature**, v. 521, p. 436-444, 2015.

LEE, J.; DAVARI, H.; SINGH, J.; PANDHARE, V. Industrial Artificial Intelligence for industry 4.0-based manufacturing systems. **Manufacturing Letters**, v. 18, p. 20-23, 2018.

LEWIS, J. A. Statistical principles for clinical trials (ICH E9): an introductory note on an international guideline. **Statistics in Medicine**, v. 18, n. 15, p. 1905-1942, 1999.

LOPEZ-PAZ, D., OQUAB, M. Revisiting classifier two-sample test. In: International Conference on Learning Representations, **Proceedings...** Toulon, France, p. 1-15, 2017.

MA, Y.; ZHONG, G.; LIU, W.; WANG, Y.; JIANG, P.; ZHANG, R. ML-CGAN: Conditional Generative Adversarial Network with a meta-learner structure for high-quality image generation with few training data. **Cognitive Computation**, v. 13, p. 418-430, 2021.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115-133, 1943.

MCHUGH, M. L. The Chi-square test of independence. **Biochemia Medica**, v. 23, n. 2, p. 143-149, 2013.

MEYNERS, M. Equivalence tests - A review. **Food Quality and Preference**, v. 26, p. 231–245, 2012.

MIGUEL, P. A. C.; FLEURY, A.; MELLO, C. H. P.; NAKANO, D. N.; TURRIONI, J. B.; LEE HO, L.; MORABITO, R.; MARTINS, R. A.; PUREZA, V. **Metodologia de pesquisa em engenharia de produção e gestão de operações**. 2 ed. Rio de Janeiro: Elsevier, 2014.

MINAR, M. R.; NAHER, J. Recent advances in deeplearning: an overview. **Advances in Neural Information Processing Systems (NIPS)**, p. 1-31, 2018.

MIRZA, M; OSINDERO, S. Conditional Generative Adversarial Nets. **Advances in Neural Information Processing Systems (NIPS)**, p. 1-7, 2014.

MIYATO, T.; KOYAMA, M. cGANs with projection Discriminator. In: International Conference on Learning Representations, **Proceedings...** Vancouver, Canada, p. 1-21, 2018.

MOHER, D.; SHAMSEER, L.; CLARKE, M.; GHERSI, D.; LIBERATI, A.; PETTICREW, M.; SHEKELLE, P.; STEWART, L. A.; PRISMA-P Group. Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement. **Systematic Reviews**, v. 4, n. 1, p. 1-9, 2015.

MONTEVECHI, J. A. B.; PINHO, A. F. de; LEAL, F.; MARINS, F. A. S. Application of design of experiments on the simulation of a process in an automotive industry. In: Winter Simulation Conference, **Proceedings...** Washington, DC, USA, p. 1601-1609, 2007.

MONTEVECHI, J. A. B.; LEAL, F.; PINHO, A. F. de; COSTA, R. F. da S.; OLIVEIRA, M. L. M. de; SILVA, A. L. F. da. Conceptual modeling in simulation projects by mean adapted IDEF: an application in a Brazilian tech company. In: Winter Simulation Conference, **Proceedings...** Baltimore, MD, USA, p. 1624-1635, 2010.

MONTEVECHI, J. A. B.; PEREIRA, T. F.; SILVA, C. E S. da; MIRANDA, R. de C.; SCHEIDEGGER, A.P.G. Identification of main methods used in simulation projects. In: Winter Simulation Conference, **Proceedings...** Huntington Beach, CA, USA, p. 3469-3480, 2015.

MONTGOMERY, D. C.; RUNGER, G. C. **Applied Statistics and Probability for Engineers**. 7 ed. Wiley, 2018.

MORADI, S.; NASIRZADEH, F.; GOLKHOO, F. A hybrid SD–DES simulation approach to model construction projects. **Construction Innovation**, v. 15, n. 1, p. 66–83, 2015.

MORREL, K. The Narrative of 'Evidence Based' Management: A Polemic. Journal of **Management Studies**, v. 45, n. 3, p. 613-635, 2008.

NASCIMENTO, I.; SOUZA, R.; LINS, S.; SILVA, A.; KLAUTAU, A. Deep reinforcement learning applied to congestion control in fronthaul networks. In: IEEE Latin-American Conference on Communications (LANTICOM), **Proceedings...** Salvador, Brasil, p. 1-6, 2019.

