UNIVERSIDADE FEDERAL DE ALFENAS

ERICK VERONESI BOCZAR

PREVISÃO DE PREÇOS DE AÇÕES COM TÉCNICAS DE APRENDIZADO PROFUNDO: UM ESTUDO COM REDES NEURAIS RECORRENTES EM EMPRESAS BRASILEIRAS

ALFENAS/MG

2025

ERICK VERONESI BOCZAR

PREVISÃO DE PREÇOS DE AÇÕES COM TÉCNICAS DE APRENDIZADO PROFUNDO: UM ESTUDO COM REDES NEURAIS RECORRENTES EM EMPRESAS BRASILEIRAS

Dissertação apresentada ao Programa de Pós-Graduação em Estatística Aplicada e Biometria da Universidade Federal de Alfenas - MG, como parte dos requisitos para a obtenção do título de Mestre.
Linha de Pesquisa: Biometria
Orientador: Prof. Dr. Denismar Alves Nogueira

ALFENAS/MG

2025

Sistema de Bibliotecas da Universidade Federal de Alfenas
Biblioteca Central

Boczar, Erick Veronesi.
Previsão de preços de ações com técnicas de aprendizado profundo : um estudo com redes neurais recorrentes em empresas brasileiras / Erick Veronesi Boczar. - Alfenas, MG, 2025.
103 f. : il. -

Orientador(a): Denismar Alves Nogueira.
Dissertação (Mestrado em Estatística Aplicada e Biometria) - Universidade Federal de Alfenas, Alfenas, MG, 2025.
Bibliografia.

1. GRU. 2. LSTM. 3. MLP. 4. Séries Temporais. 5. Mercado Financeiro. I. Nogueira, Denismar Alves, orient. II. Título.

Ficha gerada automaticamente com dados fornecidos pelo autor.

ERICK VERONESI BOCZAR

Previsão de Preços de Ações com Técnicas de Aprendizado Profundo: Um Estudo com Redes Neurais recorrentes em Empresas Brasileiras

A Banca examinadora abaixo-assinada aprova a Dissertação apresentada como parte dos requisitos para a obtenção do título de Mestre em Estatística Aplicada e Biometria pela Universidade Federal de Alfenas. Área de concentração: Estatística Aplicada e Biometria.

Aprovada em: 30 de setembro de 2025.

Prof. Dr. Denismar Alves Nogueira
Instituição: Universidade Federal de Alfenas - UNIFAL-MG

Profa. Dra. Patrícia de Siqueira Ramos
Instituição: Universidade Federal de Alfenas - UNIFAL-MG

Prof. Dr. Luiz Eduardo da Silva
Instituição: Universidade Federal de Alfenas - UNIFAL-MG

Documento assinado eletronicamente por **Denismar Alves Nogueira**, **Professor do Magistério Superior**, em 30/09/2025, às 15:55, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.unifal-mg.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1629550** e o código CRC **514EDD3C**.

Dedico este trabalho à minha amada esposa Jordana, por seu amor e apoio incondicional, e ao meu querido filho Matteo, cuja alegria e energia são minha maior inspiração

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão à minha esposa Jordana e ao meu filho Matteo, por sua alegria contagiante e fonte diária de inspiração; aos meus pais Levi e Dalva, por gerarem a base e proverem meus ensinamentos fundamentais; à minha irmã Thaís e sobrinha Alice pelo amor e suporte; ao meu estimado amigo Ivan, por sua amizade leal e companheirismo; e ao meu orientador Denismar, por sua orientação e sabedoria inestimáveis. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001. A todos vocês, meu mais sincero obrigado.

RESUMO

A previsão de preços de ações é um desafio central no mercado financeiro devido à natureza volátil e não linear das séries temporais. Com a crescente demanda por ferramentas preditivas mais precisas, técnicas de aprendizado profundo, como as Redes Neurais Recorrentes (RNNs), têm se destacado. Este estudo realiza uma análise comparativa do desempenho dos modelos *Gated Recurrent Unit* (GRU) e *Long Short-Term Memory* (LSTM), além de *ensemble* que combina ambos, na tarefa de prever os preços de fechamento de 16 ações do mercado brasileiro, distribuídas em quatro setores distintos (Varejo, Bancário, Energia e Tecnologia). O desempenho dos modelos propostos foi comparado a um *baseline*: uma rede neural não recorrente, a *Multi-Layer Perceptron* (MLP). A metodologia foi conduzida em duas fases: uma com hiperparâmetros padrão e outra com otimização via *Grid Search*. Os resultados demonstram que os modelos recorrentes propostos são estatisticamente superiores ao *baseline* (valor-p $< 0{,}05$). A fase de otimização resultou em uma melhora média notável no Erro Percentual Absoluto Médio (MAPE), com uma redução de até 32,13% para o modelo LSTM. Dentre os modelos otimizados, o GRU emergiu como o mais robusto, apresentando o melhor equilíbrio entre baixo erro e estabilidade de desempenho em todos os setores analisados. Conclui-se que as arquiteturas GRU após a otimização, representam uma ferramenta altamente eficaz para a predição de preços de ações, oferecendo *percepções* valiosas que podem auxiliar na tomada de decisão de investidores e mitigar riscos no mercado financeiro.

Palavras-chave: GRU; LSTM; MLP; Séries Temporais; Mercado Financeiro.

ABSTRACT

Stock price forecasting is a central challenge in the financial market due to the volatile and non-linear nature of time series. With the growing demand for more accurate predictive tools, deep learning techniques, such as Recurrent Neural Networks (RNNs), have gained prominence. This study conducts a comparative analysis of the performance of *Gated Recurrent Unit* (GRU) and *Long Short-Term Memory* (LSTM) models, as well as an ensemble model that combines both, in the task of forecasting the closing prices of 16 stocks from the Brazilian market, distributed across four distinct sectors (Retail, Banking, Energy, and Technology). The performance of the proposed models was compared against a baseline: a non-recurrent neural network, the *Multi-Layer Perceptron* (MLP). The methodology was carried out in two phases: one with standard hyperparameters and another with optimization via *Grid Search*. The results demonstrate that the proposed recurrent models are statistically superior to the baselines (p-value < 0.05). The optimization phase led to a remarkable average improvement in the Mean Absolute Percentage Error (MAPE), with a reduction of up to 32.13% for the LSTM model. Among the optimized models, the GRU emerged as the most robust, exhibiting the best balance between low error and performance stability across all analyzed sectors. It is concluded that GRU architectures represent a highly effective tool for stock price prediction, offering valuable insights that can aid investor decision-making and mitigate risks in the financial market.

Keywords: GRU; LSTM; MLP; Time Series; Financial Market.

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A origem das bolsas de valores está associada ao desenvolvimento do comércio internacional e à necessidade de criar mecanismos formais para a negociação de títulos e participações societárias. O primeiro registro de uma instituição organizada com esse propósito remonta ao início do século XVII, com a fundação da Bolsa de Amsterdã, em 1602, pela Companhia das Índias Orientais (VOC), considerada a primeira empresa do mundo a emitir ações negociáveis publicamente. Esse evento é amplamente reconhecido como o marco inicial do mercado de capitais moderno, ao possibilitar a captação de recursos por meio da venda de participações societárias e a liquidez dos investimentos (Neal, 2012).

Nos séculos seguintes, o modelo de bolsa de valores se expandiu pela Europa e pelos Estados Unidos, com destaque para a Bolsa de Londres (1801) e a Bolsa de Nova Iorque (NYSE), fundada em 1792, que se consolidou como o maior e mais influente mercado acionário do mundo. No Brasil, o desenvolvimento desse tipo de instituição teve início com a criação da Bolsa de Valores do Rio de Janeiro (BVRJ) em 1845, seguida por diversas bolsas regionais que, posteriormente, foram integradas. O processo de modernização e unificação culminou, em 2017, na criação da B3 – Brasil, Bolsa, Balcão, a partir da fusão entre a BM&FBOVESPA e a CETIP, consolidando um ambiente único de negociação de ativos financeiros e derivativos no país (COMISSÃO DE VALORES MOBILIÁRIOS (BRASIL), 2020).

Essa trajetória histórica demonstra a evolução estrutural e tecnológica das bolsas de valores, que passaram de pregões presenciais, conhecidos como *viva-voz*, para plataformas totalmente eletrônicas. Tais avanços ampliaram a eficiência, a segurança e a acessibilidade dos mercados, reforçando o papel central das bolsas como elo entre poupadores e empresas no financiamento da atividade econômica (COMISSÃO DE VALORES MOBILIÁRIOS (BRASIL), 2020).

A previsão de preços no mercado de ações é uma atividade crucial e desafiadora para investidores e profissionais do setor financeiro. A busca por métodos eficazes de previsão tem sido objeto de interesse contínuo, especialmente diante da natureza volátil e complexa dos mercados financeiros (Shen; Shafiq, 2020; Kamalov, 2020; Chandra; He, 2021). Nesse contexto, o uso de técnicas avançadas, como inteligência artificial, tem despertado grande interesse devido à sua capacidade de capturar padrões complexos nos dados e fornecer previsões mais precisas e adaptáveis (Tiwari; Srivastava; Gera, 2020; Houssein *et al.*, 2021).

Esta dissertação propõe uma comparação de modelos de previsão de preços de ações, explorando o potencial da inteligência artificial nesse cenário. A previsão de preços de ações é um campo de estudo multidisciplinar que combina elementos da análise financeira, estatística e ciência de dados. A capacidade de prever com precisão os movimentos do mercado pode influenciar significativamente as estratégias de investimento e as decisões financeiras (Shen; Shafiq, 2020; Kamalov, 2020).

Como há diversas arquiteturas que podem ser utilizadas para essa tarefa, neste estudo abordam-se as Redes Neurais Recorrentes (RNNs), que foram especificamente projetadas para modelar sequências e memorizar padrões ao longo do tempo, como a LSTM (*Long Short Term Memory*) e a GRU (*Gated Recurrent Unit*). A proposta deste estudo é comparar estas com uma rede neural do tipo *feedforward*, a MLP (Multilayer Perceptron), que é uma rede neural de outro tipo, que será utilizada como *baseline*. Além de tudo isso, estuda-se o efeito da combinação de arquiteturas, ou *ensemble*, para as redes LSTM e GRU, bem como o ganho que uma otimização de hiperparâmetros proporciona em um estudo de predição de valores.

Um modelo é considerado robusto quando seu desempenho é consistente em diferentes ativos e condições de mercado, enquanto a adaptabilidade refere-se à sua capacidade de lidar com a natureza não estacionária e a volatilidade inerente aos dados financeiros (Houssein *et al.*, 2021; Yu; Yan, 2020). A proposta é explorar a capacidade das redes neurais em capturar padrões não lineares e complexos nos dados (Bishop, 2006; Olah, 2015), buscando uma melhoria na precisão das previsões de preços de fechamento. A novidade não reside na criação de uma nova arquitetura neural, mas na aplicação de um *framework* metodológico completo e rigoroso para comparar, otimizar e validar modelos de *deep learning* em um contexto prático e relevante, gerando conclusões específicas sobre a sua eficácia no mercado de ações brasileiro.

2 OBJETIVO

2.1 OBJETIVO GERAL

Realizar uma análise comparativa do desempenho de modelos de aprendizado profundo de Redes Neurais Recorrentes (RNN) para a previsão de preços de ações no curto prazo (horizonte H = 1 dia), confrontando-os com uma *baseline*, além de explorar os ganhos na otimização de hiperparâmetros e o uso de combinação de modelos (*ensemble*).

2.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral, os seguintes objetivos específicos foram traçados e executados:

a) **análise exploratória de dados (AED):** Realizar uma análise aprofundada sobre a carteira de 16 ativos de diferentes setores da economia (Varejo, Bancário, Energia e Tecnologia), para identificar e compreender suas características fundamentais, como tendências, volatilidade, sazonalidade e correlações entre os diferentes papéis e setores;

b) **estabelecimento de *baseline*:** Implementar modelo de referência (*baseline*) para criar um padrão de comparação: uma rede neural não recorrente, a MLP;

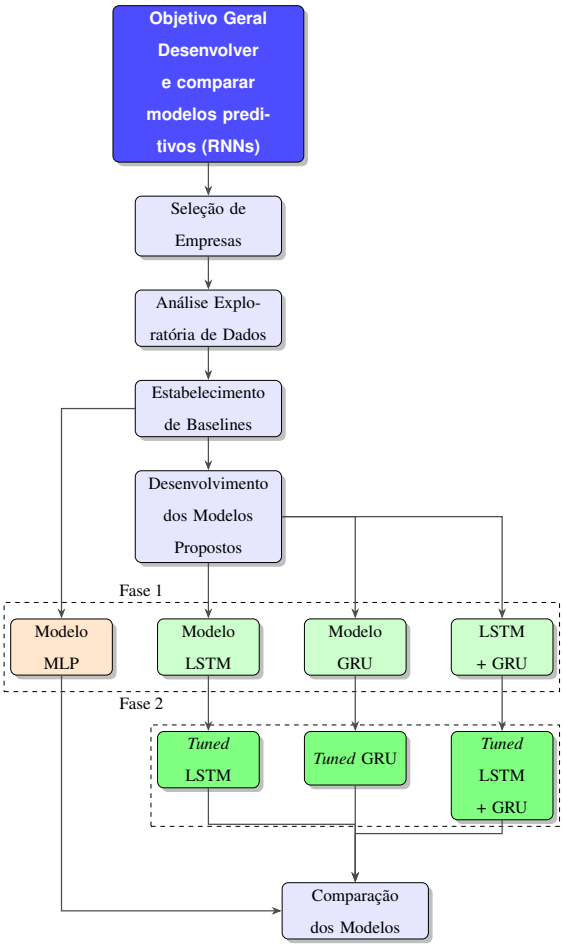
c) **desenvolvimento dos modelos propostos (Fase 1):** implementar e avaliar os modelos de redes neurais recorrentes, especificamente a GRU, a LSTM, e um modelo de *Ensemble* (Combinado) que agrega, neste caso, as médias das predições de ambos, utilizando um conjunto de hiperparâmetros padrão;

d) **otimização dos modelos (Fase 2):** aplicar uma técnica de ajuste fino de hiperparâmetros, o *Grid Search*, aos modelos GRU e LSTM para encontrar suas configurações de melhor performance e maximizar seu poder preditivo;

e) **análise comparativa e validação estatística:** realizar uma análise comparativa abrangente do desempenho de todos os modelos, utilizando as métricas MAPE,

RMSE e R^2. Avaliar o impacto da otimização (Fase 1 vs. Fase 2) e o desempenho segmentado por setor, concluindo com uma validação estatística.

Figura 1 – Fluxograma dos objetivos do trabalho



Fonte: Autor (2025).

3 REVISÃO DE LITERATURA

A compreensão dos conceitos teóricos fundamentais é essencial para embasar a aplicação prática das técnicas de previsão de preços de ações propostas neste trabalho. Assim, esta seção apresenta uma revisão da literatura relacionada aos principais tópicos envolvidos na pesquisa, abordando o funcionamento do mercado financeiro, os fundamentos da inteligência artificial aplicada à previsão de séries temporais e a revisão de trabalhos similares que servem como referência para a construção da metodologia adotada.

3.1 BOLSA DE VALORES

A bolsa de valores representa um ambiente regulado e organizado, seja físico ou eletrônico, onde investidores podem negociar ativos financeiros específicos, conhecidos como valores mobiliários, mediante a participação de instituições autorizadas, os intermediários (COMISSÃO DE VALORES MOBILIÁRIOS (BRASIL), 2020). Nesse espaço, ocorre a negociação de diferentes ativos, como ações, debêntures, cotas de fundos imobiliários, fundos fechados, ETFs (*Exchange Traded Fund*) e fundos de índices (COMISSÃO DE VALORES MOBILIÁRIOS (BRASIL), 2020).

As negociações ocorrem no pregão da bolsa, onde ordens de compra e venda são submetidas, e transações são concretizadas conforme os procedimentos e regras estabelecidas. Anteriormente, o pregão acontecia fisicamente nas instalações da bolsa, onde corretores se reuniam, recebiam ordens, buscavam contrapartes e registravam as operações de forma presencial, no chamado pregão viva-voz (COMISSÃO DE VALORES MOBILIÁRIOS (BRASIL), 2020).

No entanto, a partir de 2005 na Bovespa e de 2009 na BM&F (Bolsa de Mercadorias e Futuros), o pregão presencial viva-voz foi abolido, e as negociações passaram a ser exclusivamente eletrônicas. Apesar da mudança no método, o conceito permanece similar: a bolsa oferece um ambiente de negociação eletrônico onde as ordens são registradas, e os negócios são fechados ao melhor preço disponível (COMISSÃO DE VALORES MOBILIÁRIOS (BRASIL), 2020).

No mercado brasileiro, esses ambientes de bolsa devem ser administrados por entidades autorizadas pela Comissão de Valores Mobiliários (CVM). Atualmente, até a data de publicação

do guia, apenas uma instituição, a B3 – Brasil, Bolsa e Balcão, detém essa autorização. A B3 oferece diversos serviços, incluindo a administração e operacionalização do sistema de negociação de ativos conhecido como PUMA (Plataforma Unificada MultiAtivos) (COMISSÃO DE VALORES MOBILIÁRIOS (BRASIL), 2020).

