UNIVERSIDADE PRESBITERIANA MACKENZIE
PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA ELÉTRICA E COMPUTAÇÃO

Mauricio Cruz Vidotto

Previsão de preços de ações utilizando uma rede neural TimeGAN

Dr. Pedro Paulo Balbi de Oliveira - Orientador (FCI-UPM)
Dr. Eli Hadad Junior - Coorientador (CCSA-UPM)

São Paulo
2024

UNIVERSIDADE PRESBITERIANA MACKENZIE
PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA ELÉTRICA E COMPUTAÇÃO

Mauricio Cruz Vidotto

Previsão de preços de ações utilizando uma rede neural TimeGAN

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica e Computação da Universidade Presbiteriana Mackenzie como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica e Computação.

Dr. Pedro Paulo Balbi de Oliveira - Orientador (FCI-UPM)
Dr. Eli Hadad Junior - Coorientador (CCSA-UPM)

São Paulo
2024

V654p Vidotto, Mauricio

Previsão de preços de ações utilizando uma rede neural TimeGAN / Mauricio Cruz Vidotto

3.13 KB

Dissertação (Mestrado em Engenharia Elétrica e Computação) – Universidade Presbiteriana Mackenzie, São Paulo, 2024.

Orientador: Prof Dr. Pedro Paulo Balbi de Oliveira

Bibliografia: f. 49-55

1. Aprendizado de máquina 2. Redes neurais 3. Redes neurais recorrentes, 4. Redes adversárias generativas 5. Mercado de capitais 6. Previsão de séries temporais. I. Oliveira, Pedro Paulo Balbi de, orientador II. Título

CDD 006.37

Bibliotecária responsável: Maria Gabriela Brandi Teixeira – CRB 8 / 6339

MAURICIO CRUZ VIDOTTO

Previsão de preços de ações utilizando uma rede neural TimeGAN

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica e Computação da Universidade Presbiteriana Mackenzie como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica e Computação.

Orientador: Prof. Dr. Pedro Paulo Balbi de Oliveira
Co-orientador: Prof. Dr. Eli Hadad Junior

Aprovada em 6 de agosto de 2024.

BANCA EXAMINADORA

Prof. Dr. Pedro Paulo Balbi de Oliveira (Orientador)
Universidade Presbiteriana Mackenzie

Prof. Dr. Eli Hadad Junior (Co-orientador)
Universidade Presbiteriana Mackenzie

Prof. Dr. Rogério de Oliveira (Titular Interno)
Universidade Presbiteriana Mackenzie

Dr. Paulo André de Castro (Titular Externo)
Instituto Tecnológico de Aeronáutica

RESUMO

Prever o comportamento de ativos no mercado financeiro sempre foi um grande desafio devido à sua imprevisibilidade. Existem muitos fatores que podem influenciar o preço tornando o mercado muito volátil. Com a evolução das técnicas de inteligência artificial, cada vez mais modelos de aprendizado automático são desenvolvidos e utilizados para esta finalidade. Por exemplo, na área financeira, essas técnicas têm sido aplicadas em previsões de mercado de ações, otimização de carteiras, processamento de dados financeiros e estratégias de negociação. Este estudo aborda a utilização de um modelo generativo capaz de aprender com os dados reais e produzir preços como previsões. Foi utilizada a estrutura *TimeGAN* porque combina a versatilidade das Redes Adversárias Generativas (GAN) não supervisionadas com o controle sobre a dinâmica temporal condicional que é proporcionada pelos modelos autorregressivos supervisionados. A *TimeGAN* foi especificamente projetada para séries temporais e é normalmente empregada para capturar características temporais complexas dos dados e gerar séries sintéticas. O autoencoder e a rede adversária que compõem o *TimeGAN* utilizaram redes neurais recorrentes (RNN) do tipo *Long Short-Term Memory* (LSTM). A metodologia proposta é aplicada a duas série temporais de preços que possuem uma relação de causalidade, demonstrando eficácia do modelo em capturar tendências de preços e volatilidades em gerar séries sintéticas que são úteis para serem utilizadas para predição.

Palavras-chave: *Aprendizado de máquina, redes neurais, redes neurais recorrentes, redes adversárias generativas, mercado de capitais, previsão de séries temporais*

ABSTRACT

Predicting the behavior of assets in the financial market has always been a significant challenge due to their unpredictability. Numerous factors can influence prices, making the market highly volatile. With the evolution of artificial intelligence techniques, an increasing number of machine learning models are being developed and utilized for this purpose. For instance, in the financial domain, these techniques have been applied to stock market predictions, portfolio optimization, financial data processing, and trading strategies. This study explores the use of a generative model capable of learning from real data and producing price forecasts. We employed the *TimeGAN* framework because it combines the versatility of unsupervised Generative Adversarial Networks (GANs) with control over conditional temporal dynamics provided by supervised autoregressive models. *TimeGAN* is specifically designed for time series data and is commonly used to capture complex temporal features and generate synthetic series. The autoencoder and adversarial network components within *TimeGAN* utilize Long Short-Term Memory (LSTM) recurrent neural networks. The proposed methodology is applied to two price time series with a causal relationship, demonstrating the model's effectiveness in capturing price trends and volatilities for predictive purposes.

Keywords: *Machine Learning, Neural Networks, Recurrent Neural Networks, Generative Adversarial Networks, Capital Markets, Time Series Forecasting*

AGRADECIMENTOS

Agradeço ao Professor Dr. Nizam Omar pelo apoio e incentivo para realizar o curso, aos Professores Dr. Pedro Paulo Balbi e Dr. Eli Hadad Junior por todos os conselhos, pela ajuda, pela paciência e pelas orientações para realização deste trabalho. Ao amigo Richard Williams Macedo Junior pela ajuda para a realização deste trabalho.

A minha esposa e filha, que me incentivaram nos momentos difíceis e compreenderam a minha ausência enquanto eu me dedicava à realização deste trabalho.

Esta dissertação se beneficiou dos projetos de pesquisa concedidos a meu orientador, pela CAPES (projeto STIC-AmSud - CAMA, no. 88881.694458/2022- 01) e CNPq (bolsa de pesquisa PQ 303356/2022-7).

Sumário

1 INTRODUÇÃO

No domínio financeiro, a aspiração de antecipar a dinâmica dos ativos negociados em bolsa de valores é indubitavelmente tão antiga quanto a própria concepção do mercado acionário. Para atingir esse objetivo, os investidores empregam instrumentos como a análise fundamentalista, na qual, a partir de informações públicas de uma empresa, buscam compreender a sua condição financeira, avaliando se a organização é sustentável e possui recursos monetários disponíveis. Uma abordagem alternativa é por meio da análise técnica, onde o histórico das ações é examinado graficamente com o propósito de avaliar se ocorreram alterações estruturais na oferta e demanda, bem como na identificação de tendências.

Em um labirinto de números e tendências, o mercado de ações é o lugar onde mesmo as menores flutuações podem levar a grandes variações. Se pudéssemos prever alguns destes movimentos de mercados para tomarmos decisões bem informadas vislumbrando o futuro, poderíamos percorrer este labirinto com mais segurança.

Eugene Francis Fama, Nobel de Economia de 2013, foi um dos primeiros a estudar como as cotações das ações respondem a um evento. Em sua tese publicada em 1965, *The Behavior of Stock Market Prices*, (FAMA, 1965) apresenta a **Hipótese do Mercado Eficiente** onde afirma que é muito difícil prever os movimentos dos preços dos ativos no curto prazo, porque os mercados incorporam rapidamente qualquer informação relevante sobre os preços. Em plena era dos dados, este desejo de prever os movimentos futuros do mercado cada vez mais se aproxima da realidade devido a disponibilidade de informações e ao uso de métodos de aprendizado automático.

Assim, alguns críticos de Eugene Fama afirmam que o movimento dos preços das ações podem ser previstos até certo ponto com modelos apropriados e o uso das variáveis adequadas.

Nos dias atuais, com o aumento da capacidade computacional aliada aos avanços das técnicas de aprendizado, é possível lidar com grandes quantidades de dados e transformá-los em informações valiosas que podem fornecer *insights* relevantes que podem complementar os métodos tradicionais de pesquisa.

Esta pesquisa utilizará técnicas de aprendizagem de máquina com o objetivo de gerar

séries sintéticas de dados que serão utilizadas para prever preços futuros de ativos utilizando uma rede neural generativa treinada com os dados da série histórica real. Será utilizada a estrutura *TimeGAN* proposta por Yoon, Jarrett e Schaar (2019) pelo fato de combinar a flexibilidade do paradigma não supervisionado com o controle proporcionado pelo treinamento supervisionado para geração de dados temporais sintéticos realistas. Uma vez treinado, o gerador será utilizado para geração de preços sintéticos. A previsão será avaliada utilizando o RMSE (Raiz Do Erro Quadrático Médio) que é uma métrica que nos diz a raiz quadrada da diferença quadrada média entre os valores previstos e os valores reais em um conjunto de dados. O RMSE é frequentemente usado na prática porque é medido nas mesmas unidades que a variável de resposta.

Este estudo será organizado da seguinte forma: Na seção 2, tem-se a revisão da literatura, na seção 3 o referencial teórico, na seção 4 a metodologia, na seção 5 os resultados e a conclusão na seção 6. A última seção contém as referências bibliográficas utilizadas para a realização da pesquisa.

2 REVISÃO DA LITERATURA

Com o objetivo de avaliar a contribuição deste artigo para preencher lacunas de conhecimento ainda não exploradas por estudos anteriores, conduzimos uma análise bibliométrica e uma revisão sistemática da literatura. A amostra final é composta por 16 artigos publicados entre 2018 e 2023.

A análise bibliométrica envolveu uma abordagem quantitativa, incluindo a contagem de frequências e co-citações. Por outro lado, a revisão da literatura adotou uma abordagem qualitativa, considerando a correlação entre temas relevantes, ainda pouco explorados pela comunidade acadêmica.

O estoque de pesquisas analisadas é originário das base de dados *Web of Science (WoS)* e, tanto a análise bibliométrica quanto a revisão da literatura, utilizaram a linguagem de programação *R* (R Core Team, 2020), a interface de programação R *RStudio* (RStudio Team, 2020), os softwares *Biblioshiny* para análises de mapeamento científico (ARIA; CUCCURULLO, 2017) e *VOSViewer* para construção e visualização de redes bibliométricas (R Core Team, 2023).

A revisão da literatura foi realizada com a identificação de métodos para a previsão do preço de ações com uso de redes generativas adversárias (*GANs*).

A pesquisa foi conduzida em 7 etapas, conforme descritas a seguir. As etapas de 1 a 5 abrangem tanto a metodologia da análise bibliométrica quanto a revisão sistemática, enquanto as etapas 6 e 7 estão relacionadas exclusivamente à revisão sistemática.

Etapa 1: Escolha da base de dados. Os artigos da amostra foram coletados da *Web of Science* (WoS), a principal base de dados de citações global, que indexa os periódicos mais citados em suas respectivas áreas. Atualmente, a *WoS* possui mais de 9.000 periódicos indexados.

Etapa 2: Uso de parâmetros iniciais de pesquisa a partir da *WoS*. Foram selecionados os sub-DB's (*Subject Classifications Databases*) relacionados a área de estudo da pesquisa.

Para nossa análise bibliométrica, selecionamos três sub-DB's do *WoS*: "Science Citation Index Expanded"(SCI-EXPANDED), "Emerging Sources Citation Index"(ESCI) e "Social Sciences Citation Index"(SSCI). Essa escolha foi baseada na área de interesse desses sub-DB's e por sua relevância a área de estudo da pesquisa. Estes sub-DB's contém dados a partir dos anos de 2019, 1956 e 1945 respectivamente. Inicialmente, foram identificados 50 artigos com base em variações das palavras-chave "**gan** (All Fields) AND **stock** (All Fields) AND **prediction** (All Fields)". Em seguida, aplicaram-se filtros na base de dados da *WoS*, resultando em uma amostra intermediária de 18 artigos, conforme apresentado na tabela 1.

Etapa 3: Exclusão de artigos não relacionados. Após a análise inicial dos resumos, introduções e conclusões dos artigos, excluímos 2 dos 18 artigos da amostra intermediária. Essas exclusões ocorreram porque esses artigos não estavam diretamente relacionados a redes generativas adversárias (*GANs*), mas especificamente a um método de otimização utilizado para resolver problemas complexos relacionados à resistência a colisões (1) e a análise de inteligência de dados do setor financeiro com base em algoritmo genético (1). A amostra final será composta pelos 16 artigos restantes. A Figura 1 apresenta a distribuição dos artigos por categorias no *WoS*. É importante ressaltar que um mesmo artigo pode estar associado a mais de uma categoria.

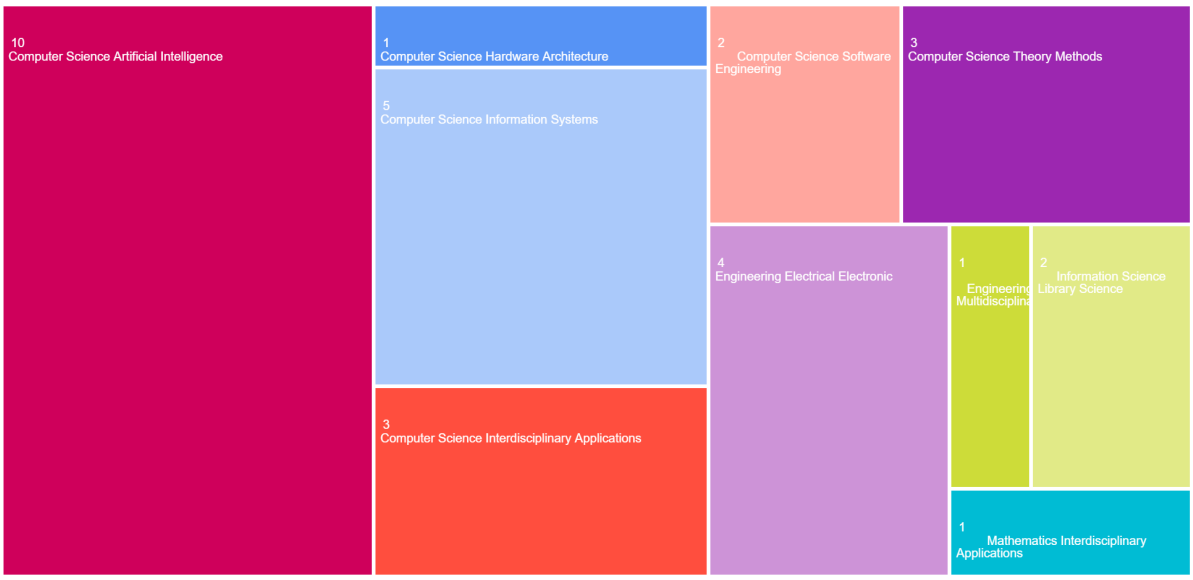
Etapa 4: Criação do banco de dados e coleta de artigos. Os artigos da amostra

Tabela 1: Evolução da amostra por meio de filtros da *WoS*

Sinal	Descrição	Artigos
(+)	Palavras-chave iguais a: “gan AND stock AND prediction”	50
(-)	Tipo de documento: diferente de “articles”	2
(-)	Idioma: diferente de “english”	0
(-)	Categorias da *WoS* diferentes de ”Computer Science Artificial Intelligence”, ”Computer Science Information Systems”, ”Engineering Electrical Electronic”, ”Computer Science Interdisciplinary Applications”, ”Computer Science Theory Methods”, ”Operations Research Management Science”, ”Computer Science Software Engineering”, ”Information Science Library Science”, ”Computer Science Hardware Architecture”, ”Engineering Multidisciplinary”, ”Mathematics Interdisciplinary Applications”	30
(=)	Amostra intermediária	18
(-)	Exclusão de artigos não relacionados	2
(=)	Amostra final	16

Fonte: Tabela criada pelo autor utilizando dados extraídos do Clarivate Analytics (2023)

Figura 1: Distribuição dos artigos artigos por categoria



Fonte: Figura de coautoria de artigos científicos gerada usando o Clarivate Analytics (2023)

final são obtidos por meio de pesquisa nas bases ”*Science Citation Index Expanded*”(SCI-EXPANDED), ”*Emerging Sources Citation Index*”(ESCI) e ”*Social Sciences Citation Index*”(SSCI) da *Web of Science*. As seguintes informações foram coletadas para a captura

dos dados gerais do artigo: título, nome do autor, instituição afiliada e país de origem dos autores/pesquisadores, nome do periódico, volume e número da edição, página inicial e final, ano da publicação, país de origem dos dados e número de anos de dados da amostra, palavras-chave, Identificador Digital de Objetos (DOI), *Journal of Economic Literature* (JEL) e número de citações de artigos na base de dados *WoS*.

Etapa 5: Análise bibliométrica. Por meio dos softwares *R*, *RStudio*, *Biblioshiny* e *VOSviewer*, são analisados os dados dos artigos para a elaboração e a análise das tabelas e mapas de relacionamento/co-citação.