NEGAHBAN, A.; YILMAZ, L. Agent-based simulation applications in marketing research: an integrated review. **Journal of Simulation**, v. 8, n. 2, p. 129–142, 2014.

OCA, I. M.-M. de; SNOECK, M.; REIJERS, H. A.; RODRÍGUEZ-MORFFI, A. A systematic literature review of studies on business process modeling quality. **Information and Software Technology**, v. 58, p. 187-205, 2015.

OLDEN, J. D.; JOY, M. K.; DEATH, R. G. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. **Ecological Modelling**, v. 178, p. 389-397, 2004.

OLSEN, M.; RAUNAK, M. A method for quantified confidence of DEVS validation. In: Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium, **Proceedings…** Alexandria, VA, USA, p. 135-142, 2015.

OÑA, J. de; GARRIDO, C. Extracting the contribution of independent variables in neural network models: a new approach to handle instability. **Neural Computing and Applications**, v. 25, p. 859-869, 2014.

PAN, Z.; YU, W.; YI, X.; KHAN, A.; YUAN, F.; ZHENG, Y. Recent Progress on Generative Adversarial Networks (GANs): A Survey. **IEEE Access**, v. 7, p. 36322-36333, 2019.

PISANIELLO Jr, A.; da SILVA, W. B.; CHWIF, L.; PEREIRA, W. I. Discrete event simulation of appointments handling at a children's hospital call center: Lessons learned from V&V process. In: Winter Simulation Conference, **Proceedings...** Gothenburg, Suécia, p. 3861-3872, 2018.

POPOVICS, G.; PFEIFFER, A.; MONOSTORI, L. Generic data structure and validation methodology for simulation of manufacturing systems. **International Journal of Computer Integrated Manufacturing**, v. 29, n. 12, p. 1272-1286, 2016.

RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In: International Conference on Learning Representations, **Proceedings...** San Juan, Puerto Rico, p. 1-16, 2015.

RAJU, V. N. G.; LAKSHMI, K. P.; JAIN, V. M.; KALIDINDI, A.; PADMA, V. Study the Influence of Normalization/Transformation process on the Accuracy of Supervised Classification. In: Third International Conference on Smart Systems and Inventive Technology (ICSSIT), **Proceedings...** Tirunelveli, Índia, p. 729-735, 2020.

RASCHKA, S.; JULIAN, D.; HEARTY, J. **Python: Deeper Insights into Machine Learning**. 1 ed. Packt Publishing, 2016.

RAUNAK, M.; OLSEN, M. Quantifying validation of discrete event simulation models. In: Winter Simulation Conference, **Proceedings...** Savannah, GA, USA, p. 628-639, 2014.

ROBINSON, S. General concepts of quality for discrete-event simulation. **European Journal of Operational Research**, v. 138, n. 1, p. 103-117, 2002.

ROBINSON, S. **Simulation: The Practice of Model Development and Use**. 2. ed. Palgrave Macmillan, 2014.

ROBINSON, S.; BROOKS, R. J. Independent Verification and Validation of an Industrial Simulation Model. **Simulation-Transactions of the Society for Modeling and Simulation**, v. 86, n. 7, p. 405-416, 2010.

ROCHA, F. **Simulação e Realidade Virtual: Uma Pesquisa Experimental em Engenharia de Produção.** 2020. 144 f. Tese (Dissertação em Engenharia de Produção). Universidade Federal de Itajubá (UNIFEI), Itajubá, MG, 2020.

RODIČ, B.; KANDUČ, T. Optimisation of a complex manufacturing process using discrete event simulation and a novel heuristic algorithm. **International Journal of Mathematical Models and Methods in Applied Sciences**, v. 9, p. 320–329, 2015.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 4 ed. Pearson Series in Artifical Intelligence, 2020.

SADOUN, B. Applied system simulation: a review study. **Information Sciences**, v. 124, p. 173-192, 2002.

SALIMANS, T.; GOODFELLOW, I.; ZAREMBA, W.; CHEUNG, V.; RADFORD, A.; CHEN, X. Improved techniques for training GANs. **Advances in Neural Information Processing Systems (NIPS)**, p. 2226-2234, 2016.

SAMUEL A. L. **Some Studies in Machine Learning Using the Game of Checkers. II - Recent Progress**. In: Levy D.N.L. (eds) Computer Games I. Springer, New York, NY, 1988.