No contexto do mercado acionário, duas abordagens clássicas são amplamente utilizadas para a avaliação e previsão do comportamento dos preços: a análise fundamentalista e a análise técnica. A análise fundamentalista baseia-se em indicadores econômicos, contábeis e setoriais, buscando estimar o valor intrínseco de uma empresa a partir de fundamentos como lucro, fluxo de caixa, endividamento, perspectivas de crescimento e governança corporativa (Martins; Assaf, 2019). Seu pressuposto central é que, no longo prazo, o preço de mercado tende a convergir para o valor justo do ativo.

Por outro lado, a análise técnica concentra-se no comportamento histórico dos preços e volumes negociados, partindo da premissa de que todas as informações relevantes já estão refletidas nos preços e que padrões passados tendem a se repetir (Murphy, 1999). Essa abordagem utiliza gráficos, tendências e indicadores como médias móveis, bandas de Bollinger e índices de força relativa (RSI) para identificar oportunidades de compra e venda.

Embora ambas as metodologias ofereçam subsídios importantes à tomada de decisão, elas apresentam limitações em mercados altamente voláteis e não lineares. Nesse contexto, a aplicação de técnicas de aprendizado profundo surge como uma alternativa promissora, ao permitir a identificação de padrões complexos nos dados sem depender de suposições lineares ou estáticas sobre o comportamento do mercado.

3.2 VOLATILIDADE DE AÇÕES

A volatilidade é uma das métricas mais importantes no mercado financeiro, utilizada para quantificar o grau de variação ou dispersão dos retornos de um ativo ao longo de um determinado período. Em essência, ela serve como um indicador primário do risco associado a um ativo: uma alta volatilidade implica em maior incerteza e risco, enquanto uma baixa volatilidade sugere maior estabilidade nos preços. A abordagem mais comum para mensurar a volatilidade histórica é através do desvio padrão dos retornos do ativo (Hull, 2018).

No contexto deste trabalho, a volatilidade foi calculada com base nos retornos diários

das ações, obtidos a partir dos preços de fechamento (*Close*). Os retornos diários (r_t) foram calculados como a variação percentual do preço de um dia para o outro:

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{3.1}$$

onde P_t é o preço de fechamento no dia t, e P_{t-1} é o preço de fechamento no dia anterior.

A partir da série de retornos diários de cada ativo, foi calculada a **volatilidade histórica anualizada**. Esta é uma métrica padrão que mensura o desvio padrão dos retornos diários e o projeta para uma base anual, permitindo uma comparação mais consistente entre diferentes ativos e mercados. O cálculo foi realizado em duas etapas: primeiro, calculou-se o desvio padrão ($\sigma_{\text{diária}}$) de toda a série de retornos diários. Em seguida, este valor foi anualizado multiplicando-o pela raiz quadrada de 252, que é o número aproximado de dias de negociação em um ano no mercado brasileiro. A fórmula é dada por:

$$\sigma_{\text{anualizada}} = \sigma_{\text{diária}} \times \sqrt{252} \tag{3.2}$$

onde:

$$\sigma_{\text{diária}} = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(r_t - \bar{r})^2} \tag{3.3}$$

sendo N o número total de retornos no período, r_t o retorno no dia t, e $\bar{r}$ a média dos retornos.

É importante notar que a volatilidade em mercados financeiros não é constante. Uma característica empírica bem documentada é a presença de "*clusters* de volatilidade", onde períodos de alta instabilidade tendem a ser seguidos por outros períodos de alta instabilidade, e períodos de calmaria são seguidos por calmaria. Este fenômeno foi formalizado por Engle (1982), que introduziu os modelos ARCH (*Autoregressive Conditional Heteroskedasticity*) para capturar essa dinâmica (Engle, 1982). A análise da volatilidade anualizada, embora não modele essa dinâmica, permite criar um ranking de risco entre as empresas e os setores estudados, fornecendo informações valiosas sobre a adequação dos diferentes modelos de previsão, uma vez que ativos mais voláteis tendem a apresentar maior dificuldade de modelagem.

3.3 INTELIGÊNCIA ARTIFICIAL

A Inteligência Artificial (IA) é um campo multidisciplinar da ciência da computação que visa desenvolver sistemas capazes de executar tarefas que normalmente exigiriam inteligência humana, como raciocínio, aprendizado, percepção e tomada de decisão (Russell; Norvig, 2022). Ao emular aspectos do comportamento cognitivo humano, a IA busca não apenas automatizar processos, mas também adaptá-los, melhorá-los e otimizar sua execução de forma autônoma. Com raízes que remontam aos anos 1950, a IA evoluiu de simples programas baseados em regras para sistemas complexos capazes de aprender com dados massivos e melhorar seu desempenho ao longo do tempo.

O avanço da IA tem impulsionado inovações em diversas áreas, como medicina, indústria automotiva, segurança cibernética, robótica, processamento de linguagem natural, entre outras. Dentro desse vasto espectro, destaca-se o aprendizado de máquina (*Machine Learning*), uma subárea da IA focada em desenvolver algoritmos que aprendem automaticamente a partir de dados, sem serem explicitamente programados para cada tarefa específica. Em particular, técnicas de aprendizado profundo (*Deep Learning*), baseadas em redes neurais artificiais, tornaram-se essenciais para enfrentar problemas de alta complexidade e grande volume de dados (Tiwari; Srivastava; Gera, 2020).

No contexto financeiro, a aplicação de Inteligência Artificial para a previsão de tendências do mercado de ações tem se tornado um tema central de estudo e prática. Essa linha de pesquisa opera sob a hipótese de que informações públicas disponíveis — como preços históricos, volumes de negociação e indicadores macroeconômicos — carregam padrões e correlações que podem ser extraídas e modeladas para antecipar movimentos futuros dos preços dos ativos (Kolarik; Rudorfer, 1997). Assim, a análise preditiva não busca apenas replicar tendências passadas, mas identificar sinais sutis e relações não lineares que escapariam a métodos estatísticos tradicionais.

Prever tendências no mercado de ações, no entanto, é uma tarefa desafiadora e sujeita a inúmeras variáveis imprevisíveis, como mudanças súbitas na política monetária, eventos geopolíticos e comportamento humano irracional, que frequentemente fogem das estruturas padronizadas dos modelos. Devido a essa alta volatilidade e à natureza estocástica dos mercados financeiros, a previsão exata do valor futuro de um ativo é extremamente difícil. Estudos mostram que é mais viável — e muitas vezes mais útil — focar na previsão da direção dos preços

(alta ou baixa) do que tentar estimar o valor exato (Kolarik; Rudorfer, 1997).

Ainda assim, previsões precisas, mesmo que restritas à direção do movimento, podem gerar vantagens significativas em estratégias de investimento, gestão de risco e alocação de portfólio. Por essa razão, o uso de modelos de aprendizado profundo, especialmente Redes Neurais Recorrentes (RNNs), Redes LSTM (Long Short-Term Memory) e GRUs (Gated Recurrent Units), tem ganhado destaque, devido à sua capacidade de capturar dependências temporais e padrões complexos em séries financeiras.

O destaque dessas arquiteturas se deve a mecanismos específicos. As redes **LSTM**, por exemplo, foram explicitamente projetadas para resolver o problema da dependência de longo prazo, utilizando um sistema de portões (*gates*) que permite à rede reter informações por longos períodos, uma limitação crítica das RNNs tradicionais (Hochreiter; Schmidhuber, 1997). Já as redes GRU, uma evolução mais recente, simplificam a arquitetura da LSTM e têm demonstrado em diversos estudos um desempenho comparável com maior eficiência computacional, tornando-se uma alternativa robusta e atrativa (Chung *et al.*, 2014). A recomendação desses modelos na literatura, portanto, baseia-se em sua capacidade estrutural de modelar a dinâmica complexa do mercado de forma mais eficaz do que os métodos estatísticos clássicos.

Além disso, técnicas modernas de otimização de hiperparâmetros e abordagens híbridas — combinando diferentes arquiteturas de redes e métodos estatísticos — estão expandindo as fronteiras do que é possível na previsão financeira. Apesar dos avanços, o desafio de criar modelos robustos e generalizáveis para previsão de preços de ações permanece aberto, exigindo uma contínua evolução teórica e prática nas técnicas de Inteligência Artificial.

Para uma melhor compreensão das características e diferenças entre esses modelos, bem como de conceitos essenciais relacionados ao treinamento de redes neurais, como a técnica de *dropout*, as funções de perda e a distinção entre parâmetros e hiperparâmetros, as próximas subseções apresentam uma revisão detalhada desses tópicos.

3.4 REDES NEURAIS ARTIFICIAIS (RNA)

As Redes Neurais Artificiais (RNA) constituem uma classe de modelos computacionais inspirados no funcionamento do cérebro humano, projetadas para reconhecer padrões e modelar relações complexas entre variáveis. Esses modelos são compostos por unidades fundamentais

chamadas *neurônios artificiais*, organizados em camadas conectadas entre si, que processam e transformam informações de entrada por meio de operações matemáticas (Bishop, 2006; Goodfellow; Bengio; Courville, 2016).

Em sua forma mais básica, uma RNA é composta por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio recebe um conjunto de entradas ponderadas, aplica uma soma linear e, em seguida, uma função de ativação não linear, responsável por introduzir a capacidade de modelar relações complexas. Matematicamente, o funcionamento de um neurônio pode ser expresso por:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3.4}$$

em que x_i representa as entradas, w_i os pesos associados a cada conexão, b o termo de viés (*bias*), e $f(\cdot)$ a função de ativação, que pode ser linear ou não linear (como a função sigmoide, tangente hiperbólica ou ReLU).

O aprendizado de uma rede neural ocorre por meio de um processo iterativo de ajuste dos pesos e vieses, com o objetivo de minimizar uma função de perda que quantifica o erro entre as previsões da rede e os valores observados. Esse processo é geralmente conduzido pelo algoritmo de retropropagação (*backpropagation*), que utiliza o gradiente do erro em relação aos parâmetros da rede para realizar atualizações segundo um otimizador, como o método do gradiente descendente (Goodfellow; Bengio; Courville, 2016).

A principal vantagem das RNAs em relação aos modelos estatísticos tradicionais reside em sua capacidade de aproximar funções não lineares complexas e aprender representações hierárquicas dos dados, mesmo quando as relações entre variáveis são altamente não triviais. Entretanto, essa flexibilidade vem acompanhada de desafios, como a necessidade de grandes volumes de dados para treinamento, alto custo computacional e risco de sobreajuste (*overfitting*) se não houver regularização adequada (Bishop, 2006).

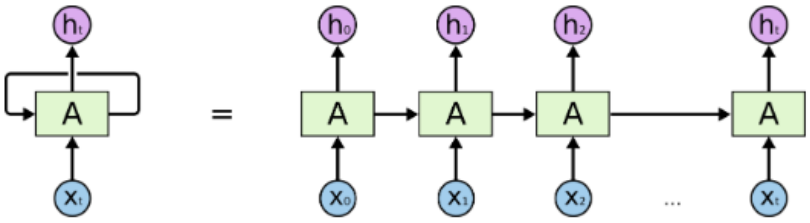
No contexto de previsão de séries temporais financeiras, as RNAs são amplamente utilizadas por sua capacidade de capturar relações não lineares entre retornos passados e preços futuros. Contudo, as redes neurais tradicionais do tipo *feedforward*, como o Perceptron Multicamadas (MLP), tratam cada observação de forma independente, sem levar em conta a dependência temporal entre os dados. Essa limitação motivou o desenvolvimento das Redes Neurais Recorrentes (RNN), apresentadas na seção seguinte, que introduzem conexões internas capazes de reter informações sobre estados passados, tornando-as mais adequadas para o processamento

de séries temporais.

3.4.1 Redes neurais recorrentes (RNN)

As Redes Neurais Recorrentes (RNNs) são uma classe de redes neurais especializadas para lidar com sequências de dados, como séries temporais, texto ou áudio (Goodfellow; Bengio; Courville, 2016). Sua característica principal é a capacidade de manter um estado interno ou "memória"que captura informações sobre os elementos anteriores da sequência. Isso é conseguido através de *loops* internos dentro da rede, permitindo que as informações persistam de um passo de processamento para o próximo (Olah, 2015).

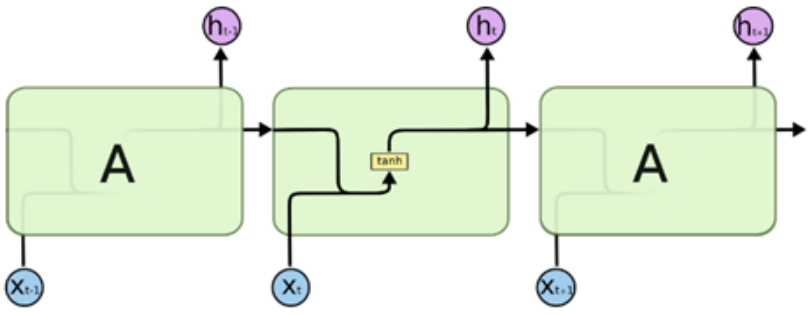
Figura 2 – Representação gráfica de uma Rede Neural Recorrente padrão e a representação das entradas e saídas do modelo



Fonte: Adaptado de Olah (2015).

Na Figura 2, observa-se o funcionamento básico de uma RNN padrão. À esquerda há uma célula da rede denotada por "A", com um dado de entrada "X_t"e um dado de saída "h_t". Esta rede é composta por várias células de processamento de dados, sendo que em cada célula o valor anterior é multiplicado por um ponderador associado a essa conexão (Goodfellow; Bengio; Courville, 2016). Essa operação é geralmente mediada por uma função de ativação, como a função sigmoide, que atribui uma ponderação ao valor anterior entre 0 e 1, ou, dependendo da função utilizada, entre -1 e 1.

Figura 3 – Função de ativação (Sigmoide) utilizada em RNN padrão

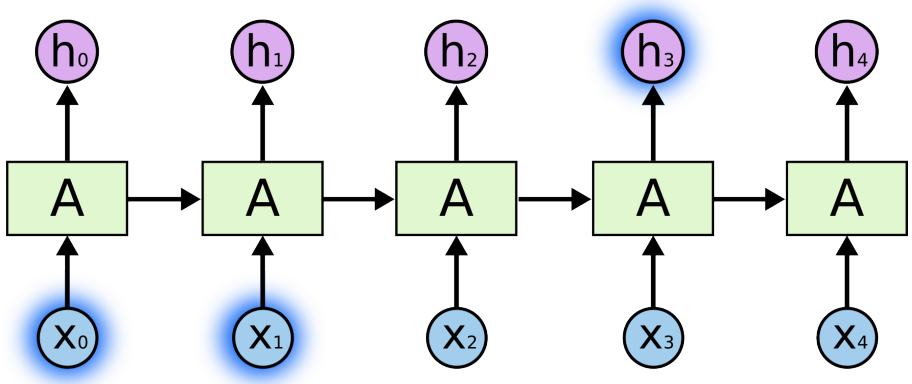


Fonte: Adaptado de Olah (2015).

Para RNNs padrão, cada célula possui apenas uma camada com função de ativação. Na Figura 3, utilizou-se a função tangente hiperbólica como função de ativação, ponderando os valores entre -1 e 1.

Essa estrutura permite que as RNNs sejam utilizadas para examinar informações recentes e chegar a uma conclusão final. Por exemplo, considere um modelo de linguagem tentando prever a próxima palavra com base nas anteriores. Se o objetivo é prever a última palavra em "as nuvens estão no", não é necessário mais contexto — é intuitivo que a próxima palavra será "céu". Nesses casos, onde a lacuna entre a informação relevante e o local onde ela é necessária é pequena, as RNNs podem aprender a utilizar a informação passada eficientemente (Olah, 2015).

Figura 4 – RNN padrão com pequeno número de camadas.

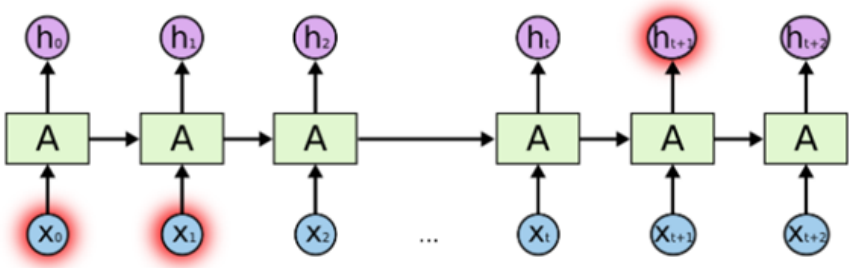


Fonte: Adaptado de Olah (2015).

Na Figura 4, observa-se que, quando há apenas uma pequena distância entre as células — isto é, com redes neurais de pequena profundidade —, a informação passada não se perde, permitindo que o modelo alcance a saída correta esperada.

O problema torna-se mais crítico à medida que o número de camadas e o volume de dados aumentam, ampliando essa lacuna de forma expressiva. Esse aumento da distância temporal dificulta que as RNNs padrão capturem relações de longo prazo, tornando-as incapazes de aprender conexões entre informações distantes (Hochreiter; Schmidhuber, 1997). Para o objetivo deste trabalho, compreende-se que esta limitação restringe severamente o uso de RNNs tradicionais na tarefa de predição de preços futuros de ações.