Etapa 6: Leitura e codificação dos artigos. Trata-se da identificação dos objetivos, amostra, métodos e contribuições dos artigos. Além disso, eles são classificados e codificados em categorias e subcategorias estruturadas.

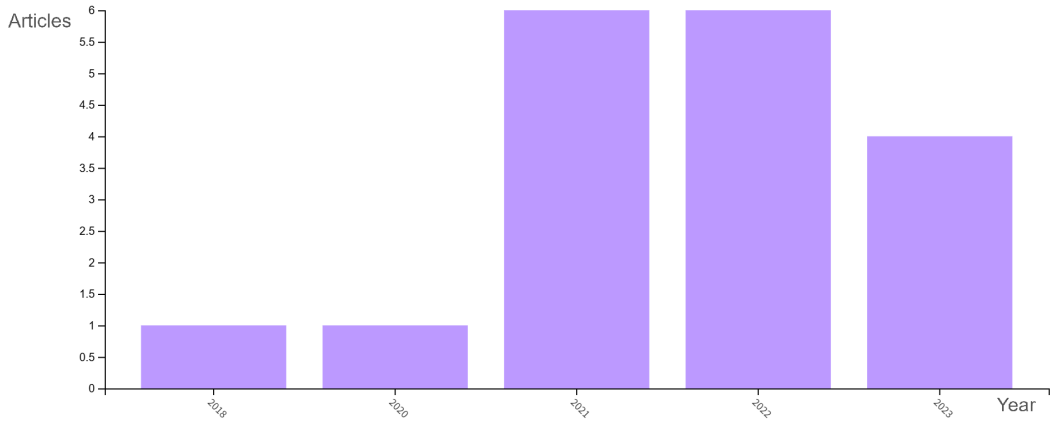
Etapa 7: Revisão da literatura. Após a categorização da matriz de (sub)categorias com base nos artigos da amostra final, procede-se à contagem de frequência das subcategorias. Esse processo visa facilitar a identificação das lacunas de conhecimento. Em seguida, essas lacunas são comparadas com as subcategorias, que representam possíveis direções para estudos futuros, com o objetivo de identificar áreas passíveis de novas pesquisas.

A figura 2 exibe a amostra final, composta pelos artigos selecionados, distribuídos entre os anos de 2018 e 2023, obtidos a partir da base *WoS*, sendo possível notar um grande aumento do interesse por parte dos pesquisadores nesse período de cinco anos, acentuadamente a partir de 2021 onde é identificada uma média de aproximadamente seis publicações de artigos por ano, que tratam especificamente sobre tema *generative adversarial network* destinada a previsões.

A figura 3 apresenta o mapa de coocorrências das palavras-chave mais utilizadas nos artigos da amostra final. Destacam-se as palavras *generative adversarial network*, *deep learning*, *time-series*, *predictionl* e *neural network*. O tamanho dos nós indica maior relevância dessas palavras nos artigos e a espessura das linhas indica a força da ligação entre elas, enquanto as cores destacam os grupos.

Considerando-se a escassez de pesquisas sobre o tema, este estudo realiza uma análise bibliométrica e a revisão da literatura, a respeito da previsão do preço de ações com uso do modelo *generative adversarial network* (GAN). Como resultado, há a identificação de

Figura 2: Distribuição anual dos artigos

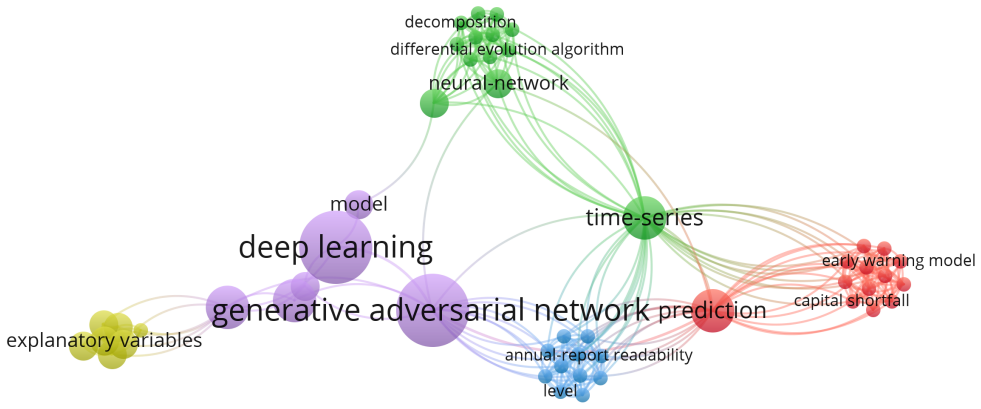


Fonte: Figura de coautoria de artigos científicos gerada usando o Clarivate Analytics (2023)

lacunas de conhecimento a serem preenchidas por uma proposta de futura agenda de pesquisas relacionadas a esse tema.

A amostra inicial foi composta por 50 artigos, que após a adoção dos critérios de exclusão mencionados acima, é reduzida para 16 estudos finais, obtidos na base de dados *WoS*. A análise bibliométrica apresenta dados quantitativos por meio de gráficos e mapas de relacionamento. Essas verificações ocorrem através dos softwares *RStudio*, *Biblioshiny* e *VOSviewer*. Por sua vez, a revisão sistemática identifica a frequência das (sub) categorias definidas para a amostra final. Sua verificação permite a percepção de quais combinações de subcategorias são viáveis para investigações futuras.

Figura 3: Mapa de co-ocorrências das palavras-chave



Fonte: Figura gerada usando o *VosViewer*

Como é possível depreender através dos resultados obtidos através da análise bibliométrica e revisão da literatura, este artigo traz novos resultados ao acrescentar o conceito de causalidade preditiva proposto por Granger analisando se as séries de tempo apresentam correlação e causalidade e, sua utilização para treinamento de um modelo generativo que possa aprender o comportamento real da séries e, assim, gerar dados sintéticos que poderão ser utilizados realizar predições.

O artigo traz como inovação o uso de grande quantidade de dados que correspondem a todas as operações realizados durante o período de negociação dos ativos entre 2019 e 2021 no mercado brasileiro, utilizando séries de dados dos ativos Índice Bovespa (IBOV) e Petrobras (PETR3) com mais de 20 milhões de observações no período analisado. Quantidades volumosas de dados como essas, permitem análises que garantem níveis mais altos de significância estatística.

3 REFERENCIAL TEÓRICO

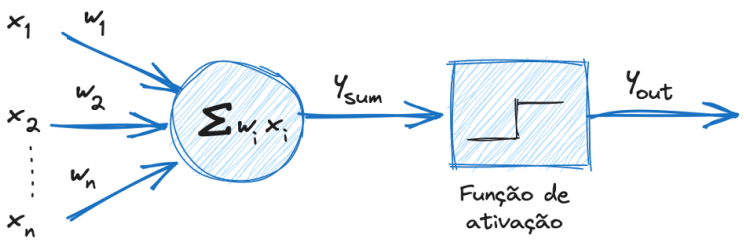
As redes neurais nasceram com o objetivo de criar um sistema computacional capaz de resolver problemas como um cérebro humano. Com o passar do tempo, o foco passou a ser resolver tarefas especificas, com abordagens menos biológicas e oferecendo suporte a uma variedade maior de tarefas, incluindo visão computacional, reconhecimento de fala, tradução de máquina, filtragem de redes sociais, jogos de tabuleiro, videogames e diagnósticos médicos.

A primeira tentativa de modelar o comportamento de um neurônio biológico ocorreu na década de 40 quando Mcculloch e Pitts (1943), da Universidade de Illinois, publicam no *Bulletin of Mathematical Biophysics* o artigo com o título "*A logical calculus of the ideas immanent in nervous activity*"onde fazem uma analogia entre as células nervosas e o processo eletrônico. A pesquisa realizada por eles procurou entender como o cérebro humano poderia produzir padrões complexos por meio de células cerebrais conectadas, ou neurônios. Propuseram um modelo matemático para um neurônio artificial baseado no funcionamento do neurônio humano e modelaram uma rede neural simples usando circuitos elétricos.

O neurônio proposto por Mcculloch e Pitts (1943) é um dispositivo que possui um con-

junto de entradas $x = [x_1, x_2, ..., x_n]$. Associado a cada entrada existe um peso sináptico $w = [w_1, w_2, ..., w_n]$ que pode ser excitatório ($w_i > 0$) ou inibitório ($w_i < 0$). Todas as entradas têm o mesmo peso fixo. O dispositivo também possui ativação é binária representando um de dois estados possíveis, 0 e 1 ou -1 e 1. Na figura 4 temos a representação gráfica de um neurônio.

Figura 4: Neurônio de McCulloch-Pitts



Fonte: Adaptado de Rothman (2020), capítulo 2, página 33

Para determinar a saída de um neurônio realiza-se a integração sináptica, ou seja, calcula-se a soma ponderada das entradas com seus respectivos ganhos sinápticos. A formulação matemática (1), na sua forma mais simples, é a seguinte:

$$y_{sum} = \sum_{i=1}^{n} w_i x_i \tag{1}$$

Onde $x_1, x_2, ..., x_n$ são os valores das entradas binárias $\in [0, 1]$, $w_1, w_2, ..., w_n$ são os pesos associados a cada entrada $\in [-1, 1]$ e θ é um valor de limiar predefinido para ativação do neurônio. Cada neurônio possui um limiar fixo (θ), definido pela função (2) abaixo:

$$y_{out} = f(y_{sum}) = \begin{cases} 1, \text{ se } y_{sum} >= \theta \\ 0, \text{ se } y_{sum} < \theta \end{cases} \tag{2}$$

A função f é chamada de função de transferência e, é normalmente aproximada por uma função degrau $f(x) = h(x)$ ou sinal $f(x) = sgn(x)$.

Em 1949, o biólogo e psicólogo Donald Hebb publica o livro *The organization of behavior: A neuropsychological theory* (HEBB, 1949) onde apresenta um principio de aprendizado em sistemas nervosos complexos fornecendo uma estrutura geral para relacionar o

comportamento a organização sináptica por meio da dinâmica das redes neurais. A obra apontou o fato de que os caminhos neurais são fortalecidos cada vez que são utilizados, um conceito essencial para descrever a forma como os seres humanos aprendem.

O próximo grande avanço ocorreu em 1958. Frank Rosenblatt na Universidade de Cornell, baseando-se no trabalho de Mcculloch e Pitts (1943) introduziu pesos na equação e desenvolveu o seu modelo matemático para sinapse humana criando uma rede de múltiplos neurônios do tipo discriminadores, a rede **Perceptron**. O *Perceptron* é caracterizado pela sua estrutura de camada única de neurônios artificiais, onde cada neurônio opera como um classificador linear binário, ajustando seus pesos sinápticos em resposta a estímulos externos durante o processo de aprendizado supervisionado. Sua pesquisa esta documentada no artigo "*The perceptron: a probabilistic model for information storage and organization in the brain.*"(ROSENBLATT, 1958). Muitos o consideram a pai da neuro-computação devido a suas complexas pesquisas e inúmeras contribuições técnicas.

Subsequentemente a este marco, vários pesquisadores direcionaram seus esforços para esta linha de pesquisa, impulsionados pelos avanços tecnológicos da década de 1950. Finalmente torna-se possível simular uma hipotética rede neural.

A ideia básica de retropropagação contínua foi derivada no contexto da teoria de controle já em 1961 por J. Kelly, Henry Arthur e E. Bryson. Esse trabalho é considerado um precursor dos métodos de aprendizado de máquina modernos. Em 1969, dez anos após a descoberta do perceptron que mostrou que uma máquina poderia ser ensinada a realizar certas tarefas usando exemplos, Minsky e Papert (1969) publicaram o livro "*Perceptrons: An Introduction to Computational Geometry*" onde analisaram as capacidades computacionais dos perceptrons para tarefas específicas abordando os pontos fortes e limitações do modelo.

Embora vários pesquisadores tenham contribuído para a ideia de propagação retroativa, Paul Werbos foi o primeiro a perceber a sua aplicação dentro das redes neurais em sua tese "*Beyond regression: New tools for prediction and analysis in the behavior science*"(WERBOS, 1974). O artigo de Hopfield (1982) também foi um marco importante no campo das redes neurais. Hopfield introduziu uma classe de redes neurais com propriedades associativas, que são capazes de armazenar informações e reconhecer padrões de forma estável.

Em 1986, o professor de psicologia da Universidade de Stanford, David E. Rumelhart e James L McClelland, professor de psicologia da Universidade de CarnegieMellon publicam o livro "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations"(RUMELHART; MCCLELLAND, 1986), onde apresentam um modelo matemático e computacional que proporciona o treinamento supervisionado dos neurônios artificias. Surge o algorítimo *Backpropagation* ou Retropropagação, que é um conjunto de regras de aprendizado para redes neurais que impactou significativamente o desenvolvimento do aprendizado profundo (*deep learning*).

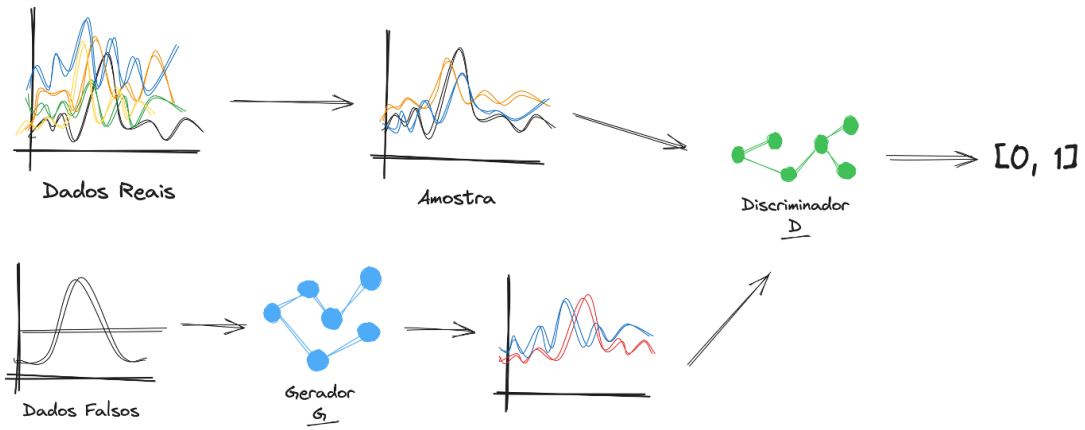
Introduzidas por Hochreiter e Schmidhuber (1997), as redes LSTM (*Long Short-Term Memory*) são uma variação das redes neurais recorrentes (RNNs) projetadas para superar as limitações de aprendizado de dependências de longo prazo. Utilizam uma arquitetura de células de memória que permite a retenção e o esquecimento seletivo de informações ao longo do tempo. Essa característica é particularmente útil em tarefas sequenciais, como processamento de linguagem natural, tradução automática e previsão de séries temporais. As LSTMs são compostas por três portas principais: a porta de entrada (*Input Gate*), que controla a quantidade de informação a ser armazenada na célula; a porta de esquecimento (*Forget Gate*), que decide quais informações antigas devem ser descartadas; e a porta de saída (*Output Gate*), que determina quais informações da célula serão usadas para a próxima etapa. Essa estrutura permite que as LSTMs mantenham informações relevantes por longos períodos, tornando-as eficazes em capturar padrões complexos em dados sequenciais.

Em 2014, Ian J. Goodfellow em seu artigo *Generative Adversarial Networks* (GOODFELLOW et al., 2014) apresenta uma nova abordagem utilizando técnicas de aprendizado profundo para gerar imagens. A abordagem proposta usa duas redes neurais para melhorar competitivamente a qualidade de uma imagem. Na sua configuração mais elementar, uma *GAN* é constituída por duas redes neurais em oposição: uma geradora e uma discriminadora. Estas redes operam em um contexto de competição mútua, onde a rede geradora tem como objetivo produzir dados indistinguíveis dos reais, enquanto a rede discriminadora esforça-se para diferenciar os dados genuínos dos sintetizados pela geradora. Juntas, estas redes executam uma análise profunda, capturam e replicam as nuances e variações intrínsecas a um conjunto de dados específico (figura 5). Por este motivo, estas redes exibem um potencial significativo, pois são capazes de aprender a replicar qualquer

distribuição de dados.

Nas redes *GANs*, o gerador (G) e o discriminador (D) são treinados em um jogo de soma zero onde o gerador tenta criar dados que sejam indistinguíveis dos dados reais, enquanto o discriminador tenta distinguir entre os dados reais e os gerados. Esse processo é fundamentado na teoria dos jogos, especificamente no conceito de equilíbrio de *Nash*.

Figura 5: Rede generativa adversária



Fonte: Adaptado de Goodfellow et al. (2014)

O equilíbrio de *Nash* é um conceito dentro da teoria dos jogos em que o resultado ideal de um jogo é aquele em que nenhum jogador têm incentivos para mudar sua estratégia levando em consideração as decisões de seus oponentes, ou seja, nenhum jogador tem a ganhar mudando apenas sua própria estratégia, desde que os outros jogadores mantenham as suas inalteradas. É um sistema estável de competição.

