UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
BACHARELADO EM ENGENHARIA FÍSICA

THEODORA LUÍSA BRANDÃO

MODELAGEM E PREVISÃO DO ÍNDICE S&P 500 UTILIZANDO REDES NEURAIS RECORRENTES

SÃO CARLOS

2025

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
BACHARELADO EM ENGENHARIA FÍSICA

THEODORA LUÍSA BRANDÃO

MODELAGEM E PREVISÃO DO ÍNDICE S&P 500 UTILIZANDO REDES NEURAIS RECORRENTES

Trabalho Final de Curso apresentado ao Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, para obtenção do título/grau de bacharel em Engenharia Física.
Orientação: Prof. Dr. Pedro Augusto Franco Pinheiro Moreira

SÃO CARLOS
2025

Brandão, Theodora Luísa

Modelagem e previsão do índice S&P 500 utilizando Redes Neurais Recorrentes / Theodora Luísa Brandão -- 2025.
71f.

TCC (Graduação) - Universidade Federal de São Carlos, campus São Carlos, São Carlos
Orientador (a): Pedro Augusto Franco Pinheiro Moreira
Banca Examinadora: Claudio Antonio Cardoso, Raphael Santarelli
Bibliografia

1. Séries Temporais Financeiras. 2. Redes Neurais Recorrentes. 3. Previsão do Índice S&P 500. I. Brandão, Theodora Luísa. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Arildo Martins - CRB/8 7180

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA - CCET

Coordenação de Graduação do Curso de Engenharia Física
Departamento de Física, Rodovia Washington Luiz, Km 235 - Caixa Postal 676 - CEP 13565-905, São Carlos - SP – Brasil - Fone: (16) 3351-8222
e-mail: enfi@df.ufscar.br

FICHA DE AVALIAÇÃO DE TRABALHO FINAL DE CURSO
ENGENHARIA FÍSICA

Aluno: Theodora Luísa Brandão
Título: "Modelagem e previsão do índice S&P 500 utilizando redes neurais recorrentes."
Prof(a). Orientador(a): Pedro Augusto Franco Pinheiro Moreira
Prof(a). Examinador(a) 1: Claudio Antonio Cardoso
Prof(a). Examinador(a) 2: Raphael Santarelli

Itens Avaliados	Orientador	Examinador 1	Examinador 2
Redação **(atribuir notas de 0 a 2)**	1,5	1,5	2,0
Apresentação oral **(atribuir notas de 0 a 2)**	1,0	1,5	1,5
Conteúdo desenvolvido no trabalho **(atribuir notas de 0 a 4)**	3,0	3,0	3,0
Arguição **(atribuir notas de 0 a 2)**	1,5	2,0	1,5

Observações: Não consta.

São Carlos, 1º de julho de 2025.

gov.br Documento assinado digitalmente
PEDRO AUGUSTO FRANCO PINHEIRO MOREIRA
Data: 01/07/2025 11:40:34-0300
Verifique em https://validar.iti.gov.br

_____

Prof(a). Orientador(a)

gov.br Documento assinado digitalmente
CLAUDIO ANTONIO CARDOSO
Data: 01/07/2025 11:46:49-0300
Verifique em https://validar.iti.gov.br

_____

Prof(a). Examinador(a) 1

gov.br Documento assinado digitalmente
RAPHAEL SANTARELLI
Data: 01/07/2025 11:57:37-0300
Verifique em https://validar.iti.gov.br

_____

Prof(a). Examinador(a) 2

Este trabalho é dedicado aos meus pais, Idemar Luís Brandão e Rosangela Aparecida Parra.

Agradecimentos

Primeiramente, agradeço ao meu orientador, Prof. Dr. Pedro Augusto Franco Pinheiro Moreira, pela oportunidade concedida para a realização deste trabalho. Estendo meus agradecimentos aos colegas de trabalho pelo auxílio e apoio ao longo do projeto. Por último, agradeço especialmente ao Gabriel de Albuquerque, pois sem seu suporte este trabalho não teria sido possível.

Resumo

O presente trabalho tem como objetivo desenvolver um modelo preditivo para o índice S&P 500 utilizando Redes Neurais Recorrentes do tipo Long Short-Term Memory (LSTM), dada sua capacidade de capturar padrões temporais de longo prazo em séries temporais financeiras. O índice é composto pelas 500 maiores empresas de capital aberto dos Estados Unidos, sendo um importante indicador do desempenho do mercado acionário norte-americano e, por extensão, da economia global, servindo como referência para investidores institucionais, gestores de fundos e analistas econômicos em todo o mundo. A metodologia adotada neste trabalho envolveu a coleta e o tratamento de dados históricos do S&P 500, decomposição da série temporal, modelagem e previsão do índice. Os resultados obtidos demonstram que o modelo LSTM foi capaz de capturar com razoável precisão a dinâmica do índice S&P 500, acompanhando suas tendências e variações no conjunto de teste. A comparação gráfica entre os valores reais e previstos revelou uma boa aderência nas fases de estabilidade e crescimento, embora com maior dificuldade em períodos de alta volatilidade. A decomposição da série temporal reforçou a presença de uma tendência de longo prazo ascendente, sazonalidade estável e impacto significativo de eventos macroeconômicos, evidenciando a complexidade da série e a importância de considerar seus componentes estruturais na modelagem preditiva.

Palavras-chave: Redes Neurais Recorrentes. Séries Temporais Financeiras. Modelagem Numérica. Previsão de Ativos. Índice S&P 500.

Abstract

The present study aims to develop a predictive model for the S&P 500 index using Long Short-Term Memory (LSTM) Recurrent Neural Networks, due to their ability to capture long-term temporal patterns in financial time series. The index comprises the 500 largest publicly traded companies in the United States and serves as a key indicator of the performance of the U.S. stock market and, by extension, the global economy. It is widely used as a benchmark by institutional investors, fund managers, and economic analysts around the world. The methodology adopted in this work involved the collection and preprocessing of historical S&P 500 data, time series decomposition, and the modeling and forecasting of the index. The results demonstrate that the LSTM model was able to capture the dynamics of the S&P 500 index with reasonable accuracy, effectively tracking its trends and variations in the test set. The graphical comparison between actual and predicted values revealed strong alignment during periods of stability and growth, although performance declined during times of heightened volatility. The time series decomposition reinforced the presence of a long-term upward trend, stable seasonality, and a significant influence of macroeconomic events, highlighting the complexity of the series and the importance of considering its structural components in predictive modeling.

Keywords: Recurrent Neural Networks. Financial Time Series. Numerical Modeling. Asset Forecasting. S&P 500 Index.

Lista de ilustrações

Lista de quadros

Lista de abreviaturas e siglas

COVID-19	Coronavirus Disease 2019.
LSTM	Long Short-Term Memory – Memória de Longo e Curto Prazo.
NYSE	New York Stock Exchange – Bolsa de Valores de Nova Iorque.
PIB	Produto Interno Bruto.
RMSE	Root Mean Squared Error – Raiz Quadrada do Erro Médio.
RNN	Rede Neural Recorrente.
S&P 500	Standard & Poor's 500 – Índice das 500 maiores empresas listadas nas bolsas NYSE e Nasdaq.
MAPE	Mean Absolute Percentage Error – Erro Percentual Absoluto Médio.
API	Application Programming Interface – Interface de Programação de Aplicações.
FED	Federal Reserve – Banco Central dos Estados Unidos.
VaR	Value at Risk – Valor em Risco.

Lista de símbolos

Y_t	Valor da série temporal no instante t.
$\mathbb{R}$	Conjunto dos números reais.
β_0	Intercepto da regressão temporal (valor inicial da tendência).
β_1	Coeficiente angular da tendência linear.
β_k	Coeficientes da tendência polinomial de ordem k.
η_t	Termo de erro da tendência determinística.
T_t	Componente de tendência da série temporal.
ε_t	Choque aleatório (ruído branco) no modelo de passeio aleatório.
σ_η^2	Variância do erro da tendência determinística.
σ_ε^2	Variância dos choques no modelo estocástico.
$\mathbb{E}$	Operador de valor esperado (esperança matemática), aplicado a uma variável aleatória X como $\mathbb{E}[X]$.
Var	Operador de variância, indicado como $\mathrm{Var}(X)$ para uma variável X.
Cov	Operador de covariância entre duas variáveis X e Y, denotado por $\mathrm{Cov}(X, Y)$.
S_t	Componente sazonal da série temporal.
C_t	Componente cíclica (não periódica).
λ	Frequência angular em componentes harmônicos.
ψ, ϕ	Coeficientes harmônicos usados na modelagem de ciclos.
R_t	Componente de ruído (ruído branco).
α_k	Coeficientes associados às variáveis dummy sazonais.
D_{kt}	Variável dummy indicadora da sazonalidade no período k.
v_k	Potencial de ativação do neurônio k.
x_j	Entrada j de um neurônio.
w_{kj}	Peso sináptico da entrada x_j no neurônio k.

b_k	Viés (bias) do neurônio k.
$\varphi(\cdot)$	Função de ativação (como tanh ou sigmoide).
h_t	Estado oculto da rede no tempo t.
x_t	Entrada da rede no tempo t.
W_{ih}	Matriz de pesos das entradas.
W_{hh}	Matriz de pesos das conexões recorrentes.
b_h	Vetor de bias da camada oculta.
$\sigma(\cdot)$	Função sigmoide.
$\tanh(\cdot)$	Função tangente hiperbólica.
$\text{ReLU}(z)$	Unidade Linear Retificada: $\text{ReLU}(z) = \max(0, z)$.
f_t	Porta de esquecimento (forget gate).
i_t	Porta de entrada (input gate).
$\tilde{C}_t$	Candidato à memória da célula no tempo t.
C_t	Estado da célula LSTM no tempo t.
o_t	Porta de saída (output gate).
$\hat{y}_i$	Valor previsto pelo modelo para a i-ésima observação.
y_i	Valor real da i-ésima observação.
MSE	Mean Squared Error – Erro quadrático médio.
RMSE	Root Mean Squared Error – Raiz do erro quadrático médio.
MAPE	Mean Absolute Percentage Error – Erro percentual absoluto médio.
$\odot$	Produto de Hadamard (multiplicação elemento a elemento).

Sumário

1 Introdução

O mercado financeiro é um dos pilares da economia global, operando como um ecossistema dinâmico onde se negociam diversos ativos — ações, títulos, commodities e derivativos. Sua função principal é intermediar o fluxo de capital entre investidores e instituições, promovendo o crescimento econômico através do financiamento de empresas e governos, além de oferecer mecanismos para gestão de riscos e otimização de investimentos (MISHKIN, 2019; FABOZZI, 2021).

Nesse cenário, as bolsas de valores exercem papel crucial como plataformas regulamentadas que asseguram transparência, segurança e liquidez nas transações. Elas possibilitam que empresas captem recursos pela emissão de ações, enquanto investidores podem participar do crescimento dessas companhias e obter retornos financeiros (BREALEY; MYERS; ALLEN, 2020).

Os índices financeiros são utilizados para monitorar o desempenho do mercado acionário, servindo como medidores de setores específicos ou da economia como um todo. Nesse contexto, o S&P 500 destaca-se como um dos indicadores mais relevantes mundialmente. Criado e mantido pela Standard & Poor's (S&P Global), engloba as 500 maiores empresas de capital aberto listadas nas bolsas norte-americanas Nasdaq e NYSE (New York Stock Exchange), cobrindo diversos setores: tecnologia, saúde, finanças e consumo (S&P GLOBAL, 2024).

O S&P 500 não só reflete a saúde econômica dos Estados Unidos — país que responde por cerca de 25% do PIB mundial (BLANCHARD, 2017) — como também influencia diretamente decisões de investidores, políticas monetárias e estratégias corporativas globalmente (BLANCHARD, 2017). Sua importância é tal que oscilações significativas no índice podem provocar efeitos em cascata em mercados emergentes, como o brasileiro, cuja economia está intrinsecamente ligada aos ciclos internacionais (SHILLER, 2015).

O S&P 500 representa aproximadamente 80% da capitalização do mercado acionário dos Estados Unidos, sendo considerado um espelho da economia americana e referência crucial para investidores globais (AVENUE, 2025). Sua previsão constitui, assim, ferramenta essencial para mitigar riscos sistêmicos, otimizar portfólios e assegurar a estabilidade financeira, principalmente em contextos de alta volatilidade e interdependência econômica.

A evolução histórica do S&P 500, ilustrada na Figura 1, demonstra como eventos macroeconômicos e geopolíticos significativos influenciam o desempenho do índice. Nas últimas décadas, períodos de crise global resultaram em quedas acentuadas seguidas de recuperações, evidenciando a resiliência do mercado acionário.

Entre os episódios mais marcantes está a Crise das Empresas de Tecnologia, conhecida

Figura 1 – Evolução Histórica do S&P 500 com Eventos Relevantes.

Fonte: Elaboração própria a partir de dados do (Yahoo Finance, 2025a).

como Crise Dot-com (2000), quando a supervalorização de empresas do setor levou a uma bolha especulativa e, posteriormente, a uma forte correção nos preços das ações (SHILLER, 2015). Em 2008, a Crise do Subprime desencadeou uma das maiores recessões desde a Grande Depressão, com a falência de grandes instituições financeiras e um colapso no mercado imobiliário, resultando em uma queda acentuada do S&P 500 (BLANCHARD, 2017; MISHKIN, 2019).

Outros eventos, como o impasse do Teto da Dívida nos Estados Unidos em 2011, o referendo do Brexit em 2016, e a pandemia de COVID-19 em 2020, também provocaram volatilidade significativa no índice, refletindo a interconexão entre fatores econômicos, políticos e sociais globais (FABOZZI, 2021; S&P GLOBAL, 2024). Mais recentemente, a escalada da inflação em 2022, impulsionada por choques de oferta e estímulos monetários, trouxe novos desafios para a estabilidade dos mercados, levando a ajustes nas expectativas de investidores e bancos centrais (BODIE; KANE; MARCUS, 2023).

Em momentos de turbulência, é comum observar maior oscilação nos preços, com quedas pronunciadas sendo sucedidas por recuperações, muitas vezes impulsionadas por intervenções governamentais. Para investidores e gestores de risco, compreender e antecipar esses movimentos é fundamental para a proteção de portfólios e a busca por oportunidades em meio à adversidade (BREALEY; MYERS; ALLEN, 2020).

A análise quantitativa desses ciclos, especialmente com o uso de modelos baseados em aprendizado de máquina, permite identificar padrões recorrentes e avaliar a influência de variáveis externas sobre o comportamento do índice. Assim, a modelagem e previsão do S&P

500, especialmente em cenários de crise, tornam-se ferramentas estratégicas para a tomada de decisão no mercado financeiro global.

Dentre essas variáveis externas, a taxa de juros norte-americana apresenta influência na dinâmica do S&P 500, dada sua relação direta sobre o custo do capital e a atratividade para ativos financeiros de alto risco, influenciando estratégias de alocação de capital em períodos de incerteza econômica (MANKIW, 2015). Quando o banco central dos Estados Unidos - Federal Reserve (FED) - aumenta as taxas de juros, os rendimentos de títulos públicos (como o Treasury Bill) tornam-se mais atraentes em comparação com ações. Isso desvia o capital de investidores para ativos de menor risco, pressionando negativamente a demanda por ações e, consequentemente, os preços do S&P 500 (MANKIW, 2015). Além disso, as taxas de juros elevadas aumentam o custo de empréstimos para empresas, reduzindo investimentos em expansão, pesquisa e desenvolvimento. Isso pode impactar os lucros futuros e a avaliação das empresas listadas no S&P 500 (GITMAN; JOEHNK, 2013).

1.1 Objetivos

Este trabalho tem como objetivo principal desenvolver um modelo preditivo para o índice S&P 500 utilizando redes neurais recorrentes com arquitetura Long Short-Term Memory (LSTM - Memória de Curto e Longo Prazo em tradução livre), reconhecida por sua capacidade de capturar dependências de longo prazo em séries temporais. Busca-se analisar as características temporais do índice, identificar padrões relevantes para a modelagem e implementar um modelo capaz de prever os preços futuros do S&P 500. Além disso, pretende-se avaliar a robustez do modelo em períodos de alta volatilidade e crise.

A metodologia adotada baseia-se na previsão de séries temporais por meio de redes neurais recorrentes do tipo LSTM, uma arquitetura avançada projetada para superar limitações das redes neurais recorrentes convencionais, especialmente o problema do desvanecimento do gradiente. Esse problema ocorre durante o treinamento de redes recorrentes tradicionais, quando os gradientes calculados nas etapas iniciais da sequência tornam-se muito pequenos, dificultando a atualização eficaz dos pesos e, consequentemente, a aprendizagem de dependências de longo prazo em dados sequenciais (HAYKIN, 2009).

As LSTM superam essa limitação por meio de um sistema de portas (gates) que controlam seletivamente o fluxo de informações, e permitem que a rede retenha ou esqueça informações relevantes ao longo do tempo, facilitando a captura de padrões temporais complexos e prolongados. Dessa forma, as LSTM possibilitam uma modelagem mais robusta e precisa de séries temporais financeiras, como o índice S&P 500, que apresentam características não lineares e dependências temporais de longo alcance (BHANDARI et al., 2022).

Os objetivos deste trabalho são organizados em etapas que visam a análise, modelagem e validação da previsão do índice S&P 500:

- **Análise e preparação dos dados**: coleta dos dados históricas do índice S&P 500, tratamento dos dados, normalização e criação de janelas temporais para alimentar o modelo.
- **Decomposição da série temporal**: separação da série em componentes de tendência, sazonalidade e ruído para melhor compreensão e modelagem.
- **Implementação das redes neurais recorrentes LSTM**: definição da arquitetura da rede, configuração dos hiperparâmetros, treinamento e validação do modelo com os dados preparados.
- **Análise dos resultados**: avaliação do desempenho do modelo utilizando a métrica Raiz Quadrada do Erro-Médio (Root Mean Squared Error - RMSE) para medir a precisão das previsões.
- **Previsão futura de 30 dias**: aplicação do modelo treinado para realizar previsões sequenciais "às cegas"dos próximos 30 dias, permitindo a comparação e validação dos resultados após esse período.