Figura 5 – RNN padrão com grande número de camadas



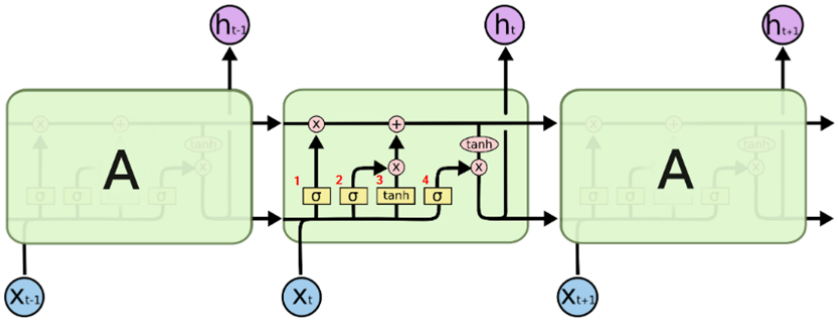
Fonte: Adaptado de Olah (2015).

Visando solucionar esta limitação, foram criadas as LSTM (*Long Short Term Memory*).

Redes LSTM (*Long Short Term Memory*)

Introduzidas por Hochreiter e Schmidhuber (1997), as LSTM foram explicitamente desenvolvidas para evitar o problema de dependência a longo prazo (*Long-term dependency*) explicado mais adiante. A principal mudança entre uma LSTM e uma RNN comum é que, em cada módulo, em vez de possuir apenas uma camada, haverá quatro camadas interagindo de uma forma especial (Goodfellow; Bengio; Courville, 2016; Olah, 2015).

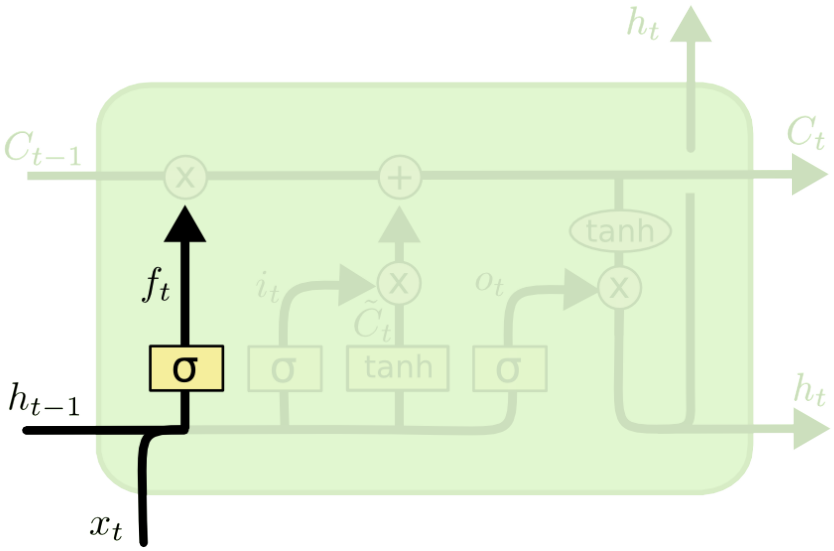
Figura 6 – Rede LSTM com quatro camadas por módulo interagindo entre si.



Fonte: Adaptado de Olah (2015).

Na Figura 6, pode-se observar quatro funções sigmoides (em amarelo) representando as camadas internas. As LSTM possuem a habilidade de remover ou adicionar informação ao estado da célula, regulada por estruturas chamadas portões (*gates*) (Goodfellow; Bengio; Courville, 2016). Cada portão é uma combinação de uma função sigmoide e uma operação de multiplicação (denotada por "x"rosa na figura), controlando o fluxo de informações.

Figura 7 – Passo 1 – Funcionamento de uma LSTM: portão de esquecimento.



Fonte: Adaptado de Olah (2015).

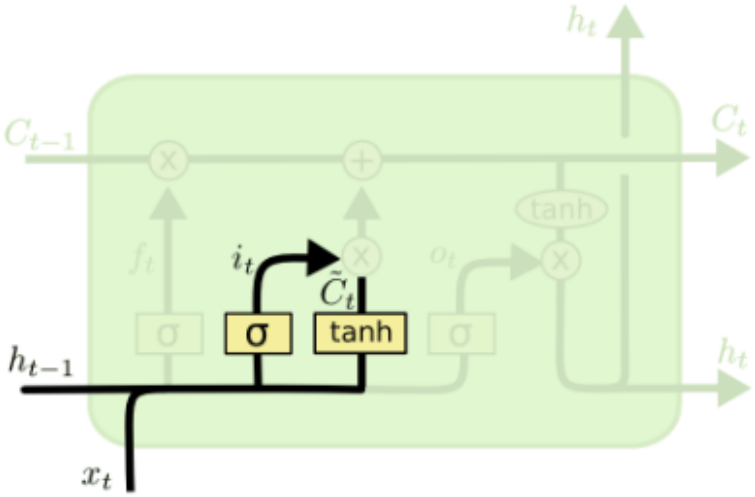
Nesta primeira etapa, a função sigmoide decide qual informação será descartada do estado anterior, através do portão de esquecimento, com base em h_{t-1} e x_t.

$$f_t = \sigma(W_f \cdot [h_{t-1},x_t] + b_f) \tag{3.5}$$

Em que f_t representa o portão de esquecimento (*forget gate*), responsável por decidir quais informações do estado da célula anterior (C_{t-1}) devem ser mantidas ou descartadas. Esse controle é feito com base na combinação linear entre o estado oculto anterior (h_{t-1}) e a entrada atual (x_t), ponderada pela matriz de pesos W_f e ajustada pelo viés b_f. A função de ativação sigmoide $\sigma(\cdot)$ restringe os valores resultantes ao intervalo $[0,1]$, em que valores próximos de 1 indicam preservação da informação e próximos de 0 indicam esquecimento.

O próximo passo é definir qual nova informação será armazenada no estado da célula C_t. Esse processo é dividido em duas partes: (i) o Portão de Entrada calcula i_t, e (ii) a camada $tanh$ gera novos candidatos $\tilde{C}_t$:

Figura 8 – Passo 2 – Funcionamento de uma LSTM: portão de entrada.



Fonte: Adaptado de Olah (2015).

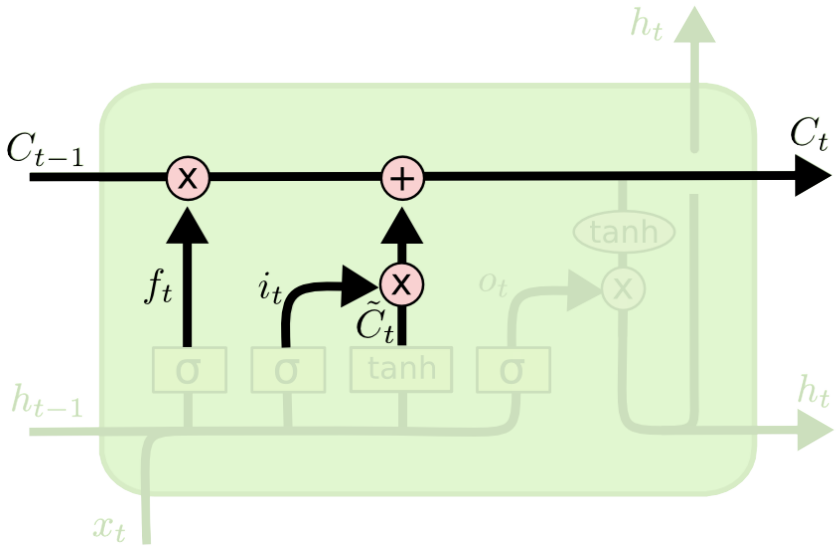
$$i_t = \sigma(W_i \cdot [h_{t-1},x_t] + b_i) \tag{3.6}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1},x_t] + b_C) \tag{3.7}$$

Nas equações (3.6) e (3.7), o portão de entrada (i_t) e o vetor candidato $\tilde{C}_t$ definem quais novas informações serão adicionadas ao estado da célula. O portão i_t, obtido também por meio de uma função sigmoide, determina a proporção de nova informação a ser incorporada. Já $\tilde{C}_t$, calculado pela função tangente hiperbólica, gera um vetor de candidatos que contém possíveis novos conteúdos a serem armazenados. As matrizes de pesos W_i e W_C, juntamente com seus vieses b_i e b_C, controlam como as entradas e o histórico influenciam a atualização do estado interno.

Atualiza-se então o estado da célula:

Figura 9 – Passo 3 – Atualização do estado da célula na LSTM.



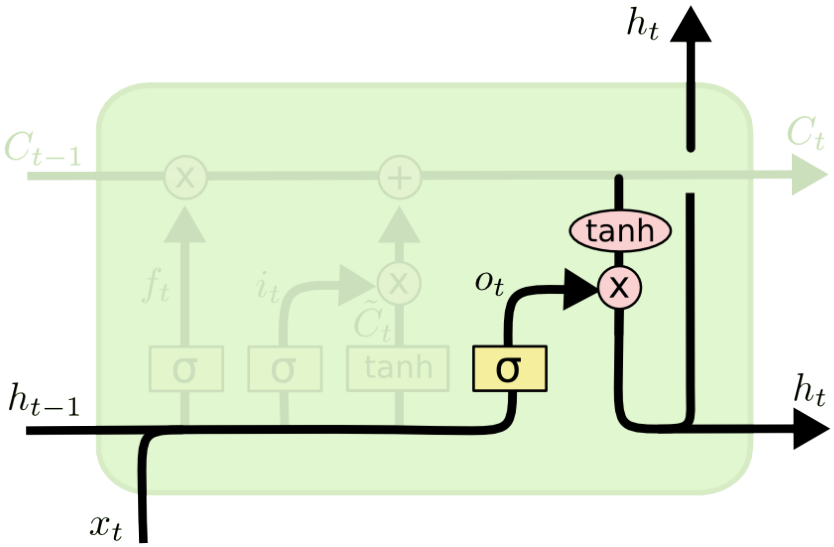
Fonte: Adaptado de Olah (2015).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.8}$$

Na equação (3.8), o novo estado de célula C_t é atualizado combinando as duas informações anteriores: a memória anterior C_{t-1}, ponderada pelo portão de esquecimento f_t, e os novos candidatos $\tilde{C}_t$, ponderados pelo portão de entrada i_t. Essa soma ponderada define o equilíbrio entre a retenção do conhecimento passado e a incorporação de novas informações, garantindo que o modelo mantenha dependências de longo prazo relevantes e descarte padrões obsoletos.

Por fim, determina-se a saída h_t do módulo LSTM, combinando o estado da célula filtrado por um portão de saída:

Figura 10 – Passo 4 – Funcionamento de uma LSTM: geração da saída.



Fonte: Adaptado de Olah (2015).

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.9}$$

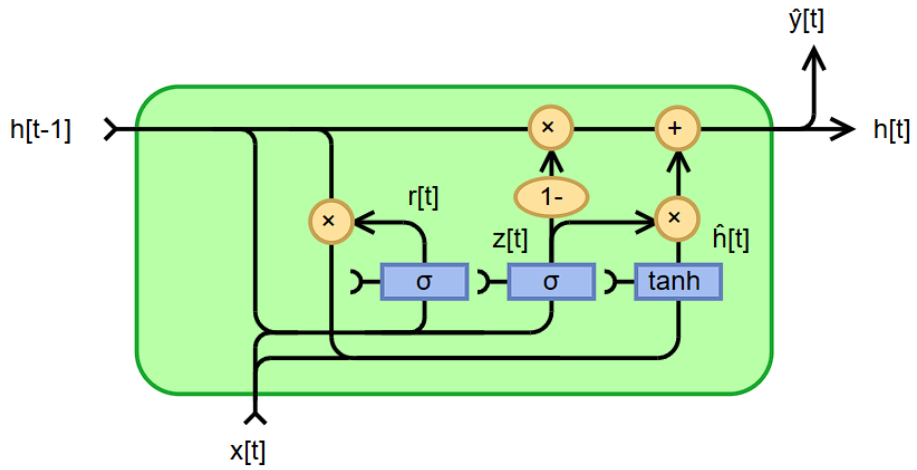
$$h_t = o_t * tanh(C_t) \tag{3.10}$$

As equações (3.9) e (3.10) definem o portão de saída (o_t) e o estado oculto atual (h_t). O portão o_t controla quais partes da memória atual C_t serão expostas como saída do módulo LSTM. Ele é obtido pela combinação de h_{t-1} e x_t, seguida da função sigmoide, enquanto o estado oculto h_t é calculado multiplicando o valor de saída o_t pela transformação $\tanh(C_t)$. Essa estrutura garante que a célula possa decidir, a cada passo, o quanto da informação armazenada será propagada para o próximo instante de tempo.

Redes GRU (*Gated Recurrent Unit*)

Segundo Dey e Salem (), a rede neural GRU (*Gated Recurrent Unit*) reduz a quantidade de portões ("*Gates*") para dois em relação ao modelo LSTM. Os dois portões são chamados Portão de Atualização ("*Update Gate*") z_t e Portão de Reinício ("*Reset Gate*") r_t.

Figura 11 – Funcionamento de uma GRU



Fonte: Adaptado de WIKIMEDIA COMMONS (2018).

A RNN GRU é bastante similar à LSTM, porém com menor número de sinais de portões externos na interpolação da equação 3.4.

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{3.11}$$

$$\tilde{h}_t = g(W_h x_t + U_h(r_t * h_{t-1}) + b_h) \tag{3.12}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{3.13}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{3.14}$$

Nas equações (3.11) a (3.14), são descritos os mecanismos internos da GRU (*Gated Recurrent Unit*). Nessa arquitetura, o portão de atualização z_t (equação 3.13) controla o quanto do estado anterior h_{t-1} será mantido, enquanto o portão de reinicialização r_t (equação 3.14) determina o quanto do estado passado será ignorado ao gerar a nova ativação. Ambos são calculados por funções sigmoides aplicadas às combinações lineares das entradas (x_t) e do estado anterior (h_{t-1}), com pesos W_z, U_z, W_r, U_r e respectivos vieses b_z e b_r.

A equação (3.12) define o vetor candidato $\tilde{h}_t$, calculado a partir da entrada ponderada $W_h x_t$ e da contribuição do estado anterior modulado por r_t. A função $g(\cdot)$, geralmente $\tanh(\cdot)$, introduz a não linearidade que permite capturar dependências complexas entre as observações. Por fim, a equação (3.11) combina o estado anterior e o vetor candidato por meio do portão de atualização z_t, produzindo o novo estado oculto h_t. Essa operação permite à GRU controlar

dinamicamente o fluxo de informações e reduzir o número de parâmetros em relação à LSTM, mantendo desempenho comparável com menor custo computacional (Chung *et al.*, 2014; Józefowicz; Zaremba; Sutskever, 2015).

GRU simplifica a estrutura do LSTM combinando o Portão de Esquecimento e de Entrada em um único Portão de Atualização ("*Update Gate*") z_t e utilizando o Portão de Reinício ("*Reset Gate*") r_t para controlar o fluxo de informação. Não há estado de separação de célula, fazendo uso de h_t para este propósito.

Estudos comparativos mostram ainda que essa simplificação reduz o número de parâmetros e o tempo de treinamento. Chung *et al.* (2014) demonstram que, sob o mesmo número de neurônios, GRUs convergem mais rapidamente do que LSTMs, enquanto análises comparativas indicam que a GRU apresenta menos parâmetros e maior eficiência computacional em relação à LSTM (Józefowicz; Zaremba; Sutskever, 2015).

3.4.2 Modelo de comparação - *Baseline*

Após a apresentação das arquiteturas de redes neurais recorrentes, é fundamental contextualizar e validar seu desempenho. Uma prática robusta para essa validação é a comparação com um modelo de referência (*baseline*) estabelecido na literatura, como no caso de Nelson, Pereira e Oliveira (2017), Shen e Shafiq (2020), Shah *et al.* (2022).

Para este fim, foi selecionado um modelo com uma arquitetura de rede neural fundamental, o Perceptron Multicamadas (MLP). A inclusão desse modelo permite não apenas avaliar a eficácia das redes LSTM, GRU e *Ensemble* mas também quantificar o ganho de desempenho em relação a redes neurais não recorrentes. A subseção a seguir detalha o funcionamento desse modelo.

3.4.2.1 Perceptron multicamadas (MLP)

O MLP é uma classe fundamental de rede neural artificial do tipo *feedforward*, servindo como base para muitas arquiteturas mais complexas em aprendizado profundo (Bishop, 2006; Goodfellow; Bengio; Courville, 2016). Sua estrutura é organizada em uma sequência de camadas de neurônios.

A camada de entrada (*Input Layer*), recebe os dados brutos que alimentam o modelo.

As camadas ocultas (*Hidden Layers*) realizam transformações lineares e não lineares nos dados. Cada neurônio aplica uma função de ativação (como Sigmoid ou ReLU) para capturar padrões complexos. Já a camada de saída (*Output Layer*) gera o resultado final da previsão.

O aprendizado em um MLP ocorre por meio do algoritmo de retropropagação (*back-propagation*), onde os pesos das conexões entre os neurônios são ajustados iterativamente para minimizar o erro entre as previsões do modelo e os valores reais (Goodfellow; Bengio; Courville, 2016).