Nas redes *GANs*, o equilíbrio é alcançado quando o gerador produz dados tão realistas que o discriminador não consegue diferenciá-los dos dados reais, e qualquer mudança na estratégia de um dos jogadores (gerador ou discriminador) não resulta em uma melhoria significativa no desempenho. Esse estado de equilíbrio é crucial para a eficácia das *GANs*, pois indica que o gerador está criando dados de alta qualidade e o discriminador está suficientemente treinado para avaliar a autenticidade dos dados.

Para Goodfellow et al. (2014), ao utilizar a técnica de **Minimax** e, com tempo e recursos computacionais suficientes, espera-se que o Discriminador (D) e Gerador (G) entrem em equilíbrio.

O procedimento de treinamento para G é maximizar a probabilidade de D cometer um erro. Desta forma, o modelo se beneficia da incapacidade de D em reconhecer se os dados são verdadeiros ou não. A perda de D é minimizada quando ele classifica corretamente as entradas de G como falsas e os dados reais como verdadeiros. Esta competição entre as redes faz com que o desempenho seja melhorado até que os dados reais não sejam distinguíveis dos gerados.

Desde o seu surgimento, houveram significativos avanços na utilização das Redes Generativas Adversárias (*GANs*). Entre suas as principais aplicações destacam-se a geração de exemplos como versões de alta resolução de imagens de entrada, geração de novas imagens sintéticas ou até mesmo a capacidade de traduzir fotografias de dia para noite ou para outras estações do ano e, muito outros casos de uso. Reconhecidamente, as redes *GAN* são muito boas para o processamento de imagens sendo possível avaliar as imagens sintéticas geradas visualmente. Contudo, sua aplicação alguns tipos de dados ainda é um desafio não solucionado. Por exemplo, as *GANs* não são naturalmente projetadas para lidar com sequências temporais ou dados dependentes do tempo.

Com o objetivo de solucionar este problema, atualmente existem diversas abordagens propostas baseadas em *GAN* para geração de diferentes tipos de dados. As redes *WGAN*, ou *Wasserstein Generative Adversarial Networks*, de Arjovsky, Chintala e Bottou (2017) são uma variante das *GANs* tradicionais que utilizam uma função de custo diferente, baseada na distância de *Wasserstein*, para medir a diferença entre a distribuição de dados gerada e a real. As redes *RGAN*, ou *Recurrent Generative Adversarial Network*, foram desenvolvidas para gerar séries temporais multidimensionais de valores reais, com um foco especial na aplicação em dados médicos. Esta rede utiliza redes neurais recorrentes (*RNNs*) tanto no gerador quanto no discriminador permitindo que o modelo capture dinâmicas temporais complexas e produza dados sintéticos que se assemelham a séries temporais reais para área médica (ESTEBAN; HYLAND; RäTSCH, 2017). Algo similar foi utilizado por Fekri, Ghosh e Grolinger (2019) para o setor de energia onde o objetivo era que o modelo aprendesse a gerar dados de consumo realísticos capturando as dependências de tempo presente nos dados.

No contexto de previsão de preços, Zhang et al. (2019) propuseram uma arquitetura de rede generativa adversária utilizando *Multi-Layer Perception* (MLP) como discriminador

e *Long Short-Term Memory* (LSTM) como gerador. Essa abordagem obteve resultados promissores em comparação com métodos clássicos de previsão como *Support Vector Regression* (SVR), *Artificial Neural Network* (ANN) e LSTM. Os valores baixos para os indicadores estatísticos MAE, RMSE e MAPE indicaram que a previsão do preço de fechamento se aproximou dos dados reais.

As redes generativas adversárias foram propostas para gerar dados a partir de uma distribuição alvo. Desta forma, se elas podem gerar dados de mercado realistas, estes dados gerados podem ser utilizados por modelos de aprendizagem de máquina como intuito de prever um comportamento futuro (GOODFELLOW et al., 2014).

Como parte deste estudo foi utilizada a promissora pesquisa de Yoon, Jarrett e Schaar (2019) que propõe uma nova estrutura, o *Time-Series Generative Adversarial Network* (*TimeGAN*), que visa contabilizar correlações temporais combinando treinamento supervisionado e não-supervisionado.

A chave para produzir dados de séries temporais de alta fidelidade é preservar a dinâmica temporal. Isso significa que as sequências geradas devem respeitar a relação entre variáveis ao longo do tempo como nos dados originais. A estrutura *TimeGAN* combina a flexibilidade do paradigma não-supervisionado da estrutura *GAN* com o controle sobre a dinâmica temporal condicional proporcionado pelo treinamento supervisionado em modelos autorregressivos para geração de dados de séries temporais sintéticas realísticas.

O *TimeGAN* incorpora explicitamente a natureza autorregressiva das séries temporais, combinando a perda adversária não supervisionada em sequências reais e sintéticas sincronizadas por meio de uma perda supervisionada passo-a-passo em relação aos dados originais. O objetivo é recompensar o modelo por aprender a distribuição sobre as transições de um ponto no tempo, para o próximo valor nos dados históricos.

De acordo com Plesner (2021), o *TimeGAN* é um dos primeiros modelos utilizando redes neurais adversárias que produzem dados sintéticos de séries temporais de alta confiança porque considera a dependência do tempo nos dados de treinamento.

A aplicação do *TimeGAN* para geração de dados sintéticos foi explorada na pesquisa de Zhang et al. (2022) para aumento de dados (*data augmentation*) através da geração de dados sintéticos para melhorar a precisão de seu modelo de previsão de aquecimento em

subestações de energia elétrica. Também foi utilizado por Li et al. (2022) em um modelo hibrido para expandir dados históricos de energia fotovoltaica para aprimorar a precisão da previsão e, assim, melhorar a qualidade de geração de energia para dias nublados e chuvosos. A estratégia de utilizar dados sintéticos para previsão foi explorada em estudos para prever o desgaste do diâmetro das rodas de trens devido a danos por fadiga com o objetivo de assegurar que os trens operem sem falhas (SHANGGUAN et al., 2023). Uma outra abordagem para previsão de falhas em discos rígidos utilizando dados sintéticos obteve, experimentalmente, uma taxa média de detecção de falha de 95% e uma taxa de alarmes falsos de 0,2% na previsão de falhas de disco (HAI et al., 2022).

Para avaliação dos dados sintéticos, um dos métodos utilizados por Yoon, Jarrett e Schaar (2019) foi a análise visual através da Análise de Componentes Principais, PCA (da sigla em inglês *Principal Component Analysis*) e t-SNE (da sigla em inglês *T-distributed Stochastic Neighbor Embedding*). A PCA é um método estatístico multivariado que foi introduzido por Pearson com o objetivo de identificar padrões ocultos no conjunto de dados e reduzir dimensão removendo ruídos (PEARSON, 1901). No entanto, por conta da alta sensibilidade do PCA à presença de *outliers* aplica-se também a técnica de t-SNE, que lida melhor com esse problema e, é menos sensível. O t-SNE, assim como a PCA, é uma técnica de redução de dimensionalidade que pode ser usada para visualizar a similaridade entre os dados reais e sintéticos em um espaço de duas ou três dimensões.

Quando abordamos um problema de previsão, nosso objetivo é formular uma equação que permita fazer previsões sobre a variável dependente, com base nos valores observados das variáveis independentes. Em uma análise causal, consideramos as variáveis independentes como causadoras da variável dependente. Por essa razão, frequentemente encontramos a expressão "correlação não implica causalidade" em pesquisas acadêmicas.

Sabemos que a correlação é um parâmetro estatístico que quantifica a relação entre variáveis e descreve a associação entre diferentes tipos de variáveis. Quando uma variável sofre alterações, a outra também apresenta mudanças, indicando a presença de covariação. No entanto, é importante ressaltar que essa covariação não implica necessariamente um vínculo causal direto ou indireto. Por outro lado, a causalidade implica que alterações em uma variável resultam em mudanças na outra. Existe uma relação de causa e efeito entre essas variáveis, e ambas estão correlacionadas entre si. Além disso, há um nexo

causal entre elas. Logo, podemos afirmar que correlação não implica causalidade, mas causalidade sempre implica correlação.

Em seu artigo de 1969, o Professor Clive W.J. Granger, Ganhador do Prêmio Nobel de Economia de 2003, propôs uma definição estatística de causalidade entre processos estocásticos, baseados na chamada cointegração, para diferenciar e combinar análise das flutuações de curto prazo e tendências de longo prazo (GRANGER, 1969). A causalidade de Granger é um teste de hipótese estatística para determinar se uma série temporal e suas defasagens oferecem informações úteis para explicar os valores da outra série.

A causalidade de Granger tornou-se uma ferramenta essencial em econometria e previsões econômicas, permitindo aos analistas identificar potenciais relações *lead-lag* entre indicadores econômicos. Estas relações refere-se à relação temporal entre duas variáveis, onde uma precede a outra e, são fundamentais para o desenvolvimento de modelos econômicos e ferramentas de previsão utilizadas, por exemplo, nos mercados financeiros.

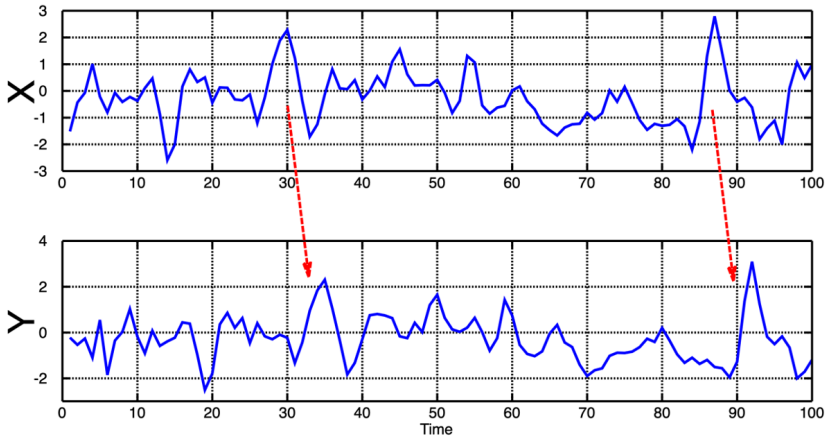
Compreender a dinâmica preditiva entre as variáveis pode ajudar os decisores a conceber intervenções oportunas e permitir que as tendências do mercado sejam antecipadas. Além disso, nos mercados financeiros, a identificação de ativos cujos movimentos de preços preveem outros pode criar estratégias lucrativas.

O conceito de causalidade para séries temporais baseia-se no conceito de previsibilidade, X_t causa Y_t se o presente de Y_t pode ser melhor previsto usando valores passados de X_t do que usando somente o valor de Y_t (figura 6). Entretanto, a relação de causalidade entre X e Y não implica na relação de causa e efeito entre elas pois ambas podem estar associadas a uma terceira variável.

No entanto, é crucial lembrar que a causalidade de Granger tem a ver com previsão, e não com causalidade direta, e deve ser interpretada com cautela nos processos de tomada de decisão.

A utilização de redes neurais aliada ao conceito de causalidade foi explorada por Apte, Vaishampayan e Palshikar (2021) ao abordar a detecção de anomalias em séries temporais que são causadas por eventos específicos, ao invés de anomalias aleatórias. Este é um problema relevante em diversas áreas, como finanças, saúde e segurança. No estudo

Figura 6: X_t causa Y_t



Fonte: Commons (2020)

utilizaram algoritmos não-supervisionados e o conceito estatístico de causalidade preditiva, a Causalidade de Granger (*G-causality*). Isso é particularmente útil para detectar fraudes, falhas de sistema ou outros eventos críticos que podem ter um impacto significativo. Adotar uma abordagem similar no contexto de previsão de preços para dados que possuem uma correlação significativa foi discutida em um artigo anterior Vidotto, Balbi e Junior (2023).

Entretanto não foram identificados casos de uso em cenários de previsão. Este estudo irá explorar o conceito de causalidade preditiva proposta por Granger utilizando séries que cointegram para produzir séries sintéticas que sejam úteis para realizar predições.

4 METODOLOGIA

Para este estudo foram utilizados os dados dos ativos Índice Bovespa (IBOV) e Petrobras (PETR3) negociados na Bolsa de Valores brasileira (B3).

A B3 (Brasil, Bolsa, Balcão) é a bolsa de valores oficial do Brasil, localizada em São Paulo. É responsável por intermediar negociações de diversos ativos financeiros, como ações, títulos públicos, fundos imobiliários, derivativos, entre outros.

Criado em 1968, o Índice Bovespa (IBOV) é o principal índice da B3 e reflete o desempenho das ações das maiores empresas de capital aberto do Brasil. Empresas como

Vale, Petrobras, Itaú, Bradesco, Banco do Brasil e Eletrobras fazem parte desse índice. A composição do Índice Bovespa é atualizado a cada quatro meses e serve como referência para investidores em todo o mundo.

Fundada em 1953, a Petrobras é uma das maiores empresas de energia do mundo e a maior do Brasil atuando na exploração, produção, refino e distribuição de petróleo e gás natural. A Petrobras desempenha um papel crucial na economia brasileira e é listada na B3 como PETR3 (ações ordinárias) e PETR4 (ações preferenciais). Suas ações também são negociadas nas bolsas de valores de Nova Iorque (NYSE), Madri (BME) e Buenos Aires (BYMA).

As informações utilizadas na pesquisa foram obtidas a partir do sinal de difusão de dados de mercado da B3 denominado UMDF para o período compreendido entre 08/04/2019 a 30/12/2021. A plataforma UMDF (*Unified Market Data Feed*) é composta por vários canais que fornecem informações em tempo real de eventos que ocorrem para todos os ativos negociados na B3, como o livro de ofertas e suas atualizações, negócios realizados, estados de instrumentos, dados estatísticos e serviços para sincronização e recuperação de mensagens. A plataforma é baseada no protocolo FIX 5.0 (*Financial Information eXchange*) com compressão FAST (*FIX Adapted for STreaming*) para otimização do consumo de banda, e os dados são publicados via *multicast* UDP.

O acesso a plataforma UMDF é restrito a corretoras, clientes, fornecedores e instituições que possuem acesso a infraestrutura tecnológica da B3 via Rede de Comunicação B3 (RCB). Para a pesquisa, os dados foram obtidos através da parceria entre a Universidade Presbiteriana Mackenzie (UPM) e a B3.

Os dados da pesquisa foram capturados e disponibilizados por uma empresa coligada da B3 especializada em *electronic & algorithmic trading* e desenvolvimento de softwares e algoritmos para negociação de alta frequência para os mercados de capitais e derivativos financeiros.

O Modelo Vetorial Autorregressivo (VAR) será utilizado conforme estabelecido no artigo seminal de (SIMS, 1980). Na Autorregressão Vetorial (VAR), o $valor - p$ é usado para testar se uma variável defasada tem um efeito significativo em outra variável no sistema. Se o valor p for menor ou igual a um nível de significância escolhido, geralmente 0.05, rejeitamos a hipótese nula de que não há efeito e, concluímos que há evidência de

uma relação causal entre as duas variáveis.

O $valor - p$ foi introduzido por Fisher (1922) para dar alguma formalidade à análise dos dados coletados em suas investigações científicas. Neyman e Pearson (1933) posteriormente modificaram para um procedimento pelo qual um desvio observado com p menor que 5%, ou outro valor pequeno, levaria à rejeição da hipótese, com aceitação em contrário. Assim, tradicionalmente, o valor de corte para rejeitar a hipótese nula é de 0.05, o que significa que, quando não há nenhuma diferença, um valor tão extremo para a estatística de teste é esperado em menos de 5% das vezes.

Se as séries temporais não forem estacionárias, a estrutura VAR precisa ser modificada para permitir uma estimativa consistente das relações entre as séries. O Modelo de Correção de Erro Vetorial (VEC) é um caso especial do VAR para variáveis que são estacionárias em suas diferenças e também pode levar em consideração qualquer relação de cointegração entre as variáveis. O objetivo é constatar a existência de uma relação de causalidade entre as séries.

A estrutura *TimeGAN* será treinada com dados reais de duas séries temporais deslocadas em "n"defasagens obtidas no teste de causalidade e, em seguida, o modelo será utilizado para gerar séries temporais sintéticas para previsão.

Para codificação foi utilizada a linguagem de programação *Python* (Python Software Foundation, 2023) por ser *Open-Source*, muito utilizada em aplicações de *data science* e aprendizagem de máquina, ser de fácil aprendizado e o código é escrito de forma fluente e natural. Possui uma grande variedade de bibliotecas disponíveis para análise de dados, estatística, gráficos entre outras. Segundo o ranking de popularidade da IEEE (2021), a linguagem de programação *Python* é a primeira no ranking de popularidade.

Alguns dos gráficos apresentados neste estudo foram gerados utilizando a biblioteca *Matplotlib* (HUNTER, 2007), uma ferramenta poderosa e amplamente utilizada para visualização de dados em *Python*.