Este trabalho busca contribuir para o avanço das técnicas de previsão de séries temporais financeiras, explorando o potencial das redes neurais recorrentes LSTM na modelagem do índice S&P 500. Embora os resultados possam oferecer contribuições relevantes sobre o comportamento do mercado, é importante destacar que este estudo tem caráter exclusivamente acadêmico e de pesquisa.

2 Fundamentação Teórica

A modelagem de séries temporais financeiras, como a do índice S&P 500, demanda métodos capazes de identificar padrões não lineares, captar dependências de longo prazo e lidar com eventos abruptos característicos de mercados voláteis. Com o avanço das técnicas de deep learning, as Redes Neurais Recorrentes (RNNs), especialmente as arquiteturas Long Short-Term Memory (LSTM), vêm se destacando por sua habilidade em aprender relações temporais complexas e adaptar-se a mudanças não estacionárias nos dados financeiros (FISCHER; KRAUSS, 2018).

2.1 Séries Temporais

As séries temporais correspondem a um conjunto de observações ordenadas no tempo, nas quais a ordem cronológica assume papel fundamental na análise e modelagem (MORETTIN, 2006). Formalmente, uma série temporal é definida como uma sequência de variáveis aleatórias $\{Y_t\}_{t\in T}$, em que T representa um conjunto de índices temporais discretos ou contínuos. Na prática, as observações são registradas em intervalos equidistantes, como diários, mensais ou anuais (ENDERS, 2015).

Em termos matemáticos, uma série temporal é representada por uma sequência de valores reais ordenados no tempo, expressa pela Equação 2.1:

$$\{Y_t\}_{t=1}^{T}, \quad \text{onde } Y_t \in \mathbb{R} \text{ e } t \in \{1,2,\ldots,T\} \tag{2.1}$$

em que Y_t é o valor observado no instante t e T é o horizonte temporal (MORETTIN, 2006).

A decomposição de uma série temporal em suas componentes estruturais é fundamental para compreender seu comportamento subjacente e aprimorar a modelagem preditiva (MORETTIN, 2006), podendo ser decomposta em tendência, sazonalidade e resíduo, conforme proposto por (BOX et al., 2015) e (COWPERTWAIT; METCALFE, 2009). A separação dessas estruturas permite isolar movimentos de longo prazo, identificar padrões cíclicos regulares e mensurar a presença de ruídos aleatórios, facilitando a aplicação de técnicas estatísticas e de aprendizado de máquina com maior precisão.

A tendência (T_t) representa mudanças de longo prazo em uma série temporal, capturando direcionalidade persistente e sua modelagem divide-se em duas categorias fundamentais: determinística e estocástica.A Determinística caracteriza-se por uma relação funcional explícita com o tempo, geralmente de forma polinomial. A forma mais comum é a tendência linear representada pela Equação 2.2:

$$T_t = \beta_0 + \beta_1 t + \eta_t \tag{2.2}$$

onde β_0 é o intercepto (valor inicial da tendência em $t = 0$), β_1 é o coeficiente angular (taxa de crescimento por unidade de tempo), t é o índice temporal discreto ($t = 1, 2, \ldots, n$) e η_t é o erro de ajuste, frequentemente assumido como $\eta_t \sim \mathcal{N}(0, \sigma_\eta^2)$.

Em casos não lineares, utiliza-se a tendência polinomial de ordem k, conforme a Equação 2.3:

$$T_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \cdots + \beta_k t^k \tag{2.3}$$

A tendência determinística é considerada previsível e não aleatória, pois depende unicamente do tempo t. Para removê-la da série original, aplica-se uma regressão temporal e subtrai-se $\widehat{T}\_t$ da série observada (COWPERTWAIT; METCALFE, 2009).

A tendência estocástica representa a evolução de uma série ao longo do tempo por meio da acumulação de eventos aleatórios. Diferentemente da tendência determinística, essa forma não possui relação funcional fixa com o tempo, e seus efeitos são permanentes. Preços de ativos financeiros frequentemente exibem tendência estocástica, onde choques, como notícias econômicas, têm efeito permanente (TSAY, 2005). O passeio aleatório (random walk) é um dos modelos mais fundamentais da literatura de séries temporais estocásticas, especialmente utilizado para representar tendências de longo prazo em séries financeiras, dado pela Equação 2.4:

$$T_t = T_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim WN(0, \sigma_\varepsilon^2) \tag{2.4}$$

onde T_t é o valor da tendência no tempo t, que representa a acumulação dos choques passados, T_{t-1} é o valor da tendência no instante anterior ($t-1$), refletindo dependência serial total — memória perfeita, ε_t é o termo de erro (choque aleatório) no tempo t, modelado como ruído branco (white noise).

O termo ε_t, por sua vez, é modelado como um ruído branco (white noise), que corresponde a uma sequência de variáveis aleatórias com média zero, variância constante σ^2 e ausência de autocorrelação temporal.

$$\begin{aligned} \mathbb{E}[\varepsilon_t] &= 0 \quad \text{(média zero)} \\ \text{Var}(\varepsilon_t) &= \sigma_\varepsilon^2 \quad \text{(variância constante)} \\ \text{Cov}(\varepsilon_t, \varepsilon_{t+k}) &= 0, \quad \forall k \neq 0 \quad \text{(não autocorrelacionado)} \end{aligned}$$

O ruído branco representa a parte puramente aleatória de uma série temporal, sendo considerado um processo estatisticamente não estruturado. Tal característica reforça a importância

da decomposição para isolar e compreender os componentes estruturais de uma série, já que a presença de passeio aleatório impede o uso direto de modelos que requerem estacionariedade (BOX et al., 2015).

A Sazonalidade (S_t) e uma série temporal representa flutuações periódicas e previsíveis que se repetem em intervalos fixos, como ciclos mensais, trimestrais ou anuais. Esses padrões são intrínsecos a fenômenos econômicos, climáticos e sociais, e sua modelagem adequada é crucial tanto para isolar componentes não sazonais (tendência, ruído), quanto para melhorar a acurácia de previsões e pode ser representada por variáveis dummy ou funções harmônicas (BOX et al., 2015).

As Variáveis Dummy são ferramentas de codificação binária que identificam a presença de um efeito categórico em períodos específicos. É ideal para capturar padrões sazonais, como feriados anuais ou picos de vendas em datas específicas. A principal vantagem dessa abordagem está na interpretação direta dos coeficientes, que quantificam o efeito fixo de cada período (por exemplo, um mês ou trimestre) em relação à categoria de referência (COWPERTWAIT; METCALFE, 2009). Para uma série com sazonalidade de período s, utilizam-se $s-1$ variáveis, conforme a Equação 2.5:

$$S_t = \sum_{k=1}^{s-1} \alpha_k D_{kt} \tag{2.5}$$

onde D_{kt} é a variável binária que assume valor 1 se a observação t pertence à estação k e 0 caso contrário; e α_k é o coeficiente que quantifica o desvio médio da estação k em relação à categoria de referência (HAMILTON, 1994).

Entretanto, o uso de variáveis dummy torna-se inviável em séries com períodos longos, como dados diários com $s = 252$, devido ao grande número de parâmetros a serem estimados. Além disso, essa técnica pressupõe sazonalidade constante, não acomodando variações temporais nos padrões sazonais. Por isso, recomenda-se seu emprego em contextos onde os padrões são estáveis e conhecidos, como em vendas mensais com sazonalidade fixa (COWPERTWAIT; METCALFE, 2009).

As Funções Harmônicas, são funções trigonométricas que permitem modelar sazonalidade por meio de combinações de ondas senoidais e cossenoidais, adequadas a padrões suaves ou não fixos. Sua principal vantagem é a redução do número de parâmetros, especialmente útil para períodos longos, através da combinação de ondas com diferentes frequências (BOX et al., 2015). A Equação 2.6 representa a decomposição de Fourier:

$$S_t = \sum_{k=1}^{m} \left[\gamma_k \cos\left(\frac{2\pi kt}{s}\right) + \delta_k \sin\left(\frac{2\pi kt}{s}\right) \right], \tag{2.6}$$

onde γ_k, δ_k são os coeficientes que definem a amplitude e fase de cada harmônico, s é

o período sazonal (ex.: 12 para dados mensais) e m é o número de harmônicos (HAMILTON, 1994).

O Ruído (R_t), também denominado componente aleatória, resíduo ou erro, representa a parcela não explicada da série temporal, capturando variações não sistemáticas e imprevisíveis (BOX et al., 2015). Assume-se que R_t segue um processo de ruído branco (white noise, WN), caracterizado por média zero, variância constante e não autocorrelação (HAMILTON, 1994).

As componentes fundamentais de uma série temporal, tendência (T_t), sazonalidade (S_t) e ruído (R_t), podem ser estruturadas por meio de dois modelos principais: aditivo ou multiplicativo. A escolha entre esses modelos depende da forma como os componentes interagem e da estabilidade da variância ao longo da série (MORETTIN, 2006).

No modelo aditivo, os componentes são somados linearmente, conforme a Equação 2.7:

$$Y_t = T_t + S_t + R_t \tag{2.7}$$

onde o ruído R_t é assumido como um processo de ruído branco, caracterizado por média zero ($\mathbb{E}[R_t] = 0$), variância constante ($\text{Var}(R_t) = \sigma^2$) e ausência de autocorrelação ($\text{Cov}(R_t, R_{t+k}) = 0$ para $k \neq 0$). Esse modelo é indicado quando a amplitude da sazonalidade e do ciclo não depende do nível da tendência. Por exemplo, em dados climáticos, como a temperatura mensal, onde a amplitude sazonal é aproximadamente constante (por exemplo, variações em torno de ±5°C) e não é afetada por uma tendência gradual de aquecimento (COWPERTWAIT; METCALFE, 2009).

O modelo multiplicativo considera que os componentes interagem de forma proporcional, sendo representado pela Equação 2.8:

$$Y_t = T_t \times S_t \times R_t \tag{2.8}$$

Neste caso, o ruído R_t apresenta variância que escala com o nível da série. Para facilitar a análise, aplica-se uma transformação logarítmica que lineariza o modelo, conforme a Equação 2.9:

$$ln(Y_t) = \ln(T_t) + \ln(S_t) + \ln(R_t) \tag{2.9}$$

Esse modelo é especialmente útil quando a amplitude da sazonalidade ou do ciclo cresce proporcionalmente à tendência, como em vendas de uma empresa em expansão, onde os picos sazonais (por exemplo, no Natal) aumentam conforme o crescimento anual (TSAY, 2005).

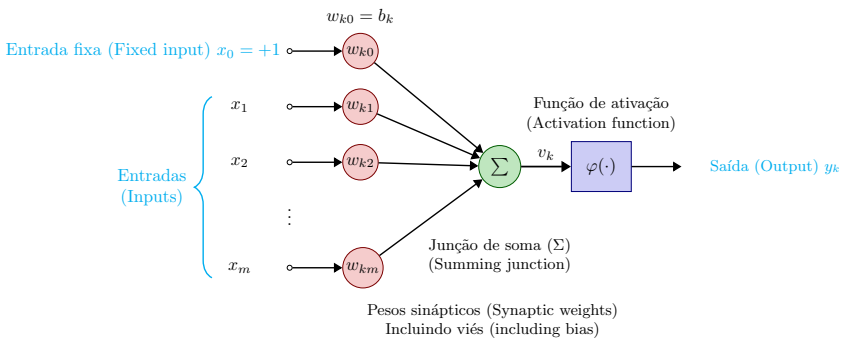
2.2 Redes Neurais Recorrentes (RNNs)

As Redes Neurais Recorrentes (RNNs) constituem modelos computacionais especificamente projetados para o processamento eficaz de dados sequenciais, tais como séries temporais. Essa capacidade decorre de seu mecanismo intrínseco que permite ao sistema assimilar padrões temporais de forma incremental, habilitando-o a realizar previsões ou inferências fundamentadas na natureza ordenada das entradas recebidas (IBM, 2024). Em contrapartida às redes neurais tradicionais, que processam cada entrada de maneira independente e descontextualizada, as RNNs incorporam conexões recorrentes que desempenham papel crucial na retenção de informação proveniente de estados precedentes, estabelecendo assim um mecanismo de memória interna. Essa memória endógena exerce influência direta sobre o processamento aplicado a cada nova entrada subsequente na sequência (IONOS, 2025).

Observa-se ainda uma analogia estrutural e funcional relevante entre as RNNs e o processamento de informação no cérebro humano, conforme documentado na literatura especializada (HAYKIN, 2009). Primariamente, a aquisição de conhecimento pela rede ocorre mediante suas interações contínuas com o ambiente, processo análogo à aprendizagem adaptativa observada em sistemas biológicos. Secundariamente, o conhecimento assimilado é armazenado e representado nas conexões entre as unidades de processamento, denominadas pesos sinápticos, que modulam dinamicamente a intensidade das interações internas da rede. Esse duplo paralelismo - tanto no mecanismo de aquisição contínua de conhecimento quanto em seu armazenamento nas forças de conexão - consolida a fundamentação neurobiológica subjacente à arquitetura recorrente (HAYKIN, 2009).

Conforme descrito por (HAYKIN, 2009), o modelo não linear de um neurônio é composto por elementos essenciais que simulam o comportamento de um neurônio biológico e está apresentado na Figura 2.

Figura 2 – Modelo não linear de um neurônio: w_{k0} representa o viés b_k



Fonte: Elaboração própria a partir de (HAYKIN, 2009).

As entradas $(x_1, x_2, \ldots, x_n)$ representam os sinais recebidos, que podem ser oriundos de dados externos ou de outros neurônios na rede. Uma entrada adicional fixa $x_0 = +1$ é incluída

para ajustar o viés b_k do neurônio, permitindo maior flexibilidade no aprendizado. Esse viés, representado pelo peso w_{k0} atua como um termo de ajuste, deslocando a função de ativação para melhorar a capacidade de modelagem.

O processo de soma ponderada ocorre na junção somadora, onde cada entrada é multiplicada por seu respectivo peso sináptico $(w_{k1}, w_{k2}, \ldots, w_{kn})$, conforme a Equação 2.10, e o resultado é combinado com o viés $w_{k0} = b_k$.

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{2.10}$$

Essa operação gera o potencial de ativação v_k, que é subsequentemente transformado por uma função de ativação não linear $\phi(\cdot)$, como a sigmoide ou ReLU. Essa função introduz não linearidade no modelo, o que permite ao neurônio representar padrões complexos que não seriam possíveis com uma combinação puramente linear. A saída y_k do neurônio é então propagada para outras unidades na rede ou serve como resultado final, dependendo da arquitetura (HAYKIN, 2009).

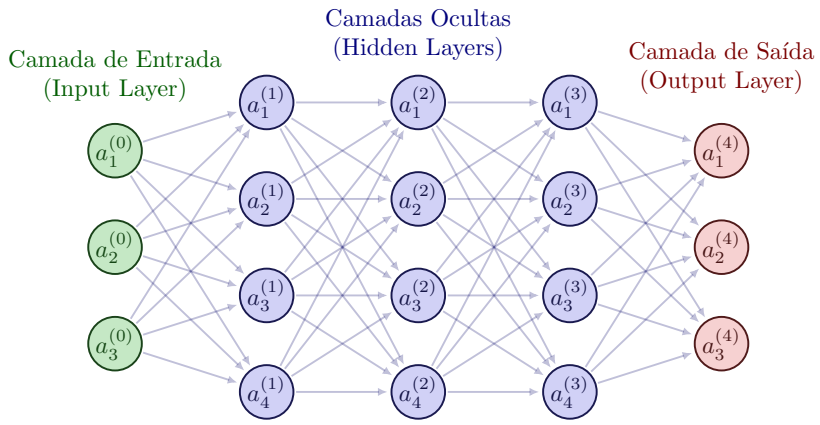
Partindo do modelo básico de um neurônio artificial não linear, a rede neural feedforward totalmente conectada surge como uma extensão natural, organizando múltiplas unidades desse tipo em camadas sequenciais (HAYKIN, 2009). Enquanto o neurônio isolado representa uma transformação não linear elementar, a interconexão dessas unidades em arquiteturas multicamadas permite a modelagem de relações hierárquicas e padrões complexos nos dados (GOODFELLOW YOSHUA BENGIO, 2016).

Nessa estrutura, a informação flui unidirecionalmente das camadas de entrada para as de saída, sem ciclos ou retroalimentações, caracterizando o processamento feedforward, conforme a Figura 3. Cada camada é composta por múltiplos neurônios que operam em paralelo, sendo que cada unidade em uma camada está conectada a todas as unidades da camada subsequente através de pesos sinápticos ajustáveis (RUMELHART; HINTON; WILLIAMS, 1986).

Embora as redes feedforward totalmente conectadas representem um avanço significativo em relação ao neurônio isolado, sua arquitetura apresenta uma limitação referente a incapacidade de processar informações sequenciais mantendo dependências temporais. Esta restrição motivou o desenvolvimento das Redes Neurais Recorrentes (RNNs), que estendem o conceito básico de redes neurais ao introduzir conexões recorrentes que atuam como memória interna, permitindo que a informação persista ao longo do tempo (GRAVES, 2012).