A principal limitação de um MLP padrão no contexto de séries temporais é que ele trata cada entrada de forma independente, não possuindo um mecanismo inerente para reter informações de observações passadas. Essa falta de "memória temporal" o torna menos adequado para capturar as dependências sequenciais e dinâmicas presentes nos preços das ações, em contraste direto com as arquiteturas recorrentes como LSTM e GRU (Goodfellow; Bengio; Courville, 2016).

3.4.3 Modelos combinados (*Ensemble*)

Essas técnicas consistem na combinação de múltiplos modelos preditivos com o objetivo de aumentar a robustez e a acurácia das previsões. A ideia central é que diferentes modelos capturam distintos aspectos dos dados e, ao combiná-los, é possível reduzir a variância, o viés ou melhorar a estabilidade do sistema.

Existem diversas estratégias, como *bagging* (agregação de previsões por meio de amostras reamostradas do conjunto de treino), *boosting* (treinamento sequencial de modelos com foco em corrigir erros anteriores) e *stacking* (combinação hierárquica de modelos, geralmente com um modelo de nível superior para integrar as saídas). No contexto de redes neurais profundas, isso pode envolver a combinação de arquiteturas distintas — como LSTM e GRU — em uma única rede, como forma de potencializar os pontos fortes de cada abordagem.

A utilização de modelos combinados tem sido explorada com sucesso na literatura, especialmente em contextos com alta volatilidade ou estruturas temporais complexas, como os encontrados no mercado financeiro (Zhang; Patuwo; Hu, 2001).

3.5 ANÁLISE EXPLORATÓRIA DE DADOS (AED)

A análise exploratória de dados (AED) é uma etapa crítica no processo de análise de dados que permite uma compreensão profunda das características fundamentais dos dados, facilita a limpeza e preparação dos dados e ajuda a identificar padrões, anomalias e correlações (Anderson; Sweeney; Williams, 2002). Esta compreensão inicial é vital para a construção de modelos analíticos robustos, orienta a escolha das técnicas estatísticas a aplicar e melhora a comunicação dos resultados, tornando a AED indispensável para a tomada de decisões com base em dados confiáveis e precisos.

A AED constitui um pilar fundamental deste estudo, realizada antes do pré-processamento e do treinamento dos modelos. Esta fase é crucial, pois as percepções geradas informam diretamente as decisões de modelagem — como a necessidade de transformar dados ou a escolha de parâmetros — e fornecem um contexto indispensável para interpretar o desempenho de cada algoritmo. Há diversas formas de fazer essa etapa. Para análises do escopo deste trabalho, pode-se utilizar estatísticas descritivas, que são tabelas consolidadas com as principais métricas estatísticas para os preços de fechamento de todos os ativos. Os dados podem ser agrupados ainda em diferentes grupos (*clusters*). Estas métricas são:

- Média: oferece uma noção do nível central de preço de cada ativo no período.
- Desvio Padrão (std): serve como uma primeira medida da dispersão dos preços; valores mais altos indicam maiores flutuações e, consequentemente, maior risco histórico.
- Valores Mínimo (min) e Máximo (max): definem a faixa de negociação e destacam a magnitude dos eventos extremos ocorridos.

Também podem ser utilizadas diversas representações gráficas para avaliar o comportamento dos dados.

3.5.1 Alguns conceitos e termos

Grid Search É um método de busca exaustiva utilizado para encontrar a melhor combinação de hiperparâmetros em modelos de aprendizado de máquina. Consiste em definir um conjunto

finito de valores possíveis para cada hiperparâmetro e testar todas as combinações possíveis entre eles. Cada configuração é avaliada com base em uma métrica de desempenho previamente escolhida - neste trabalho, o MAPE (*Mean Average Percetual Error*), ou, Erro Percentual Médio, e o modelo com melhor resultado é selecionado. Apesar de ser computacionalmente custosa quando o espaço de busca é grande, a *Grid Search* garante uma cobertura sistemática das possibilidades e é especialmente útil quando o número de hiperparâmetros ou valores testados é reduzido. Essa abordagem oferece transparência e controle total sobre o processo de ajuste fino, contribuindo para a reprodutibilidade dos resultados (Probst; Wright; Boulesteix, 2019).

Overfitting* e *underfitting *Overfitting* ocorre quando um modelo de aprendizado de máquina se ajusta excessivamente aos dados de treinamento, capturando ruídos e variações específicas que não generalizam bem para novos dados, resultando em um desempenho ruim no conjunto de teste. Por outro lado, *underfitting* acontece quando o modelo é muito simples para capturar os padrões subjacentes dos dados, apresentando desempenho insatisfatório tanto nos dados de treinamento quanto nos dados de teste. Ambos os fenômenos comprometem a capacidade do modelo de generalizar, sendo essencial encontrar um equilíbrio adequado por meio de técnicas como regularização, validação cruzada e ajuste de hiperparâmetros para obter um modelo robusto e eficaz (Bishop, 2006)

***Dropout*:** É uma técnica de regularização utilizada em redes neurais para reduzir o *overfitting*, ou seja, o problema do modelo memorizar excessivamente o conjunto de treinamento em vez de aprender padrões generalizáveis. Essa técnica foi introduzida por Srivastava *et al.* (2014) como uma abordagem simples e eficaz para melhorar a capacidade de generalização das redes neurais.

Durante o treinamento, o *dropout* desativa (ou "desliga") aleatoriamente uma fração dos neurônios em cada camada da rede a cada passo de atualização dos pesos. Essa fração é determinada pela taxa de *dropout*, por exemplo, 0,2 ou 0,5, indicando que 20% ou 50% dos neurônios serão desativados aleatoriamente. Quando um neurônio é desativado, ele não contribui para o cálculo da saída nem para o processo de retropropagação do erro naquele ciclo de treinamento (Goodfellow; Bengio; Courville, 2016).

Vazamento de dados (*data leakage*) Vazamento de dados ocorre quando informações do conjunto de teste são indevidamente utilizadas durante o processo de treinamento do modelo,

comprometendo a avaliação realista do desempenho do modelo. Em outras palavras, o modelo acaba aprendendo padrões que, na prática, só estariam disponíveis no momento da avaliação, o que reduz sua capacidade de generalização. Para evitar esse problema, é essencial que as transformações de pré-processamento sejam calibradas exclusivamente com os dados de treinamento e aplicadas posteriormente ao conjunto de teste (Kaufman *et al.*, 2012).

***Epochs* e o processo de treinamento** Um *epoch* representa uma passagem completa por todo o conjunto de dados de treinamento durante o processo de aprendizado do modelo. Em cada *epoch*, o modelo processa todas as amostras de dados, ajustando seus pesos com base no erro encontrado em cada amostra, com o objetivo de minimizar a função de perda. Geralmente, várias *epochs* são necessárias para que o modelo aprenda padrões complexos nos dados e melhore seu desempenho ao longo do tempo. Durante a fase de treinamento, o modelo utiliza os dados de treinamento para ajustar seus parâmetros (pesos e viés), minimizando uma função de perda, que mede o erro entre as previsões do modelo e os valores reais. Esse ajuste é realizado iterativamente ao longo de vários *epochs*, refinando o modelo para capturar padrões nos dados (Goodfellow; Bengio; Courville, 2016).

Early Stopping *Early Stopping* é uma técnica utilizada durante o treinamento de modelos de aprendizado de máquina para prevenir o *overfitting*. Consiste em monitorar o desempenho do modelo em um conjunto de validação ao longo das *epochs* e interromper o treinamento assim que o erro no conjunto de validação começar a aumentar ou deixar de melhorar, indicando que o modelo está começando a se ajustar demais aos dados de treinamento. Dessa forma, o *Early Stopping* ajuda a encontrar um ponto ótimo de treinamento, equilibrando a aprendizagem suficiente dos padrões dos dados e evitando a perda de capacidade de generalização para dados novos (Goodfellow; Bengio; Courville, 2016).

Dependência de Longo Prazo (*Long-Term Dependency*) em Previsão de Séries Temporais: Na previsão de séries temporais, valores futuros podem depender de padrões formados meses ou anos antes — como ciclos econômicos, sazonalidades anuais ou choques exógenos raros. Capturar essas relações distantes é desafiador, pois, durante o treinamento de redes neurais recorrentes, o gradiente — que é a medida matemática usada para ajustar os pesos do modelo por meio do algoritmo de retropropagação — tende a se dissipar (tornar-se muito pequeno) ou explodir (crescer exageradamente) quando propagado por muitas etapas temporais. Esses

fenômenos, conhecidos como *vanishing* e *exploding gradients*, dificultam o aprendizado de dependências de longo alcance. Embora arquiteturas recorrentes avançadas, como as LSTM, atenuem parcialmente esse problema ao manter retenção de contexto — isto é, a capacidade de preservar, ao longo do processamento, informações relevantes sobre eventos passados que podem influenciar previsões futuras, ainda se investigam estratégias para ampliar essa preservação de informação e, assim, melhorar a acurácia em horizontes de previsão extensos (Bengio; Simard; Frasconi, 1994).

Função de Perda (*Loss Function*): A função de perda quantifica a discrepância entre as previsões do modelo e os valores reais, atuando como uma medida de quão bem o modelo está performando. Em aprendizado profundo, o objetivo é minimizar essa função ajustando os parâmetros da rede para reduzir o erro de previsão ao longo das *epochs* (Goodfellow; Bengio; Courville, 2016).

O acompanhamento do valor da função de perda ao longo das *epochs* é uma prática comum para monitorar o treinamento, permitindo detectar problemas como o *overfitting*. Este fenômeno ocorre quando a perda no conjunto de validação começa a aumentar, enquanto a perda no conjunto de treinamento continua diminuindo (Bishop, 2006).

Diferença entre Parâmetros e Hiperparâmetros: Em aprendizado profundo, é essencial compreender a diferença entre parâmetros e hiperparâmetros. Os parâmetros são os valores aprendidos pelo modelo durante o treinamento, como pesos e vieses, ajustados pelo modelo com o objetivo de minimizar a função de perda e otimizar o desempenho do modelo (Goodfellow; Bengio; Courville, 2016).

Por outro lado, os hiperparâmetros são definidos manualmente pelo pesquisador antes do início do treinamento e controlam o comportamento do processo de aprendizado. Exemplos incluem a taxa de aprendizado, o número de camadas, o número de neurônios por camada, o tamanho dos lotes (*batch size*) e a taxa de *dropout*.

A escolha adequada dos hiperparâmetros é fundamental para alcançar um bom desempenho do modelo. Para otimizar esses valores, técnicas como a busca em grade (*grid search*) e a busca aleatória (*random search*) são amplamente empregadas (Bergstra; Bengio, 2012).

Neste trabalho, inicialmente foram utilizados hiperparâmetros de referência e, posteriormente, foi conduzido um ajuste fino desses hiperparâmetros utilizando a técnica de *random search*, comparando-se os resultados obtidos nas fases estudadas (fase 1 e 2).

3.6 METODOLOGIAS E DADOS DOS ESTUDOS RELACIONADOS

A previsão de preços de ações com aprendizado profundo tem recebido ampla atenção nos últimos anos, com diversos modelos sendo aplicados a diferentes mercados e horizontes.

Rahmadeyan e Mustakim (2024) analisaram ações do *Bank Rakyat Indonesia* (BBRI.JK) com cotações diárias de 1 jan 2018 a 30 dez 2022, obtidas via YAHOO!, utilizando arquiteturas de LSTM e GRU, com métricas-chave de RMSE e MAPE. O trabalho é estruturado em janelas de 10, 20 e 30 passos antes de treinar LSTM e GRU sob diferentes combinações de hiperparâmetros. O melhor desempenho surgiu com GRU, atingindo um MAPE de 1,17%, superando consistentemente o LSTM em todos os cenários testados.

Winardi *et al.* (2023) constroem um *pipeline* em quatro etapas do papel de níquel (ANTM. JK) no mercado indonésio, utilizando dados do YAHOO!, e fazendo a predição de valores com uma rede GRU multicamada, obtendo resultados como MAPE de 2,94%.

Já Selvin *et al.* (2017) investigaram predição de preços de ações em alta frequência, recorrendo a três arquiteturas profundas — RNN (Redes Neurais Recorrentes), LSTM e CNN (Redes Neurais Convolucionais). O estudo parte de uma base de dados de minuto a minuto de papéis da Bolsa Nacional da Índia (NSE) no intervalo de Julho de 2014 a Junho de 2015. Concentram-se em duas empresas de TI (Infosys e TCS) e uma do setor farmacêutico (Cipla) e utilizaram como *benchmark* um método estatístico tradicional, o ARIMA (*AutoRegressive Integrated Moving Average*), método estatístico clássico muito utilizado para previsão de séries temporais, cujo erro percentual ultrapassou 30% em algumas séries.

Fischer e Krauss (2018) empregaram redes LSTM para prever o movimento direcional diário das ações do S&P 500 (mercado dos EUA) entre 1992 e 2015. Nesse estudo, as LSTM superaram claramente classificadores sem memória como *Random Forest*, redes neurais profundas convencionais (DNNs), muito utilizadas no processamento de imagens, e regressão logística, obtendo retornos diários médios de 0,46% – desempenho significativamente superior ao de modelos-base e ao desempenho do próprio índice S&P 500 na maior parte do período analisado.

Khanpuri, Darapaneni e Paduri (2024) criaram um pipeline que faz uso de métricas contábeis, como demonstrações de fluxo de caixa e preços históricos mensais de quatro blue-chips indianas (Reliance, TCS, ITC e Tata Motors). Estes atributos alimentam um LSTM de duas camadas para prever o preço médio de fechamento dos ativos nos 6 meses seguintes. Este

artigo também faz uso do MAPE como métrica chave, avalia ações em um mercado emergente e faz uso de uma rede neural LSTM para a predição.

De forma geral, a Tabela 1 apresenta uma síntese comparativa dos principais trabalhos utilizados como referência neste estudo, permitindo identificar contextos de aplicação, metodologias empregadas, métricas de avaliação e pontos de convergência ou divergência em relação à presente pesquisa.

Tabela 1 – Resumo comparativo de estudos similares de trabalhos sobre previsão de preços de ações com redes neurais

Estudo (ano)	Mercado período granularidade	Modelos testados	Melhor resultado	Semelhanças
Selvin *et al.* (2017)	NSE Índia; jul. 2014–jun. 2015; 1 min	RNN, LSTM, CNN, ARIMA	CNN: MAPE 2,36 % (Infosys); DL $< 5\,\%$	Arquiteturas DL diversas, janelas fixas, uso de MAPE
Rahmadeyan & Mustakim (2024)	BBRI.JK; 2018–2022; diário	LSTM vs. GRU (grid de hiperparâmetros)	GRU + RMSprop, janela 30: MAPE 1,17 %	Horizonte diário, comparação LSTM/GRU, teste de tamanho de janela
Winardi *et al.* (2023)	4 blue-chips IDX; 2016–2022; diário	GRU (janelas 15–90)	Janela 15: MAPE 2,94 %	GRU como núcleo, análise de múltiplas janelas e robustez
Fischer & Krauss (2018)	S&P 500; 1992–2015; diário	LSTM, Logit, Random Forest	LSTM: retorno 0,46 %/dia; Sharpe 5,8	LSTM em série diária, avaliação de ganhos econômicos
Khanapuri *et al.* (2024)	Blue-chips indianas; 2003–2023; mensal/diário + fundamentos	LSTM com *feature fusion*	Relata MAE/RMSE (sem MAPE)	Combina indicadores técnicos e fundamentalistas
Este trabalho	16 ações B3; 2016–2024; diário	MLP, LSTM, GRU	—	—

Fonte: Autor (2025).

4 MATERIAL E MÉTODOS

Nesta seção detalha-se o processo metodológico usado durante a pesquisa, explicando desde a seleção e preparação dos dados até a construção, treinamento e avaliação dos modelos. Esse processo foi dividido em duas fases experimentais:

- **Fase 1 - Modelagem com Hiperparâmetros Padrão:** O objetivo desta fase foi estabelecer uma linha de base (*baseline*) de desempenho, implementando e comparando os modelos propostos com uma arquitetura e hiperparâmetros fixos, comumente utilizados na literatura.
- **Fase 2 - Otimização de Hiperparâmetros e Modelagem Ajustada:** A segunda fase focou em refinar os modelos através de um processo de busca automatizada pelos melhores hiperparâmetros que otimizem o poder preditivo para cada ativo individualmente.

4.1 SOBRE A ESCOLHA DOS MODELOS DE PREDIÇÃO

Nesta pesquisa foram utilizados modelos de redes neurais recorrentes (RNNs), como LSTM e GRU, devido à sua capacidade de lidar com séries temporais financeiras que apresentam alta volatilidade e dependências temporais de curto e longo prazo.

Para comparar esses modelos, foi definida como *baseline* uma rede neural fundamental, o Perceptron Multicamadas (MLP). Esse modelo não é capaz de processar as relações entre diferentes entradas ao longo do tempo (Goodfellow; Bengio; Courville, 2016).