O ambiente de teste foi montado no *Google Colab* (GOOGLE, 2018) utilizando uma máquina virtual com 12 processadores Intel(R) Xeon(R) CPU @ 2.20GHz, 83.5 GB de RAM, 40.0 GB de GPU (NVIDIA A100) e 166.8 GB de disco. Para a execução dos experimentos, optou-se por não utilizar unidades de processamento gráfico (GPUs), de modo

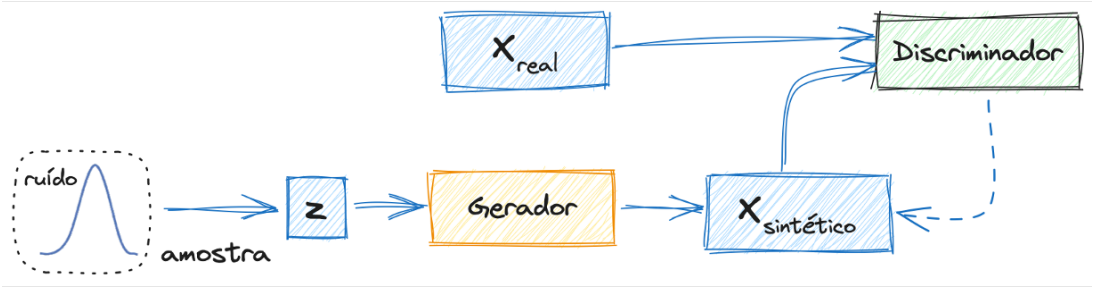
a avaliar o desempenho do modelo de aprendizado em um ambiente de processamento exclusivamente baseado em CPU.

O estudo será realizado nas etapas a seguir: (1) entendendo o *TimeGAN*, (2) preparação dos dados, (3) análise de causalidade (4) treinamento do modelo, (5) geração de dados sintéticos, e (6) previsão de movimento de preços.

4.1 ENTENDENDO O TIMEGAN

A Rede Generativa Adversária proposta por Goodfellow et al. (2014) consiste basicamente de um Gerador (G) e um Discriminador (D) (figura 7).

Figura 7: Estrutura GAN



Fonte: Elaborado pelo próprio autor

G é a rede responsável por gerar as amostras ao receber um ruído aleatório z e gerar amostras usando o ruído chamado $G(z) = X_{sintetico}$. D é a rede responsável por classificar se a amostra é verdadeira ou falsa. A amostra será verdadeira quando a saída for igual a 1 o que indica que amostra é real. A amostra será falsa quando a saída for igual a 0 o que indica que amostra é sintética. Durante o treinamento as amostras sintéticas $X_{sintetica}$ e as amostras reais X_{real} serão enviadas para D que deverá separar as amostras geradas por G das amostras reais. Portanto, G e D competirão entre si e o processo de treinamento pode ser expresso pela equação 3.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3}$$

Na equação 3, D e G representam o gerador e o discriminador respectivamente, $V(D,G)$ representa a função de valor do gerador e do discriminador, $\mathbb{E}$ representa esperança matemática, z e x representam os dados ruidosos e os dados reais respectivamente,

$p_z(z)$ e $p_{data}(x)$ representam a distribuição de ruído e distribuição de dados reais respectivamente. Em um cenário ótimo G produzirá uma amostra $G(z)$ que se parece muito com uma amostra real. Quando D torna-se difícil determinar se a amostra gerada por G é real ou não, sua saída será 0.5 e o modelo esta pronto para produzir amostras sintéticas como dados reais.

O *TimeGAN* é uma rede neural generativa especializada para trabalhar com séries de tempo com o objetivo de criar séries sintéticas de dados suavizados. É composto por duas partes, uma rede *GAN* e um *Autoencoder*.

A rede *GAN* é formada por um Gerador responsável por produzir os dados sintéticos e um Discriminador que irá tentar distinguir entre dados reais e falsos produzidos pelo Gerador. Gerador e Discriminador irão aprender competindo entre si resultando na perda adversária.

O *Autoencoder* é formando por uma rede de incorporação (*embedding network*) e uma rede de reconstrução (*recovery network*) que serão treinadas em conjunto com os componentes adversários. Com isso, o modelo aprende simultaneamente a codificar recursos, gerar representações e iterar ao longo do tempo.

Consequentemente, em relação a um modelo *GAN* tradicional, o *TimeGAN* possui duas perdas adicionais. A primeira é a **perda de reconstrução** usada para avaliar quão bem o gerador consegue reproduzir as sequências originais. Ela mede a diferença entre as sequências originais e as sequências geradas pelo modelo. O objetivo é minimizar essa diferença, para que as sequências geradas se assemelhem o máximo possível às originais. A segunda é a **perda supervisionada** responsável por forçar o gerador a aprender a dinâmica temporal dos dados avaliando quão bem o gerador se aproxima do próximo passo de tempo no espaço latente capturando essa relação e contribuindo para o treinamento do modelo. Além disso, a perda não supervisionada reflete a competição entre as redes geradora e discriminadora. Ao adicionar perdas e componentes adicionamos sobrecarga computacional o que resulta em um maior tempo de treinamento.

O treinamento é realizado em três etapas. A primeira etapa é o pré-treinamento de séries temporais e, é neste momento que o *Autoencoder* é treinado para aprender representações latentes das séries temporais reais. O objetivo é capturar as características importantes das sequências originais. O *Autoencoder* consiste em um codificador que

mapeia as sequências para o espaço latente e um decodificador que reconstrói as sequências a partir desse espaço latente.

A rede de incorporação (*embedding network*) irá mapear os dados de entrada (séries temporais) para um espaço latente (vetor de características). A dinâmica latente de dados reais e sintéticos será sincronizada por uma perda supervisionada introduzida pelo treinamento passo-a-passo utilizando os dados originais. Assim, incentivamos o modelo a capturar as distribuições condicionais presentes nos dados. A incorporação irá capturar as informações relevantes e reduzir a dimensionalidade dos dados. Com isso o modelo se beneficia extraindo mais informações dos dados de treinamento do que simplesmente se preocupando se cada dado é real ou sintético e, desta forma, pode aprender com a dinâmica de transição das sequências reais.

Após a incorporação, a rede de reconstrução (*recovery network*) reconstrói os dados originais a partir do espaço latente. Isso permite gerar novas séries temporais que se assemelham às originais. As redes de incorporação e reconstrução que formam o autoencoder podem ser parametrizadas para qualquer tipo de arquitetura tendo como única condição serem auto-regressivas e obedecerem a ordenação causal, ou seja, a saída de cada etapa deve depender exclusivamente das informações anteriores.

A segunda etapa consiste no treinamento adversário onde o gerador tenta produzir sequências sintéticas que se assemelhem às reais, enquanto o discriminador tenta distinguir entre as duas. A perda supervisionada produzida pelo supervisor é minimizada pelo treinamento conjunto das redes de incorporação e geração de forma que o espaço latente não sirva apenas para promover a eficiência dos parâmetros mas também esteja condicionado a facilitar o aprendizado das relações temporais do gerador guiando o aprendizado adversário. O supervisor é um modelo que aprende a prever o próximo passo de tempo no espaço latente e é treinado usando as representações latentes geradas pelo autoencoder. Essa etapa tem como objetivo melhorar a coerência temporal das sequências geradas.

Na terceira etapa teremos o treinamento de séries temporais sintéticas onde o gerador e supervisor são treinados juntos para gerar sequências sintéticas que se assemelham às reais no espaço latente. Essas sequências são então decodificadas para obter as séries temporais sintéticas finais. O treinamento é iterativo, ajustando os pesos do gerador e do supervisor para otimizar a qualidade das sequências geradas. O objetivo é garantir que

as sequências geradas sejam estatisticamente semelhantes às reais.

Resumidamente, o *TimeGAN* irá incorporar dados em um espaço latente e, em seguida, recuperará essas informações para gerar séries temporais sintéticas.

4.2 PREPARAÇÃO DOS DADOS

O processo de captura de dados é realizado estabelecendo-se uma conexão *multicast* UDP com a B3 para recebimento de atualizações através do sinal de difusão de *Market Data*. Os dados são recebidos no formato FIX com compressão FAST. Para cada informação recebida o dado é decodificado e as informações de interesse são extraídas e armazenadas em arquivos, um para cada ativo.

O número de observações é influenciado pela frequência de disponibilização de dados que, é diferente para cada ativo e, em alguns casos, depende da dinâmica do mercado. Os dados para o Índice Bovespa (IBOV) são disponibilizados de forma contínua pela B3 a cada 30 segundos durante todo o período de negociação e, os dados para Petrobras (PETR3) a cada negócio realizado. Para o período analisado, o volume de dados para os ativos citados são respectivamente 517,909 observações e 20,503,734 observações. Os dados capturados foram armazenados em arquivos do tipo texto no formato CSV (*comma-separated values*) conforme a estrutura de dados descrita na tabela 2. Como separador de colunas foi utilizado o caractere *virgula*.

Tabela 2: Estrutura do arquivo de dados

Índice	Atributo	Tipo	Descrição	Exemplo
1	Coluna 1	Data e Hora	Data e hora da observação	2019-04-08 10:04:00
2	Coluna 2	Numérico	Preço do ativo	97203.89

Fonte: Elaborado pelo próprio autor

O campo data e hora da observação não é contínuo devido a limitação do horário de negociação e pelo fato de que não há negociação aos finais de semana e feriados.

Os arquivos gerados não contém dados dados nulos, inválidos ou ausentes e, cada linha corresponde a uma observação. Foi realizado um pré-processamento dos arquivos com o objetivo de criar um arquivo único contendo os preços dos dois ativos em intervalos de

tempo de 1 minuto. Para isso, os dados foram organizados em ordem ascendente por data e hora e, o último preço disponível para cada um dos ativos no fechamento do minuto foi armazenado no novo arquivo.

A inexistência de um novo preço implicará na utilização do valor imediatamente anterior, ou seja, se apenas o preço de um dos ativos variar, o outro ativo permanecerá com o último preço disponível. Como a quantidade de de observações entre as séries é muito diferente, esta abordagem tem como objetivo construir duas séries de dados com o mesmo número de observações. As novas séries organizados por data e hora foram armazenados em um único arquivo texto no formato CSV (*comma-separated values*) onde as colunas são delimitadas pelo caractere *ponto e vírgula*.

Devido à extrema volatilidade e queda significativa dos preços observadas durante a pandemia de Covid-19, optamos por desconsiderar os dados anteriores a 01/06/2020. Esta decisão visa garantir a precisão e a relevância das análises, evitando distorções causadas por eventos atípicos e de grande impacto econômico.

Com esta abordagem, o conjunto de dados inicial contendo 273,672 observações foi reduzido a um conjunto final contendo 160,286 observações e 3 atributos que seguem listados na tabela 3.

Tabela 3: Formato do conjunto final de dados

Índice	Atributo	Tipo	Descrição	Exemplo
1	Coluna 1	Data e Hora	Data e hora da observação	2020-06-01 10:12:00
2	Coluna 2	Numérico	Preço do IBOV	87353.06
3	Coluna 3	Numérico	Preço do PETR3	20.67

Fonte: Elaborado pelo próprio autor

A tabela 4 contém uma amostra das cinco primeiras observações do conjunto de dados.

4.3 ANÁLISE DE CAUSALIDADE

O teste de causalidade de Granger é uma maneira de verificarmos se uma série temporal X ajuda a prever a outra série Y, ou vice-versa. Para o teste de causalidade, nossa hipótese nula é que, dadas duas séries temporais, elas não estão relacionadas, o que im-

Tabela 4: Amostra contendo os 5 primeiras observações

Coluna 1	Coluna 2	Coluna 3
2020-06-01 10:12:00	87353.06	20.67
2020-06-01 10:13:00	86912.26	20.71
2020-06-01 10:14:00	86928.39	20.79
2020-06-01 10:15:00	86971.37	20.83
2020-06-01 10:16:00	87001.54	20.78

Fonte: Elaborado pelo próprio autor

plica que o coeficiente dos modelos autorregressivos é zero. Se o Teste de Causalidade de Granger produzir uma estatística de teste com $valor - p$ significativamente pequeno (< 0.05), podemos rejeitar a hipótese nula e especular que pode haver relação entre as séries temporais.

Para validar as hipóteses, as séries temporais foram analisadas para identificação da defasagem, ou seja, o quanto um ponto esta atrasado no tempo em relação a outro. Isto pode ser feito utilizando-se um gráfico de defasagem. Através dele é possível descobrir se a série segue algum padrão como aleatoriedade, tendências ou sazonalidade. Para a análise de causalidade foi utilizado o software Stata (StataCorp LLC, 2019), do fornecedor *Timberlake Analytics Inc.*, que é utilizado para ciência de dados, incluindo manipulação de dados, visualização, estatísticas e relatórios automatizados.

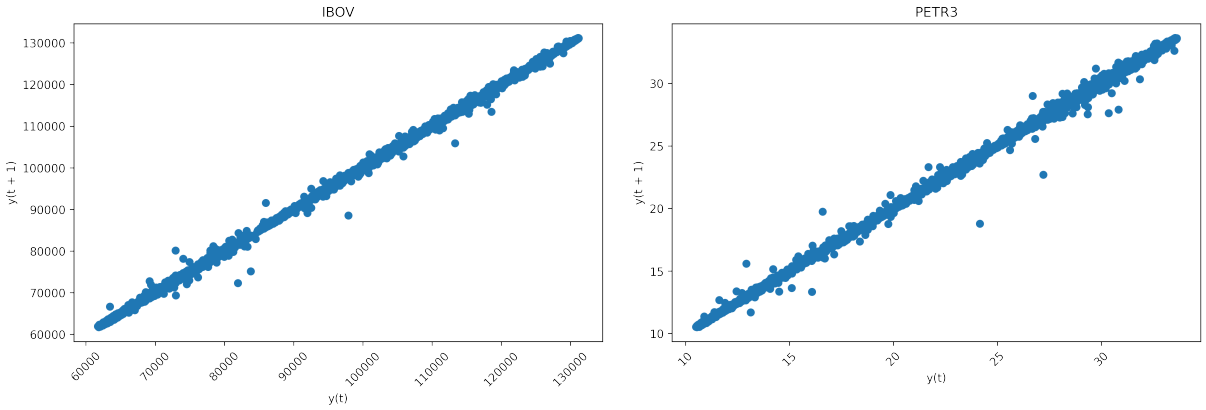
A figura 8 mostra que há alguma autocorrelação para defasagem igual a 1, ou seja, para um ponto de dados o atraso será o ponto de dados anterior. No eixo x temos os dados da série temporal e no eixo y o atraso do ponto de dados da série temporal.

A visualização das séries no decorrer do tempo pode ser observado na figura 9.

4.4 TESTE DE ESTACIONARIEDADE

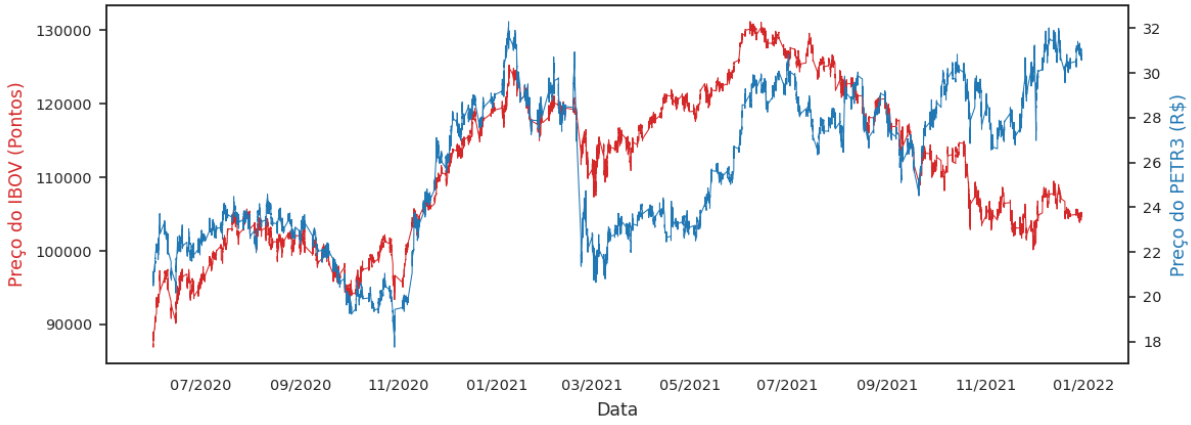
Uma das hipóteses necessárias a respeito de uma série temporal é que ela seja estacionária. Podemos dizer que a estacionariedade é um importante conceito na modelagem de séries temporais e é caracterizada por uma variável que se comporta de forma aleatória ao longo do tempo ao redor de uma média constante refletindo alguma forma de equilíbrio

Figura 8: Autocorrelação para defasagem em sua primeira diferença (lag=1)



Fonte: Gerado no *Google Colab* utilizando a biblioteca *Matplotlib*

Figura 9: Histórico de preços dos ativos



Fonte: Gerado no *Google Colab* utilizando a biblioteca *Matplotlib*

estável. Isto permite que as propriedades estatísticas da série tornem-se mais interessantes. Séries temporais que possuem tendência e/ou sazonalidade não são estacionárias.