A arquitetura característica das Redes Neurais Recorrentes (RNNs) encontra-se ilustrada na Figura 4, evidenciando a presença de conexões recorrentes que habilitam o fluxo de informações através de intervalos temporais sucessivos. Esse mecanismo de retroalimentação temporal é implementado mediante operadores de atraso unitário, denotados por z^{-1} (HAYKIN, 2009), os quais funcionam como componentes mnemônicos essenciais. Tais operadores têm como função

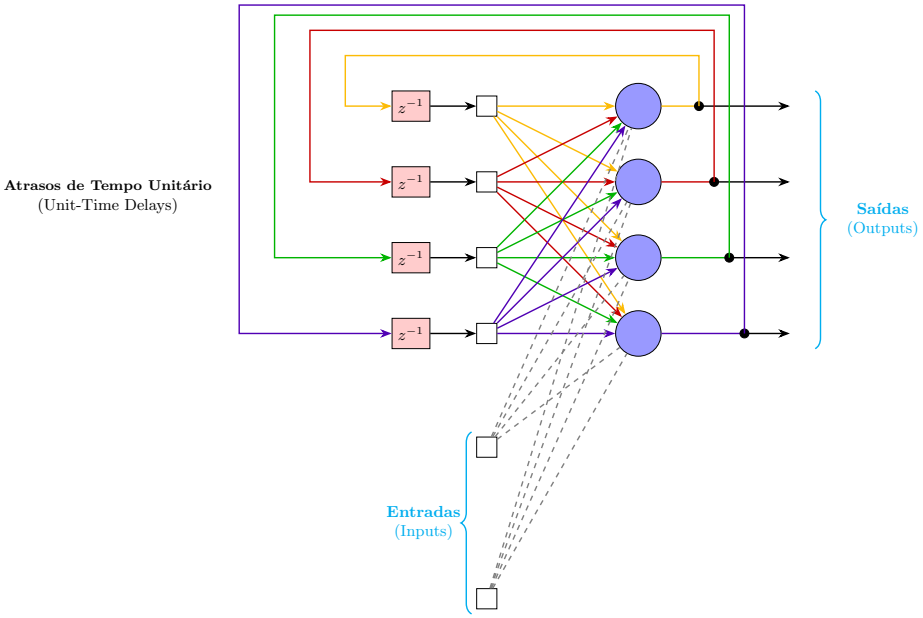
Figura 3 – Rede Neural feedforward totalmente conectada, com três camadas ocultas e uma camada de saída.



Fonte: Elaboração própria a partir de (HAYKIN, 2009).

primordial armazenar o estado oculto precedente e reintroduzi-lo na estrutura de processamento no instante temporal subsequente (GRAVES, 2012).

Figura 4 – Representação esquemática de Rede Neural Recorrente com neurônios ocultos.



Fonte: Elaboração própria a partir de (HAYKIN, 2009).

Em cada passo temporal discreto t, o estado oculto corrente, designado por h_t, é computado através de uma transformação não linear aplicada à combinação linear entre a entrada instantânea x_t e o estado oculto imediatamente anterior h_{t-1}, conforme formalizado pela Equação

2.11 (HAYKIN, 2009):

$$h_t = \phi\left(W_{ih}x_t + W_{hh}h_{t-1} + b_h\right) \tag{2.11}$$

Nesta expressão matemática, W_{ih} e W_{hh} correspondem, respectivamente, às matrizes de pesos sinápticos associadas às conexões de entrada e às ligações recorrentes. O termo b_h representa o vetor de polarização (bias) inerente à camada oculta, enquanto ϕ denota uma função de ativação não linear, frequentemente implementada pela função tangente hiperbólica (*tanh*), responsável por introduzir não-linearidade na transformação.

A matriz W_{ih} é responsável por transformar a entrada atual x_t em uma contribuição para o estado oculto h_t. Ela realiza o mapeamento do espaço de entrada, que possui dimensão d_x, para o espaço oculto, de dimensão d_h (HAYKIN, 2009).

Por exemplo, se a entrada x_t possui 3 características ($d_x = 3$) e a camada oculta contém 5 neurônios ($d_h = 5$), então W_{ih} é uma matriz 5×3:

$$W_{ih} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \end{bmatrix}.$$

Cada elemento w_{ij} indica a influência da j-ésima característica da entrada sobre o i-ésimo neurônio da camada oculta.

A matriz W_{hh} determina como o estado oculto anterior h_{t-1} influencia o estado atual h_t, configurando a conexão recorrente que permite à rede manter memória temporal (HAYKIN, 2009). Para $d_h = 5$, a matriz W_{hh} é dada por:

$$W_{hh} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \end{bmatrix}.$$

Cada peso w_{ij} representa a influência do j-ésimo neurônio do estado anterior sobre o i-ésimo neurônio do estado atual.

O vetor de bias b_h é adicionado à combinação linear antes da aplicação da função de ativação, permitindo ajustar a saída mesmo quando as entradas x_t e h_{t-1} são zero (HAYKIN, 2009). Para $d_h = 5$, temos:

$$b_h = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}.$$

Cada componente b_i é aprendido durante o treinamento para ajustar a ativação do neurônio i.

A função ϕ é aplicada elemento a elemento à soma ponderada $W_{ih}x_t + W_{hh}h_{t-1} + b_h$ para introduzir não linearidade ao modelo, essencial para que a rede possa aprender relações complexas nos dados. Sem essa não linearidade, a RNN seria equivalente a uma transformação linear simples (HAYKIN, 2009).

As funções de ativação mais comuns em RNNs são:

- **Sigmoide** (σ), que gera valores entre 0 e 1, definida pela Equação 2.12:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2.12}$$

  usada para controlar o fluxo de informações nas portas.

- **Tangente Hiperbólica** (tanh), que normaliza valores entre -1 e 1, definida pela Equação 2.13:

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.13}$$

  ajudando a evitar a explosão de valores durante o processamento, devido a sua simetria em torno de zero, o que ajuda na estabilização dos gradientes durante o treinamento.

- **ReLU (Rectified Linear Unit):** definida pela Equação 2.14:

$$\mathrm{ReLU}(z) = \max(0, z) \tag{2.14}$$

  é menos comum em RNNs clássicas devido ao problema do desvanecimento dos gradientes, mas é amplamente utilizada em outras arquiteturas de redes neurais.

As RNNs clássicas enfrentam o problema de desvanecimento de gradientes, onde gradientes tornam-se extremamente pequenos durante o Backpropagation Through Time (BPTT), dificultando o aprendizado de dependências de longo prazo (HAYKIN, 2009). Para resolver isso, variantes como LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Unit) foram desenvolvidas, onde essas arquiteturas introduzem portas que regulam o fluxo de informações, permitindo que a rede decida quais dados reter ou descartar em cada passo (GHAREHBAGHI, 2023).

2.3 Long Short-Term Memory (LSTM)

A arquitetura Long Short-Term Memory (LSTM) é uma evolução das Redes Neurais Recorrentes (RNNs) projetada para aprender dependências de longo prazo em dados sequenciais, superando limitações como o desvanecimento de gradientes por meio de uma estrutura interna com mecanismos de portas que regulam o fluxo de informações (HOCHREITER; SCHMIDHUBER, 1997). Essas portas permitem que a célula LSTM decida quando reter, atualizar ou descartar informações, adaptando-se a contextos complexos e sequências longas (GOODFELLOW YOSHUA BENGIO, 2016).

A entrada de uma célula LSTM é composta por dois elementos fundamentais: a entrada atual x_t, que representa o dado no instante t — como uma palavra em uma frase ou um valor em uma série temporal —, e o estado oculto anterior h_{t-1}, que corresponde à saída gerada pela célula no passo imediatamente anterior (GOODFELLOW YOSHUA BENGIO, 2016). Esses dois componentes são combinados para atualizar a memória interna da rede e produzir uma saída contextualizada.

O processamento dentro da célula ocorre em três etapas principais. Inicialmente, a porta de esquecimento avalia quais informações do estado da célula anterior C_{t-1} são irrelevantes para o contexto atual e, portanto, devem ser descartadas. Essa avaliação é feita por meio de uma função sigmoide, que gera valores entre 0 (descartar completamente) e 1 (reter integralmente) (BROWNLEE, 2018). Em seguida, a porta de entrada determina quais novas informações, derivadas da entrada atual x_t e do estado oculto anterior h_{t-1}, serão incorporadas ao estado atualizado da célula C_t. Essa modulação é realizada por uma combinação das funções sigmoide e tangente hiperbólica, que controlam a relevância das informações adicionadas (GOODFELLOW YOSHUA BENGIO, 2016).

Por fim, a porta de saída define quais partes do estado da célula C_t serão utilizadas para gerar a saída h_t. Essa operação aplica uma transformação não linear (função *tanh*) e filtra o resultado com base no contexto aprendido, produzindo uma saída que alimenta tanto a próxima etapa temporal quanto pode ser usada para previsões imediatas, como classificação de sentimentos em texto ou estimativa de valores futuros em séries temporais (BROWNLEE, 2018).

O desvanecimento de gradientes é um desafio em RNNs tradicionais, no qual os gradientes utilizados para ajustar os pesos durante o treinamento tornam-se extremamente reduzidos à medida que são propagados retroativamente em sequências longas. Esse fenômeno ocorre porque o cálculo do gradiente envolve o produto sucessivo de derivadas parciais em cada passo temporal. Quando funções de ativação como a sigmoide — que geram valores entre 0 e 1 — são aplicadas repetidamente, as multiplicações contínuas por valores menores que 1 resultam em uma redução exponencial da magnitude do gradiente, inviabilizando o aprendizado de relações temporais de longo prazo (HOCHREITER; SCHMIDHUBER, 1997). Para resolver esse problema, a arquitetura LSTM supera essa limitação por meio dos mecanismos:

2.3.1 Porta de Esquecimento (Forget Gate)

A porta de esquecimento tem a função de decidir quais informações do estado anterior da célula C_{t-1} devem ser descartadas. Essa decisão é feita por meio da função sigmoide, que gera valores entre 0 e 1, atuando como "interruptores" que regulam o fluxo de informação (GOODFELLOW YOSHUA BENGIO, 2016). A Equação 2.15 representa essa porta:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right), \tag{2.15}$$

onde W_f é a matriz de pesos da porta de esquecimento, h_{t-1} é o estado oculto do passo anterior, x_t é a entrada atual, b_f é o vetor de bias da porta de esquecimento e σ é a função sigmoide.

Se $f_t \approx 1$, o estado anterior C_{t-1} é mantido, facilitando a propagação do gradiente. Já a porta de entrada controla a incorporação de novas informações ao estado da célula, modulando a relevância dos dados. Por exemplo, se $f_t = 0,9$, isso indica que 90% da informação do estado anterior será mantida (GOODFELLOW YOSHUA BENGIO, 2016).

2.3.2 Porta de Entrada (Input Gate) e Candidato à Memória

A porta de entrada regula quais partes das novas informações candidatas serão adicionadas ao estado da célula. Ela é composta pelas Equações 2.16 e 2.17:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right), \tag{2.16}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right), \tag{2.17}$$

onde i_t é o vetor que controla a importância das novas informações, $\tilde{C}_t$ representa as informações candidatas à memória, W_i e W_C são as matrizes de pesos correspondentes, b_i e b_C são os vetores de bias e tanh é a função tangente hiperbólica, que normaliza valores entre -1 e 1 (GOODFELLOW YOSHUA BENGIO, 2016).

2.3.3 Atualização do Estado da Célula (Cell State)

O estado da célula é atualizado combinando as informações filtradas pela porta de esquecimento e as novas informações moduladas pela porta de entrada, conforme a Equação 2.18:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{2.18}$$

onde $\odot$ representa a multiplicação elemento a elemento (produto de Hadamard). Essa operação permite descartar informações irrelevantes e adicionar dados novos, preservando a memória de longo prazo (HOCHREITER; SCHMIDHUBER, 1997).

O gradiente em relação ao estado da célula é calculado pela Equação 2.19:

$$\frac{\partial C_t}{\partial C_{t-1}} = f_t + \text{termos adicionais} \tag{2.19}$$

Isso evita a multiplicação repetida de matrizes de pesos, característica das RNNs tradicionais. Essa estrutura aditiva permite que o gradiente se propague sem sofrer decaimento exponencial, mesmo em sequências muito longas (HOCHREITER; SCHMIDHUBER, 1997).

2.3.4 Porta de Saída (Output Gate) e Estado Oculto

A porta de saída determina quais partes do estado da célula serão expostas como saída, que também serve como entrada para o próximo passo temporal (HOCHREITER; SCHMIDHUBER, 1997), conforme as Equações 2.20 e 2.21:

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{2.20}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.21}$$

onde o_t regula a saída da célula, h_t é o estado oculto atualizado, W_o é a matriz de pesos da porta de saída e b_o é o vetor de bias da porta de saída.

Dessa forma, as LSTMs mantêm o fluxo estável dos gradientes durante o treinamento, possibilitando o aprendizado eficaz de dependências temporais de longo prazo em dados sequenciais.

2.4 Raiz Quadrada do Erro-Médio (RMSE)

A Raiz Quadrada do Erro Médio (RMSE, Root Mean Square Error) é uma medida utilizada para mensurar a acurácia de modelos preditivos em estatística e aprendizado de máquina. Seu objetivo principal é expressar a magnitude média dos desvios entre as previsões geradas por um modelo ($\hat{y}_i$) e os valores reais observados (y_i). Ao elevar os resíduos ao quadrado antes de calcular a média, o RMSE atribui maior peso a erros de maior magnitude, o que o torna particularmente útil para identificar inconsistências relevantes em previsões (KUTNER; NACHTSHEIM; NETER, 2005; JAMES DANIELA WITTEN, 2021).

O RMSE é originado do Erro Médio Quadrático (MSE, Mean Squared Error), que representa a média aritmética dos quadrados dos resíduos, conforme a Equação 2.22:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.22}$$

onde y_i é o valor real, $\hat{y}_i$ é a previsão do modelo, e n é o número de observações.

Ao calcular a raiz quadrada do MSE, obtém-se o RSME, dado pela Equação 2.23:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.23}$$

A operação de quadratização no MSE garante que erros positivos e negativos não se anulem, enquanto a raiz quadrada final retorna a métrica à mesma unidade da variável original, facilitando sua interpretação prática (JAMES DANIELA WITTEN, 2021). Por exemplo, em um modelo de previsão de demanda com RMSE igual a 50 unidades, conclui-se que as previsões desviam, em média, 50 unidades dos valores reais.

A escolha do RMSE como métrica de avaliação fundamenta-se em três propriedades essenciais. Primeiramente, sua sensibilidade a outliers — decorrente da natureza quadrática da fórmula — atribui penalidades mais severas a desvios extremos, característica crítica para capturar adequadamente a volatilidade inerente aos mercados financeiros, onde erros significativos implicam consequências econômicas substanciais. Em segundo lugar, o RMSE oferece comparabilidade direta ao manter a mesma unidade de medida da variável-alvo original, o que permite aos analistas quantificar imediatamente o impacto econômico dos erros de previsão em termos tangíveis. Finalmente, alinha-se com protocolos consolidados na literatura financeira, sendo amplamente adotado como métrica padrão para avaliação de modelos preditivos de ativos (AHMED et al., 2010).

Para expressar os erro da previsão em termos relativos, o Erro Percentual Absoluto Médio (MAPE - Mean Absolute Percentage Error) é adotado por sua simplicidade interpretativa e aplicabilidade prática nos contextos preditivos (ARMSTRONG, 2001). O MAPE é definido como a média aritmética dos erros percentuais absolutos, sendo calculado conforme a Equação 2.24:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{2.24}$$

onde y_i representa o valor real, $\hat{y}_i$ o valor previsto, e n o número total de observações. Ao expressar o erro como uma fração da própria observação real, o MAPE torna-se uma medida relativa que permite comparações diretas entre séries de diferentes escalas, sem depender das unidades da variável-alvo (MAKRIDAKIS; WHEELWRIGHT; HYNDMAN, 1997).

3 Metodologia

A metodologia é fundamental para estabelecer os procedimentos e técnicas, assegurando a validade, confiabilidade e a possibilidade de replicação dos resultados (SHUMWAY; STOFFER, 2025). No âmbito da análise e previsão de séries temporais financeiras, sua importância é amplificada pela complexidade inerente aos dados econômicos, que envolvem componentes como tendência não estacionária, sazonalidade variável, ciclos econômicos e ruído aleatório (TSAY, 2005).

3.1 Tratamento dos Dados

Dentro da metodologia empregada, a base de dados assume um papel central, sendo a fonte primária das informações utilizadas para o treinamento, validação e teste do modelo preditivo (PCS - Programa de Computação Científica, 2023). Neste estudo, a base de dados foi construída a partir de dados históricos do índice S&P 500 (`^GSPC`) (Yahoo Finance, 2025a) e da taxa de juros americana implícita a Letra do Tesouro Americano de curto prazo (13 semanas - 13-week Treasury Bill, (`^IRX`)) (Yahoo Finance, 2025b), coletadas via API do Yahoo Finance utilizando a biblioteca yfinance em Python. A integração e manipulação dos dados foram realizadas com o auxílio da biblioteca pandas, garantindo alinhamento temporal e tratamento de valores ausentes. O código completo utilizado em todo o trabalho encontra-se disponível em (BRANDÃO, 2025)

Para o S&P 500, foram coletadas duas variáveis diárias: o preço de fechamento ajustado (SP500_Close), registrado em pontos, e o volume financeiro negociado (SP500_Volume), expresso em milhões de dólares. Esses dados abrangem o período de 01 de janeiro de 2000 a 01 de maio de 2025 e representam o desempenho agregado das 500 maiores empresas listadas nas bolsas de valores dos Estados Unidos. Para a taxa de juros americana (Interest_Rate), foi coletada uma variável diariamente, expressa em porcentagem.

É importante ressaltar que a qualidade da base de dados é crucial para o desempenho do modelo preditivo. Por isso, foram realizados procedimentos de tratamento e pré-processamento dos dados, como a remoção das datas que não continham informações, garantindo que apenas registros completos fossem utilizados no treinamento. As datas sem valores do índice S&P 500 podem ocorrer por diversos motivos, como feriados em que a bolsa de valores está fechada (BENZINGA, 2024), interrupções temporárias nas negociações devido a eventos extraordinários ou ainda indisponibilidade de dados históricos (RANAROUSSI, 2020).