A escolha do modelo MLP como *baseline* se justifica por sua natureza intermediária entre abordagens estatísticas tradicionais e redes neurais mais complexas. Enquanto modelos clássicos como ARIMA ou Prophet apresentam excelente desempenho em séries temporais com comportamento linear e padrões sazonais bem definidos, sua capacidade de capturar relações não lineares e dinâmicas de alta frequência é limitada, o que os torna menos adequados para séries financeiras caracterizadas por volatilidade, rupturas estruturais e comportamento estocástico (Hyndman; Athanasopoulos, 2018).

A expectativa é que os arquiteturas especializadas como LSTM e GRU demonstrem um desempenho melhor na identificação de padrões, assim demostrando os benefícios de sua

aplicação em séries financeiras complexas.

4.2 FERRAMENTAS, BIBLIOTECAS, FRAMEWORKS E BANCO DE DADOS UTILIZADOS

Este trabalho utilizou diferentes ferramentas, bibliotecas e bancos de dados que permitiram a manipulação de grandes volumes de dados e a construção dos modelos de aprendizado de máquina e elas estão listadas na Tabela 10, no Apêndice A, com seus nomes, usos e versões.

A linguagem de programação Python foi escolhida como base para o desenvolvimento e execução dos experimentos deste estudo devido à sua ampla adoção na comunidade científica, versatilidade e integração com bibliotecas especializadas em análise de dados e aprendizado profundo. Seu ecossistema maduro e de código aberto, aliado à simplicidade sintática e extensa documentação, permite a implementação eficiente de modelos complexos, desde o pré-processamento de dados até a avaliação estatística dos resultados (Rossum; Drake Jr., 2010). Além disso, o Python possui bibliotecas consolidadas, como *NumPy*, *Pandas*, *TensorFlow* e *Scikit-learn*, que oferecem suporte otimizado para operações matriciais e computação em GPU, tornando-o uma escolha natural para aplicações em aprendizado de máquina e séries temporais (Oliphant, 2007).

O código utilizado, bem como o repositório do github onde estará disponível, pode ser encontrado no Anexo B.

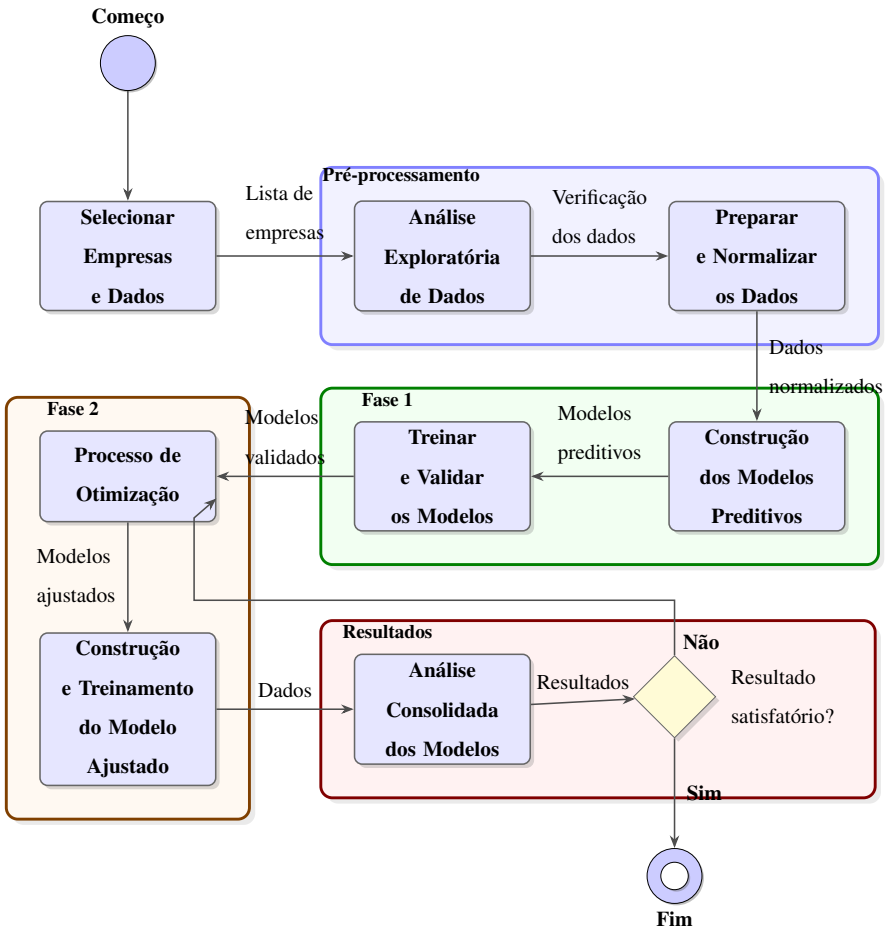
Os experimentos foram conduzidos em um notebook *Dell Vostro 15 3515*, equipado com processador *AMD Ryzen 5 3450U* com gráficos integrados *Radeon Vega Mobile Gfx* (2.10 GHz), 16 GB de memória RAM e sistema operacional *Windows 11 Pro*, versão 24H2. Essa configuração foi suficiente para realizar o treinamento dos modelos recorrentes em tempo hábil, garantindo desempenho estável mesmo em tarefas de otimização de hiperparâmetros.

4.3 *FRAMEWORK* PROPOSTO

O *framework* proposto neste estudo adota uma abordagem estruturada e sequencial para a construção e avaliação de modelos de previsão de preços de ações. Ele abrange a seleção e

a análise exploratória dos dados, a normalização, a construção e ajuste dos modelos preditivos, bem como a avaliação de desempenho. Esse *framework* foi definido para fazer uma análise que permita tanto a comparação entre diferentes arquiteturas de redes neurais quanto uma otimização de hiperparâmetros que melhore as previsões. A Figura 12 apresenta o fluxograma da sequência adotada e cada uma das etapas são detalhadas ao longo dessa seção.

Figura 12 – Fluxograma das etapas realizadas no processo de construção e avaliação dos modelos de previsão.



Fonte: Autor (2025).

4.3.1 Seleção de empresas e dados

Foram escolhidas 16 empresas de alta liquidez na B3, distribuídas em quatro setores econômicos distintos — Varejo, Bancário, Energia e Tecnologia — de forma a capturar diferentes dinâmicas do mercado de capitais brasileiro. A diversidade setorial permitiu testar os modelos em cenários com diferentes níveis de volatilidade e estabilidade, avaliando sua adap-

tação e precisão em condições variadas do mercado brasileiro. Foram utilizados dados diários de Abertura, Máxima, Mínima, Fechamento e Volume, entre 1º de janeiro de 2021 e 31 de dezembro de 2024, com o preço de fechamento (*Close*) como variável-alvo ou variável a ser predita. A seleção das empresas seguiu critérios técnicos de representatividade setorial, liquidez e disponibilidade de dados históricos. Buscou-se contemplar companhias pertencentes a setores economicamente relevantes e com comportamentos distintos em termos de risco e volatilidade, de forma a permitir uma análise comparativa abrangente entre diferentes dinâmicas de mercado. Foram, portanto, selecionadas empresas de quatro setores da economia brasileira — Varejo, Bancário, Energia e Tecnologia — que, em conjunto, oferecem um panorama diversificado do mercado acionário nacional.

Dentro de cada setor, priorizaram-se empresas com elevada liquidez na B3, observada pelo volume médio diário de negociações e pela presença recorrente no índice Ibovespa durante o período de 2021 a 2024. Essa escolha assegura o uso de ativos amplamente negociados, reduzindo o impacto de ruídos associados à baixa liquidez e contribuindo para uma maior estabilidade estatística dos modelos de previsão. Além disso, a seleção restringiu-se a companhias com dados completos e consistentes disponíveis na plataforma *Yahoo Finance*, garantindo reprodutibilidade e qualidade das séries temporais utilizadas. Essa combinação de critérios qualitativos e quantitativos resultou em um portfólio representativo e robusto para os experimentos realizados.

4.3.2 Pré-processamento e análise exploratória

Os passos da preparação dos dados estão listados na sequência que foram organizados e executados segundo a Figura 12.

Análise Exploratória de Dados (AED)

A Análise Exploratória de Dados (AED) é feita para entender as principais características dos dados, verificar sua qualidade e identificar alguns padrões que possam facilitar ou prejudicar a modelagem (Seward; Doane, 2014). As seguintes análises foram realizadas para garantir que os dados estivessem consistentes e adequados para as fases subsequentes do *framework*.:

- **Análise Visual:** Usando gráficos de séries temporais para os preços de fechamento e para

a volatilidade. Eles permitiram uma inspeção da dispersão dos dados e das diferenças de comportamento entre os ativos.

- **Análise de Volatilidade:** Foram calculadas medidas de volatilidade para contextualizar os erros de previsão em função do risco dos ativos. Essa análise também serviu como um "teste de estresse" para avaliar a robustez dos modelos em cenários de baixa e alta instabilidade, destacando sua capacidade de generalização.

Divisão dos Dados

A série de preços de fechamento de cada ativo foi dividida em três subconjuntos distintos — treinamento, validação e teste — respeitando rigorosamente a ordem temporal dos dados. Essa divisão sequencial garante que as observações mais recentes sejam utilizadas apenas na etapa de teste, preservando o princípio de causalidade e evitando qualquer tipo de vazamento de informação entre os conjuntos (*data leakage*). Em outras palavras, o modelo é treinado apenas com dados do passado e avaliado sobre dados futuros, simulando o processo real de previsão. As proporções adotadas (70% para treinamento, 15% para validação e 15% para teste). Essa proporção foi definida com base em práticas consolidadas na literatura de aprendizado profundo aplicado a séries temporais financeiras, frequentemente adotada em estudos similares, pois assegura um equilíbrio adequado entre a quantidade de dados disponível para o ajuste dos parâmetros do modelo e a reserva de observações independentes para avaliação da generalização (Selvin *et al.*, 2017; Fischer; Krauss, 2018). O conjunto de validação é utilizado para monitorar o desempenho durante o treinamento e prevenir o sobreajuste (*overfitting*), enquanto o conjunto de teste permite uma mensuração imparcial da capacidade preditiva final dos modelos.

Divisão e Normalização dos Dados

A normalização dos dados garante que as variáveis tenham uma influência equilibrada no treinamento. Utilizou-se o *MinMaxScaler* da biblioteca `scikit-learn`, que transforma os valores para o intervalo $[0, 1]$, conforme a fórmula:

$$Y_{\text{normalizado}} = \frac{Y - \min(Y)}{\max(Y) - \min(Y)} \tag{4.1}$$

em que Y denota os valores originais, e $\min(Y)$ e $\max(Y)$ são, respectivamente, os menores e maiores valores da série.

Para evitar vazamento de dados (*data leakage*), o `scaler` foi ajustado (`fit`) apenas sobre o conjunto de treinamento. A mesma transformação (`transform`) foi então aplicada aos conjuntos de validação e teste.

4.4 METODOLOGIA EXPERIMENTAL

O desenvolvimento deste estudo foi estruturado em duas fases, seguidas por uma análise para avaliação de desempenho. O objetivo foi definir uma comparação justa entre o *baseline* e os modelos de aprendizado propostos, e quantificar o impacto da otimização de hiperparâmetros.

4.5 FASE 1: MODELAGEM COM HIPERPARÂMETROS PADRÃO

Nesta fase inicial, todos os modelos foram construídos e treinados com uma arquitetura e hiperparâmetros fixos. Esses hiperparâmetros empregados na Fase 1 foram definidos a partir de valores de referência amplamente utilizados em estudos aplicados à previsão de séries temporais financeiras. A arquitetura adotada, composta por duas camadas recorrentes com 50 unidades cada, segue a prática recomendada na literatura (Selvin *et al.*, 2017; Fischer; Krauss, 2018), pois representa um equilíbrio entre redes muito simples, que podem sofrer de *underfitting*, e redes excessivamente complexas, mais suscetíveis a *overfitting*. A taxa de *dropout* de 0,2 foi escolhida por ser um valor intermediário, frequentemente sugerido para problemas de alta volatilidade, de modo a reduzir o risco de sobreajuste sem comprometer a capacidade de aprendizado (Srivastava *et al.*, 2014). O otimizador Adam foi utilizado com taxa de aprendizado inicial de 0,001, valor padrão em implementações de *deep learning* pela sua estabilidade e boa convergência em problemas não lineares. Além disso, o tamanho da janela temporal foi fixado em 30 dias úteis, aproximadamente equivalente a um mês de pregão da B3, de forma a capturar dependências de curto prazo. O treinamento foi conduzido por até 100 épocas, com parada antecipada (*early stopping*) quando não havia melhora no conjunto de validação, prevenindo o sobreajuste e garantindo maior capacidade de generalização.

4.5.1 Construção dos modelos de referência (*Baseline*)

Como *baseline* de rede neural, foi utilizada uma MLP (Perceptron Multicamadas). O modelo foi construído com duas camadas ocultas (`Dense` com 64 e 32 unidades e ativação 'relu'), seguidas por camadas de `Dropout` para regularização. A inclusão do MLP permite avaliar especificamente o ganho de desempenho obtido pela "memória" e pela capacidade sequencial das arquiteturas recorrentes - GRU e LSTM (Nelson; Pereira; Oliveira, 2017; Shen; Shafiq, 2020).

4.5.2 Construção dos modelos propostos

Os modelos de redes recorrentes foram definidos com uma arquitetura idêntica de duas camadas:

Modelos GRU e LSTM Ambos os modelos foram construídos com:

- Camada 1: Uma camada (GRU ou LSTM) com **50 unidades** e `return_sequences=True`.
- Regularização: Uma camada de `Dropout` com taxa de 0,2
- Camada 2: Uma segunda camada (GRU ou LSTM) com 50 unidades
- Regularização: Uma segunda camada de `Dropout` com taxa de 0,2
- Camada de Saída: Uma camada `Dense` com 1 unidade para a predição.

Modelo Combinado (*Ensemble*) Este modelo não passa por um processo de treino próprio. Sua predição é calculada, pós-treinamento, como a média aritmética simples das previsões geradas pelos modelos GRU e LSTM individuais desta fase.

4.6 FASE 2: OTIMIZAÇÃO DE HIPERPARÂMETROS VIA *GRID SEARCH*

Visando maximizar a performance dos modelos recorrentes, a Fase 2 introduziu um processo de ajuste fino de hiperparâmetros. Diferentemente de uma busca aleatória, optou-se pela

técnica de Busca em Grade (*Grid Search*), que testa exaustivamente todas as combinações de um conjunto pré-definido de hiperparâmetros. Essa abordagem implica um aumento significativo no custo computacional, uma vez que múltiplos modelos são treinados e avaliados para cada ativo.

Para cada um dos 16 ativos, um processo de busca em grade independente foi executado tanto para o modelo GRU quanto para o LSTM, a fim de encontrar a combinação de hiperparâmetros que minimizasse o Erro Percentual Absoluto Médio (MAPE) no conjunto de teste. O espaço de busca explorado foi:

- Unidades por Camada: {32, 50}.
- Taxa de Dropout: {0,1, 0,2}.
- Taxa de Aprendizagem (Learning Rate): {0,001, 0,0005}.

Ao final do processo, o modelo com a melhor combinação de parâmetros para cada ação foi salvo e utilizado como a versão "otimizada"para a análise de resultados.

4.7 TREINAMENTO E ESTRUTURA DOS DADOS

Para todos os modelos neurais (MLP, GRU, LSTM) em ambas as fases, o treinamento foi conduzido de forma consistente. Os modelos foram compilados com o otimizador Adam e a função de perda de Erro Quadrático Médio (MSE). O treinamento foi executado por um máximo de 100 épocas (100 passagens completas pela base de dados de treinamento), com a utilização da técnica de Parada Antecipada (*Early Stopping*) com 10 épocas antes do final previsto para evitar sobreajuste (*overfitting*) e garantir a generalização.

Os dados foram estruturados em janelas deslizantes com um tamanho de sequência (`SEQ_LENGTH`) fixo de 30 dias. A escolha do tamanho da janela deslizante influencia diretamente a capacidade do modelo em capturar padrões temporais relevantes. Uma janela menor tende a privilegiar relações de curto prazo, tornando o modelo mais sensível a oscilações recentes, mas também mais suscetível ao ruído e à perda de contexto histórico. Em contrapartida, o aumento do tamanho da janela amplia o horizonte de observação e possibilita ao modelo identificar tendências e dependências de longo prazo, porém eleva o custo computacional e o risco de sobreajuste, uma vez que o modelo precisa processar um maior número de parâmetros

e relações redundantes (Goodfellow; Bengio; Courville, 2016). Assim, a definição de 30 dias representa um compromisso entre capturar padrões mensais típicos do mercado e manter a eficiência computacional e a capacidade de generalização dos modelos. Para prevenir o vazamento de dados (*data leakage*), a divisão em conjuntos de treino (70%), validação (15%) e teste (15%) respeitou rigorosamente a ordem cronológica dos dados.

4.8 CRITÉRIOS DE AVALIAÇÃO

A avaliação final de todos os modelos foi realizada no conjunto de teste, que representa 15% dos dados mais recentes e não foi utilizado em nenhuma etapa de treinamento ou otimização. O desempenho foi medido quantitativamente por três métricas principais:

- **Raiz do Erro Quadrático Médio (RMSE):** Mede a magnitude média dos erros, na mesma unidade do preço da ação (R$).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{4.2}$$

- **Erro Percentual Absoluto Médio (MAPE):** Expressa o erro médio como uma porcentagem, facilitando a comparação entre ativos de diferentes escalas.

$$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{4.3}$$

- **Coeficiente de Determinação (R^2):** Indica a proporção da variância na série de preços explicada pelo modelo.