Para entender se há presença significativa de tendência nas séries temporais foi utilizado o teste de *Dickey-Fuller* Aumentado, ADF (do acrônimo em inglês *Augmented Dickey-Fuller*) (DICKEY; FULLER, 1979), que é um teste de raiz unitária, que detecta se a série tem comportamento com tendência através de um teste de hipóteses. Para verificar se a série é estacionária ou não verificamos as hipóteses:

- **Hipótese nula (H0)**: se não for rejeitada, sugere que a série temporal possui uma raiz unitária, ou seja, não é estacionária;

- **Hipótese alternativa (H1)**: se a hipótese nula é rejeitada, sugere que a série

temporal não possui uma raiz unitária, ou seja, é estacionária.

Inicialmente, precisamos realizar a seleção adequada de defasagens (lags) e, para isso executamos o comando VARSOC (*Vector Autoregressive Specification Order Criterion*) no Stata. Este comando retorna a função de probabilidade (LL, *log-likelihood*) e o teste de razão de probabilidade (LR, *likelihood-ratio test*), além de quatro critérios de informação, o *Final Prediction Error* (FPE), o critério de Akaike (AIC), o *Hannan Quinn Information Criterion* (HQIC) e o *Schwarz Bayesian Information Criterion* (SBIC).

O resultado obtido está disponível na tabela 5 e, o número ótimo de defasagens do sistema é igual a 4, confirmado pelos testes LR, FPE, AIC, HQIC, SBIC. Os valores com asteriscos geralmente indicam o menor valor do critério de informação, sugerindo a ordem de defasagem mais adequada para o modelo.

Tabela 5: Seleção do número ótimo de defasagens

lag	LL	LR	df	p	FPE	AIC	HQIC	SBIC
0	-3.0e+06				2.5e+09	27.3052	27.3052	27.3053
1	-854991	4.3e+06	4	0.000	8.45532	7.81055	7.81063	7.81083
2	-841338	27306	4	0.000	7.46412	7.68586	7.686	7.68633
3	-840887	902.19	4	0.000	7.4337	7.68178	7.68197	7.68244
4	-840824	127.11*	4	0.000	7.42965*	7.68123*	7.68148*	7.68208*

Fonte: Resultados obtidos do *Stata Statistical Software* (StataCorp LLC, 2019)

Podemos agora realizar o teste de estacionariedade utilizado *Dickey-Fuller Aumentado*. Se o $valor - p$ obtido no teste de ADF for menor que o nível de significância (0.05), a hipótese nula será rejeitada. Os resultados são apresentados nas tabelas 6 e 7.

Tabela 6: Teste ADF para IBOV

	Teste estatístico	Valor Crítico (1%)	Valor Crítico (5%)	Valor Crítico (10%)
Z(t)	-1.741	-3.960	-3.410	-3.120
MacKinnon aproximado valor-p para Z(t) = 0.7326				

Fonte: Resultados obtidos do *Stata Statistical Software* (StataCorp LLC, 2019)

A partir dos resultados das tabelas 6 e 7, observamos que as variáveis estão acima do nível de significância $Z(t) > 0.05$, indicando que não existem evidências estatísticas

Tabela 7: Teste ADF para PETR3

	Teste estatístico	Valor Crítico (1%)	Valor Crítico (5%)	Valor Crítico (10%)
Z(t)	-1.680	-3.960	-3.410	-3.120
MacKinnon aproximado valor-p para Z(t) = 0.7596				

Fonte: Resultados obtidos do *Stata Statistical Software* (StataCorp LLC, 2019)

suficientes para rejeitar a hipótese nula de não estacionariedade, ou seja, as séries não são estacionárias.

Em geral, a maior parte dos processos não são estacionários mas, podem se tornar estacionários utilizando-se transformações. Estas transformações das observações originais tem como objetivo obter uma distribuição mais simétrica e próxima da normal ou estabilizar a variância e tornar o efeito sazonal aditivo. Tornar o efeito sazonal aditivo significa que as flutuações sazonais na série temporal são constantes ao longo do tempo, independentemente do nível da série. Em outras palavras, a magnitude das variações sazonais não muda conforme o valor da série aumenta ou diminui.

A transformação mais comum é a diferenciação por ser o mecanismo mais simples. A diferenciação é a diferença entre um valor no tempo t, menos o valor anterior, no tempo $t-1$ (equação 4). Séries não estacionárias podem se transformar em séries estacionárias a partir de sua primeira diferença:

$$\Delta y_t = y_t - y_{t-1} \tag{4}$$

Caso necessário, se a série transformada permanecer não estacionária, podem ser calculadas diferenças adicionais utilizando a equação 5.

$$\Delta^d y_t = \Delta^{d-1} y_t - \Delta^{d-1} y_{t-1} \tag{5}$$

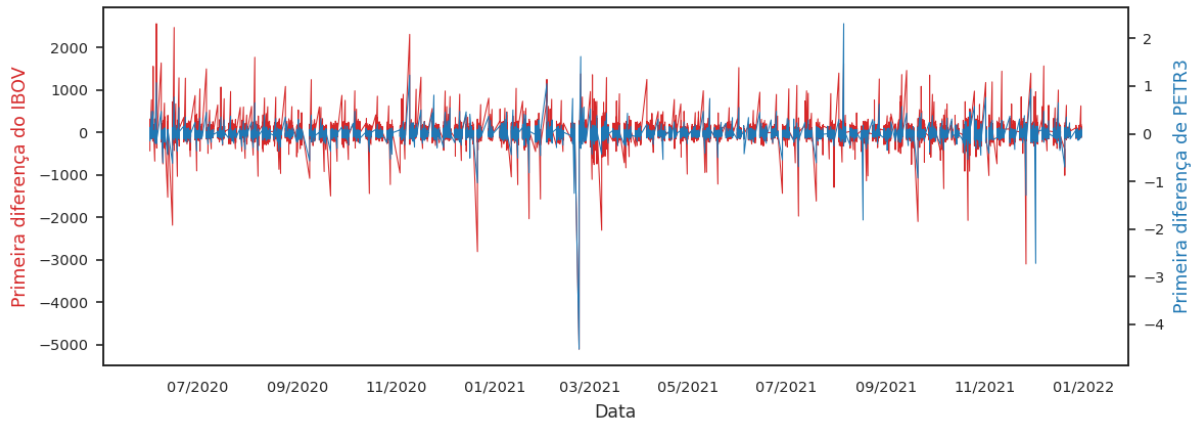
Os modelos de previsão autorregressivos são essencialmente modelos de regressão linear que utilizam os atrasos da própria série como preditores. Sabemos que na regressão linear, é desejável que os preditores (variáveis independentes) não estejam fortemente correlacionados entre si.

Quando os preditores estão correlacionados, ocorre o fenômeno conhecido como multicolinearidade, que pode dificultar a interpretação dos coeficientes de regressão e reduzir

a precisão das estimativas. Assim, tornar uma série estacionária resolve esse problema, pois remove qualquer autocorrelação persistente, tornando os preditores (defasagens da série) nos modelos de previsão quase independentes.

Como as séries avaliadas não são estacionárias, foi aplicado o método da diferenciação e, novas séries foram construídas com a diferença entre a série original, subtraída de sua defasagem de um período. A figura 10 exibe as séries após a primeira diferenciação.

Figura 10: Séries temporais após a primeira diferenciação



Fonte: Gerado no *Google Colab* utilizando a biblioteca *Matplotlib*

O teste ADF foi refeito para as séries diferenciadas e o resultado pode ser observado nas tabelas 8 e 9.

Tabela 8: Teste ADF para IBOV em sua primeira diferença

	Teste estatístico	Valor Crítico (1%)	Valor Crítico (5%)	Valor Crítico (10%)
Z(t)	-200.116	-3.960	-3.410	-3.120
MacKinnon aproximado valor-p para Z(t) = 0.0000				

Fonte: Resultados obtidos do *Stata Statistical Software* (StataCorp LLC, 2019)

Após a diferenciação, os valores de $valor - p$ obtidos no teste de ADF estão abaixo do nível de significância (0.05) o que indica que as séries se tornaram estacionárias.

Tabela 9: Teste ADF para PETR3 em sua primeira diferença

	Teste estatístico	Valor Crítico (1%)	Valor Crítico (5%)	Valor Crítico (10%)
Z(t)	-213.147	-3.960	-3.410	-3.120
MacKinnon aproximado valor-p para Z(t) = 0.0000				

Fonte: Resultados obtidos do *Stata Statistical Software* (StataCorp LLC, 2019)

4.5 MODELOS VAR E CORRELAÇÃO RESIDUAL

O Modelo de Vetor Autorregressivo (VAR) é uma ferramenta estatística usada para capturar as relações entre múltiplas variáveis à medida que elas mudam ao longo do tempo (SIMS, 1980). Ele generaliza o modelo autorregressivo univariado, permitindo a análise de séries temporais multivariadas, ou seja, sistemas de séries temporais com duas ou mais variáveis. A fórmula do modelo VAR (6) pode ser representada como:

$$VAR(p) = c + A_1 X_{(t-1)} + A_2 X_{(t-2)} + \cdots + A_p X_{(t-p)} + \varepsilon_{(t)} \tag{6}$$

onde,

- VAR(p) é o modelo VAR de ordem p.
- c é um termo constante.
- $A_1, A_2, \ldots, A_p$ são matrizes de coeficientes.
- $X_{(t-1)}, X_{(t-2)}, \ldots, X_{(t-p)}$ são valores defasados das variáveis.
- $\varepsilon_{(t)}$ é o termo de erro.

Quando as séries são estacionárias em nível, ou seja, suas propriedades estatísticas, como média, variância e autocorrelação, permanecem constantes ao longo do tempo, escolhemos empiricamente o número adequado de defasagens com base no critério de informação de Akaike (1974) (AIC). O AIC é uma métrica que avalia a qualidade de um modelo estatístico, sendo preferível o modelo com menor valor de AIC.

Se as séries não forem estacionárias, o modelo VAR não é apropriado, pois não estimará corretamente as relações entre as variáveis. Nesse caso, o Modelo de Correção de Erro Vetorial (VEC), uma extensão do VAR, é recomendado para estimar os coeficientes das variáveis que são estacionárias em suas diferenças.

Considerando o menor valor de AIC obtido para os ativos **IBOV e PETR3 (AIC = 7.68123)** com **4 defasagens**, a matriz de correlação dos resíduos indica um relacionamento fraco ou correlação positiva fraca entre as variáveis ($0.25 < r = 0.4870 < 0.50$).

A análise dos resíduos é crucial para identificar observações discrepantes. Utilizamos a estatística de Durbin e Watson (1950), que testa a presença de autocorrelação nos resíduos de uma regressão. No caso de defasagem igual a 1, o valor obtido foi 2.0, indicando ausência de correlação significativa.

4.6 COINTEGRAÇÃO

A cointegração é um método estatístico usado para testar a correlação entre duas ou mais séries temporais não estacionárias no longo prazo ou para um período especificado. Ela ajuda a identificar parâmetros de longo prazo ou equilíbrio para duas ou mais variáveis. A cointegração ocorre quando duas ou mais séries temporais não estacionárias têm um equilíbrio de longo prazo e se movem juntas de tal forma que sua combinação linear resulta em uma série temporal estacionária.

Para verificar se as séries diferenciadas cointegram foi utilizado o comando *VECRANK* no Stata. O *VECRANK* produz estatísticas usadas para determinar o número de equações de cointegração em um modelo vetorial de correção de erros (VECM). Ele implementa três métodos principais para essa determinação:

- Estatística de traço de Johansen
- Estatística do máximo autovalor de Johansen
- Critérios de informação

Esses métodos ajudam a identificar quantas relações de longo prazo existem entre as variáveis no modelo. O resultado obtido esta disponível na tabela 10 onde r indica a classificação máxima, ER a estatística de traço e LL a função de probabilidade (log-likelihood).

Como a estatística de traço em $r = 0$ é de $1.06e + 05$ e excede seu valor crítico de 15.41, rejeitamos a hipótese nula de nenhuma equação de cointegração. Da mesma forma, como a estatística de traço em $r = 1$ é de $4.60e + 04$ e excede seu valor crítico de 3.76, rejeitamos a hipótese nula de que existe uma ou menos equações de cointegração. Em

Tabela 10: Cointegração entre IBOV e PETR3 em sua primeira diferença

r	Parâmetros	LL	Autovalor	ER	Valor Crítico (5%)
0	14	-893527.57	.	1.06e+05	15.41
1	17	-863667.75	0.23874	4.60e+04	3.76
2	18	-840667.62	0.18951		

Fonte: Resultados obtidos do *Stata Statistical Software* (StataCorp LLC, 2019)

suma, a hipótese nula de não cointegração foi rejeitada e a hipótese de que existe pelo menos uma equação de cointegração, não pode ser rejeitada.

O sistema VEC foi estimado usando as **4 defasagens** indicadas pelos testes LR, FPE, AIC, HQIC, SBIC e, em cada equação foram utilizados os dados em sua primeira diferença com o objetivo de melhorar a estimação (tabela 5).

Como resultado temos uma equação de cointegração ($_ce1$) que, na equação de IBOV está funcionado ao nível de 1% ($valor - p = 0.0000$) e está corrigindo a trajetória em 49.37% (-0.4937). Na equação de PETR3, está funcionado ao nível de 1% ($valor - p = 0.0000$) e está corrigindo a trajetória de IBOV em 0,02% (-0.00028) contribuindo pouco para o equilíbrio do sistema.

A equação de cointegração $_ce1$ está funcionando ao nível de 1%, entretanto, observa-se que para o sistema se corrigir (tender à média de longo prazo) para cada 1 ponto de IBOV, são necessários 2993.825 pontos no sentido contrário e a constante indica a velocidade de correção do sistema, neste caso de 18.58 períodos de tempo.

A tabela 11 contém os resultados obtidos no Teste de Causalidade de Granger para os ativos IBOV e PETR3 que, evidencia uma **causalidade bidirecional**, na qual a hipótese de não causalidade é rejeitada ao nível de 1%, para as duas variáveis.

O teste de Granger verifica se uma variável (X) é útil para prever outra variável (Y) avaliando a causalidade entre as séries temporais, mas não considera a possibilidade de múltiplas relações de cointegração.

O teste de Johansen (1995) é usado para verificar se há cointegração entre várias séries temporais avaliando se as séries compartilham tendências de longo prazo e se estão ligadas por relações estáveis. Como permite mais de uma relação de cointegração, torna-se mais

Tabela 11: Resultado do Teste de Causalidade de Granger

	IBOV-PETR3		PETR3-IBOV	
F(4, 273659)	5455.59		44.39	
Prob > F	0.0000		0.0000	
chi2(4)	21823.09	(assintótico)	177.57	(assintótico)
Prob > chi2	0.0000	(assintótico)	0.0000	(assintótico)

Fonte: Resultados obtidos do *Stata Statistical Software* (StataCorp LLC, 2019)

geral do que o teste de Granger.

Logo, enquanto o teste de Causalidade de Granger permite verificar se as variáveis são co-integradas ou não, o teste de *Johansen* é uma etapa adicional realizada após o teste de Granger e permite identificar quantos vetores de co-integração existem entre as variáveis.

Assim, para constatar a existência de cointegração, foi executado o teste de cointegração de *Johansen* para testar as hipóteses sobre o número de vetores de cointegração:

- **Hipótese nula (H0)**: Não há vetores de cointegração
- **Hipótese alternativa (H1)**: Existe pelo menos um vetor de cointegração

A tabela 12 contém o resultado do teste de *Johansen*. Comparando a estatística calculada com os valores críticos e como o $valor - p$ obtido é menor que 0.05, rejeitamos a hipótese nula e confirmamos que as séries de fato se cointegram.

Tabela 12: Resultados do teste de Johansen: IBOV e PETR3

t-statistic	Valor-p	Valores críticos
		01%: -3.896480
-63.302545	0.0	05%: -3.336152
		10%: -3.044466

Fonte: Resultados obtidos do *Stata Statistical Software* (StataCorp LLC, 2019)

4.7 TREINAMENTO DO MODELO

Os modelos utilizados nos componentes da estrutura *TimeGAN* para geração dos dados sintéticos realísticos serão compostos por redes neurais recorrentes (RNN) do tipo *Long Short-Term Memory* (LSTM).

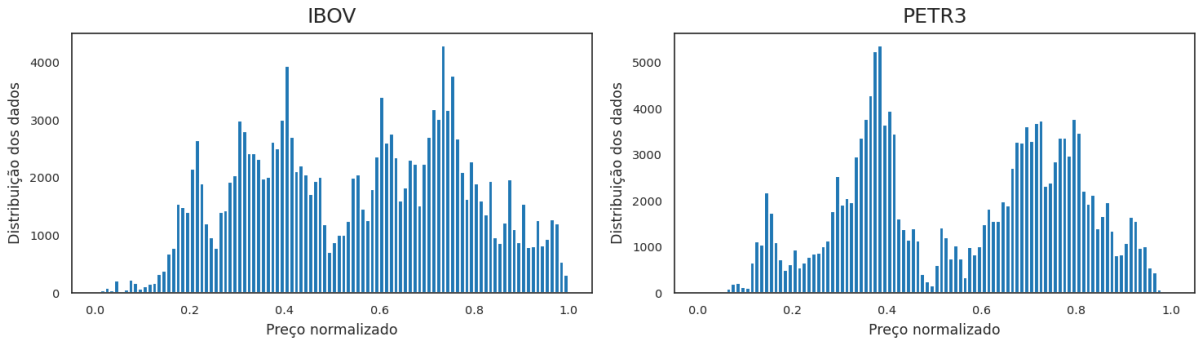
As redes neurais recorrentes (RNN) são conhecidas por seus ciclos de retroalimentação ou *feedback*. Esta característica faz com que sejam mais especializadas e eficientes para alguns tipos de aplicação como reconhecimento de sequências e previsões sobre resultados futuros, como previsões de mercados de ações. Este mecanismo de retroalimentação atua como uma via expressa permitindo que os sinais de erro passem de uma etapa para a anterior. Podemos dizer que a RNN possui duas entradas, o presente e o passado recente. Isto é importante porque a sequência de dados contém informações importantes sobre o que está vindo a seguir.