A remoção dos valores em branco foi realizado em python, utilizando o método $dropna()$, que elimina todas as linhas que contenham valores ausentes nas colunas selecionadas, garantindo

Quadro 1 – Trecho de código em Python para coleta de dados do S&P 500 e taxa de juros

```
import pandas as pd
import yfinance as yf

# Coleta de dados
start_date = '2000-01-01'
end_date = '2025-05-01'

sp500 = yf.download('^GSPC', start=start_date, end=end_date,
    interval='1d')[['Close', 'Volume']]
sp500.dropna(inplace=True)
sp500.columns = ['SP500_Close', 'SP500_Volume']

interest_rate = yf.download('^IRX', start=start_date, end=
    end_date, interval='1d')[['Close']]
interest_rate.columns = ['Interest_Rate']
interest_rate.dropna(inplace=True)
```

Fonte: Elaborado pelo autor.

que a base final não tenha dados incompletos que possam comprometer o treinamento do modelo. Os dados foram combinados em um único DataFrame usando a função *join* do pandas, com junção interna ($how =' inner'$) para garantir correspondência exata de datas.

Quadro 2 – Trecho de código em Python para junção dos dados do S&P 500 com a taxa de juros

```
# Juncao dos dados
df = sp500.join(interest_rate, how='inner')
```

Fonte: Elaborado pelo autor.

Além do tratamento de valores ausentes, a normalização foi realizada utilizando o *MinMaxScaler* do *sklearn*, que escala as variáveis para o intervalo $[0,1]$, essencial para modelos de rede neural como LSTM para estabilizar e acelerar o treinamento. Ao aplicar a normalização, cada variável é transformada de forma que seus valores fiquem dentro de uma mesma escala, permitindo que o modelo LSTM capture padrões temporais de maneira mais eficaz e evite que variáveis com escalas maiores tenham influência desproporcional no processo de aprendizado (LEE, 2025).

A transformação aplicada segue a Equação 3.1:

$$x' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{3.1}$$

onde $x_{\min}$ e $x_{\max}$ são os valores mínimo e máximo da variável no conjunto de dados (PEDRE-

Quadro 3 – Trecho de código em Python para normalização dos dados com MinMaxScaler

```
from sklearn.preprocessing import MinMaxScaler

# Normalizacao
scaler = MinMaxScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), index=df.
    index, columns=df.columns)
```

Fonte: Elaborado pelo autor.

GOSA et al., 2011).

3.2 Decomposição da Série Temporal do Índice S&P 500

A decomposição da série temporal do S&P 500 foi realizada para identificar e isolar componentes essenciais que estruturam seu comportamento histórico, sendo implementada através da função *seasonal_decompose* do pacote *statsmodels*. Apresenta-se o código completo no Apêndice A.

Adotou-se o modelo multiplicativo adequado para dados financeiros onde a amplitude das variações sazonais apresenta dependência direta do nível da série, proporcional ao nível de preços (BROCKWELL; DAVIS, 2016). A convenção dos mercados globais estabelece o período sazonal em 252 dias úteis — o ano financeiro padrão. Isso viabiliza a análise da recorrência de padrões conforme os calendários disponibilizados pelas principais bolsas de valores, como a NYSE e a NASDAQ (Nasdaq Trader, 2025).

Quadro 4 – Decomposição sazonal da série temporal do S&P 500

```
from statsmodels.tsa.seasonal import seasonal_decompose

# Decomposicao sazonal
decomposicao = seasonal_decompose(df['SP500_Close'], model='
    multiplicative', period=252)
```

Fonte: Elaborado pelo autor.

Os resultados da decomposição foram apresentados por meio de gráficos gerados com a biblioteca matplotlib, com o objetivo de facilitar a interpretação visual dos componentes. A Figura 5 apresenta quatro gráficos: a série observada, a tendência estimada, o componente sazonal e o resíduo aleatório. Essa visualização permite a análise separada de cada componente, o que é fundamental para investigações mais detalhadas de comportamento cíclico ou eventos atípicos.

3.3 Modelagem Long Short-Term Memory (LSTM)

A modelagem do índice S&P 500 foi realizada através da arquitetura de Redes Neurais Recorrentes do tipo Long Short-Term Memory (LSTM), que capturam dependências temporais de longo prazo em séries financeiras. Inicialmente, os dados foram estruturados em formato supervisionado utilizando a técnica de janelas deslizantes (windowed dataset), onde cada observação (valor de fechamento do S&P 500) é prevista com base em uma sequência histórica de 60 períodos anteriores. Essa abordagem transforma a série temporal univariada em um problema de aprendizado de máquina multivariado, preservando a ordem cronológica dos dados (BROWNLEE, 2018). Os dados foram então organizados em três subconjuntos temporais: aprendizado (01/01/2000 até 31/12/2022), validação (01/01/2023 até 31/12/2023) e teste (01/01/2024 até 01/05/2025).

A modelagem preditiva utilizando redes neurais LSTM foi implementada por meio da biblioteca TensorFlow e Keras, conforme apresentado no Apêndice B, onde o TensorFlow fornece o framework fundamental para construção de modelos de deep learning, enquanto as importações específicas do Keras permitem acessar componentes especializados. A classe Sequential é utilizada para criar modelos de rede neural com camadas empilhadas sequencialmente, formando um fluxo de dados unidirecional. As camadas LSTM, Dense e Dropout fornecem as funcionalidades essenciais para processamento temporal, saída de regressão e regularização, respectivamente. A camada Input permite especificar explicitamente a forma dos dados de entrada, uma prática recomendada para modelos complexos (CHOLLET, 2021).

O modelo LSTM é construído como uma pilha sequencial de camadas, onde cada camada recebe a saída da anterior. Essa abordagem é adequada para redes neurais *feedforward* e recorrentes com fluxo linear de dados (CHOLLET, 2021). A construção do modelo foi iniciada com a criação de uma instância da classe `Sequential`, e a primeira camada adicionada ao modelo é a de entrada (`Input`), que define o formato dos dados de entrada. A entrada possui dimensão tridimensional, conforme exigido pelas camadas `LSTM`, com o formato `[amostras, passos temporais, variáveis]`. Neste caso, `X.shape[1]` corresponde ao número de passos temporais (60 observações anteriores), e `X.shape[2]` ao número de variáveis por observação.

A primeira camada contém 64 unidades (células de memória) e está configurada com `return_sequences=True`, o que faz com que ela retorne uma sequência de saídas ao longo do tempo, permitindo o empilhamento de mais camadas LSTM.

Em seguida, é inserida uma camada de regularização do tipo Dropout, Essa técnica, proposta por (SRIVASTAVA et al., 2014), consiste em desativar aleatoriamente uma fração (20%, neste caso) das conexões da rede durante o treinamento. Isso reduz o sobreajuste (overfitting), forçando o modelo a generalizar melhor, evitando dependência excessiva de certos neurônios.

A segunda camada LSTM é então definida, diferentemente da primeira, essa camada não possui o parâmetro `return_sequences`, e portanto retorna apenas o último estado oculto da

Quadro 5 – Trecho de código em Python para definição e compilação de uma rede LSTM com Keras

```
model = Sequential([
    Input(shape=(X.shape[1], X.shape[2])),
    LSTM(64, return_sequences=True),
    Dropout(0.2),
    LSTM(64),
    Dropout(0.2),
    Dense(1)
])
model.compile(optimizer='adam', loss='mean_squared_error')
```

Fonte: Elaborado pelo autor.

sequência. Isso é apropriado para tarefas de regressão univariada, onde se deseja prever apenas o próximo valor da série com base no histórico anterior. Após, outra camada de Dropout é aplicada reforça a regularização da rede, contribuindo para a robustez do modelo durante o treinamento. Por fim, a camada Dense com um neurônio produz a previsão do valor do S&P 500 utilizando função de ativação linear implícita, padrão para problemas de regressão (GÉRON, 2019).

Concluída a definição da arquitetura, o modelo é compilado com a função compile(). O otimizador 'adam' implementa o algoritmo Adam (Adaptive Moment Estimation), que ajusta automaticamente as taxas de aprendizado durante o treinamento, oferecendo eficiência computacional e bom desempenho em problemas complexos (KINGMA; BA, 2014).

O método `fit()` é responsável por treinar a rede neural utilizando os dados de entrada `X_train` e os respectivos valores-alvo `y_train`. O número de épocas foi definido como 30, ou seja, o modelo percorre todo o conjunto de treinamento 30 vezes. O parâmetro `batch_size=32` indica que a atualização dos pesos do modelo será feita a cada 32 amostras processadas, o que promove estabilidade no gradiente e melhora a convergência do treinamento. Além disso, foi fornecido um conjunto de validação (`validation_data=(X_val, y_val)`) para monitoramento do desempenho do modelo ao longo das épocas, a fim de detectar possíveis sinais de overfitting ou underfitting. O argumento `verbose=1` ativa a exibição de logs informativos durante o treinamento, permitindo acompanhar o progresso e métricas do processo.

Após o treinamento, o método `predict()` é utilizado para inferir valores do conjunto de teste (`X_test`). A saída (`y_pred_test`) consiste em valores escalados na mesma transformação aplicada aos dados originais. Como o modelo opera em sequências temporais, cada previsão corresponde ao fechamento previsto do S&P 500 para um dia específico, baseado na janela de 60 dias anteriores.

A etapa de pós-processamento inicia-se com a identificação da posição da coluna-alvo no conjunto de dados original. Através do comando `sp500_close_idx = df.columns.`

Quadro 6 – Trecho de código em Python para treinamento do modelo LSTM

```
history = model.fit(X_train, y_train,
                    epochs=30,
                    batch_size=32,
                    validation_data=(X_val, y_val),
                    verbose=1)
```

Fonte: Elaborado pelo autor.

Quadro 7 – Trecho de código em Python para geração de previsões com o modelo treinado

```
y_pred_test = model.predict(X_test)
```

Fonte: Elaborado pelo autor.

`get_loc("SP500_Close")`, obtém-se o índice numérico correspondente à variável de fechamento do S&P 500 no `DataFrame`. Esta informação é crucial para o procedimento subsequente de reescalonamento, pois permite isolar seletivamente a variável-alvo durante a transformação inversa (BROWNLEE, 2018).

A função `inverse_scale()` executa a conversão dos valores normalizados de volta à escala original. Primeiramente, cria uma matriz temporária `temp` com dimensões idênticas ao dataset escalonado original (`número_de_amostras`, `número_de_variáveis`), preenchida com zeros. Em seguida, os valores escalados da variável-alvo (`y_scaled`) são inseridos exclusivamente na coluna correspondente ao S&P 500 (`sp500_close_idx`). Esta matriz reconstruída é submetida ao método `inverse_transform()` do objeto `scaler` (originalmente utilizado na normalização), que restaura todas as variáveis às suas magnitudes originais. Finalmente, extrai-se apenas a coluna do S&P 500 (`[:, sp500_close_idx]`), descartando as demais variáveis que serviram como *placeholders* durante o processo (GÉRON, 2019).

Na aplicação prática, `y_test_real = inverse_scale(y_test)` converte os valores reais do conjunto de teste para a escala interpretável (e.g., dólares), enquanto `y_pred_real = inverse_scale(y_pred_test.flatten())` realiza a mesma transformação nas previsões do modelo. O método `flatten()` é empregado para transformar a estrutura bidimensional das previsões (`[amostras, 1]`) em um vetor unidimensional, garantindo compatibilidade com a função de reescalonamento. Este passo final viabiliza a análise comparativa direta entre valores reais e previstos em unidades financeiras significativas (BROWNLEE, 2018).

Após a validação do modelo nos conjuntos históricos, procedeu-se à previsão futura para os 30 dias úteis a partir de 1º de maio de 2025, utilizando a sequência final da série histórica como base, ajustada com um componente de sazonalidade e ruído estocástico. Essa abordagem busca refletir não apenas a tendência aprendida pelo modelo LSTM, mas também variações

Quadro 8 – Trecho de código em Python para reescalonamento inverso da variável-alvo

```
def inverse_scale(y_scaled):
    temp = np.zeros((len(y_scaled), df_scaled.shape[1]))
    temp[:, sp500_close_idx] = y_scaled
    return scaler.inverse_transform(temp)[:, sp500_close_idx]

sp500_close_idx = df.columns.get_loc('SP500_Close')
y_test_real = inverse_scale(y_test)
y_pred_real = inverse_scale(y_pred_test.flatten())
```

Fonte: Elaborado pelo autor.

esperadas com base em padrões sazonais observados nos retornos diários históricos do índice, bem como flutuações aleatórias representadas pela volatilidade anualizada.

Inicialmente, utilizou-se a última sequência temporal de 60 dias normalizados — correspondente aos dias úteis imediatamente anteriores à data de início da projeção — como entrada do modelo. Essa janela deslizante de entrada é fundamental para manter a coerência temporal e estrutural dos dados em relação ao padrão aprendido pela rede recorrente (HOCHREITER; SCHMIDHUBER, 1997).

Para cada novo dia previsto, o modelo gera uma estimativa do S&P 500 em escala normalizada, que é então revertida à escala original por meio da inversão do processo de normalização. Com o objetivo de tornar a previsão mais realista e aderente ao comportamento típico do mercado, esse valor foi ajustado por dois componentes adicionais: (i) sazonalidade, capturada por meio da média dos retornos diários históricos agrupados por dia do ano (últimos 252 dias úteis), e (ii) ruído estocástico, representado por um valor aleatório extraído de uma distribuição normal com média zero e desvio padrão igual à volatilidade anualizada dos retornos diários mais recentes.

Este processo cíclico repete-se por 30 iterações consecutivas, correspondentes aos dias úteis projetados, armazenando progressivamente cada previsão em um vetor acumulativo (predictions). Concluída a fase de projeção, aplica-se a transformação inversa para reconverter os valores à escala original, adaptando a mesma lógica utilizada anteriormente nos conjuntos de teste, porém otimizada para a natureza estritamente univariada das previsões futuras. Finalmente, constrói-se um índice temporal rigoroso ($future\_dates$) mediante a função $pd.date\_range$ com frequência business day, garantindo alinhamento entre cada previsão e sua respectiva data calendária no horizonte projetado.

Utilizou-se a biblioteca Matplotlib para criar três gráficos com visualizações claras e informativas que facilitam a análise dos dados de séries temporais. Cada gráfico é configurado com tamanho adequado, títulos, legendas, eixos rotulados e estilos diferenciados para distinguir entre dados reais, previstos e históricos.

A Figura 8 apresenta a comparação entre os valores reais e previstos pelo modelo no conjunto de teste ao longo do tempo, enquanto a Figura 11 mostra o histórico do índice S&P 500 junto com a previsão para os próximos 30 dias. O histórico é representado por uma linha contínua, enquanto a previsão futura é destacada por uma linha tracejada laranja. Esse gráfico ajuda a visualizar a tendência projetada pelo modelo em relação aos dados passados, enquanto a Figura 12 foca no período a partir de 2024 até o final da previsão futura, oferecendo um detalhamento maior da projeção às cegas.

Para avaliar a capacidade preditiva do modelo em um cenário real, realizou-se uma previsão futura de 30 dias úteis do índice S&P 500, com início em 1º de maio de 2025. A previsão foi conduzida de forma autoregressiva, utilizando como entrada a última sequência de 60 dias úteis anteriores à data de início da projeção. A cada passo, o modelo previa o valor de fechamento do índice para o dia seguinte, e essa previsão era incorporada à sequência, substituindo a observação mais antiga, permitindo a geração sequencial das 30 estimativas.

Ao final do mês de maio, os dados reais do S&P 500 foram coletados via Yahoo Finance e utilizados para comparação com as previsões realizadas. A comparação foi feita por meio de análise gráfica e cálculo de métricas de erro, possibilitando identificar o grau de aderência do modelo à trajetória real do índice durante o período previsto.

3.4 Raiz Quadrada do Erro-Médio (RMSE)

A avaliação do desempenho preditivo do modelo LSTM foi realizada por meio de duas métricas amplamente utilizadas em problemas de regressão financeira: a Raiz Quadrada do Erro Médio (RMSE) e o Erro Percentual Absoluto Médio (MAPE). A métrica RMSE quantifica o erro em unidades absolutas do índice S&P 500, penalizando com mais intensidade os desvios maiores, enquanto o MAPE fornece uma medida relativa do erro, expressando a média das diferenças percentuais absolutas entre os valores previstos e os valores reais observados (HYNDMAN et al., 2008).

A implementação inicia com a função `mean_squared_error` da biblioteca `scikit-learn`, que calcula a média dos quadrados das diferenças entre os valores reais observados (`y_test_real`) e as previsões geradas (`y_pred_real`) (PEDREGOSA et al., 2011).

Após, foi realizada uma análise mais segmentada do erro preditivo ao longo do tempo. Para isso, calculou-se a Raiz do Erro Quadrático Médio (RMSE) de forma separada para cada mês do conjunto de teste, com o objetivo de identificar variações no desempenho do modelo em diferentes contextos temporais. Essa estratégia permite detectar períodos em que o modelo apresentou maior ou menor precisão, o que pode ser reflexo de diferentes regimes de mercado, eventos macroeconômicos ou volatilidade específica em determinados anos.

O procedimento iniciou-se com a criação de um *DataFrame* contendo três colunas: a

data das observações, os valores reais do índice S&P 500 (Real) e os valores previstos pela rede (Previsto). Em seguida, adicionou-se uma nova coluna `AnoMes`, extraída diretamente da variável `Data` utilizando o método `.dt.to_period('M')` do Pandas, que permite converter uma coluna de datas para o formato ano-mês.

Quadro 9 – Trecho de código em Python para criação do DataFrame de resultados e extração dos meses

```
df_resultados = pd.DataFrame({
'Data': test_dates,
'Real': y_test_real,
'Previsto': y_pred_real
})
df_resultados['AnoMes'] = df_resultados['Data'].dt.to_period('M')
```

Fonte: Elaborado pelo autor.

Após a preparação dos dados, o conjunto foi agrupado por mês com a função `groupby()`, e para cada grupo foi calculado o MSE e, em seguida, o RMSE.