SANTOS, C. H. dos; GABRIEL, G. T.; AMARAL, J. V. S. dos; MONTEVECHI, J. A. B.; QUEIROZ, J. A. de; Decision-making in a fast fashion company in the Industry 4.0 era: a Digital Twin proposal to support operational planning. **International Journal of Advanced Manufacturing Technology**, v. 116, p. 1653-1666, 2021.

SANTOS, C. H. dos; MONTEVECHI, J. A. B.; QUEIROZ, J. A. de; MIRANDA, R. C. de; LEAL, F. Decision support in productive processes through DES and ABS in the Digital Twin era: a systematic literature review. **International Journal of Production Research**, p. 1-20, 2021.

SANTOS, C. H. dos; QUEIROZ, J. A. de; LEAL, F.; MONTEVECHI, J. A. B. Use of simulation in the industry 4.0 context: Creation of a Digital Twin to optimise decision making on non-automated process. **Journal of Simulation**, 2020.

SANTOS, R. P. dos; DEAN, D. L.; WEAVER, J. M.; HOVANSKI, Y. Identifying the relative importance of predictive variables in artificial neural networks based on data produced through a discrete event simulation of a manufacturing environment. **International Journal of Modelling and Simulation**, v. 39, n. 4, p. 234-245, 2019.

SARAVANAN, R.; SUJATHA, P. A state of art techniques on machine learning algorithms: a perspective of supervised learning approaches in data classification. In: Second International Conference on Intelligent Computing and Control Systems, **Proceedings...** Madurai, Índia, p. 945-949, 2018.

SARGENT, R. G. The Use of Graphical Models in Model Validation. In: Winter Simulation Conference, **Proceedings...** Washington, DC, USA, p. 237-241, 1986.

SARGENT, R. G. Verification and validation of simulation models. **Journal of Simulation**, v. 7, n. 1, p. 12-24, 2013.

SARGENT, R. G. An interval statistical procedure for use in validation of simulation models. **Journal of Simulation**, v. 9, n. 3, p. 232-237, 2015.

SARGENT, R. G.; BALCI, O. History of verification and validation of simulation models. In: Winter Simulation Conference, **Proceedings...** Las Vegas, NV, USA, p. 292-307, 2017.

SARGENT, R. G.; GOLDSMAN, D. M.; YAACOUB, T. Use of the Interval Statistical Procedure for Simulation Model Validation. In: Winter Simulation Conference, **Proceedings...** Huntington Beach, CA, USA, p. 60-72, 2015.

SARGENT, R. G.; GOLDSMAN, D. M.; YAACOUB, T. A tutorial on the operational validation of simulation models. In: Winter Simulation Conference, **Proceedings...** Arlington, VA, USA, p. 163-177, 2016.

SCHEIDEGGER, A. P. G.; PEREIRA, T. F.; OLIVEIRA, M. L. M.; BANERJEE, A.; MONTEVECHI, J. A. B. An introductory guide for hybrid simulation modelers on the primary simulation methods in industrial engineering identified through a systematic review of the literature. **Computers & Industrial Engineering**, v. 124, p. 474-492, 2018.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural Networks**, v. 61, p. 85-117, 2015.

SCHRIBER, T. J.; BRUNNER, D. T.; SMITH, J. S. Inside discrete-event simulation software? How it works and why it matters. In: Winter Simulation Conference, **Proceedings...** Washington, DC, USA, p. 735-749, 2017.

SCHUIRMANN, D. L. On Hypothesis Testing to Determine If the Mean of a Normal Distribution Is Contained in a Known Interval. **Biometrics**, v. 37, p. 617, 1981.

SCHUIRMANN, D. J. A Comparison of the Two One-Sided Tests Procedure and the Power Approach for Assessing the Equivalence of Average Bioavailability. **Journal of Pharmacokinetics and Biopharmaceutics**, v. 15, n. 6, p. 657-680, 1987.

SHI, D.; FAN, W.; XIAO, Y.; LIN, T.; XING, C. Intelligent scheduling of discrete automated production line via deep reinforcement learning. International Journal of Production Research, v. 58, n. 11, p. 3362-3380, 2020.

SKOOGH, A.; JOHANSSON, B.; STAHRE, J. Automated input data management: evaluation of a concept for reduced time consumption in discrete event simulation. **The Society for Modeling and Simulation International**, v. 88, n. 11, p. 1279-1293, 2012.

SOKOLOWSKI, J. A.; BANKS, C. M. **Modeling and Simulation Fundamentals: Theoretical Underpinnings and Practical Domains**. 1 ed. Hoboken, New Jersey: John Wiley & Sons, Inc., 2010.