Entretanto, quando treinamos uma RNN, o processo na qual o algoritmo ajusta seus pesos é por meio da descida do gradiente permitindo que o modelo determine a direção a ser tomada para reduzir as perdas (ou minimizar a função de custo). A dissipação do gradiente é um dos principais problemas pois, se o gradiente da função de custo decai exponencialmente com o tempo, o modelo para de aprender. Neste caso, o gradiente pode se tornar muito pequeno (desvanecimento do gradiente) ou muito grande (explosão do gradiente) o que torna o treinamento difícil. Esses problemas podem dificultar o treinamento da rede, pois:

- **Desvanecimento do gradiente (*Vanishing*)**: os gradientes se tornam muito pequenos fazendo com que as atualizações dos pesos sejam insignificantes e a rede tenha dificuldade em aprender dependências de longo prazo;
- **Explosão do gradiente (*Exploding*)**: os gradientes se tornam muito grandes resultando em atualizações de pesos muito grandes o que pode fazer com que a rede se torne instável.

O utilização de arquiteturas como LSTM e a aplicação de regularização podem ajudar a mitigar esses problemas. Em uma rede LSTM, a descida do gradiente ajuda a atualizar os pesos dos neurônios para melhorar a precisão das previsões ao longo do tempo minimizando a diferença entre as previsões da rede e os valores reais esperados (perdas). Isso é crucial para lidar com dependências de longo prazo em dados sequenciais. Além disso, por ser capaz de aprender dependências de longo prazo em problemas de predição de sequência torna-se ideal para classificar e prever séries temporais. Isto é possível devido a mecanismos chamados *Gates* que são diferentes operações de tensor que podem aprender quais informações adicionar ou remover do estado oculto. Por causa dessa capacidade, a memória de curto prazo é um problema menor.

Figura 11: Histograma das séries normalizadas



Fonte: Gerado no *Google Colab* utilizando a biblioteca *Matplotlib*

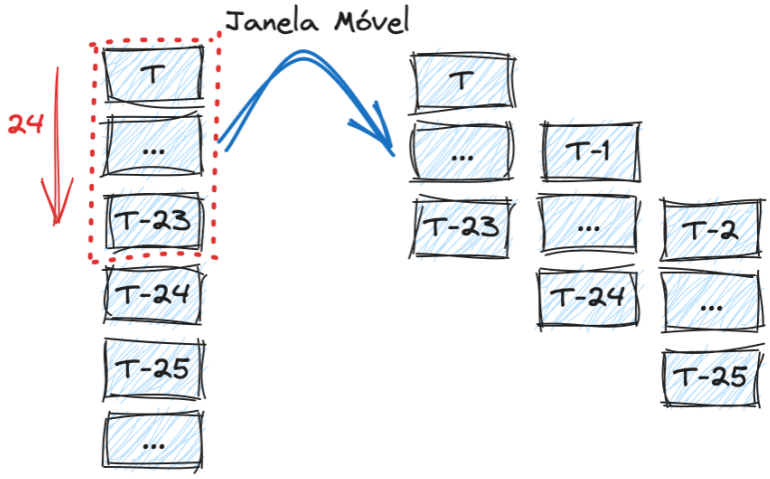
Inicialmente o conjunto de dados (tabela 4) foi normalizados para que os preços das séries estejam em uma mesma escala. Foi utilizado *MinMaxScaler* para dimensionar e traduzir cada recurso individualmente de modo que estejam no intervalo $X = [0, 1]$. Dimensionando os dados deixamos todos os atributos comparáveis conforme observado na figura 11 pois este tipo de normalização garante que todas as *features* estão na mesma escala. Entretanto, tem a limitação de tratar valores discrepantes, sendo sensível à presença de *outliers*.

O conjunto de dados preparado previamente foi dividido em duas partes, uma para treinamento e outra para validação na proporção 80% e 20% respectivamente (128,226 e 32,080 observações). Os dados de treinamento foram divididos em 128,202 janelas móveis com sequências sobrepostas contendo 24 observações conforme recomendação do artigo original (YOON; JARRETT; SCHAAR, 2019). A figura 12 ilustra como as janelas móveis foram constituídas.

Com as séries de dados normalizadas, o gerador deve ser capaz gerar sequências aleatórias com distribuição uniforme com valores neste mesmo intervalo. Para esta tarefa foi utilizada a biblioteca *NumPy* (HARRIS et al., 2020) pelo fato de oferecer um conjunto de funções matemáticas abrangentes, geradores de números aleatórios, rotinas de álgebra linear, transformadas de *Fourier* e muito mais.

Para a geração das sequências aleatórias com distribuição uniforme no intervalo entre 0 e 1 foi utilizada a função *np.random.uniform* da biblioteca *NumPy*. O código utilizado esta descrito na listagem 1.

Figura 12: Diagrama da janela móvel



Fonte: Elaborado pelo próprio autor

```
def generateRandomSeries():
    while True:
        # seq_len = tamanho da janela; n_seq = número de colunas
        yield np.random.uniform(low=0, high=1, size=(seq_len, n_seq))

random_series_iterator =
    iter(tf.data.Dataset
         .from_generator(generateRandomSeries, output_types=tf.float64)
         .batch(batch_size)
         .repeat())
```

Listagem 1: Código para geração de sequências falsas

Como mencionado anteriormente, a estrutura *TimeGAN* é composta por quatro componentes: uma rede de incorporação, uma rede de reconstrução, um gerador e discriminador de sequência. Cada um dos componentes é composto por uma rede neural LSTM com 2 camadas, cada uma contendo 24 nós (*hidden dimension*), uma camada de *Dropout* (0.2) entre elas e uma camada para processar informações e criar conexões entre os neurônios. A listagem 2 corresponde a rotina utilizada para construção dos modelos.

A camada de *Dropout* tem como objetivo definir aleatoriamente as unidades de entrada como zero com uma frequência de taxa em cada etapa durante o tempo de treinamento, melhorando a generalização da rede e reduzindo o efeito de *overfitting* (HINTON et al., 2012). O *Dropout* é uma técnica que reduz co-adaptações complexas de neurônios, já que um neurônio não pode confiar na presença de outros neurônios em particular. Assim, é

```
def createModel(n_layers, hidden_units, output_units, name):
    model = Sequential(name=name)
    for i in range(n_layers):
        model.add(LSTM(units=hidden_units
            , return_sequences=True
            , name=f'LSTM_{i + 1}'))
        model.add(Dropout(0.2, name=f'DROPOUT_LSTM_{i + 1}'))

    model.add(Dense(units=output_units
        , activation='sigmoid'
        , name='OUT'))
    return model
```

Listagem 2: Rotina para criação dos modelos

forçado a aprender recursos mais robustos que são úteis em conjunto com muitos subconjuntos aleatórios diferentes dos outros neurônios. Pensando em uma rede onde o objetivo é realizar previsões, o *Dropout* é uma forma de garantir que o modelo seja robusto para a perda de qualquer evidência individual.

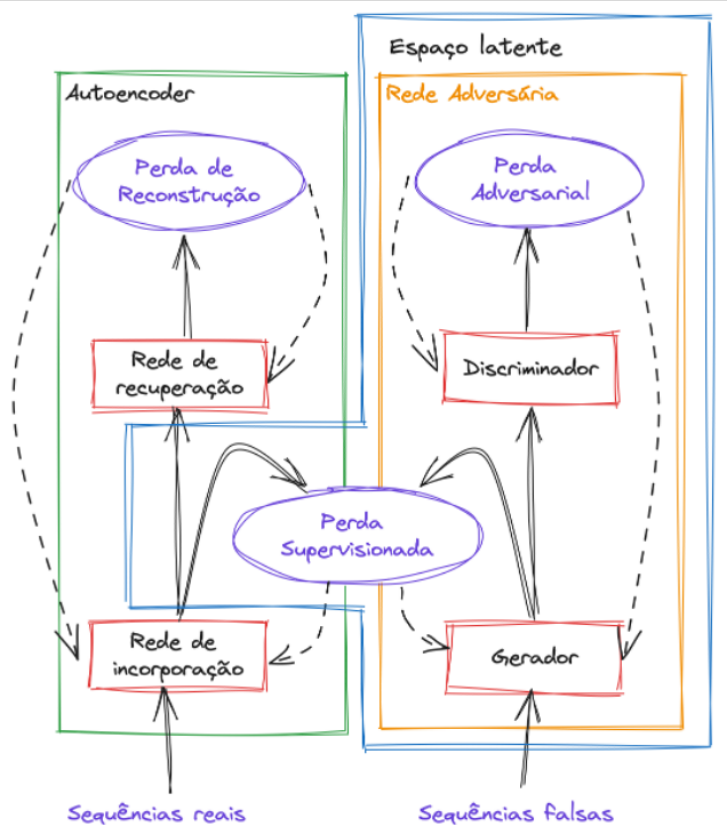
O treinamento do *TimeGAN* envolve várias etapas distintas para garantir a geração eficaz de dados sintéticos de séries temporais. Estas etapas consistem no treinamento do *Autoencoder*, Supervisor e Treinamento Conjunto (figura 13).

No treinamento do *Autoencoder* os dados de entrada serão comprimidos em uma representação latente (Codificação). Os dados serão então reconstruídos a partir da representação latente (Decodificação). A perda será calculada pela diferença entre os dados de entrada originais e os reconstruídos, ajustando os pesos para minimizar essa diferença.

Em seguida, com o treinamento do *Supervisor*, o modelo tenta prever a próxima etapa da sequência com base na representação latente. A perda supervisionada será calculada pelo Erro Quadrático Médio (MSE, da sigla em inglês *Mean Squared Error*) entre a representação latente real e a prevista, ajustando os pesos para melhorar a precisão da previsão. Isso ajuda a garantir que o modelo aprenda a estrutura temporal dos dados de forma eficaz.

Finalmente, no treinamento conjunto o Gerador cria dados sintéticos a partir de ruído aleatório. O Discriminador distingue os dados entre reais e sintéticos. A rede de incorporação transforma os dados de entrada em uma representação latente enquanto a rede de reconstrução reconstrói os dados a partir dessa representação. A perda combinada é

Figura 13: Diagrama de treinamento



Fonte: Adaptado de Tai, Wang e Huang (2023)

calculada e inclui as perdas adversarias (para o gerador e discriminador), perdas de reconstrução (para o autoencoder) e perdas supervisionadas (para o supervisor). O objetivo é equilibrar essas perdas para que o modelo aprenda a gerar dados sintéticos realistas que preservem a estrutura temporal dos dados reais.

A perda de reconstrução ($E_{loss_{T0}}$) do *Autoencoder* compara o quão bem foi a reconstrução dos dados sintéticos em relação aos dados reais, ou seja, mede a qualidade da reconstrução dos dados após a codificação. Esta perda é importante para o entendimento do quão perto a linha de regressão está dos pontos previstos, entrada (X) versus saída (X_{tilde}) (equação 7) e, é calculada utilizando o MSE.

$$E_{loss_{T0}} = (10 * \sqrt{MSE(X, X_{tilde})}) \tag{7}$$

A perda supervisionada é responsável por capturar o quão bem o gerador se aproxima do próximo passo de tempo no espaço latente. A perda combinada reflete a relação entre

as redes Geradora e Discriminadora.

Para o treinamento foram utilizados os hiper-parâmetros descritos na tabela 13.

Tabela 13: Hiper-parâmetros para o treinamento

n_seq	2	Número de *features*
seq_len	24	Tamanho da janela de dados
batch_size	128	Tamanho do *batch*
hidden_dim	24	Número de nós em cada camada oculta da rede
learning_rate	0.001	Taxa de aprendizagem
num_layers	3	Número de camadas LSTM
train_steps	10.000	Número de iterações para treinamento

Fonte: Elaborado pelo próprio autor

Inicialmente as redes de incorporação (*Embedder*) e reconstrução (*Recovery*) foram criadas para compor o *Autoencoder*. A rede de incorporação será responsável por reduzir a dimensionalidade do espaço de aprendizagem adversário. A rede de reconstrução por permitir reconstruções precisas dos dados originais e suas representações latentes. O modelo *Autoencoder* esta disponível na figura 14.

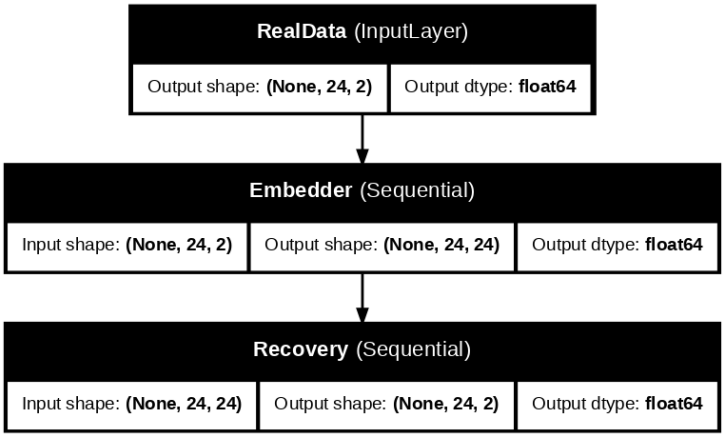
Na etapa seguinte foi realizado o treinamento da rede supervisora com o objetivo de otimizar a perda supervisionada. A rede supervisora é responsável por prever a próxima etapa de uma série temporal e, durante o treinamento, as informações anteriores da série serão utilizadas como entrada para fazer essa previsão. O código *Python* utilizado para construção da rede *Autoencoder* está disponível na listagem 3.

```
# X representa a entrada do gerador. Essa entrada é uma sequência de dados latentes
↪  (ou ruído) que o gerador transforma em uma série temporal sintética
H = embedder(X)
X_tilde = recovery(H)
autoencoder = Model(inputs=X, outputs=X_tilde, name='Autoencoder')
```

Listagem 3: Rede Autoencoder

A perda supervisionada (G_{loss_s}) no *TimeGAN* foi calculada usando o erro quadrático médio (MSE) entre duas sequências de representações latentes (equação 8) onde, H representa as representações latentes geradas pelo gerador e $H_{hat_{sup}}$ é a saída do supervisor, que tenta prever a próxima etapa da série temporal com base na representação anterior. Os pesos treináveis para a perda supervisionada incluem os pesos do gerador e os pesos

Figura 14: Rede Autoencoder



Fonte: Imagem gerada no *Google Colab* utilizando *plot_model* do TensorFlow (TENSORFLOW, 2024)

do supervisor. Embora pareça redundante, esses pesos não são atualizados durante o retro-propagação dessa perda.

$$G_{loss_s} = MSE(H[:, 1:, :], H_{hat_{sup}}[:, :-1, :]) \tag{8}$$

As figuras 15 e 16 exibem respectivamente as curvas da perda de reconstrução e da perda supervisionada obtidas durante o treinamento do modelo.