Quadro 10 – Trecho de código em Python para cálculo do RMSE agrupado por mês

```
rmse_por_mes = {}
for mes, grupo in df_resultados.groupby('AnoMes'):
    mse_mes = mean_squared_error(grupo['Real'], grupo['Previsto'])
    rmse_mes = np.sqrt(mse_mes)
    rmse_por_mes[str(mes)] = rmse_mes
```

Fonte: Elaborado pelo autor.

Essa etapa utilizou novamente a função `mean_squared_error`, da biblioteca `scikit-learn`, e aplicou-se a transformação `np.sqrt()` para obter o valor da RMSE correspondente a cada mês. Os resultados foram armazenados em um dicionário (`rmse_por_mes`) para posterior visualização.

A representação gráfica dos valores mensais de RMSE foi realizada com auxílio da biblioteca `matplotlib`. A Figura 9 apresenta o gráfico de linha mostrando os meses no eixo x e os valores de RMSE no eixo y, com marcadores para facilitar a leitura dos pontos individuais. Cada valor foi também exibido como anotação no gráfico, o que contribui para a clareza e interpretação visual dos dados.

Para complementar a análise, calculou-se o erro percentual absoluto médio (MAPE) das previsões geradas pelo modelo LSTM. Essa métrica expressa, de forma relativa, o erro

médio entre os valores previstos e os valores reais observados, sendo amplamente utilizada em modelagens financeiras por sua interpretabilidade e sensibilidade proporcional às magnitudes reais.

Quadro 11 – Trecho de código em Python para cálculo do MAPE

```
# Calculo do MAPE mensal
mape_por_mes = df_mape.groupby('AnoMes')['ErroPercentual'].mean().round(2)

# Conversao dos periodos para datetime
meses_datetime = pd.to_datetime([str(m) for m in mape_por_mes.index])
```

Fonte: Elaborado pelo autor.

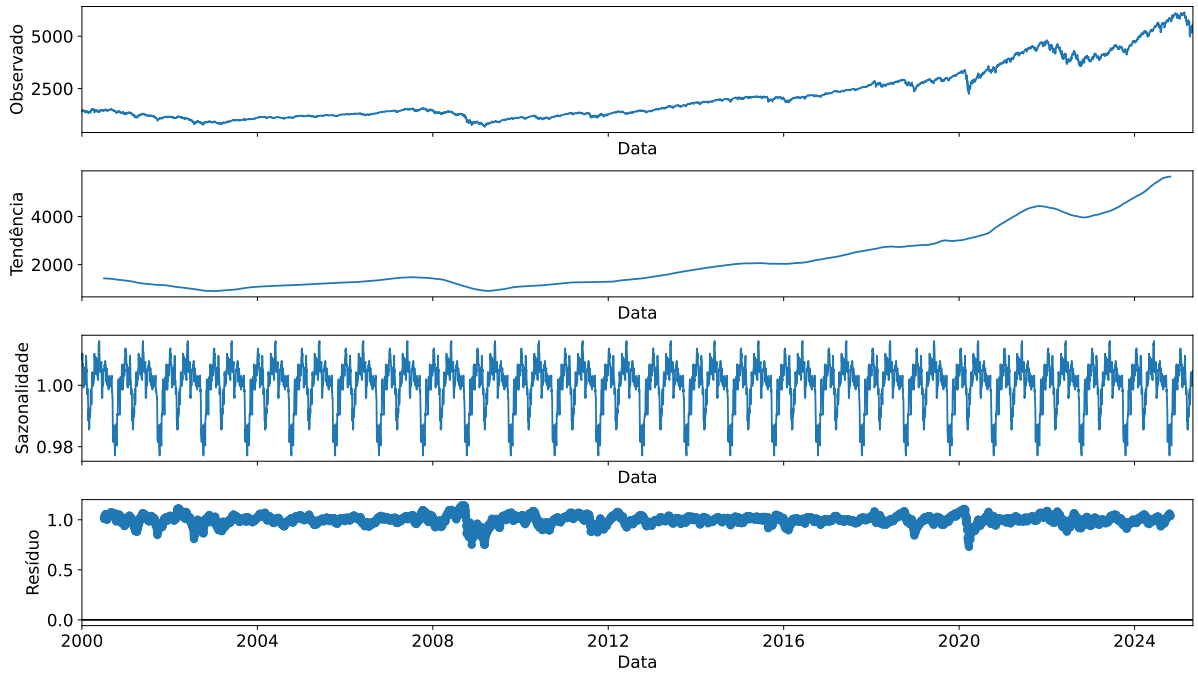
4 Resultados

4.1 Decomposição da Série Temporal

A partir da decomposição da série temporal do índice S&P 500 na separação dos componentes estruturais — tendência, sazonalidade e ruído — foi possível identificar padrões de comportamento relevantes ao longo do horizonte temporal observado. Essa análise permite compreender a dinâmica subjacente da série, interpretações econômicas, identificação de eventos de ruptura e aprimoramento de modelos preditivos.

A componente observada da série temporal representa os valores brutos do índice ao longo do tempo, refletindo sua evolução histórica sem qualquer transformação ou filtragem. Conforme ilustrado na Figura 5, observa-se uma trajetória ascendente de longo prazo, interrompida por quedas abruptas associadas a eventos econômicos significativos, que provocaram oscilações intensas na série, demonstrando a sensibilidade do mercado a choques exógenos. Além disso, é perceptível a presença de flutuações de curta duração e variações sazonais recorrentes, o que justifica a necessidade de decomposição da série para melhor compreensão de seus padrões estruturais.

Figura 5 – Decomposição da Série Temporal do Índice S&P 500.



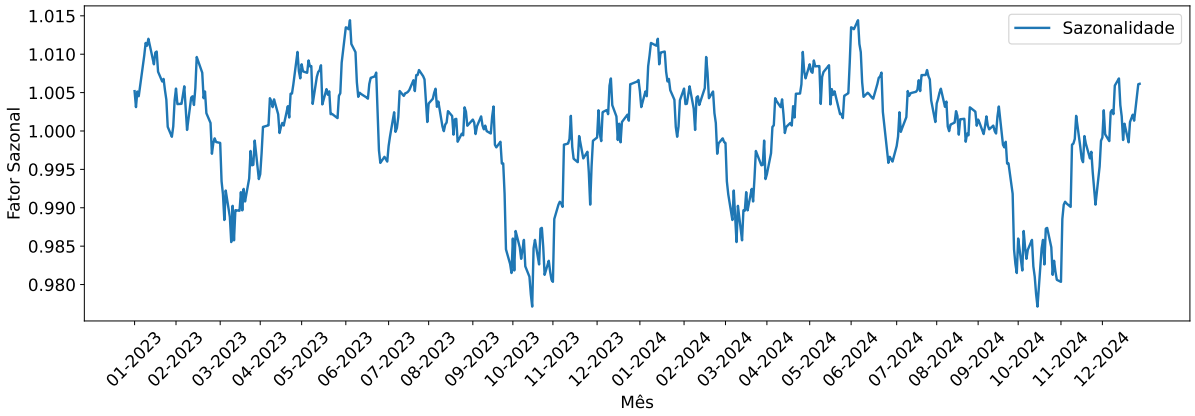
Fonte: Autoria própria (2025).

A componente de tendência da série temporal evidencia a direção de longo prazo do índice, suavizando as variações de curto prazo e os ruídos aleatórios. Observa-se uma trajetória predominantemente ascendente, interrompida por quedas pontuais em períodos de instabilidade, seguidas por retomadas consistentes. Esse comportamento é característico de séries temporais influenciadas por choques permanentes, o que indica uma tendência de natureza estocástica, conforme descrito por (TSAY, 2005). Diferente da tendência determinística, que segue uma função explícita do tempo, a tendência estocástica evolui a partir da acumulação de choques aleatórios, modelada frequentemente como um passeio aleatório com ruído branco (HAMILTON, 1994).

A análise da componente sazonal (S_t) revela padrões cíclicos que se repetem com regularidade ao longo dos anos, caracterizados por amplitude relativamente baixa e constante, oscilando em torno de um valor médio próximo a 1, comportamento típico do modelo multiplicativo Equação 2.8 aplicado a séries financeiras. A oscilação em torno de 1 indica que as variações sazonais afetam a série original de forma proporcional, e não aditiva, ou seja, os efeitos sazonais não somam um valor fixo à série, mas multiplicam seu valor por um fator levemente maior ou menor que 1 (BOX et al., 2015). Essas flutuações, modeladas adequadamente por funções harmônicas (Equação 2.6), refletem fenômenos recorrentes como fechamentos trimestrais, sazonalidade de dividendos ou efeitos de liquidez no final do ano.

A Figura 6 apresenta a componente sazonal da série temporal do índice S&P 500 ao longo do período de 2023 a 2024, para melhor visualização dos ciclos e pontos de máximo e mínimo.

Figura 6 – Sazonalidade da Série Temporal do Índice S&P 500 (2023-2024).



Fonte: Autoria própria (2025).

A análise revela padrões cíclicos recorrentes com valores máximos da sazonalidade concentrados no início do ano, janeiro e fevereiro, e um pico expressivo em junho, este último sendo o maior valor da série sazonal. Os picos sazonais observados em janeiro e fevereiro, com valores

de $S_t \approx 1.010$ a 1.015, são classicamente associados ao "January Effect", fenômeno que surge da convergência de dois mecanismos: investidores realizam perdas em dezembro para benefícios fiscais ("tax-loss harvesting"), seguido por uma realocação em ativos subvalorizados no início do ano, pressionando os preços para cima (REINGANUM, 1983). Simultaneamente, fundos de pensão e gestores institucionais rebalanceiam carteiras com novos aportes anuais, aumentando a demanda por ações de grande liquidez, como as do S&P 500 (HAUGEN; LAKONISHOK, 1988).

O pico de junho, sendo o mais acentuado do ciclo $S_t \geq 1.015$, reflete dinâmicas estruturais mais complexas. Empresas do índice concentram pagamentos semestrais de dividendos em maio-junho, atraindo investidores em busca de rendimento, particularmente fundos de renda (GU, 2015). Esse movimento fundamental superpõe-se a fluxos estratégicos, como rebalanceamentos para relatórios semestrais ("window dressing"), que artificialmente inflacionam preços de ativos-chave (LAKONISHOK; SHLEIFER; VISHNY, 1991). Adicionalmente, o fechamento do primeiro semestre fiscal estimula reposicionamentos táticos, amplificando volumes e liquidez (JACOBS; LEVY, 1988).

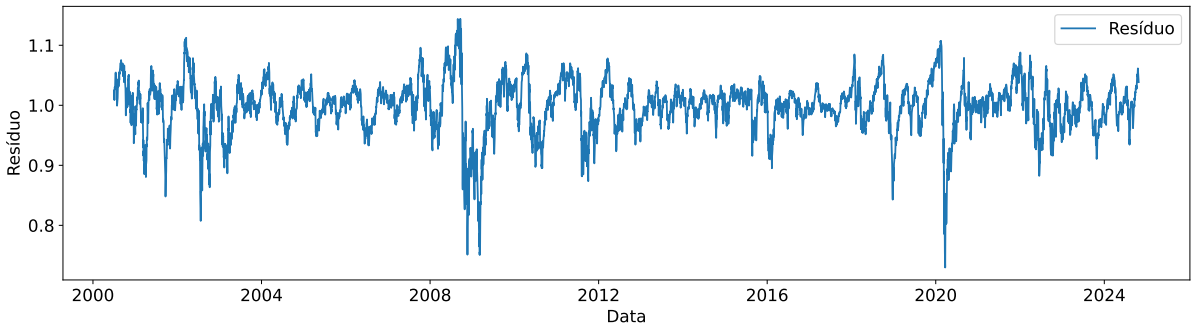
Os mínimos sazonais concentram-se em dois períodos críticos ao longo do ano: março ($S_t \approx 0.985 - 0.990$) e setembro-outubro ($S_t \approx 0.975 - 0.980$), ambos enraizados em dinâmicas de mercado históricas e estruturais. O vale de março, frequentemente subestimado, está intrinsicamente ligado ao fenômeno do "Triple Witching", que é a expiração trimestral simultânea de opções de ações, futuros de índices e opções de índices na terceira sexta-feira do mês. Esse evento gera volatilidade devido ao fechamento em massa de posições derivativas, pressionando os preços à baixa, especialmente em setores de alta liquidez (STOLL; WHALEY, 1991). Adicionalmente, março marca o fim do primeiro trimestre fiscal, quando instituições realizam ajustes táticos ("portfolio rebalancing") para otimizar alocações antes dos relatórios, muitas vezes liquidando ganhos acumulados nos meses anteriores. É também um período com menor volume de novidades corporativas ou divulgações macroeconômicas expressivas, o que pode contribuir para uma retração nos preços por ausência de catalisadores positivos (LAKONISHOK; SHLEIFER; VISHNY, 1991).

O mínimo sazonal de setembro-outubro, historicamente o mais profundo do ciclo ($S_t \leq 0.980$), mostra o notório "October Effect", cujas origens remontam a colapsos históricos como a crise de 1929 e a "Segunda-Feira Negra"de 1987 (KAMSTRA; KRAMER; LEVI, 2003; HIRSHLEIFER; JIANG; DIGIOVANNI, 2020). Esse padrão opera por meio de três mecanismos interligados que intensificam pressões vendedoras. Primeiramente, investidores realizam lucros estratégicos após ganhos acumulados no primeiro semestre, encerrando posições de forma coordenada e amplificando quedas (SEYHUN, 1990). Paralelamente, uma aversão comportamental emerge da memória coletiva de crises emblemáticas, como a do Lehman Brothers (2008), elevando a percepção de risco (HAGGARD; WITTE, 2010). Finalmente, fatores macroestruturais reforçam o ciclo, pois em outubro é o fim do ano fiscal de fundos, evento que acelera

a liquidação de ativos com resultados abaixo do esperado a fim de aprimorar métricas como indicadores de volatilidade, risco e retorno anualizado, usados pelos fundos para demonstrar eficiência nos relatórios anuais, criando ondas adicionais de venda (JACOBS; LEVY, 1988).

O Gráfico 7 apresenta os resíduos da decomposição da série temporal. O resíduo revela a parcela não explicada da série após a extração da tendência e sazonalidade. Conforme estabelecido por (BOX et al., 2015), um resíduo ideal deve comportar-se como ruído branco (média zero, variância constante e não autocorrelação), contudo, a série apresenta desvios significativos que evidenciam choques exógenos e limitações do modelo.

Figura 7 – Resíduo da Série Temporal do Índice S&P 500.



Fonte: Autoria própria (2025).

Embora o comportamento médio da série residual esteja de fato centrado em torno de 1, não se observa um padrão completamente aleatório ou estacionário, o que é esperado para séries financeiras. Os resíduos apresentam picos e quedas abruptas em diversos momentos do tempo, associados a eventos econômicos ou geopolíticos relevantes.

Um primeiro padrão observado está relacionado à ocorrência de choques sistêmicos e sua persistência ao longo do tempo. No início da série, entre 2000 e 2003, identifica-se uma queda, que atinge valores próximos de 0,8 durante o estouro da bolha das empresas pontocom. Esse comportamento indica que o modelo falha em antecipar grandes crises e a persistência desse desvio por mais de dois anos reforça a noção de que crises setoriais podem produzir efeitos duradouros e não estacionários sobre o comportamento do índice.

Entre 2008 e 2009, durante a crise do subprime, os resíduos caem novamente para valores em torno de 0,9. Nesse caso, há evidências de autocorrelação negativa por vários trimestres consecutivos, indicando que os impactos do colapso do mercado imobiliário foram absorvidos de forma gradual e não aleatória. Em 2020, a pandemia de COVID-19 produziu o desvio mais intenso da série residual, marcado por um dos maiores desvios negativos de toda a série, com resíduos abaixo de 0,75.

Após o choque da pandemia, a série residual retoma gradualmente valores próximos de 1, sugerindo uma fase de relativa estabilização. Contudo, observa-se novamente um aumento

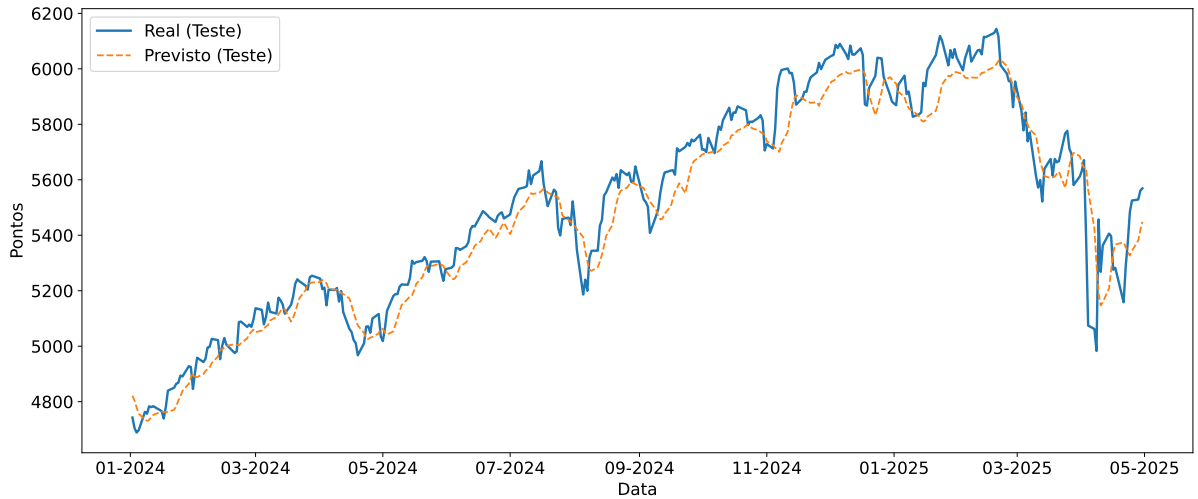
da volatilidade residual entre 2022 e 2023, período marcado por pressões inflacionárias globais, ciclos agressivos de alta de juros pelo Federal Reserve e tensões geopolíticas, como o conflito Rússia-Ucrânia. Esses episódios reforçam a natureza não estacionária e sensível ao contexto dos resíduos em séries financeiras, principalmente em horizontes de longo prazo.