SOMERS, M. J.; CASAL, J. C. Using artificial neural networks to model nonlinearity: The case of the job satisfaction - job performance relationship. **Organizational Research Methods**, v. 12, n. 3, p. 403-417, 2009.

SORIN, V.; Barash, Y.; KONEN, E.; KLANG, E. Creating artificial images for radiology applications using Generative Adversarial Networks (GANs) – A Systematic Review. **Academic Radiology**, v. 27, n. 8, p. 1175-1185, 2020.

SOUSA JUNIOR, W. T. de; MONTEVECHI, J. A. B.; MIRANDA, R. C. DE; CAMPOS, A. T. Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review. **Computers & Industrial Engineering**, v. 128, p. 526‑540, 2019.

SRIVASTAVA, M.; PALLAVI, S.; CHANDRA, S.; GEETHA, G. Comparison of optimizers implemented in Generative Adversarial Network (GAN). **International Journal of Pure and Applied Mathematics**, v. 119, n. 12, p. 16831-16836, 2018.

SWINGLER, K. **Applying Neural Networks: A Practical Guide**. 1 ed. Morgan Kaufmann, 1996.

THACKER, B. H.; DOEBLING, S. W.; HEMEZ, F. M.; ANDERSON, M. C.; PEPIN, J. E.; RODRIGUEZ, E. A. **Concepts of Model Verification and Validation**. Disponível em: https://inis.iaea.org/collection/NCLCollectionStore/_Public/36/030/36030870.pdf?r=1. 2004

TRANFIELD, D.; DENYER, D.; SMART, P. Towards a Methodology for Developing Evidence-Informed Management Knowledge by Means of Systematic Review. **British Journal of Management**, v. 14, n. 3, p. 207-222, 2003.

TSIOPTSIAS, N.; TAKO, A.; ROBINSON, S. Model validation and testing in simulation: A literature review. In: 5th Student Conference on Operational Research (SCOR 2016), **Proceedings...** p. 1-11, 2016.

WALKER, E.; NOWACKI, A. S. Understanding Equivalence and Noninferiority Testing. **Journal of General Internal Medicine**, v. 26, n. 2, p. 192-196, 2011.

WANG, Z. Selecting verification and validation techniques for simulation projects: A planning and tailoring strategy. In: Winter Simulation Conference, **Proceedings...** Washington, DC, USA, p. 1233-1244, 2013.

WANG, S.; GONG, X.; SONG, M.; FEI, C. Y.; QUAADGRAS, S.; PENG, J.; ZOU, P.; CHEN, J.; ZHANG, W.; JIAO, R. J. Smart dispatching and optimal elevator group control through real-time occupancy-aware deep learning of usage patterns. **Advanced Engineering Informatics**, v. 48, p. 1-16, 2021.

WATSON, P. F.; PETRIE, A. Method agreement analysis: A review of correct methodology. **Theriogenology**, v. 73, n. 9, p. 1167–1179, 2010.

WU, C-H.; ZHOU, F-Y., TSAI, C-H.; YU, C-J.; DAUZÈRE-PÉRÈS, S. A deep learning approach for the dynamic dispatching of unreliable machines in re-entrant production systems. International Journal of Production Research, v. 58, n. 9, p. 2822-2840, 2020.

Yi, X.; Walia, E.; Babyn, P. Generative adversarial network in medical imaging: A review. **Medical Image Analysis**, n. 58, p. 1-20, 2019.

YIN, C.; MCKAY, A. (2018) Introduction to modeling and simulation techniques. In: The 8th International Symposium on Computational Intelligence and Industrial Applications and The 12th China-Japan International Workshop on Information Technology and Control Applications, **Proceedings...** Tengzhou, Shandong, China, p. 1-6, 2018.

YURIY, G.; VAYENAS, N. Discrete-event simulation of mine equipment systems combined with a reliability assessmen model based on genetic algorithms. **International Journal of Mining, Reclamation and Environment,** v. 22, n. 1, p. 70–83, 2008.

ZEIGLER, B. P.; NUTARO, J. J. Towards a framework for more robust validation and verification of simulation models for systems of systems. **Journal of Defense Modeling and Simulation**, v. 13, n. 1, p. 3-16, 2016.

ZHANG, G.; PATUWO, B. E.; HU, M. Y. Forecasting with artificial neural networks: The state of the art. **International Journal of Forecasting**, v. 14, p. 35-62, 1998.