Figura 15: Perdas de reconstrução

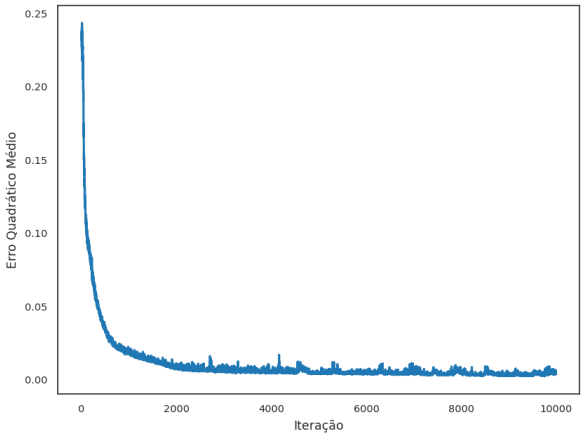
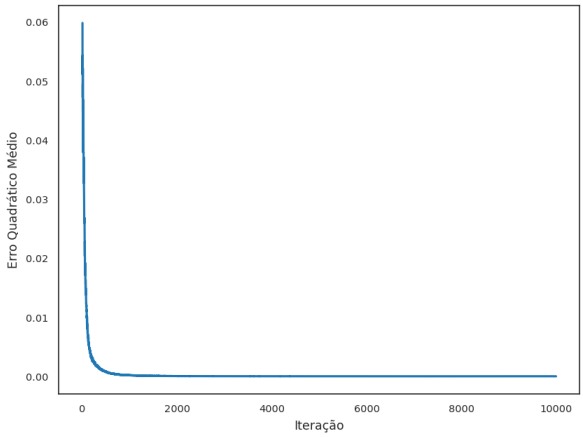


Figura 16: Perdas supervisionadas

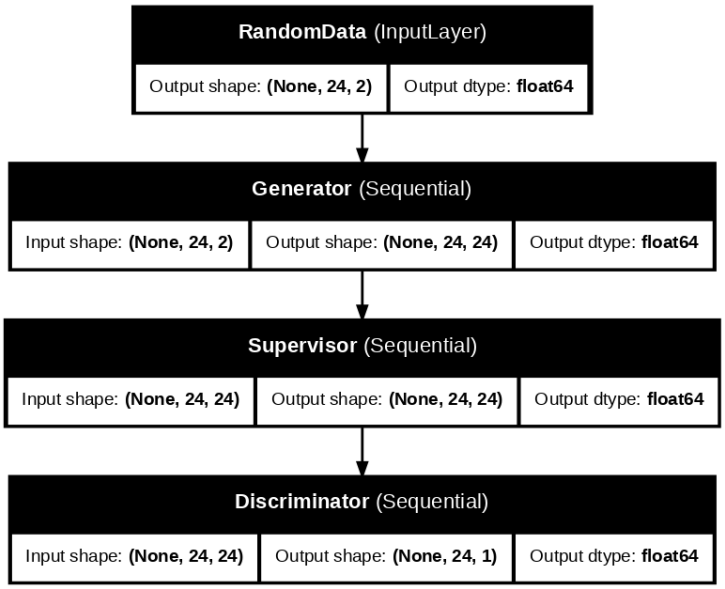


Fonte: Gerado no *Google Colab* utilizando a biblioteca *Matplotlib*

A etapa final consiste no treinamento conjunto do *TimeGAN* que envolve a combinação das abordagens adversária e supervisionada para que o modelo apenda a distribuição de dados das séries temporais.

No treinamento adversário um gerador e um discriminador são treinados de forma adversária. (figura 17).

Figura 17: Rede adversária



Fonte: Imagem gerado pelo próprio autor no *Google Colab* utilizando o *plot_model* do TensorFlow (TENSORFLOW, 2024)

O gerador tenta criar dados sintéticos que sejam indistinguíveis dos dados reais, enquanto o discriminador tenta diferenciar entre os dados reais e os sintéticos. As listagens 4 e 5 compõem a arquitetura adversária utilizada para o treinamento conjunto dos modelos.

O objetivo do treinamento conjunto é minimizar as três funções de perda e garantir que os dados sintéticos gerados pelo *TimeGAN* não apenas sejam realistas, mas também preservem as características temporais e as dependências dos dados originais.

Este processo é realizado de forma iterativa até que o modelo atinja um equilíbrio onde o gerador produz dados sintéticos de alta qualidade e o discriminador não consiga distinguir entre dados reais e sintéticos com alta precisão. A listagem 6 contém o fluxo de treinamento iterativo conjunto.

Seguindo as recomendações do artigo original de Yoon, Jarrett e Schaar (2019), o treinamento foi realizados para 10.000 iterações para cada etapa. A tabela 14 contém as perdas do treinamento conjunto (listagem 6) onde as colunas são respectivamente:

- D_{Loss} é a perda do discriminador;
- G_{Loss_U} é a perda não supervisionada;

```
# Z representa a entrada de dados falsos
# E_hat é calculado como uma estimativa da representação latente dos dados de séries
↪  temporais. Essa representação latente é usada para gerar dados sintéticos
↪  realistas por meio do gerador
E_hat = generator(Z)

# H_hat representa a matriz de transição que modela a dinâmica temporal dos dados.
↪  Essa matriz é estimada durante o treinamento do modelo e é usada para gerar
↪  sequências sintéticas que se assemelham às sequências reais
H_hat = supervisor(E_hat)

# Y_fake representa as sequências sintéticas geradas pelo modelo
Y_fake = discriminator(H_hat)
adversarial_supervised = Model(inputs=Z, outputs=Y_fake)
```

Listagem 4: Rede adversária

```
# Y_Fake_e representa as sequências sintéticas geradas pelo modelo, mas no espaço
↪  latente, ou seja, é a saída do discriminador quando alimentado com as sequências
↪  latentes geradas pelo gerador
Y_fake_e = discriminator(E_hat)
adversarial_embedder = Model(inputs=Z, outputs=Y_fake_e, name='AdversarialNetwork')

# X_hat representa as sequências sintéticas geradas pelo modelo e é obtido a partir
↪  das sequências latentes geradas pelo gerador e, em seguida, transformado de volta
↪  para o espaço original dos dados
X_hat = recovery(H_hat)
synthetic_data = Model(inputs=Z, outputs=X_hat, name='SyntheticData')

# Y_real representa os dados reais de séries temporais usados para treinamento
Y_real = discriminator(H)
discriminator_model = Model(inputs=X, outputs=Y_real, name='DiscriminatorReal')
```

Listagem 5: Rede adversária no espaço latente

- G_{Loss_S} é a perda supervisionada;
- G_{Loss_V} é a perda de correspondência de momento utilizada para melhorar a diversidade das amostras geradas;
- $E_{Loss_{T0}}$ é a perda de reconstrução.

No *TimeGAN*, a expressão matemática que representa a perda do gerador no treinamento conjunto é baseada na função de perda adversária típica das *GANs*, adaptada para séries temporais. A função de perda do gerador L_G (equação 9) pode ser descrita da seguinte forma:

```
step_g_loss_u = step_g_loss_s = step_g_loss_v = step_e_loss_t0 = step_d_loss = 0
for step in range(train_steps):
    for kk in range(2):
        X_ = next(real_series_iterator)
        Z_ = next(random_series_iterator)
        step_g_loss_u, step_g_loss_s, step_g_loss_v = train_generator(X_, Z_)
        step_e_loss_t0 = train_embedder(X_)
    X_ = next(real_series_iterator)
    Z_ = next(random_series_iterator)
    step_d_loss = get_discriminator_loss(X_, Z_)
    if step_d_loss > 0.05:
        step_d_loss = train_discriminator(X_, Z_)
```

Listagem 6: Treinamento conjunto

$$L_G = -\mathbb{E}_{z \sim p_z(z)}[D(G(z)))] \tag{9}$$

Onde:

- G é o gerador que tenta criar dados sintéticos a partir de uma distribuição de ruído $\mathbf{z}$
- D é o discriminador que tenta distinguir entre dados reais e sintéticos
- $\mathbb{E}$ denota a esperança

Essa função de perda é combinada com outras componentes para garantir que os dados sintéticos preservem as características temporais dos dados reais. A função de perda total do gerador (G_{Loss}) corresponde a soma das perdas G_{Loss_S}, G_{Loss_U} e G_{Loss_V}. As perdas observadas durante o treinamento conjunto estão disponíveis na tabela 14.

4.8 GERAÇÃO DE DADOS SINTÉTICOS

Utilizando o gerador treinado foram criadas as sequências latentes que servirão como base para as séries temporais sintéticas. O gerador irá produzir sequências que imitem as características temporais dos dados reais. As sequências latentes geradas são decodificadas pelo decodificador treinado reconstruindo as sequências temporais no espaço original dos dados garantindo que mantenham a estrutura e as dependências temporais dos dados reais. O código *Python* utilizado para geração das séries sintéticas esta na listagem 7.

Finalmente, as séries temporais sintéticas geradas foram validadas para garantir que

Tabela 14: Perdas do treinamento conjunto

Iteração	D_{Loss}	G_{Loss_U}	G_{Loss_S}	G_{Loss_V}	$E_{Loss_{T0}}$
1	2.0553	0.7131	0.0005	0.4469	0.0560
1,001	1.8465	1.2746	0.0000	0.0577	0.0145
2,001	1.7456	1.3788	0.0001	0.0854	0.0137
3,001	1.3041	1.7778	0.0002	0.0242	0.0064
4,001	1.2354	1.9799	0.0001	0.0698	0.0053
5,001	1.3165	1.8683	0.0001	0.0084	0.0061
6,001	1.5537	1.5282	0.0000	0.0365	0.0057
7,001	1.4451	1.6782	0.0000	0.0371	0.0049
8,001	1.4527	1.4798	0.0000	0.0734	0.0049
9,001	1.3113	1.5590	0.0001	0.0426	0.0038
10,000	1.3286	1.4744	0.0001	0.0480	0.0046

Fonte: Elaborado pelo próprio autor utilizando *Google Colab*

```
generated_data = []
for i in range(n_windows):
    Z_ = next(random_series_iterator)
    d = synthetic_data(Z_)
    generated_data.append(d)
```

Listagem 7: Geração de dados sintéticos

são realistas e úteis para a aplicação desejada. Isto foi feito pela comparação entre os dados sintéticos e os dados reais gerados e através das análises PCA e t-SNE. Esta abordagem também foi utilizado por Yoon, Jarrett e Schaar (2019) e, permite uma análise visual dos dados sintéticos gerados pelo *TimeGAN* em relação aos dados originais pois mostram uma sobreposição entre os dados.

4.9 PREVISÃO DE MOVIMENTO DE PREÇOS

Para previsão de preços foi construído um modelo utilizando uma rede com uma camada LSTM e uma camada densa para conectar todos os neurônios da camada anterior. O *KerasTurner* (O'MALLEY et al., 2019) é uma biblioteca para otimização de hiperparâmetros e foi utilizada para encontrar a melhor configuração para o modelo. O objetivo é melhorar o desempenho do modelo sem que seja necessário ajustar manualmente cada hiperparâmetro. O algoritmo utilizado foi o *RandomSearch* que seleciona

aleatoriamente combinações de hiperparâmetros dentro do espaço de busca definido. Este algoritmo embora não seja tão eficiente quando comparado a métodos mais sofisticados como a otimização Bayesiana, é simples de implementar e é bastante eficaz, especialmente quando o espaço de busca não é muito grande. A busca foi realizada em cinco ciclos com 3 execuções com 10 épocas com o objetivo de otimizar a métrica RMSE. Os valores dos hiperparâmetros e da métrica RSME obtidos estão disponíveis na tabela 15.

Tabela 15: Valores ótimos de hiperparâmetros obtidos com KerasTuner

Hiperparâmetro	Valor ótimo
learning_rate	0.0029925267975241636
units	384
dropout	False
Score (RMSE)	0.005460539769379348

Fonte: Resultados obtidos utilizando o KerasTuner

Os dados restantes foram normalizados entre 0 e 1 utilizando *MinMaxScaller* e divididos em treinamento e validação na proporção 80/20. O modelo utilizando os valores ótimos dos hiperparâmetros foi treinado, validado e utilizado para predição. O processo foi repetido utilizando os dados sintéticos. Os valores previstos foram comparados com os dados reais para avaliação da qualidade da previsão.

5 RESULTADOS

Nesta seção, descrevemos os resultados obtidos no experimento. Foram utilizados dados obtidos da plataforma UMDF da B3 que, após serem pré-processados formaram o conjunto final de dados. O ambiente de execução foi o *Google Colab*. O tempo total para o treinamento do modelo, geração das séries sintéticas e previsão foi de aproximadamente 8 horas e 25 minutos.

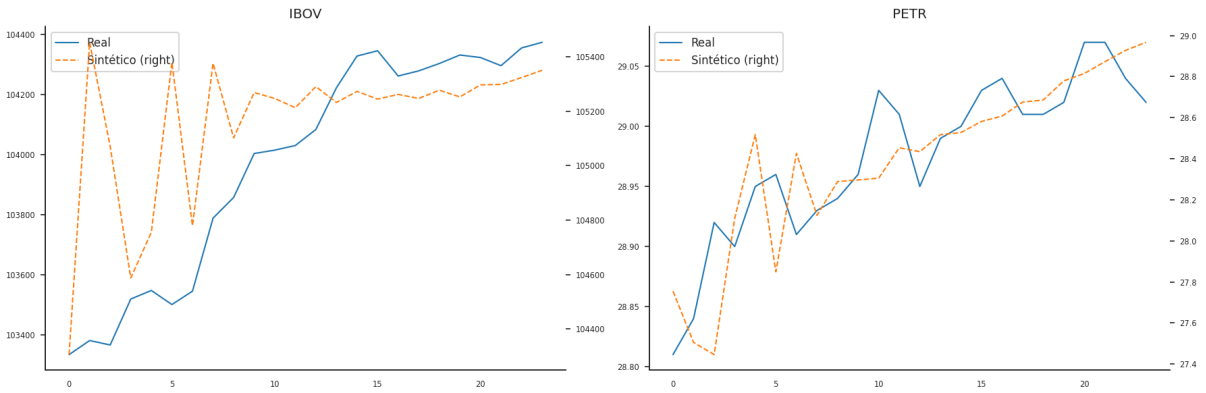
A primeira avaliação realizada consistiu em comparar os dados reais com os dados sintéticos para entender o quão próximos os valores estão. Foram selecionados aleatoriamente uma amostra contendo 24 observações e comparamos os valores reais com os valores sintéticos gerados. A figura 18 apresenta essa comparação, com o eixo x representando o

```python
def build_model(hp):
    opt = optimizers.Adam(hp.Float('learning_rate', min_value=1e-4, max_value=1e-2,
    ↪  sampling='log'))
    model = models.Sequential()
    model.add(layers.LSTM(
                units=hp.Int('units', min_value=32, max_value=512, step=32),
                input_shape=(10, 1),
                name=f'LSTM_1'))
    if hp.Boolean('dropout'):
        model.add(keras.layers.Dropout(rate=0.25))
    model.add(layers.Dense(units=1,
                    activation='sigmoid',
                    name='OUT'))
    model.compile(optimizer=opt,
                  loss="mean_squared_error",
                  metrics=[tf.keras.metrics.RootMeanSquaredError()])
    return model
```

Listagem 8: Modelo para previsão

índice observado e o eixo y representando o valor.

Figura 18: Comparação entre os dados reais e sintéticos



Fonte: Gerado no *Google Colab* utilizando a biblioteca *Matplotlib*

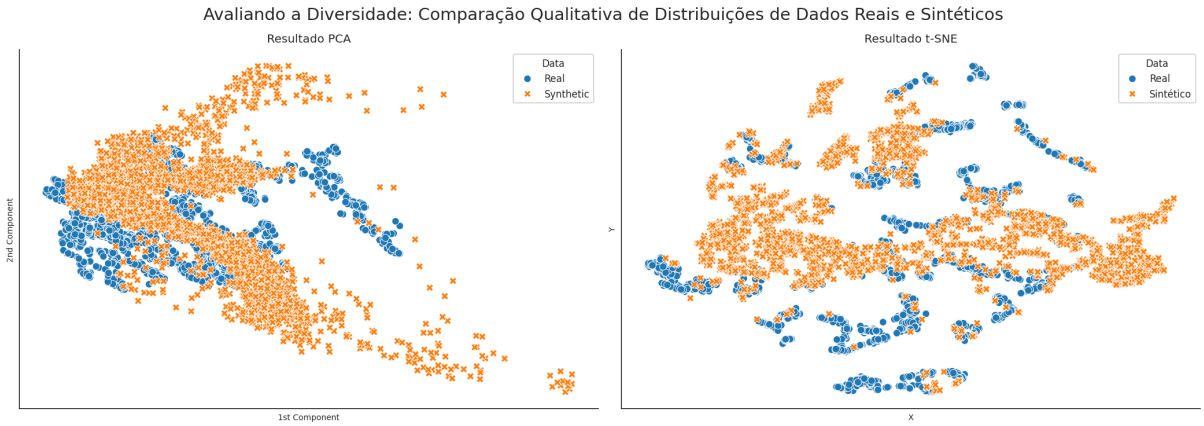
Realizamos uma primeira avaliação visual comparando os dados sintéticos com os dados reais. Naturalmente, essa não é uma forma eficaz de avaliação mas, nos fornece uma boa ideia de como está a qualidade dos dados gerados. Podemos observar que os dados sintéticos tentam manter uma semelhança estrutural e padrões em relação aos dados reais. Entretanto, a quantidade de observações da amostra é insuficiente para verificar se a distribuição de valores entre dados reais e sintéticos é adequada e se o modelo está capturando corretamente a variabilidade dos dados.

Utilizamos os métodos Análise de Componentes Principais, PCA e t-SNE para reduzir a dimensionalidade dos dados. A sobreposição entre os dados sintéticos e reais indicam que os dados gerados são úteis. Para esta análise utilizamos a PCA e o t-SNE em um subconjunto contendo 2,000 janelas de dados selecionadas aleatoriamente.

Com a PCA reduzimos a dimensionalidade dos dados e, assim, podemos ver graficamente os componentes principais dos dados reais e sintéticos em um gráfico de dispersão. Isso ajuda a visualizar se os dados sintéticos capturaram a estrutura dos dados reais.

Ao comparar os resultados de PCA e t-SNE e podemos observar como eles preservam a estrutura dos dados. Em geral, o t-SNE é melhor em preservar a estrutura local, enquanto o PCA é melhor em preservar a estrutura global. Os gráficos da figura 19 indicam em uma análise visual que o resultado obtido é promissor e a qualidade dos dados sintéticos é boa pois seguem os padrões dos dados reais. Isto é caracterizado pela fato de vermos uma boa sobreposição entre os pontos de dados sintéticos e reais.