Em contrapartida, períodos em que os resíduos permanecem próximos ao valor unitário, como observado entre 2012 e 2016, indicam fases de relativa estabilidade macroeconômica e financeira. Nessas fases o modelo de decomposição captura adequadamente a dinâmica do índice, onde a tendência e a sazonalidade explicam a maior parte da variância da série, restando apenas flutuações aleatórias compatíveis com ruído branco (BOX et al., 2015). A permanência dos resíduos em torno de 1 nessas fases reflete um mercado funcionando sob condições mais previsíveis, com menor volatilidade inesperada e sem eventos de disrupção significativa.

4.2 Modelagem Long Short-Term Memory (LSTM)

A modelagem preditiva utilizando Redes Neurais Recorrentes LSTM foi empregada com o intuito de prever o comportamento do índice S&P 500 a partir de sua série temporal. A Figura 8 apresenta os resultados da avaliação do modelo LSTM no conjunto de teste, cobrindo o período de 01 de janeiro de 2024 a 01 de maio de 2025.

Figura 8 – Avaliação do Modelo LSTM no Conjunto de Teste.



Fonte: Autoria própria (2025).

A curva azul representa os valores reais observados, enquanto a curva tracejada laranja representa os valores previstos pelo modelo. Observa-se uma forte aderência entre as curvas real e prevista, mostrando que o modelo foi capaz de capturar a tendência geral do índice ao longo do tempo, acompanhando os principais movimentos de alta e baixa com razoável precisão. No entanto, há um atraso na previsão do modelo quando comparado à curva real e também existem

momentos em que a previsão não alcança os picos de valorização e de queda. Essas diferenças nos picos podem ser atribuídas a eventos exógenos de difícil antecipação pela própria arquitetura do modelo, cuja capacidade de generalização é fortemente influenciada pelo comportamento histórico da série.

Durante o primeiro semestre de 2024, o modelo demonstra alta acurácia, reproduzindo os movimentos ascendentes com boa precisão. Essa fase de valorização do S&P 500 pode ser parcialmente explicada pela expectativa de afrouxamento monetário por parte do Federal Reserve, diante da desaceleração da inflação nos Estados Unidos e indicações de um ciclo econômico mais estável (Federal Reserve, 2024).

Um dos eventos mais impactantes para o mercado durante esse intervalo foi o anúncio das tarifas recíprocas pelo governo americano em abril de 2025. Essa política tarifária, conhecida como "Liberation Day", consistiu na imposição de tarifas universais de base de 10%, com alíquotas maiores para países que aplicam tarifas elevadas aos EUA, totalizando, por exemplo, 54% para produtos chineses, ao somar tarifas anteriores (XP Investimentos, 2025).

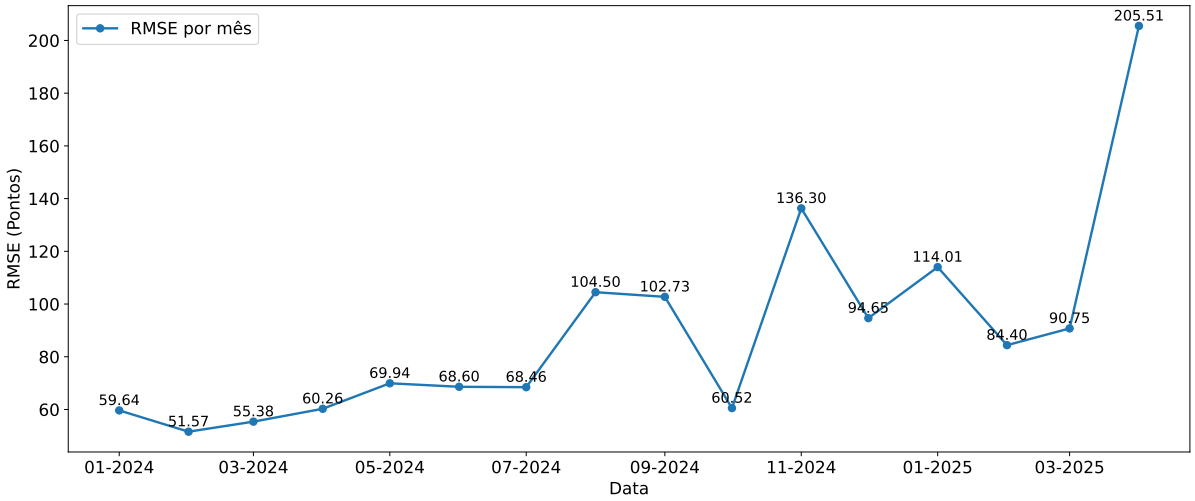
A divulgação dessas tarifas teve efeito imediato nos mercados, com queda abrupta nos futuros do S&P 500, refletindo a preocupação dos investidores com o aumento da inflação e o impacto negativo esperado sobre a atividade econômica americana (XP Investimentos, 2025). Essa instabilidade compromete o desempenho do modelo, uma vez que sua estrutura de aprendizado supervisionado depende de padrões temporais recorrentes. Como os choques tarifários introduzem rupturas nos padrões históricos, o modelo pode não ser capaz de antecipar adequadamente tais eventos.

A fim de avaliar a precisão preditiva do modelo LSTM ao longo do tempo, foi realizada uma análise mensal da Raiz Quadrada do Erro-Médio (RMSE), medida que quantifica a discrepância entre os valores previstos e os observados do índice S&P 500. O Gráfico 9 apresenta os valores de RMSE mensais para o período de janeiro de 2024 a abril de 2025, permitindo uma visão detalhada da variação do desempenho do modelo frente às oscilações do mercado.

Nos primeiros meses, de janeiro a julho de 2024, o RMSE manteve-se relativamente baixo e estável, variando entre 51,57 e 69,94 pontos. Esse comportamento indica que o modelo conseguiu realizar previsões consistentes, refletindo sua boa capacidade de capturar as dinâmicas do índice em um ambiente de mercado relativamente estável. No entanto, a partir de agosto de 2024, observa-se um aumento significativo no RMSE, que ultrapassou os 100 pontos nos meses de agosto e setembro. Esse crescimento sugere que o modelo enfrentou maiores dificuldades para prever o índice, provavelmente devido a eventos de maior volatilidade ou choques econômicos que alteraram os padrões da série temporal.

Após uma breve queda em outubro de 2024, quando o RMSE retornou a cerca de 60 pontos, o erro voltou a subir nos meses seguintes, atingindo picos em novembro de 2024 (136,30 pontos) e, principalmente, em abril de 2025, quando o RMSE alcançou 205,51 pontos. Embora a

Figura 9 – Raiz Quadrada do Erro-Médio (RMSE) por Mês do Modelo LSTM no Conjunto de Teste.



Fonte: Autoria própria (2025).

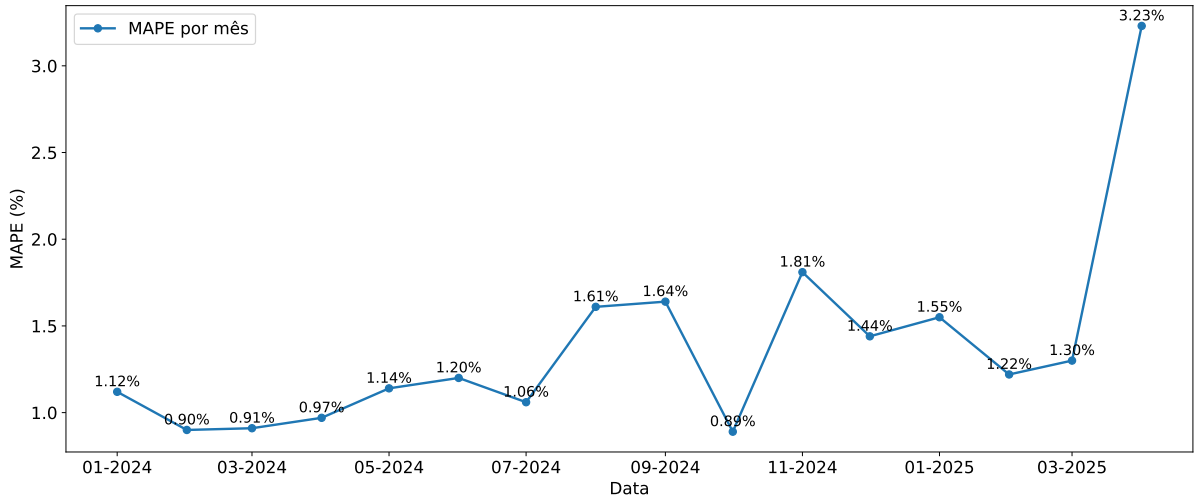
tendência geral do índice S&P 500 não tenha se alterado desde fevereiro de 2025, a Figura 8 mostra uma queda abrupta no índice em abril, causada pelo anúncio das tarifas recíprocas do governo americano. Essa movimentação acentuada, ainda que pontual, gerou forte volatilidade e dificultou o acompanhamento do modelo LSTM, contribuindo para o aumento expressivo do erro nesse período.

Esses resultados evidenciam a sensibilidade do modelo a choques externos e eventos abruptos que causam desvios significativos entre os valores previstos e observados. Embora o LSTM seja robusto para capturar padrões temporais, sua performance é naturalmente afetada quando o mercado sofre alterações rápidas e inesperadas. O RMSE, por sua característica de penalizar erros maiores de forma mais severa, é uma métrica adequada para avaliar essa precisão em séries temporais financeiras, onde grandes desvios podem ter impactos substanciais.

A análise do Erro Percentual Absoluto Médio (MAPE - Mean Absolute Percentage Error) os resultados obtidos com o RMSE ao oferecer uma medida relativa da acurácia do modelo LSTM. O Gráfico 10 apresenta os valores mensais de MAPE para o período de janeiro de 2024 a abril de 2025.

Durante o primeiro semestre de 2024, os valores de MAPE mantiveram-se abaixo de 1,20%, indicando alta precisão preditiva em um ambiente de mercado mais estável. A partir de agosto, no entanto, observa-se um aumento nos erros relativos, com destaque para os meses de novembro de 2024 (1,81%) e abril de 2025 (3,23%), sugerindo que o modelo teve maior dificuldade para capturar movimentos atípicos do índice S&P 500 em períodos de elevada volatilidade e incerteza.

Figura 10 – Erro Percentual Absoluto Médio (MAPE) por Mês do Modelo LSTM no Conjunto de Teste.

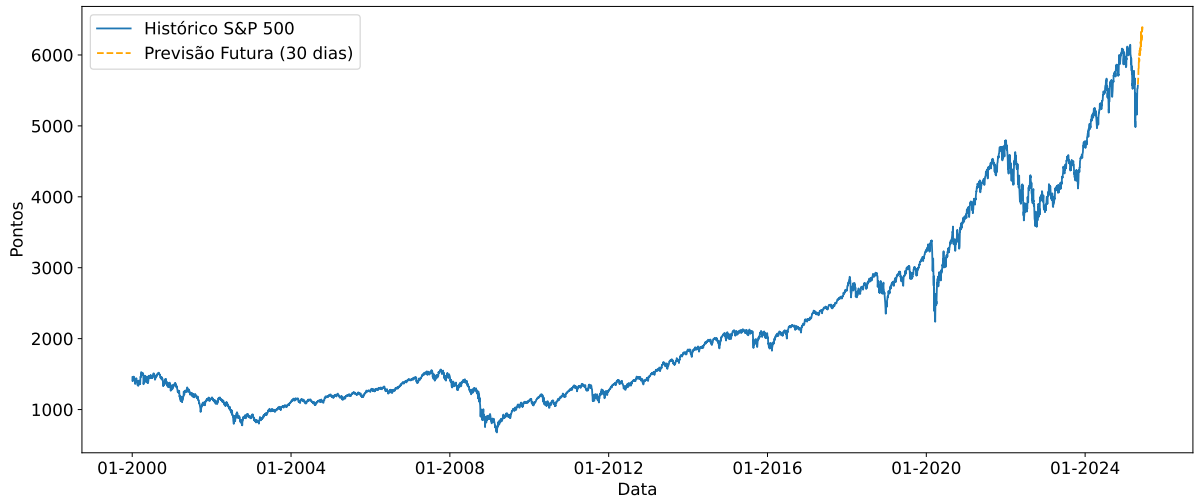


Fonte: Autoria própria (2025).

4.3 Previsão Futura

A Figura 11 apresenta os resultados da projeção futura para o índice S&P 500 ao longo de 30 dias úteis a partir de 1º de maio de 2025. Esta projeção foi gerada a partir de uma regressão ajustada com componentes de sazonalidade e ruído estocástico.

Figura 11 – Previsão Futura de 30 Dias do Índice S&P 500.

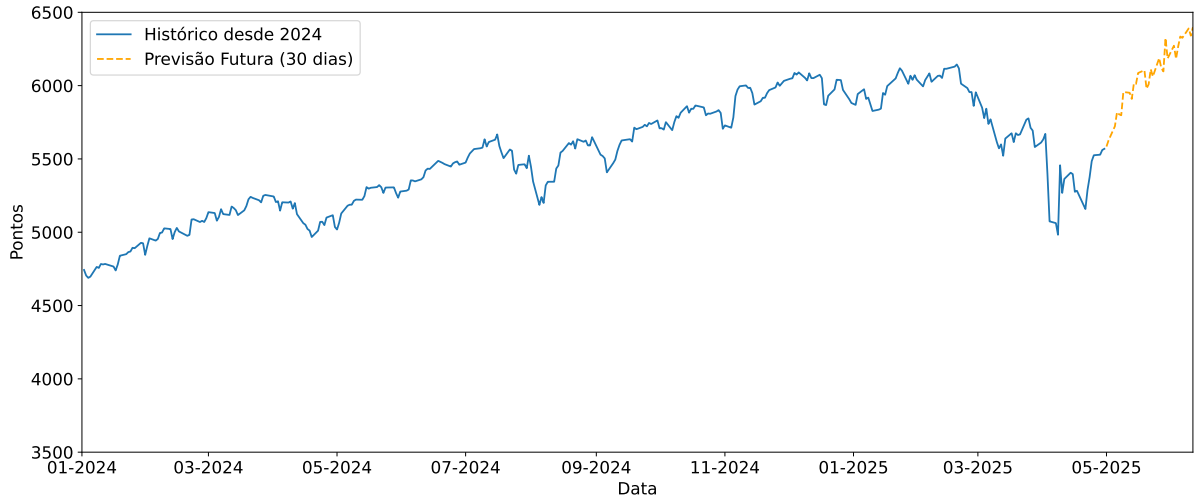


Fonte: Autoria própria (2025).

Reduzindo o espaço de tempo da Figura 11 a fim de ter um zoom projeção futura de 30 dias, apresenta-se a Figura 12. Percebe-se uma trajetória ascendente, sugerindo uma recuperação

contínua após um período de alta volatilidade nos primeiros meses do ano. A série prevista, linha tracejada laranja, se alinha de forma coerente com a tendência de retomada observada no final de abril, indicando que o modelo foi capaz de capturar o impulso de alta no curto prazo. Embora a previsão tenha mantido consistência com o comportamento recente da série, é importante destacar que projeções em horizontes futuros estão sujeitas a incertezas, especialmente diante de eventos inesperados.

Figura 12 – Zoom na Previsão Futura de 30 Dias do Índice S&P 500.



Fonte: Autoria própria (2025).

As oscilações diárias visíveis na trajetória prevista são consequência direta da metodologia adotada, onde em cada passo da previsão o valor estimado é modulado pela componente sazonal, derivada da média histórica dos retornos diários para cada dia específico do ano (considerando os últimos 252 dias úteis), e por um termo de ruído estocástico. Esta abordagem proporciona à previsão uma representação realista das flutuações observadas no mercado.

Complementando a análise da projeção futura, a Tabela 1 apresenta uma comparação detalhada entre os valores previstos pelo modelo LSTM e os valores reais observados do índice ao final do mês de maio de 2025. A inclusão da coluna de erro percentual permite avaliar, de forma quantitativa, o desempenho do modelo em um contexto de previsão futura.

Observa-se que, nos primeiros dias da projeção, o erro percentual manteve-se relativamente baixo, com desvios inferiores a 2% até aproximadamente a segunda semana. Esse comportamento sugere que o modelo conseguiu capturar adequadamente a inércia da tendência ascendente observada ao final de abril. Contudo, à medida que a projeção avança, os erros tornam-se progressivamente maiores, atingindo um pico de 6.97% no dia 29 de maio. Isso evidencia uma limitação natural dos modelos autoregressivos de longo prazo, cujo acúmulo de incertezas pode amplificar desvios à medida que a previsão se distancia da base.

Tabela 1 – Comparativo Entre os Valores Previstos e Reais do Índice S&P 500.