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2} \tag{4.4}$$

em que:

n é o número total de observações no conjunto avaliado; y_t é o valor real observado no instante temporal t; $\hat{y}_t$ é o valor previsto pelo modelo no instante temporal t e t é o índice que percorre cada observação da série, de 1 a n, que, neste caso, seriam os últimos 15% dos dados.

A escolha das métricas de avaliação adotadas neste estudo está em conformidade com recomendações amplamente difundidas na literatura sobre previsão de séries temporais finan-

ceiras. O MAPE é amplamente utilizado em estudos de previsão de preços de ações por fornecer uma medida interpretável em termos percentuais e permitir comparações diretas entre ativos de diferentes escalas (Selvin *et al.*, 2017; Rahmadeyan; Mustakim, 2024). A RMSE é reconhecida por sua sensibilidade a grandes desvios, sendo útil para identificar erros mais severos e penalizar previsões distantes dos valores observados (Fischer; Krauss, 2018; Hyndman; Koehler, 2006). Já o (R^2) fornece uma medida intuitiva da proporção da variabilidade explicada pelo modelo, sendo tradicionalmente empregado como indicador de qualidade de ajuste em modelos preditivos (Kutner *et al.*, 2005; Goodfellow; Bengio; Courville, 2016). A combinação dessas três métricas permite uma avaliação abrangente da acurácia, robustez e poder explicativo dos modelos implementados.

4.9 COMPARAÇÃO ESTATÍSTICA DE MODELOS

A comparação de algoritmos de aprendizado de máquina sobre múltiplos conjuntos de dados exige procedimentos estatísticos adequados. Segundo Demsar (2006), a abordagem recomendada é utilizar testes não paramétricos baseados em postos, uma vez que os pressupostos de normalidade e homocedasticidade dificilmente são atendidos em métricas de erro de modelos preditivos.

O *teste de Friedman* (Friedman, 1937) é uma alternativa não paramétrica à ANOVA de medidas repetidas, apropriada para situações em que vários algoritmos são comparados sobre múltiplas bases ou blocos (neste estudo, as séries de ações). Esse teste verifica a hipótese nula de que todos os algoritmos possuem desempenho equivalente em termos de suas distribuições de postos. Valores de p inferiores ao nível de significância indicam que pelo menos um modelo difere dos demais.

Caso os pressupostos de normalidade e homocedasticidade fossem atendidos, poderia ser empregada a ANOVA de medidas repetidas (Repeated Measures ANOVA), que opera sobre as médias e variâncias dos grupos, assumindo a distribuição normal das diferenças. Em contrapartida, se o número de modelos ou a estrutura dos dados violasse as condições de balanceamento necessárias ao Friedman, outras abordagens não paramétricas poderiam ser consideradas, como o teste de Quade, que incorpora pesos de importância para cada bloco, ou ainda o teste de *Aligned Rank Friedman*, que oferece maior sensibilidade em cenários com interações

entre fatores (García *et al.*, 2010).

Quando o teste de Friedman rejeita a hipótese nula, torna-se necessário aplicar testes *post-hoc* para identificar quais pares de algoritmos apresentam diferenças significativas. Uma das opções mais difundidas é o teste de Nemenyi (Nemenyi, 1963), proposto originalmente para comparações múltiplas em experimentos balanceados. O teste baseia-se na diferença crítica (CD, do inglês *critical difference*) entre as médias dos postos: sempre que a diferença entre dois modelos excede a CD, conclui-se que há diferença estatisticamente significativa entre eles. Sua aplicação é especialmente adequada em cenários com múltiplas comparações e tamanho amostral reduzido por bloco, pois controla a taxa global de erro do tipo I sem necessidade de correções adicionais (García *et al.*, 2010).

Estudos recentes em aprendizado de máquina e mineração de dados confirmam a adequação do par Friedman–Nemenyi para a avaliação comparativa de modelos. García *et al.* (2010) discutem extensões e alternativas, destacando que o teste de Nemenyi é conservador, mas amplamente aceito em áreas como bioinformática, reconhecimento de padrões e finanças.

Outras opções, como os testes de Holm ou Bergmann–Hommel, oferecem maior poder estatístico, mas implicam suposições ou etapas adicionais de ajuste sequencial (Holm, 1979). Neste trabalho, optou-se pelo Nemenyi por sua simplicidade, robustez e ampla aceitação na literatura de comparação de classificadores.

Além dos testes de Friedman e Nemenyi utilizados para comparar o desempenho entre diferentes modelos, foi aplicado o teste de Wilcoxon para avaliar as diferenças de desempenho entre as Fases 1 e 2 do estudo. O teste de Wilcoxon é um método não paramétrico para amostras pareadas, apropriado quando não se pode assumir que as diferenças entre pares seguem uma distribuição normal. Essa escolha se justifica porque as métricas de erro utilizadas, como o MAPE, tendem a apresentar assimetria e *outliers*, o que viola os pressupostos de normalidade exigidos pelo teste t pareado.

Enquanto o teste t avalia médias sob a suposição de distribuição normal das diferenças, o teste de Wilcoxon baseia-se apenas nas ordens e sinais das diferenças, tornando-se mais robusto em amostras pequenas ou com distribuições não gaussianas (Wilcoxon, 1945; Demsar, 2006). Assim, sua utilização garante uma comparação mais confiável entre as fases de modelagem, mantendo a validade estatística mesmo na presença de dispersão elevada nos erros individuais.

Em síntese, o pipeline estatístico adotado compreendeu três etapas principais: (i) a aplicação do teste de Wilcoxon para comparar as distribuições de desempenho entre as Fases 1 e

2; (ii) a aplicação do teste de Friedman, utilizado para identificar diferenças globais de desempenho entre os modelos e (iii) a realização do teste *post-hoc* de Nemenyi, conduzido sempre que o resultado do Friedman indicou significância estatística ($p < 0{,}05$), fornecendo uma base metodológica consistente para avaliar a existência de diferenças significativas e identificar, com confiança, quais modelos e configurações apresentaram o melhor desempenho geral.

5 RESULTADOS E DISCUSSÃO

Essa seção sistematiza os resultados obtidos a partir da aplicação da metodologia proposta. Contudo, nem todas as fases do procedimento demandam, por natureza, uma análise dos resultados. Algumas etapas – por exemplo, a normalização das variáveis ou a divisão temporal em conjuntos de treino, validação e teste – são estritamente operacionais; seu propósito é adequar os dados ou viabilizar a estimação dos modelos, não gerar métricas ou gráficos que mereçam discussão própria. Assim, os resultados aqui apresentados concentram-se apenas nas etapas em que há produção efetiva de evidências quantitativas ou qualitativas que permitem inferir desempenho, comparar alternativas ou extrair conclusões.

5.1 SELEÇÃO DE EMPRESAS E DADOS

A seleção dos ativos para este estudo foi pautada em critérios que visam garantir a eficácia do estudo e a generalização das conclusões. A carteira de 16 empresas, dividida em 4 setores, foi projetada para representar diferentes dinâmicas do mercado de capitais brasileiro e testar os modelos em cenários variados, conforme detalhado abaixo.

O critério fundamental foi a representação de setores distintos e importantes da economia brasileira, capturando diferentes comportamentos de mercado:

- **Varejo:** Setor cíclico e volátil, sensível a indicadores de confiança, juros e sazonalidades.
- **Bancário:** Pilar da economia, influenciado pela política macroeconômica e pela taxa Selic.
- **Energia:** Setor defensivo e resiliente, com receitas mais previsíveis, mas sujeito a fatores regulatórios e volatilidade de *commodities*.
- **Tecnologia:** Caracterizado pelo alto potencial de crescimento e maior volatilidade.

A tabela 2 demonstra as empresas selecionadas, separadas por setor e com seu *ticker* (código identificador da empresa na bolsa).

Tabela 2 – Ativos Selecionados para o Estudo, Agrupados por Setor

Setor	Nome da Empresa	*Ticker*
Varejo	Magazine Luiza	MGLU3.SA
	Lojas Renner	LREN3.SA
	Pão de Açúcar	PCAR3.SA
	Casas Bahia	BHIA3.SA
Bancário	Itaúsa	ITSA3.SA
	Banco do Brasil	BBAS3.SA
	Porto Seguro	PSSA3.SA
	Banco ABC	ABCB4.SA
Energia	Eletrobras	ELET3.SA
	Petrobras	PETR3.SA
	Equatorial	EQTL3.SA
	Taesa	TAEE11.SA
Tecnologia	TOTVS	TOTS3.SA
	Positivo Tec.	POSI3.SA
	Intelbras	INTB3.SA
	Bemobi	BMOB3.SA

Fonte: Autor (2025).

Todos os ativos selecionados possuem alta liquidez na B3, sendo a maioria componentes do índice IBOVESPA. Este critério garante a disponibilidade de dados históricos de qualidade para o treinamento dos modelos e confere maior relevância prática à análise.

A combinação dos critérios resulta em um portfólio com características distintas. No período analisado (2021-2024), os ativos apresentaram diferentes tendências, níveis de volatilidade e padrões sazonais, formando um conjunto de dados abrangente para testar os modelos. Em resumo, a carteira de ativos foi montada para criar um campo de testes desafiador e realista. A diversidade setorial e de comportamento de preços permite uma avaliação completa da capacidade de adaptação e precisão de cada algoritmo, fortalecendo as conclusões sobre sua performance em diferentes condições de mercado.

5.2 ANÁLISE EXPLORATÓRIA DE DADOS (AED)

Esta análise quantitativa permite uma comparação objetiva da escala e da variabilidade de cada ativo e setor.

Tabela 3 – Estatísticas Descritivas dos Preços, em reais, de Fechamento, Agrupadas por Setor entre 2021 e 2024

Setor	Ticker	n	média	dp	min	Q25	Q50	Q75	max	{CV (%)
Bancário	ABCB4.SA	996	15,9	3,8	10,4	12,5	15,4	19,5	23,5	23,9
	BBAS3.SA	996	18,0	5,6	10,2	12,7	16,6	24,3	27,9	31,1
	ITSA3.SA	996	8,0	0,8	6,4	7,4	7,9	8,6	9,9	10,0
	PSSA3.SA	996	24,0	5,8	15,3	19,7	22,4	27,4	40,2	24,2
Energia	ELET3.SA	996	36,2	4,4	23,4	33,0	36,2	39,6	48,4	12,2
	EQTL3.SA	996	27,1	4,3	18,4	23,4	26,3	31,3	35,0	15,9
	PETR3.SA	996	21,4	10,2	6,8	12,4	18,9	31,0	40,3	47,7
	TAEE11.SA	996	29,4	3,1	19,7	27,4	30,2	31,8	34,2	10,5
Tecnologia	BMOB3.SA	969	13,3	2,4	9,4	11,8	12,5	13,7	22,8	18,0
	INTB3.SA	972	24,0	4,0	12,8	21,1	23,7	26,9	34,7	16,7
	POSI3.SA	996	7,2	1,7	3,0	6,1	6,8	8,2	12,1	23,6
	TOTS3.SA	996	29,1	3,0	22,2	27,2	28,7	30,8	38,5	10,3
Varejo	BHIA3.SA	996	88,4	100,1	2,8	10,5	50,7	100,9	354,8	113,2
	LREN3.SA	996	20,0	6,9	10,3	14,3	18,1	24,3	37,1	34,5
	MGLU3.SA	996	62,1	67,3	6,3	19,0	31,2	61,3	238,1	108,4
	PCAR3.SA	996	15,6	10,5	2,2	3,7	17,3	22,0	41,1	67,3

Fonte: Autor (2025).

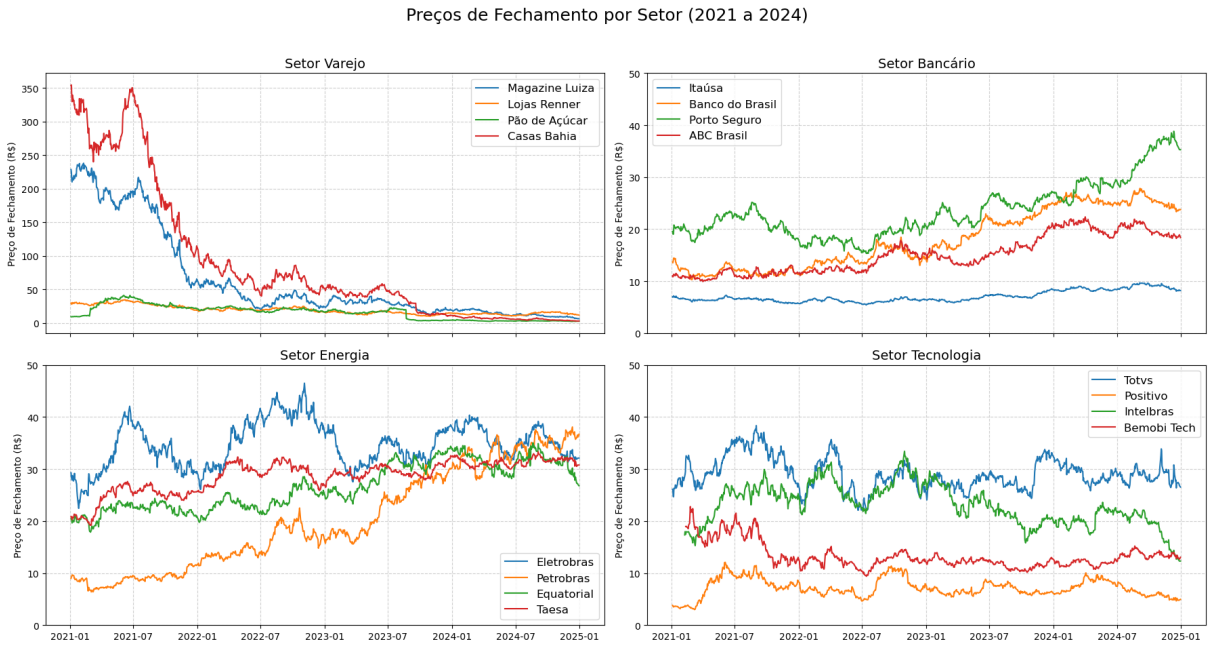
Avaliando os dados da Tabela 3, observa-se que os ativos do setor de Varejo apresentam coeficientes de variação bastante elevados (superiores a 100% em alguns casos), confirmando tratar-se do setor mais volátil. Em contraste, o setor Bancário mostra CVs significativamente menores, refletindo maior estabilidade relativa. Já os setores de Energia e Tecnologia situam-se em posição intermediária, caracterizando cenários mistos de volatilidade.

5.2.1 Gráficos de preço de fechamento por setor

Para uma análise qualitativa, foram produzidos gráficos das séries temporais dos preços de fechamento, agrupados por setor em uma única Figura 13. A inspeção visual possibilita identificar mais facilmente padrões complexos, como a presença de tendências claras (de alta

ou baixa), que indicam a não-estacionariedade da série; quebras estruturais, que são mudanças abruptas no comportamento dos preços (*thresholds* ou intervenções); e *clusters* de volatilidade, fenômeno característico de ativos financeiros onde períodos de alta instabilidade são seguidos por períodos de relativa calma.

Figura 13 – Preços de Fechamento por Setor (2021 a 2024)



Fonte: Autor (2025).

A Figura 13 evidencia contrastes claros entre os padrões de comportamento dos preços de fechamento nos diferentes setores analisados. O setor de Varejo apresenta as oscilações mais acentuadas, com quedas expressivas nas cotações ao longo de 2021 e 2022, seguidas por períodos de recuperação parcial e nova retração. Esse comportamento reflete a forte sensibilidade das empresas varejistas às condições macroeconômicas e à política monetária, em especial às elevações da taxa Selic, que reduziram o poder de compra e afetaram o consumo das famílias.

O setor Bancário, em contrapartida, mostra um padrão mais estável e ascendente, com tendência de valorização gradual entre 2022 e 2024. Essa estabilidade está associada à natureza menos volátil do segmento, que se beneficia de receitas recorrentes e maior previsibilidade financeira.

No setor de Energia, observa-se uma trajetória intermediária, com oscilações moderadas e tendência ligeiramente positiva, refletindo a resiliência de companhias com contratos de

longo prazo e influência de fatores regulatórios e de mercado internacional de petróleo e energia elétrica.