Figura 19: Comparação das distribuições de dados reais e sintéticos.



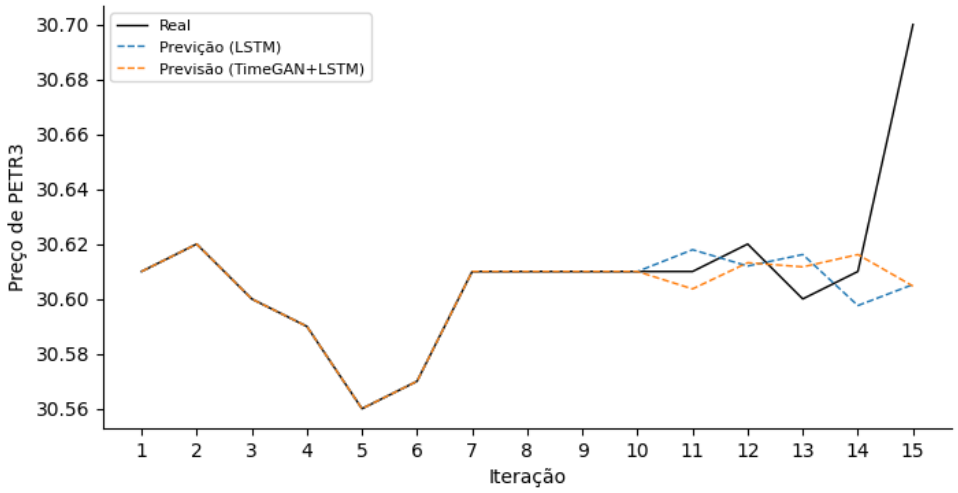
Fonte: Gerado no *Google Colab* utilizando a biblioteca *Matplotlib*

Para identificar quais previsões são mais confiáveis foi utilizado a métrica *Trustworthiness Score* (GHOBRIAL et al., 2023). Embora a métrica tenha sido inicialmente desenvolvido para avaliar previsões de redes neurais convolucionais (CNNs), ela pode ser adaptado para outras arquiteturas de redes neurais, incluindo LSTM's. Previsões com um score alto são consideradas mais confiáveis, enquanto previsões com um score baixo podem necessitar de uma revisão mais detalhada. O uso do *Trustworthiness Score* com PCA e t-SNE pode fornecer uma camada adicional de validação, ajudando a identificar

projeções que são mais confiáveis e aquelas que podem necessitar de revisão ou ajuste. Ao aplicar a métrica os valores obtidos para t-SNE e PCA são respectivamente 1.000 e 0.994. O *Trustworthiness Score* varia de 0 a 1, com valores mais altos indicando melhor preservação das semelhanças pareadas. Uma pontuação de confiabilidade de 1 indica a preservação perfeita das semelhanças entre pares. Isto indica que os dados produzidos podem ser úteis para serem utilizadas em previsões.

Os dados sintéticos e reais foram utilizado para predição utilizando uma rede neural LSTM. O modelo foi treinado com os dados reais de treinamento e validado com os dados reais de teste. Em seguida o modelo foi treinado com os dados sintéticos e validado com os dados reais de teste. O objetivo é avaliar a utilidade dos dados sintéticos gerados para predição. O resultado obtido esta disponível na tabela 16 e a figura 20 exibe a previsão realizada para os próximos 5 preços de PETR3.

Figura 20: Previsão dos próximos 5 preços de PETR3



Fonte: Gerado no *Google Colab* utilizando a biblioteca *Matplotlib*

O MAPE corresponde a média de todos os erros percentuais absolutos entre os valores previstos e reais são obtidos. Para a previsão utilizando dados sintéticos o valor foi de 0.00082362030 indicando que as previsões estão muito próximas dos valores observados. Em geral, um valor de MAPE inferior a 20% é considerado bom para previsão de séries temporais, o que indicaria que, em média, as previsões durante todo o período de tempo estavam menos de 20% distantes dos valores reais.

O valor do MAE indica que, em média, as previsões com dados sintéticos estão cerca

Tabela 16: Comparação das métricas resultados da previsão

	MAE	RMSE	MAPE
Real	0.0278570259	0.00190372525	0.00090826572
Sintético	0.0252665892	0.00186966816	0.00082362030

Fonte: Elaborado pelo próprio autor

de 0.0252665892 unidades dos valores reais indicando que o modelo esta ajustando-se aos dados observados.

O RMSE nos fornece uma medida da dispersão dos erros do modelo em relação aos dados reais e, quanto menor o valor mais preciso é o modelo em suas previsões. Um valor de RMSE de 0.0252665892 indica que as previsões estão extremamente próximas dos valores reais. Essa baixa diferença e, considerando que os valores estão normalizados entre 0 e 1, temos um indício de que o modelo está ajustando-se muito bem aos dados observados.

6 CONCLUSÃO

O propósito desta pesquisa foi verificar se a utilização de uma arquitetura *TimeGAN*, era viável na previsão de valores futuros de séries financeiras. Foi utilizada uma grande quantidade de dados dos ativos Ibovespa (IBOV) e Petrobras (PETR3), obtidos via plataforma UMDF da B3, e realizados testes de presença de raiz unitária em cada série, escolha de melhor número de defasagens (*lags*), teste de cointegração, estimação VEC e teste de causalidade. O modelo VEC mostrou-se funcional e com causalidade bidirecional entre as duas séries de tempo. Isto indica que o comportamento passado é útil na previsão do comportamento futuro. Na sequência, foi estimada a *TimeGAN* utilizando as duas séries, com geração das séries sintéticas que serão utilizadas para a previsão dos dois ativos.

Esta arquitetura mostrou-se efetiva pois ela aprendeu a dinâmica temporal de dados das séries e foi capaz de gerar dados sintéticos de aparência realística. Em um cenário, no qual os dados iniciais são insuficiente para uma previsão mais precisa, a utilização de dados sintéticos para ampliar o conjunto de dados reais, pelos resultados obtidos, demonstrou-se uma boa alternativa.

Entretanto, o processo de treinamento é complexo e pode ser computacionalmente intensivo, especialmente para séries temporais longas. O modelo também apresenta sensibilidade a hiper-parâmetros o que pode afetar a performance do treinamento. Embora projetado para preservar as dinâmicas temporais, o modelo pode não capturar nuances extremamente complexas presentes nas séries temporais.

Os resultados são promissores na utilização da arquitetura para projeção de dados de séries de tempo financeiras. As possibilidades de pesquisas futuras, dando continuação a esta, são inúmeras, pois pode-se variar os hiper-parâmetros, a frequência dos dados, os tamanhos de amostra, número de iterações, sem comprometer a qualidade dos dados sintéticos, aumentando a robustez do modelo.

A B3 possui grande movimentação financeira e muita volatilidade, sendo um lugar ideal para testes de modelos generativos, que podem trabalhar conjuntamente com os modelos econométricos para obtenção de melhores resultados de projeção de preços.

REFERÊNCIAS

AKAIKE, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, v. 19, n. 6, p. 716–723, 1974.

APTE, M.; VAISHAMPAYAN, S.; PALSHIKAR, G. K. Detection of causally anomalous time-series. *INTERNATIONAL JOURNAL OF DATA SCIENCE AND ANALYTICS*, 11, n. 2, p. 141–153, MAR 2021. ISSN 2364-415X.

ARIA, M.; CUCCURULLO, C. bibliometrix: An r-tool for comprehensive science mapping analysis. *Journal of Informetrics*, Elsevier, v. 11, n. 4, p. 959–975, 2017. Disponível em: <https://doi.org/10.1016/j.joi.2017.08.007>.

ARJOVSKY, M.; CHINTALA, S.; BOTTOU, L. Wasserstein generative adversarial networks. In: PRECUP, D.; TEH, Y. W. (Ed.). *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017. (Proceedings of Machine Learning Research, v. 70), p. 214–223. Disponível em: <https://proceedings.mlr.press/v70-/arjovsky17a.html>.

Clarivate Analytics. *Web of Science: A Comprehensive Research Platform*. [S.l.], 2023. Disponível em: <[1](https://support.clarivate.com/ScientificandAcademicResearch/s-/article/Web-of-Science-Citing-Web-of-Science-data?language=en_US)>.

COMMONS, W. *File:GrangerCausalityIllustration.svg — Wikimedia Commons, the free media repository*. 2020. [Online; accessed 02-April-2023]. Disponível em: <\urlhttps:-//commons.wikimedia.org/w/index.php?title=File:GrangerCausalityIllustration-.svg&oldid=50548627>.

DICKEY, D. A.; FULLER, W. A. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, Taylor & Francis, v. 74, n. 366a, p. 427–431, 1979.

DURBIN, J.; WATSON, G. S. Testing for serial correlation in least squares regression: I. *Biometrika*, [Oxford University Press, Biometrika Trust], v. 37, n. 3/4, p. 409–428, 1950. ISSN 00063444. Disponível em: <http://www.jstor.org/stable-/2332391>http://www.jstor.org/stable/2332391.

ESTEBAN, C.; HYLAND, S. L.; RäTSCH, G. *Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs*. arXiv, 2017. Disponível em: <https:/-/arxiv.org/abs/1706.02633>.

FAMA, E. F. The behavior of stock-market prices. *The Journal of Business*, University of Chicago Press, v. 38, n. 1, p. 34–105, 1965. ISSN 00219398, 15375374. Disponível em: <http://www.jstor.org/stable/2350752>http://www.jstor.org/stable/2350752.

FEKRI, M. N.; GHOSH, A. M.; GROLINGER, K. Generating energy data for machine learning with recurrent generative adversarial networks. *Energies*, MDPI AG, v. 13, n. 1, p. 130, Dec 2019. ISSN 1996-1073. Disponível em: <http://dx.doi.org/10.3390-/en13010130>http://dx.doi.org/10.3390/en13010130.

FISHER, R. A. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, The Royal Society London, v. 222, n. 594-604, p. 309–368, 1922.

GHOBRIAL, A.; HOND, D.; ASGARI, H.; EDER, K. A trustworthiness score to evaluate cnns predictions. *arXiv preprint arXiv:2301.08839*, 2023.

GOODFELLOW, I.; 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*, v. 27, 2014.

GOOGLE. *Google Colaboratory*. 2018. `https://colab.research.google.com/`.

GRANGER, C. W. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, JSTOR, p. 424–438, 1969.

HAI, Q.; ZHANG, S.; LIU, C.; HAN, G. Hard disk drive failure prediction based on gru neural network. In: *2022 IEEE/CIC International Conference on Communications in China (ICCC)*. [S.l.: s.n.], 2022. p. 696–701.

HARRIS, C. R.; MILLMAN, K. J.; WALT, S. J. van der; GOMMERS, R.; VIRTANEN, P.; COURNAPEAU, D.; WIESER, E.; TAYLOR, J.; BERG, S.; SMITH, N. J.; KERN, R.; PICUS, M.; HOYER, S.; KERKWIJK, M. H. van; BRETT, M.; HALDANE, A.; RÍO, J. F. del; WIEBE, M.; PETERSON, P.; GÉRARD-MARCHANT, P.; SHEPPARD, K.; REDDY, T.; WECKESSER, W.; ABBASI, H.; GOHLKE, C.; OLIPHANT, T. E. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

HEBB, D. O. *The organization of behavior: A neuropsychological theory*. New York: Wiley, 1949. Hardcover. ISBN 0-8058-4300-0.

HINTON, G. E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. Disponível em: <http://arxiv.org/abs-/1207.0580>http://arxiv.org/abs/1207.0580.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, v. 79, n. 8, p. 2554–2558, abr. 1982. ISSN 0027-8424. Disponível em: <http://view.ncbi.nlm.nih.gov/pubmed-/6953413]>http://view.ncbi.nlm.nih.gov/pubmed/6953413].

HUNTER, J. D. *Matplotlib: A 2D graphics environment*. [S.l.]: IEEE COMPUTER SOC, 2007. 90–95 p.

IEEE. *Top Programming Languages 2021*. 2021. Disponível em: <https://spectrum.ieee-.org/top-programming-languages/>.

JOHANSEN, S. *Likelihood-based inference in cointegrated vector autoregressive models*. [S.l.]: OUP Oxford, 1995.

LI, Q.; ZHANG, X.; MA, T.; LIU, D.; WANG, H.; HU, W. A multi-step ahead photovoltaic power forecasting model based on timegan, soft dtw-based k-medoids clustering, and a cnn-gru hybrid neural network. *Energy Reports*, v. 8, p. 10346–10362, 2022. ISSN 2352-4847. Disponível em: <https://www.sciencedirect.com/science/article-/pii/S2352484722016237>.

MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 127–147, 1943.

MINSKY, M.; PAPERT, S. A. *Perceptrons: An Introduction to Computational Geometry*. [S.l.]: The MIT Press, 1969. ISBN 9780262343930.

NEYMAN, J.; PEARSON, E. On the problem of the most efficient tests of statistical. 1933.

O'MALLEY, T.; BURSZTEIN, E.; LONG, J.; CHOLLET, F.; JIN, H.; INVERNIZZI, L.; OUTROS. *KerasTuner*. 2019. `https://github.com/keras-team/keras-tuner`.

PEARSON, K. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Taylor & Francis, v. 2, n. 11, p. 559–572, 1901.

PLESNER, M. *FeTGAN: Federated Time-Series Generative Adversarial Network*. 7 2021. http://resolver.tudelft.nl/uuid:5e79a3d4-4ba1-4baa-af1d-c744b545e3df.

Python Software Foundation. *Python Language Reference, version 3.x*. 2023. Acesso em: 28 ago. 2023. Disponível em: <https://www.python.org>.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2020. Disponível em: <[6](https://www.R-project.org/)>.

R Core Team. *VOSviewer: Visualizing Scientific Landscapes*. Vienna, Austria, 2023. Disponível em: <[1](https://www.vosviewer.com/)>.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958.

ROTHMAN, D. *Artificial Intelligence by Example: Acquire Advanced AI, Machine Learning, and Deep Learning Design Skills, 2nd Edition*. Packt Publishing, Limited, 2020. (Expert insight). ISBN 9781839211539. Disponível em: <https://books.google-.com.br/books?id=BVRpzQEACAAJ>.

RStudio Team. *RStudio: Integrated Development Environment for R*. Boston, MA, 2020. Disponível em: <[9](http://www.rstudio.com/)>.

RUMELHART, D. E.; MCCLELLAND, J. L. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. [S.l.]: MIT Press, 1986.

SHANGGUAN, A.; XIE, G.; FEI, R.; MU, L.; HEI, X. Train wheel degradation generation and prediction based on the time series generation adversarial network. *Reliability Engineering System Safety*, v. 229, p. 108816, 2023. ISSN 0951-8320. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0951832022004355>.

SIMS, C. A. Macroeconomics and reality. *Econometrica: journal of the Econometric Society*, JSTOR, p. 1–48, 1980.

StataCorp LLC. *Stata Statistical Software: Release 16*. 2019. College Station, TX: StataCorp LLC. Disponível em: <https://www.stata.com>.

TAI, C.-Y.; WANG, W.-J.; HUANG, Y.-M. Using time-series generative adversarial networks to synthesize sensing data for pest incidence forecasting on sustainable agriculture. *Sustainability*, v. 15, p. 7834, 05 2023.

TENSORFLOW. *plot_model utility*. 2024. `https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model`. Acessado em: 26 de abril de 2023.

VIDOTTO, M. C.; BALBI, P. P.; JUNIOR, E. H. Previsão de séries temporais de ações com dados sintéticos gerados com timegan. *Revista Eletrônica Gestão e Serviços*, Instituto Metodista de Ensino Superior, v. 14, p. 4144–4169, 2023. Disponível em: <http://www.spell.org.br/documentos/ver-/73994/previsao-de-series-temporais-de-acoes-com-dados-sinteticos-gerados-com-timegan>http://www.spell.org.br/documentos/ver/73994/previsao-de-series-temporais-de-acoes-com-dados-sinteticos-gerados-com-timegan.

WERBOS, P. Beyond regression: New tools for prediction and analysis in the behavior science. *PhD thesis, Harvard University*, 1974.

YOON, J.; JARRETT, D.; SCHAAR, M. van der. Time-series generative adversarial networks. In: WALLACH, H.; LAROCHELLE, H.; BEYGELZIMER, A.; ALCHé-BUC, F. d'; FOX, E.; GARNETT, R. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. v. 32. Disponível em: <https://proceedings-.neurips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf>.

ZHANG, K.; ZHONG, G.; DONG, J.; WANG, S.; WANG, Y. Stock market prediction based on generative adversarial network. *Procedia computer science*, Elsevier, v. 147, p. 400–406, 2019.

ZHANG, Y.; ZHOU, Z.; LIU, J.; YUAN, J. Data augmentation for improving heating load prediction of heating substation based on timegan. *Energy*, v. 260, p. 124919, 2022. ISSN 0360-5442. Disponível em: <https://www.sciencedirect.com/science/article/pii-/S0360544222018205>.