Data	Previsto (pts)	Real (pts)	Erro (%)
2025-05-01	5584.53	5604.14	0.35
2025-05-02	5627.28	5686.67	1.04
2025-05-05	5722.98	5650.38	1.28
2025-05-06	5819.05	5606.91	3.78
2025-05-07	5804.45	5631.28	3.08
2025-05-08	5797.75	5663.94	2.36
2025-05-09	5956.56	5659.91	5.24
2025-05-12	5951.11	5844.19	1.83
2025-05-13	5909.29	5886.55	0.39
2025-05-14	6015.28	5892.58	2.08
2025-05-15	6011.36	5916.93	1.60
2025-05-16	6085.94	5958.38	2.14
2025-05-19	6104.73	5963.60	2.37
2025-05-20	5978.77	5940.46	0.64
2025-05-21	6009.13	5844.61	2.81
2025-05-22	6111.73	5842.01	4.62
2025-05-23	6061.90	5802.82	4.46
2025-05-27	6119.81	5921.54	3.35
2025-05-28	6096.70	5888.55	3.53
2025-05-29	6324.29	5912.17	6.97
2025-05-30	6184.95	5911.69	4.62

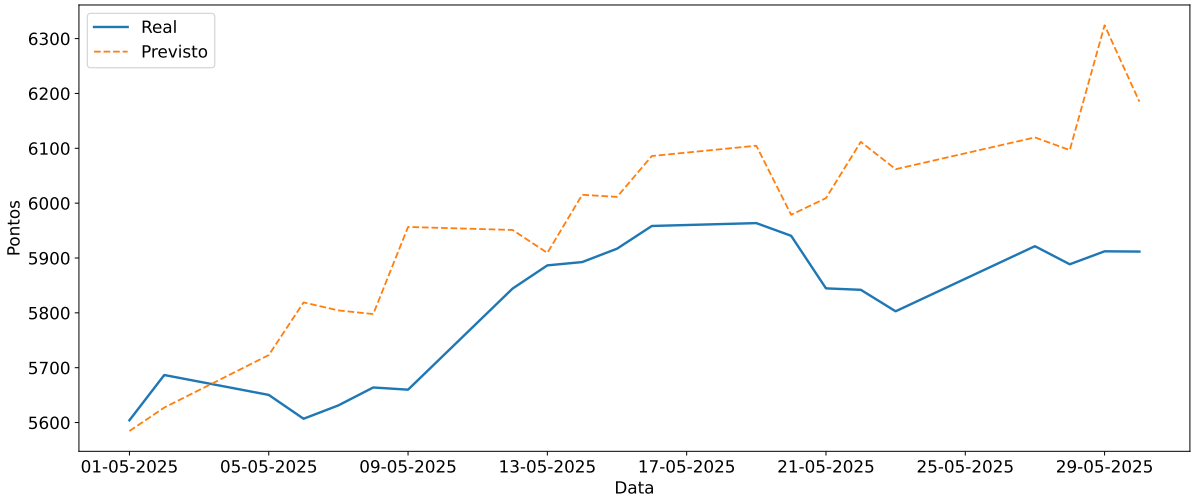
Fonte: Autoria própria (2025).

A Figura 13 ilustra a comparação direta entre os valores reais observados do índice S&P 500 e as respectivas previsões geradas ao longo do mês de maio de 2025. A linha azul representa os dados efetivos de fechamento do índice, enquanto a linha tracejada laranja indica os valores estimados pelo modelo.

Visualmente, nota-se que o modelo foi capaz de capturar a tendência geral de alta nas duas primeiras semanas do mês, embora tenha apresentado uma antecipação do movimento ascendente, com elevação mais acentuada que a observada na realidade. A partir da segunda metade de maio, entretanto, ocorre um distanciamento mais evidente entre as curvas, com o modelo superestimando de forma sistemática os valores do índice.

Esse padrão pode ser atribuído a flutuações de mercado não previstas pelo modelo, mesmo com os ajustes de sazonalidade e ruído incorporados na projeção. A presença desses desvios sugere que, embora o modelo seja eficaz para capturar movimentos gerais da série, ele ainda apresenta limitações frente a oscilações abruptas e eventos de mercado. O MAPE médio obtido para o período foi de 2,79%, reforçando a importância de futuras melhorias metodológicas, como a inclusão de variáveis exógenas e o uso de mecanismos de atenção para melhor adaptação a eventos fora do padrão histórico aprendido.

Figura 13 – Comparação Entre a Previsão Futura e os Dados Reais do Índice S&P 500.



Fonte: Autoria própria (2025).

5 Discussão dos Resultados

Os resultados obtidos ao longo deste estudo evidenciam o potencial e as limitações da modelagem preditiva do índice S&P 500 utilizando redes neurais recorrentes do tipo Long Short-Term Memory (LSTM). A partir da decomposição da série temporal e da avaliação da capacidade preditiva do modelo, foi possível realizar uma análise abrangente dos padrões estruturais do índice e da eficácia do modelo em capturá-los.

Inicialmente, a decomposição da série temporal revelou a presença de uma tendência estocástica ascendente de longo prazo, intercalada por quedas abruptas associadas a choques exógenos, como crises financeiras, pandemias e tensões geopolíticas. A sazonalidade, embora de baixa amplitude, mostrou-se recorrente e bem definida, com influências documentadas na literatura, como o efeito janeiro e o impacto de distribuições de dividendos e rebalanceamentos semestrais (REINGANUM, 1983; HAUGEN; LAKONISHOK, 1988; GU, 2015). Já os resíduos evidenciaram que, embora parte das oscilações seja explicada pelos componentes estruturais, os choques externos continuam a desempenhar um papel central na volatilidade do índice, reforçando a necessidade de modelos capazes de lidar com incerteza e eventos não lineares (BOX et al., 2015).

Em relação à previsão de 30 dias úteis realizada no início de maio de 2025, os resultados foram satisfatórios no curto prazo, com erros percentuais relativamente baixos nas primeiras semanas, conforme apresentado na Tabela 1. No entanto, à medida que o horizonte da previsão se estendia, o acúmulo de erros tornou-se evidente, com desvios crescentes nos últimos dias de maio. Essa deterioração da precisão, também observada graficamente na Figura 13, é compatível com limitações conhecidas de modelos autoregressivos que não incorporam variáveis exógenas ou mecanismos de correção de erro acumulado.

A superestimação sistemática observada nas semanas finais do mês pode estar relacionada à ausência de eventos de estabilização de mercado no conjunto de dados utilizados para treinamento, além da impossibilidade do modelo de antecipar reversões ou consolidações de preços com base apenas no histórico de preços. Tal comportamento destaca a importância de considerar a inclusão de variáveis macroeconômicas ou dados alternativos, como volume e política monetária, para aumentar a robustez preditiva em cenários mais complexos (HAMILTON, 1994).

Apesar dessas limitações, o desempenho geral do modelo reforça o potencial das redes LSTM para previsão financeira. O uso de técnicas de decomposição e visualização gráfica complementou a análise estatística, permitindo uma melhor interpretação dos dados e dos desvios, o que é essencial em estudos aplicados a mercados dinâmicos como o de ações.

6 Considerações Finais

Este trabalho teve como objetivo desenvolver e avaliar um modelo preditivo baseado em Redes Neurais Recorrentes do tipo Long Short-Term Memory (LSTM) para o índice S&P 500 através de dados históricos. A metodologia adotada combinou técnicas de aprendizado profundo com a análise estrutural da série temporal, incluindo a decomposição em tendência, sazonalidade e resíduos, permitindo uma compreensão mais robusta do comportamento histórico do índice para a modelagem preditiva.

A decomposição da série temporal foi essencial para destacar a presença de sazonalidades recorrentes e a natureza estocástica da tendência de longo prazo do índice. A análise dos resíduos mostrou que, embora parte da variação seja explicada pelos componentes estruturais, choques exógenos como crises financeiras e eventos geopolíticos continuam a exercer influência significativa sobre a dinâmica do mercado, exigindo modelos mais adaptativos e sensíveis ao contexto.

Os resultados obtidos na modelagem LSTM demonstraram que o modelo é capaz de capturar padrões temporais relevantes e realizar previsões com boa acurácia. O modelo mostra-se robusto inclusive em períodos de alta volatilidade e a subsequente recuperação do mercado, embora os picos e vales mais acentuados tendam a apresentar ligeira defasagem, o desempenho geral demonstra sua eficácia na reconstrução do padrão de comportamento da série original.

A previsão de 30 dias úteis realizada para o mês de maio de 2025 indicou bom desempenho nas primeiras semanas, com erros percentuais relativamente baixos, mas apresentou tendência de superestimação à medida que a projeção avançava, evidenciando limitações diante do acúmulo de incertezas e da ausência de variáveis exógenas.

Em conclusão, a utilização de Redes Neurais Recorrentes com arquitetura LSTM mostrou-se uma abordagem promissora para a previsão de séries financeiras, oferecendo resultados consistentes e com perspectivas para trabalhos futuros, recomendando-se a incorporação de variáveis explicativas externas, como indicadores macroeconômicos.

Referências

AHMED, N. K. et al. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, Taylor & Francis, v. 29, n. 5-6, p. 594–621, 2010.

ARMSTRONG, J. S. (Ed.). *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Dordrecht, The Netherlands: Springer Science & Business Media, 2001. 849 p. ISBN 978-0-7923-7204-8.

AVENUE. *S&P 500: entenda esse importante índice do mercado de ações*. 2025. Acesso em: 20 maio 2025. Disponível em: <https://avenue.us/blog/sp-500/>.

BENZINGA. *Is Stock Market Open Or Closed On Christmas Day 2024? Here Is What You Can Trade Today*. 2024. Acesso em: 27 maio 2025. Disponível em: <https://www.benzinga.com/markets/equities/24/12/42677225/is-the-stock-market-open-or-closed-on-christmas-day-2024-here-is-what-you-can-trade-today>.

BHANDARI, H. N. et al. Predicting stock market index using lstm. *Machine Learning with Applications*, v. 9, p. 100320, 2022. ISSN 2666-8270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2666827022000378>.

BLANCHARD, O. *Macroeconomia*. 7. ed. São Paulo: Pearson, 2017. 311-375 p.

BODIE, Z.; KANE, A.; MARCUS, A. J. *Investments*. 13. ed. New York: McGraw-Hill, 2023. 551-574 p.

BOX, G. E. P. et al. *Time Series Analysis: Forecasting and Control*. 5th. ed. [S.l.]: John Wiley & Sons, 2015.

BRANDÃO, T. L. *btheodora/TCC: A evolução histórica do SP 500*. Zenodo, 2025. Disponível em: <https://zenodo.org/doi/10.5281/zenodo.15650219>.

BREALEY, R. A.; MYERS, S. C.; ALLEN, F. *Princípios de finanças corporativas*. 13. ed. São Paulo: McGraw-Hill, 2020. 230-377 p.

BROCKWELL, P. J.; DAVIS, R. A. *Introduction to Time Series and Forecasting*. 3. ed. New York: Springer, 2016.

BROWNLEE, J. *Deep Learning for Time Series Forecasting - Predict the Future with MLPs, CNNs and LSTMs in Python*. Estados Unidos: Machine Learning Mastery, 2018.

CHOLLET, F. *Deep Learning com Python*. 2. ed. [S.l.]: Manning Publications, 2021.

COWPERTWAIT, P.; METCALFE, A. V. *Introductory Time Series with R*. [S.l.]: Springer-Verlag New York, 2009.

ENDERS, W. *Applied Econometric Time Series*. 4. ed. Hoboken: Wiley, 2015.

FABOZZI, F. J. *Foundations of financial markets and institutions*. 5. ed. Boston: Pearson, 2021. 45-78 p.

Federal Reserve. *Monetary Policy Report*. Washington, DC, 2024. Acesso em: 1 jun. 2025. Disponível em: <https://www.federalreserve.gov/monetarypolicy.htm>.

FISCHER, T.; KRAUSS, C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, v. 270, n. 2, p. 654–669, 2018.

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd. ed. Sebastopol, CA: O'Reilly Media, 2019.

GHAREHBAGHI, A. *Deep Learning in Time Series Analysis*. [S.l.]: CRC Press, 2023.

GITMAN, L. J.; JOEHNK, M. D. *Fundamentals of Investing, Global Edition*. 11. ed. São Paulo: Pearson, 2013.

GOODFELLOW YOSHUA BENGIO, A. C. I. *Deep Learning*. Cambridge: MIT Press, 2016.

GRAVES, A. *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Heidelberg: Springer, 2012.

GU, A. Y. The June Phenomenon and the Changing Month of the Year Effect. *Accounting and Finance Research*, v. 4, n. 3, p. 1–1, August 2015. Disponível em: <https://ideas.repec.org/a/jfr/afr111/v4y2015i3p1.html>.

HAGGARD, K. S.; WITTE, H. D. The halloween effect: Trick or treat? *International Review of Financial Analysis*, v. 19, n. 5, p. 379–387, 2010. ISSN 1057-5219. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1057521910000608>.

HAMILTON, J. D. *Time series analysis*. 1. ed. [S.l.]: Princeton university press, 1994.

HAUGEN, R. A.; LAKONISHOK, J. *The incredible January effect: The stock market's unsolved mystery*. [S.l.]: Dow Jones-Irwin, 1988.

HAYKIN, S. O. *Neural Networks and Learning Machines*. 3rd edition. ed. [S.l.]: Prentice Hall, 2009. ISBN 0131471392.

HIRSHLEIFER, D.; JIANG, D.; DIGIOVANNI, Y. M. Mood beta and seasonalities in stock returns. *Journal of Financial Economics*, v. 137, n. 1, p. 272–295, 2020. ISSN 0304-405X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0304405X20300362>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HYNDMAN, R. J. et al. *Forecasting with Exponential Smoothing: The State Space Approach*. New York: Springer, 2008. ISBN 3540719164.

IBM. *Redes neurais recorrentes: O que são e como funcionam*. 2024. Acesso em: 21 maio 2025. Disponível em: <https://www.ibm.com/br-pt/think/topics/recurrent-neural-networks>.

IONOS. *Recurrent Neural Network: O que é rede neural recorrente*. 2025. Acesso em: 21 maio 2025. Disponível em: <https://www.ionos.com/pt-br/digitalguide/sites-de-internet/desenvolvimento-web/recurrent-neural-network/>.

JACOBS, B. I.; LEVY, K. N. Calendar anomalies: Abnormal returns at calendar turning points. *Financial Analysts Journal*, CFA Institute, v. 44, n. 6, p. 28–39, 1988.

JAMES DANIELA WITTEN, T. H. R. T. G. *An Introduction to Statistical Learning: with Applications in R*. 2. ed. Nova York: Springer, 2021.

KAMSTRA, M. J.; KRAMER, L. A.; LEVI, M. D. Winter Blues: A SAD Stock Market Cycle. *American Economic Review*, v. 93, n. 1, p. 324–343, March 2003. Disponível em: <https://ideas.repec.org/a/aea/aecrev/v93y2003i1p324-343.html>.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KUTNER, M.; NACHTSHEIM, C.; NETER, J. *Applied Linear Statistical Models*. 5. ed. Boston: McGraw-Hill, 2005.

LAKONISHOK, J.; SHLEIFER, A.; VISHNY, R. W. Window dressing by pension fund managers. *American Economic Review*, American Economic Association, v. 81, n. 2, p. 227–231, 1991.

LEE, S. *Practical Applications of Min-Max Scaling in Machine Learning*. 2025. Acesso em: 27 maio 2025. Disponível em: <https://www.numberanalytics.com/blog/practical-min-max-scaling-machine-learning>.

MAKRIDAKIS, S.; WHEELWRIGHT, S. C.; HYNDMAN, R. J. *Forecasting: Methods and Applications*. 3rd. ed. New York: Wiley, 1997. ISBN 978-0-471-60271-5.

MANKIW, N. G. *Introdução à economia*. 7. ed. São Paulo: Cengage Learning, 2015.

MISHKIN, F. S. *The economics of money, banking, and financial markets*. 12. ed. New York: Pearson, 2019. 214-267 p.

MORETTIN, P. A. *Análise de Séries Temporais*. São Paulo: Blucher, 2006.

Nasdaq Trader. *Nasdaq Trader - Market Calendars*. 2025. Accessed: 2025-05-20. Disponível em: <https://www.nasdaqtrader.com/trader.aspx?id=calendar>.

PCS - Programa de Computação Científica. *Análise de Séries Temporais Financeiras e Trend Following*. 2023. <https://pcs.usp.br/pcspf/wp-content/uploads/sites/8/2023/12/Monografia_PCS3560_SEM_2023_Grupo_S07.pdf>. Acesso em: 26 maio 2025.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Acesso em: 27 maio 2024. Disponível em: <https://jmlr.org/papers/v12/pedregosa11a.html>.

RANAROUSSI, y. G. I. *Missing Data - Issue #525*. 2020. Acesso em: 27 maio 2025. Disponível em: <https://github.com/ranaroussi/yfinance/issues/525>.

REINGANUM, M. R. The anomalous stock market behavior of small firms in january. *Journal of Financial Economics*, Elsevier, v. 12, n. 1, p. 89–104, 1983.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986.

SEYHUN, H. N. Overreaction or fundamentals: Some lessons from insiders' response to the market crash of 1987. *Journal of Finance*, v. 45, n. 5, p. 1363–1388, 1990.

SHILLER, R. J. *Irrational exuberance*. 3. ed. Princeton: Princeton University Press, 2015. 131-193 p.

SHUMWAY, R.; STOFFER, D. *Time Series Analysis and Its Applications: With R Examples*. 4. ed. Nova York: Springer, 2025.

S&P GLOBAL. *S&P 500 Brochure*. New York, 2024. 15-22 p. Acesso em: 20 maio 2025. Disponível em: <https://www.spglobal.com/spdji/en/indices/equity/sp-500/>.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, JMLR.org, v. 15, n. 1, p. 1929–1958, jan. 2014. ISSN 1532-4435.

STOLL, H. R.; WHALEY, R. E. Program trading and expiration-day effects. *Financial Analysts Journal*, v. 47, n. 2, p. 16–28, 1991.

TSAY, R. S. *Analysis of financial time series*. 2nd. ed. [S.l.]: Wiley, 2005. (Wiley series in probability and statistics).

XP Investimentos. *Liberation Day – Trump anuncia pacote de tarifas recíprocas*. 2025. Acesso em 02 de junho de 2025. Disponível em: <https://conteudos.xpi.com.br/acoes/relatorios/trump-anuncia-novas-tarifas-como-se-posicionar-grafico-da-semana/>.

Yahoo Finance. *S&P 500 Historical Data*. 2025. Acesso em 2025-05-20. Disponível em: <https://finance.yahoo.com/quote/%5EGSPC/history/>.

Yahoo Finance. *Treasury Bill Historical Data*. 2025. Acesso em 2025-05-20. Disponível em: <https://finance.yahoo.com/quote/%5EIRX/history/>.