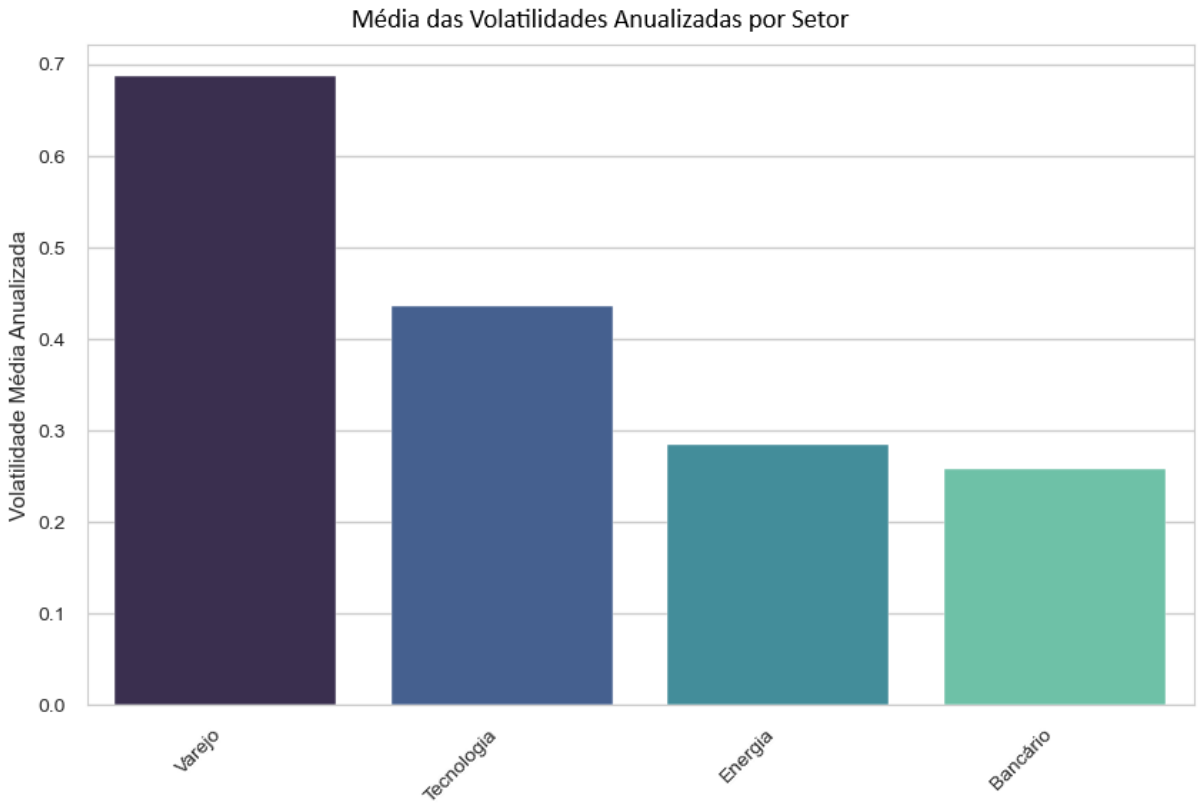
Já o setor de Tecnologia exibe comportamento misto: embora apresente crescimento em alguns períodos, especialmente no início de 2023, nota-se maior dispersão entre as empresas, sugerindo diferentes estágios de maturidade e respostas distintas às variações macroeconômicas. Esse setor mantém volatilidade superior à do Bancário e Energia, mas inferior à observada no Varejo, caracterizando-se como um segmento de risco intermediário.

Em síntese, a análise visual demonstra que os setores apresentam dinâmicas distintas de tendência e volatilidade: o Varejo é o mais instável, o Bancário o mais previsível, o de Energia o mais resiliente e o de Tecnologia o mais heterogêneo. Essas diferenças fundamentam a avaliação posterior dos modelos, pois afetam diretamente a complexidade da tarefa de previsão de preços.

5.2.2 Volatilidade média anualizada por setor

A volatilidade é uma medida do risco de um ativo e é mais bem analisada através dos seus retornos diários, que tendem a ser mais estacionários que os preços. Para quantificar e comparar o risco inerente a cada segmento, foi calculada a volatilidade anualizada, obtida através do desvio padrão dos retornos diários e multiplicada pela raiz quadrada de 252 (o número aproximado de dias de negociação no ano). Esta métrica padronizada permite criar um ranking de setores, do mais ao menos volátil, oferecendo um *insight* claro sobre a estabilidade relativa de cada um. É uma forma visual de comparar as empresas quanto à volatilidade. O setor de Varejo apresenta mais volatilidade que o de Tecnologia, seguido pelo de Energia e do setor Bancário, como representado na Figura 14. Chama a atenção o quanto o setor de Varejo é volátil em relação aos demais.

Figura 14 – Média das Volatilidades Anualizadas por Setor



Fonte: Autor (2025).

A Figura 14 apresenta a volatilidade anualizada dos setores entre 2021 e 2024, destacando diferenças relevantes associadas a fatores macroeconômicos e setoriais. O setor de Varejo revelou-se o mais instável, refletindo sua maior sensibilidade às oscilações da taxa de juros e da renda disponível em um período marcado por pressões inflacionárias e sucessivas alterações na taxa Selic, após a pandemia de Covid-19 e durante a retomada da atividade econômica. Esse comportamento está em consonância com a literatura de finanças, que associa setores cíclicos a maior risco e incerteza (Hull, 2018). O setor de Tecnologia também apresentou níveis elevados de volatilidade, acompanhando a tendência global de correção observada em companhias de crescimento a partir de 2022, cuja precificação incorpora maior incerteza futura. No setor de Energia, as oscilações foram explicadas tanto pela alta internacional das *commodities* quanto por fatores políticos e regulatórios, como as discussões sobre a Petrobras e a privatização da Eletrobras. Já o setor Bancário manteve um padrão mais estável, ainda que tenha refletido episódios de incerteza relacionados à política monetária e ao risco de inadimplência. Em todos os

casos, observa-se a presença de *clusters de volatilidade* (Engle, 1982), fenômeno característico das séries financeiras. Tais evidências reforçam a importância de considerar a volatilidade como medida de risco na previsão de preços de ações, visto que sua correta quantificação impacta diretamente o desempenho de modelos de aprendizado profundo aplicados ao mercado financeiro (Houssein *et al.*, 2021).

5.3 AVALIAÇÃO DE DESEMPENHO DE MODELOS

Nesse momento, são apresentados e analisados os resultados obtidos nas duas fases. A análise é dividida em três partes: o desempenho da abordagem com hiperparâmetros padrão (Fase 1), o desempenho após a otimização (Fase 2) e, por fim, uma análise consolidada que compara as duas fases e avalia o desempenho dos modelos de forma global e por setor.

5.3.1 Fase 1: abordagem padrão

A primeira fase do estudo serviu para estabelecer uma linha de base de desempenho para todos os modelos. A análise visual e quantitativa foi realizada de forma conjunta por setor, conforme apresentada nas Figuras 15 a 22.

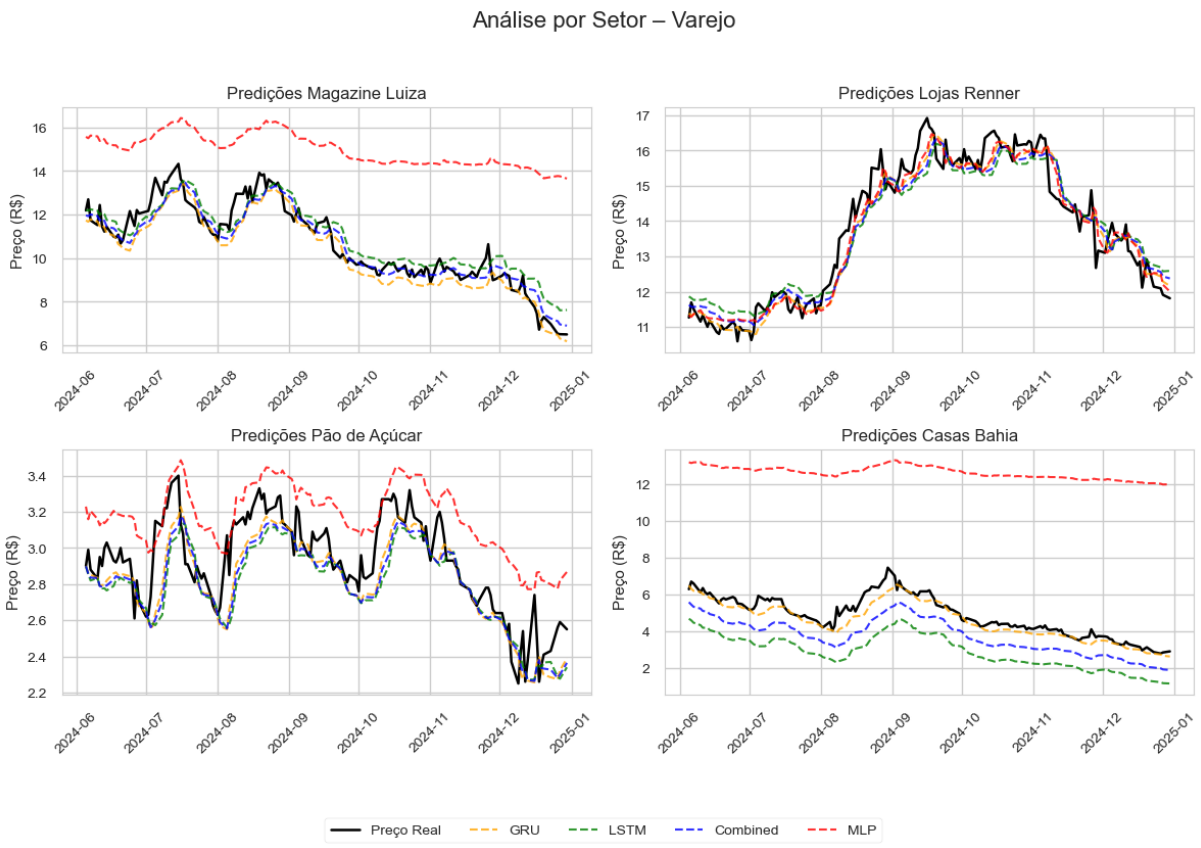
De maneira geral, uma observação consistente em todos os setores é que os modelos propostos GRU, LSTM e *Combined* já demonstram uma capacidade superior em seguir a trajetória dos preços reais em comparação com o *baseline* MLP. Quantitativamente, os gráficos de erro (Figura 16) confirmam essa observação. Em praticamente todos os ativos, os modelos recorrentes apresentaram valores de MAPE e RMSE inferiores e R^2 superiores aos *baselines*, indicando um melhor ajuste e menor erro de previsão.

Apesar disso, a performance do *baseline* não foi insatisfatória; o MLP demonstrou uma capacidade de capturar a direção geral dos preços, embora com maior ruído. Em casos onde o R^2 apresentava valores negativos, estes não foram demonstrados nos gráficos, uma vez que, independentemente da ordem de grandeza, valores negativos indicam uma incapacidade do modelo em predizer resultados melhores do que uma escolha aleatória (Kutner *et al.*, 2005).

É importante destacar que os gráficos de predição apresentados referem-se ao conjunto

de teste, composto por dados que não foram utilizados em nenhuma etapa de treinamento ou ajuste dos modelos. Dessa forma, as previsões ali exibidas refletem a capacidade de generalização dos modelos, ou seja, sua habilidade de reproduzir o comportamento dos preços de ativos nunca antes observados, baseando-se apenas nos padrões aprendidos na fase de treinamento.

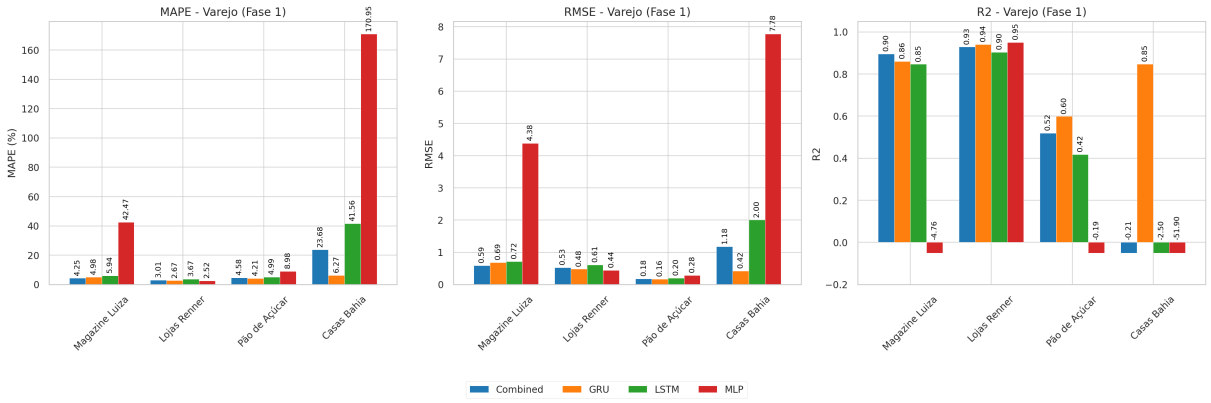
Figura 15 – Gráfico de Predições para o Setor de Varejo (Fase 1)



Fonte: Autor (2025).

Na Figura 15, pode-se observar que o setor de Varejo, os modelos conseguem acompanhar a tendência geral dos preços, mas os desvios em relação à série real são evidentes, sobretudo nos períodos de forte oscilação. O baseline MLP apresenta maior dispersão, enquanto GRU e LSTM mostram trajetória mais próxima, embora ainda com atraso em mudanças bruscas de direção.

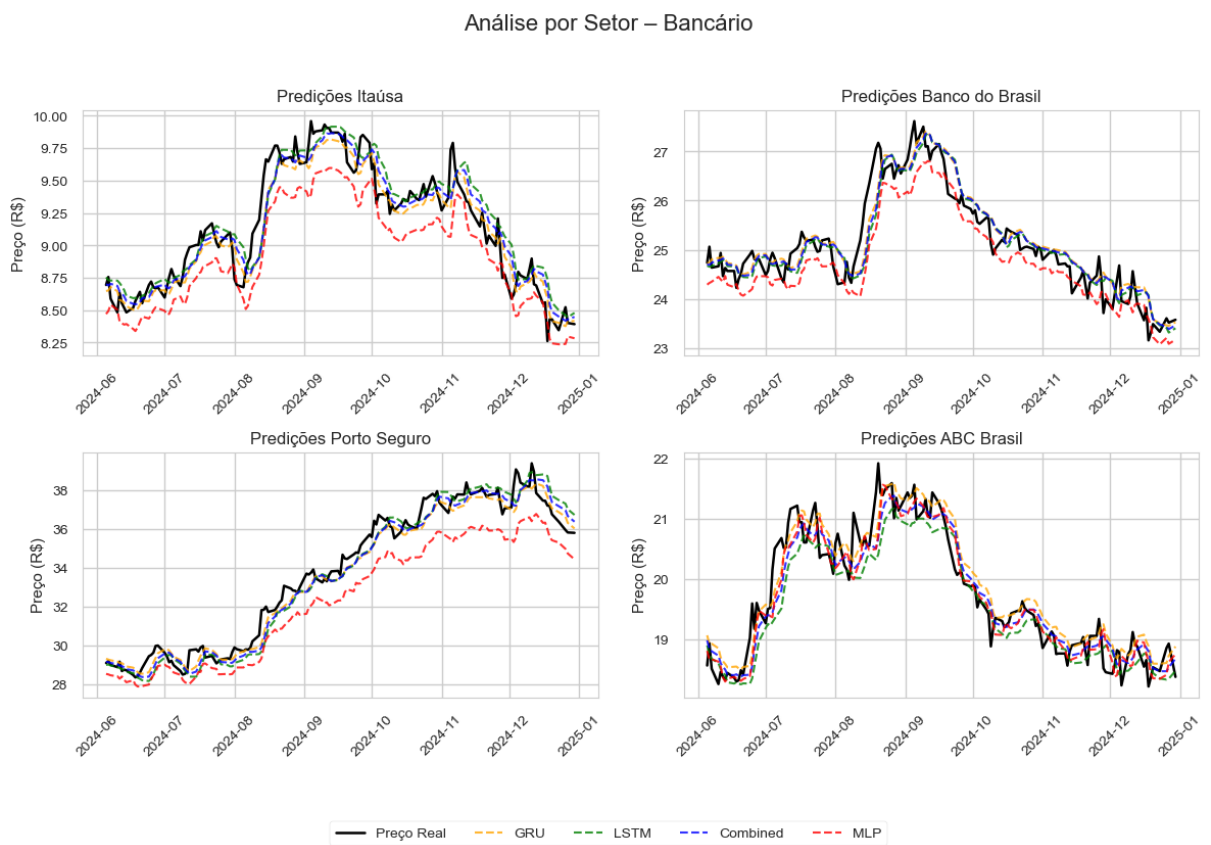
Figura 16 – Métricas de Erro para o Setor de Varejo (Fase 1)



Fonte: Autor (2025).

Os resultados da 16 indicam que os modelos recorrentes (GRU, LSTM e Combined) obtiveram erros consideravelmente menores do que o *baseline* MLP, refletindo maior aderência às séries de preços desse setor mais volátil.