APÊNDICE A – Código Python para a Decomposição da Série Temporal

```python
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.dates as mdates

# 1. Coleta de dados
df = yf.download('^GSPC', start='2000-01-01', end='2025-05-01',
    progress=False)[['Close']]
df = df.asfreq('B')  # Frequencia de dias uteis
df.dropna(inplace=True)

# 2. Decomposicao da serie (modelo multiplicativo)
decomposicao = seasonal_decompose(df['Close'], model='
   multiplicative', period=252)

# 3. Grafico da Decomposicao
plt.rcParams.update({'figure.figsize': (14, 8)})

fig = decomposicao.plot()

# Titulos dos Subgraficos
componentes = ['Observado', 'Tendencia', 'Sazonalidade', '
   Residuo']
for ax, titulo in zip(fig.axes, componentes):
    ax.set_ylabel(titulo, fontsize=14)
    ax.set_xlabel('Data', fontsize=14)
    ax.set_xlim(pd.to_datetime('2000-01-01'), pd.to_datetime('
       2025-05-01'))
    ax.xaxis.set_major_locator(mdates.YearLocator(4, month=1,
       day=1))  # Agora de 4 em 4 anos
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
    ax.tick_params(axis='both', labelsize=14)

plt.tight_layout()
```

```
plt.savefig("decomposicao_sp500_pt.pdf", dpi=300)
plt.show()

# 5. Grafico 2 - Recorte de dois anos da sazonalidade
sazonal = decomposicao.seasonal
sazonal_recorte = sazonal['2023':'2024']

plt.figure(figsize=(14, 5))
plt.plot(sazonal_recorte.index, sazonal_recorte.values, label='
    Sazonalidade', color='tab:blue', linewidth=2)

plt.xlabel('Mes', fontsize=14)
plt.ylabel('Fator Sazonal', fontsize=14)
plt.xticks(
    sazonal_recorte.index[::21],
    sazonal_recorte.index.to_series().dt.strftime('%m-%Y')
        [::21],
    rotation=45,
    fontsize=14
)
plt.yticks(fontsize=14)
plt.legend(fontsize=14)
plt.grid(False)
plt.tight_layout()
plt.savefig("sazonalidade_sp500_2023_2024.pdf", dpi=300)
plt.show()

# 6. Grafico 3 - Residuo
plt.figure(figsize=(14, 4))
plt.plot(decomposicao.resid, color='C0', label='Residuo')
plt.xlabel('Data', fontsize=14)
plt.ylabel('Residuo', fontsize=14)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.grid(False)
plt.legend(loc='upper right', fontsize=14)
plt.gca().xaxis.set_major_locator(mdates.YearLocator(4))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
plt.tight_layout()
plt.savefig("residuo_sp500_pt.pdf", dpi=300)
plt.show()
```

APÊNDICE B – Código Python para a Modelagem LSTM e Previsão Futura

```python
import numpy as np
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input
import matplotlib.dates as mdates

# Reprodutibilidade
np.random.seed(42)
tf.random.set_seed(42)

# Coleta de dados
start_date = '2000-01-01'
end_date = '2025-05-01'

sp500 = yf.download('^GSPC', start=start_date, end=end_date,
    interval='1d')[['Close', 'Volume']]
sp500.dropna(inplace=True)
sp500.columns = ['SP500_Close', 'SP500_Volume']

interest_rate = yf.download('^IRX', start=start_date, end=
    end_date, interval='1d')[['Close']]
interest_rate.columns = ['Interest_Rate']
interest_rate.dropna(inplace=True)

df = sp500.join(interest_rate, how='inner')

# Normalizacao
scaler = MinMaxScaler()
```

```
33 df_scaled = pd.DataFrame(scaler.fit_transform(df), index=df.
       index, columns=df.columns)
34
35 # Preparacao para LSTM
36 n_steps = 60
37 X, y, dates = [], [], []
38
39 for i in range(n_steps, len(df_scaled)):
40     X.append(df_scaled.iloc[i - n_steps:i].values)
41     y.append(df_scaled.iloc[i]['SP500_Close'])
42     dates.append(df_scaled.index[i])
43
44 X, y, dates = np.array(X), np.array(y), np.array(dates)
45
46 # Divisoes ajustadas: Treino, Validacao, Teste
47 train_mask = (dates >= pd.Timestamp('2000-01-01')) & (dates <=
       pd.Timestamp('2022-12-31'))
48 val_mask   = (dates >= pd.Timestamp('2023-01-01')) & (dates <=
       pd.Timestamp('2023-12-31'))
49 test_mask  = (dates >= pd.Timestamp('2024-01-01')) & (dates <=
       pd.Timestamp('2025-05-01'))
50
51 X_train, y_train = X[train_mask], y[train_mask]
52 X_val, y_val     = X[val_mask], y[val_mask]
53 X_test, y_test   = X[test_mask], y[test_mask]
54 test_dates       = dates[test_mask]
55
56 # Modelo LSTM
57 model = Sequential([
58     Input(shape=(X.shape[1], X.shape[2])),
59     LSTM(64, return_sequences=True),
60     Dropout(0.2),
61     LSTM(64),
62     Dropout(0.2),
63     Dense(1)
64 ])
65 model.compile(optimizer='adam', loss='mean_squared_error')
66
67 # Treinamento
68 history = model.fit(X_train, y_train, epochs=30, batch_size=32,
       validation_data=(X_val, y_val), verbose=1)
69
```

```
70  # Previsoes no conjunto de teste
71  y_pred_test = model.predict(X_test)
72  sp500_close_idx = df.columns.get_loc('SP500_Close')
73
74  def inverse_scale(y_scaled):
75      temp = np.zeros((len(y_scaled), df_scaled.shape[1]))
76      temp[:, sp500_close_idx] = y_scaled
77      return scaler.inverse_transform(temp)[:, sp500_close_idx]
78
79  y_test_real = inverse_scale(y_test)
80  y_pred_real = inverse_scale(y_pred_test.flatten())
81
82  # Metricas
83  mse = mean_squared_error(y_test_real, y_pred_real)
84  rmse = np.sqrt(mse)
85  print(f"\n Test MSE: {mse:.4f} | RMSE: {rmse:.4f}")
86
87  # Grafico 1 - Avaliacao
88  plt.figure(figsize=(14, 6))
89  plt.plot(test_dates, y_test_real, label='Real (Teste)',
        linewidth=2)
90  plt.plot(test_dates, y_pred_real, label='Previsto (Teste)',
        linestyle='--')
91  plt.xlabel('Data', fontsize=14)
92  plt.ylabel('Pontos', fontsize=14)
93  plt.xticks(fontsize=14)
94  plt.yticks(fontsize=14)
95  plt.legend(fontsize=14)
96  plt.grid(False)
97  plt.tight_layout()
98  ax = plt.gca()
99  ax.xaxis.set_major_formatter(mdates.DateFormatter('%m-%Y'))
100 plt.savefig("avaliacao_teste_lstmTCC.pdf", dpi=300)
101 plt.show()
102
103 # Previsao Futura com Sazonalidade e Ruido (30 dias uteis)
104 future_start = pd.Timestamp('2025-05-01')
105 last_sequence_date = future_start - pd.Timedelta(days=1)
106 index_position = df.index.get_loc(last_sequence_date)
107 last_sequence = df_scaled.iloc[index_position - n_steps + 1:
        index_position + 1].values.reshape(1, n_steps, df_scaled.
        shape[1])
```

```
108
109 # Estimativas historicas para ruido e sazonalidade
110 daily_returns = df['SP500_Close'].pct_change().dropna()
111 volatility = daily_returns[-252:].std()
112 seasonality_pattern = daily_returns[-252:].groupby(
        daily_returns[-252:].index.dayofyear).mean()
113
114 predictions = []
115 future_dates = pd.date_range(start=future_start, periods=30,
        freq='B')
116
117 for i, future_date in enumerate(future_dates):
118     next_pred_scaled = model.predict(last_sequence, verbose=0)
            [0][0]
119
120     # Converte previsao para escala real
121     temp = np.zeros((1, df_scaled.shape[1]))
122     temp[0, sp500_close_idx] = next_pred_scaled
123     predicted_price = scaler.inverse_transform(temp)[0,
            sp500_close_idx]
124
125     # Ajusta com sazonalidade e ruido
126     day_of_year = future_date.dayofyear
127     seasonal_return = seasonality_pattern.get(day_of_year, 0)
128     noise = np.random.normal(loc=0, scale=volatility)
129     predicted_price *= (1 + seasonal_return + noise)
130
131     # Armazena e atualiza sequencia
132     predictions.append(predicted_price)
133     temp_df = df.iloc[[index_position]].copy()
134     temp_df.iloc[0, sp500_close_idx] = predicted_price
135     temp_scaled = scaler.transform(temp_df)
136     new_entry = np.append(last_sequence[0, 1:, :], [temp_scaled
            [0]], axis=0)
137     last_sequence = new_entry.reshape(1, n_steps, df_scaled.
            shape[1])
138
139 # Impressao das datas e valores previstos
140 print("\n Projecao de 30 dias (com sazonalidade e ruido) a
        partir de 01/05/2025:\n")
141 for date, price in zip(future_dates, predictions):
142     print(f"{date.strftime('%Y-%m-%d')} {price:.2f} pontos")
```

```
143
144 # Grafico 2 - Previsao Futura
145 plt.figure(figsize=(14, 6))
146 plt.plot(df.index, df['SP500_Close'], label='Historico S&P 500'
    )
147 plt.plot(future_dates, predictions, label='Previsao Futura (30
    dias)', linestyle='--', color='orange')
148 plt.xlabel('Data', fontsize=14)
149 plt.ylabel('Pontos', fontsize=14)
150 plt.xticks(fontsize=14)
151 plt.yticks(fontsize=14)
152 plt.legend(fontsize=14)
153 plt.grid(False)
154 plt.tight_layout()
155 ax = plt.gca()
156 ax.xaxis.set_major_formatter(mdates.DateFormatter('%m-%Y'))
157 plt.savefig("previsao_futura_lstmTCC.pdf", dpi=300)
158 plt.show()
159
160 # Grafico 3 - Zoom
161 start_zoom = pd.Timestamp('2024-01-01')
162 end_zoom = future_dates[-1]
163
164 historical_mask = (df.index >= start_zoom)
165 historical_zoom_dates = df.index[historical_mask]
166 historical_zoom_prices = df['SP500_Close'][historical_mask]
167
168 plt.figure(figsize=(14, 6))
169 plt.plot(historical_zoom_dates, historical_zoom_prices, label='
    Historico desde 2024')
170 plt.plot(future_dates, predictions, label='Previsao Futura (30
    dias)', linestyle='--', color='orange')
171 plt.xlabel('Data', fontsize=14)
172 plt.ylabel('Pontos', fontsize=14)
173 plt.xticks(fontsize=14)
174 plt.yticks(fontsize=14)
175 plt.ylim(3500, 6500)
176 plt.xlim(start_zoom, end_zoom)
177 plt.legend(fontsize=14)
178 plt.grid(False)
179 plt.tight_layout()
180 ax = plt.gca()
```

```
181 ax.xaxis.set_major_formatter(mdates.DateFormatter('%m-%Y'))
182 plt.savefig("zoom_projecao_futura_lstmTCC.pdf", dpi=300)
183 plt.show()
184
185 #RMSE
186 meses_datetime = pd.to_datetime([mes + '-01' for mes in
       rmse_por_mes.keys()])
187
188 # Grafico RMSE por Mes
189 plt.figure(figsize=(14, 6))
190 plt.plot(meses_datetime, list(rmse_por_mes.values()), marker='o
       ', linestyle='-', linewidth=2, label='RMSE por mes')
191
192 # Rotulos de cada ponto
193 for i, v in enumerate(rmse_por_mes.values()):
194     plt.text(meses_datetime[i], v + max(rmse_por_mes.values())
          * 0.01, f'{v:.2f}',
195              ha='center', fontsize=12)
196
197 plt.xlabel('Data', fontsize=14)
198 plt.ylabel('RMSE (Pontos)', fontsize=14)
199 plt.xticks(fontsize=14)
200 plt.yticks(fontsize=14)
201 plt.grid(False)
202 plt.legend(fontsize=14)
203 ax = plt.gca()
204 ax.xaxis.set_major_formatter(mdates.DateFormatter('%m-%Y'))
205 plt.tight_layout()
206 plt.savefig("rmse_por_mes_lstmTCC.pdf", dpi=300)
207 plt.show()
208
209 #MAPE
210 # DataFrame
211 df_mape = pd.DataFrame({
212     'Data': test_dates,  # np.array de datas
213     'Real': y_test_real,  # valores reais (nao escalados)
214     'Previsto': y_pred_real  # valores previstos (nao escalados
          )
215 })
216
217 # Agrupamento por mes
218 df_mape['AnoMes'] = df_mape['Data'].dt.to_period('M')
```

```
219 df_mape['ErroPercentual'] = np.abs((df_mape['Real'] - df_mape['
    Previsto']) / df_mape['Real']) * 100
220
221 # MAPE por mes
222 mape_por_mes = df_mape.groupby('AnoMes')['ErroPercentual'].mean
    ().round(2)
223
224 meses_datetime = pd.to_datetime([str(m) for m in mape_por_mes.
    index])
225
226 # Grafico do MAPE por mes
227 plt.figure(figsize=(14, 6))
228 plt.plot(meses_datetime, mape_por_mes.values, marker='o',
    linestyle='-', linewidth=2, label='MAPE por mes')
229
230 # Rotulos de cada ponto
231 for i, v in enumerate(mape_por_mes.values):
232     plt.text(meses_datetime[i], v + max(mape_por_mes.values) *
        0.01, f'{v:.2f}%',
233              ha='center', fontsize=12)
234
235 plt.xlabel('Data', fontsize=14)
236 plt.ylabel('MAPE (%)', fontsize=14)
237 plt.xticks(fontsize=14)
238 plt.yticks(fontsize=14)
239 plt.grid(False)
240 plt.legend(fontsize=14)
241 ax = plt.gca()
242 ax.xaxis.set_major_formatter(mdates.DateFormatter('%m-%Y'))
243 plt.tight_layout()
244 plt.savefig("mape_por_mes_lstmTCC.pdf", dpi=300)
245 plt.show()
```

APÊNDICE C – Código Python para o Gráfico entre a Previsão e os Dados Reais do Índice S&P 500

```
    import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import pandas as pd

data = {
    'Data': [
        '2025-05-01', '2025-05-02', '2025-05-05', '2025-05-06',
            '2025-05-07',
        '2025-05-08', '2025-05-09', '2025-05-12', '2025-05-13',
            '2025-05-14',
        '2025-05-15', '2025-05-16', '2025-05-19', '2025-05-20',
            '2025-05-21',
        '2025-05-22', '2025-05-23', '2025-05-27', '2025-05-28',
            '2025-05-29',
        '2025-05-30'
    ],
    'Previsto': [
        5584.53, 5627.28, 5722.98, 5819.05, 5804.45,
        5797.75, 5956.56, 5951.11, 5909.29, 6015.28,
        6011.36, 6085.94, 6104.73, 5978.77, 6009.13,
        6111.73, 6061.90, 6119.81, 6096.70, 6324.29,
        6184.95
    ],
    'Real': [
        5604.14, 5686.67, 5650.38, 5606.91, 5631.28,
        5663.94, 5659.91, 5844.19, 5886.55, 5892.58,
        5916.93, 5958.38, 5963.60, 5940.46, 5844.61,
        5842.01, 5802.82, 5921.54, 5888.55, 5912.17,
        5911.69
    ]
}

# DataFrame
```

```
30 df_plot = pd.DataFrame(data)
31 df_plot['Data'] = pd.to_datetime(df_plot['Data'])
32
33 # Grafico
34 plt.figure(figsize=(14, 6))
35 plt.plot(df_plot['Data'], df_plot['Real'], label='Real (Teste)'
       , linewidth=2)
36 plt.plot(df_plot['Data'], df_plot['Previsto'], label='Previsto
       (Teste)', linestyle='--')
37
38 plt.xlabel('Data', fontsize=14)
39 plt.ylabel('Pontos', fontsize=14)
40 plt.xticks(fontsize=14)
41 plt.yticks(fontsize=14)
42 plt.legend(fontsize=14)
43 plt.grid(False)
44 plt.tight_layout()
45 ax = plt.gca()
46 ax.xaxis.set_major_formatter(mdates.DateFormatter('%d-%m-%Y'))
47 plt.savefig("avaliacao_maio2025_lstmTCC.pdf", dpi=300)
48 plt.show()
```

APÊNDICE D – Código Python para Calcular o Erro Percentual entre a Previsão e os Dados Reais do Índice S&P 500

```
    import pandas as pd

# Dados da tabela
dados = {
    "Data": [
        "2025-05-01", "2025-05-02", "2025-05-05", "2025-05-06",
            "2025-05-07",
        "2025-05-08", "2025-05-09", "2025-05-12", "2025-05-13",
            "2025-05-14",
        "2025-05-15", "2025-05-16", "2025-05-19", "2025-05-20",
            "2025-05-21",
        "2025-05-22", "2025-05-23", "2025-05-27", "2025-05-28",
            "2025-05-29", "2025-05-30"
    ],
    "Previsto": [
        5584.53, 5627.28, 5722.98, 5819.05, 5804.45,
        5797.75, 5956.56, 5951.11, 5909.29, 6015.28,
        6011.36, 6085.94, 6104.73, 5978.77, 6009.13,
        6111.73, 6061.90, 6119.81, 6096.70, 6324.29, 6184.95
    ],
    "Real": [
        5604.14, 5686.67, 5650.38, 5606.91, 5631.28,
        5663.94, 5659.91, 5844.19, 5886.55, 5892.58,
        5916.93, 5958.38, 5963.60, 5940.46, 5844.61,
        5842.01, 5802.82, 5921.54, 5888.55, 5912.17, 5911.69
    ]
}

# Criar DataFrame
df = pd.DataFrame(dados)

# Calcular erro percentual absoluto para cada linha
```

```
df["Erro %"] = abs(df["Previsto"] - df["Real"]) / df["Real"] *
    100

# Calcular MAPE
mape = df["Erro %"].mean()

# Exibir resultados
print(df.to_string(index=False))
print(f"\nMAPE: {mape:.2f}%")
```