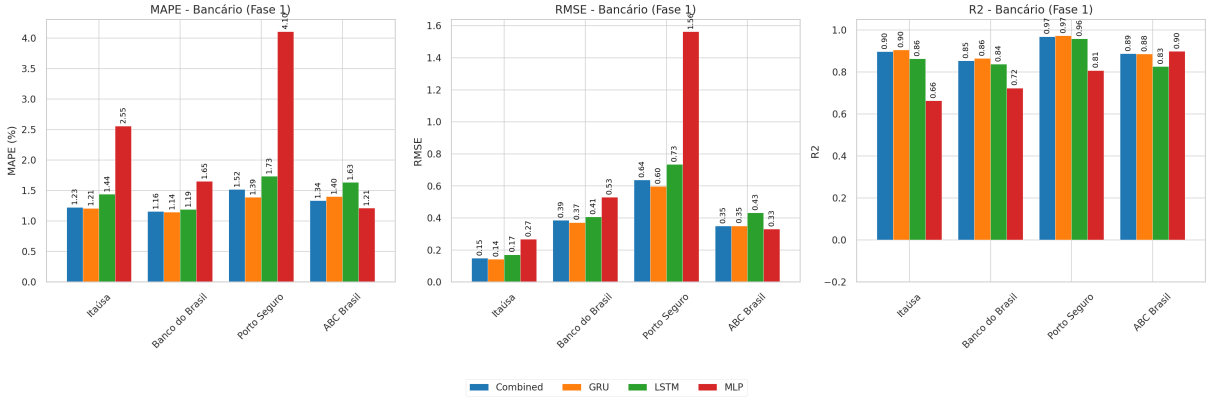
Figura 17 – Gráfico de Predições para o Setor Bancário (Fase 1)



Fonte: Autor (2025).

Na 17, pode-se observar uma boa aderência às séries reais, com curvas projetadas que quase se sobrepõem em grande parte do horizonte. A previsibilidade desse setor, mais estável, favorece a performance dos modelos recorrentes, que superam o *baseline* principalmente em períodos de leve tendência de alta.

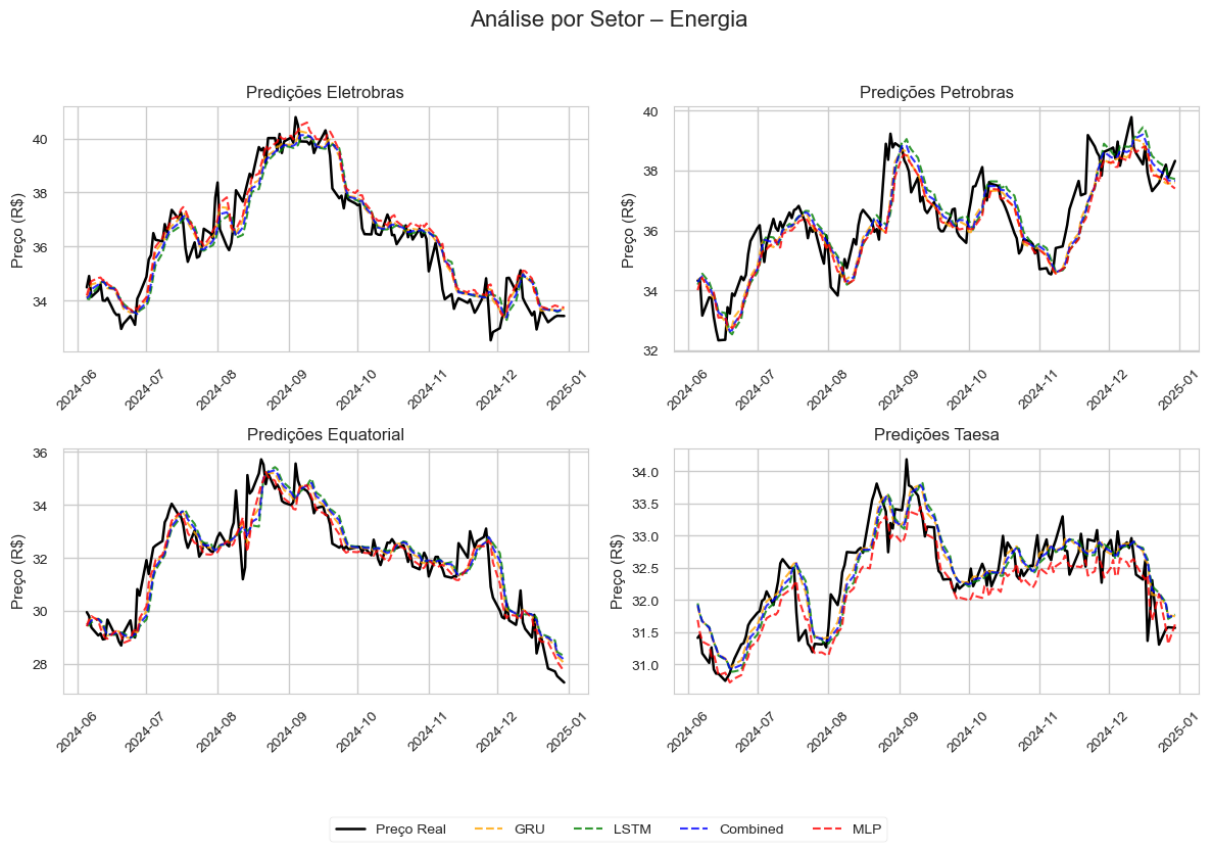
Figura 18 – Métricas de Erro para o Setor Bancário (Fase 1)



Fonte: Autor (2025).

Na Figura 18, vê-se uma diferença entre os modelos recorrentes e o *baseline* ainda mais clara, com MAPE e RMSE reduzidos e R^2 próximos de 1, sugerindo que as séries desse setor, menos voláteis, são bem capturadas pelas arquiteturas com memória.

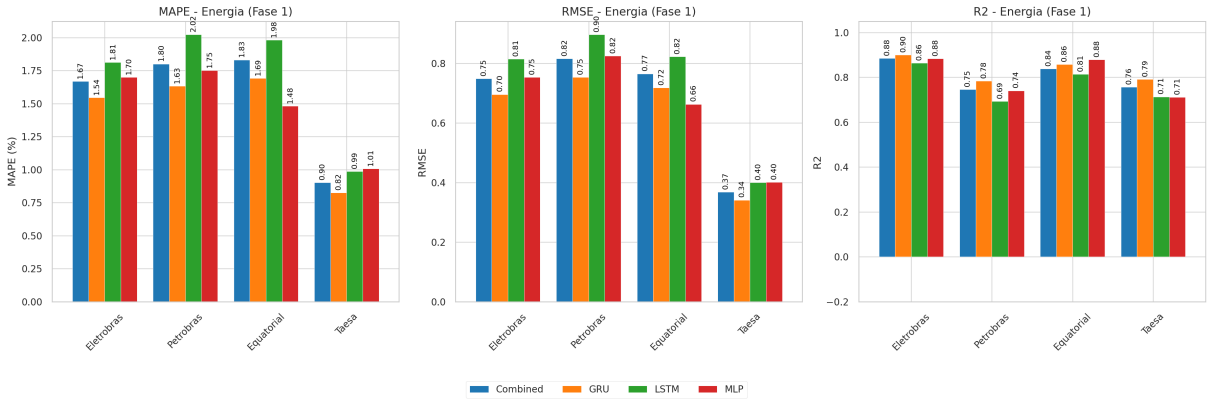
Figura 19 – Gráfico de Predições para o Setor de Energia (Fase 1)



Fonte: Autor (2025).

Na Figura 19, observa-se que os modelos reproduzem de forma satisfatória a dinâmica dos preços, mas há pequenas diferenças entre eles. A GRU mostra alinhamento consistente, enquanto o MLP apresenta oscilações maiores. Em geral, o desempenho é melhor que no Varejo, reflexo da menor volatilidade setorial.

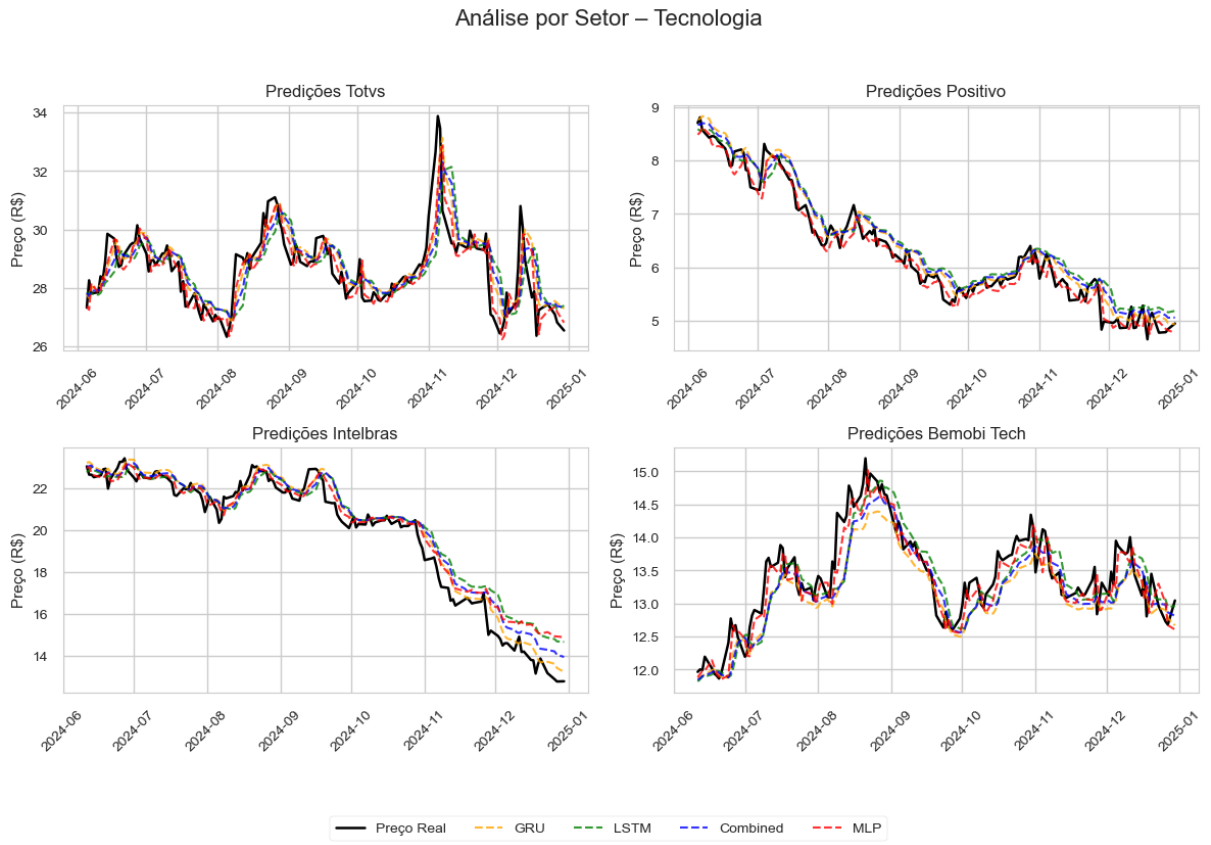
Figura 20 – Métricas de Erro para o Setor de Energia (Fase 1)



Fonte: Autor (2025).

Os modelos avaliados no setor de Energia (Figura 20) também superaram o *baseline*, embora a margem em relação ao MLP seja ligeiramente menor do que em outros setores, possivelmente devido à maior estabilidade das séries.

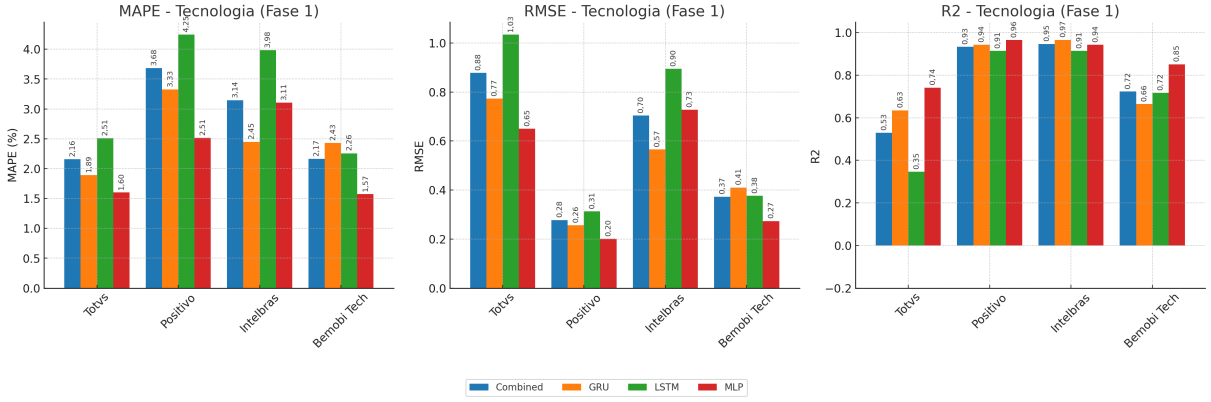
Figura 21 – Gráfico de Predições para o Setor de Tecnologia (Fase 1)



Fonte: Autor (2025).

Por fim, as previsões do setor de Tecnologia (Figura 21) refletem a capacidade dos modelos em captar tendências de médio prazo, mas ainda revelam dificuldades em episódios de maior instabilidade. O GRU mantém previsões mais suaves e próximas da série real, enquanto o MLP mostra divergências significativas.

Figura 22 – Métricas de Erro para o Setor de Tecnologia (Fase 1)



Fonte: Autor (2025).

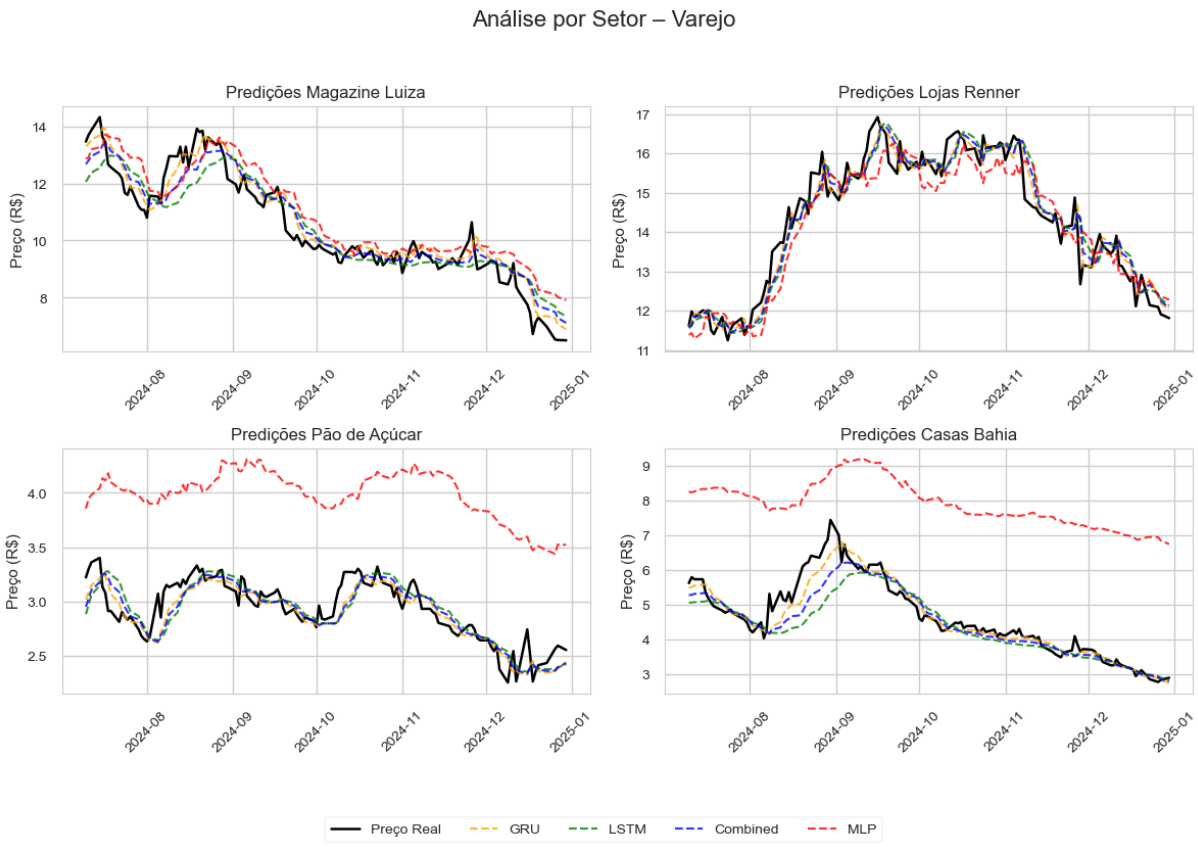
Observa-se na Figura 22 novamente a superioridade das arquiteturas GRU e LSTM, que alcançam menores erros e maior poder explicativo (R^2), mesmo em um segmento caracterizado por volatilidade intermediária.

5.3.2 Fase 2: Abordagem otimizada com *Grid Search*

Na segunda fase, os modelos GRU e LSTM foram submetidos a um processo de otimização de hiperparâmetros. Os resultados, apresentados para cada setor nas Figuras 23 a 30, mostram um refinamento no desempenho. As linhas de previsão dos modelos otimizados aparentam estar mais aderentes aos preços reais e menos suscetíveis a desvios abruptos quando comparadas às da Fase 1.

As métricas de erro da Fase 2, também apresentadas por setor (ex: Figura 24), quantificam essa melhoria. Observa-se uma redução geral nos valores de MAPE e RMSE para os modelos GRU, LSTM e, consequentemente, para o *ensemble Combined*. O R^2 também apresenta valores mais altos e consistentes, indicando que a otimização não só reduziu o erro, mas também melhorou a capacidade dos modelos em explicar a variância dos dados.

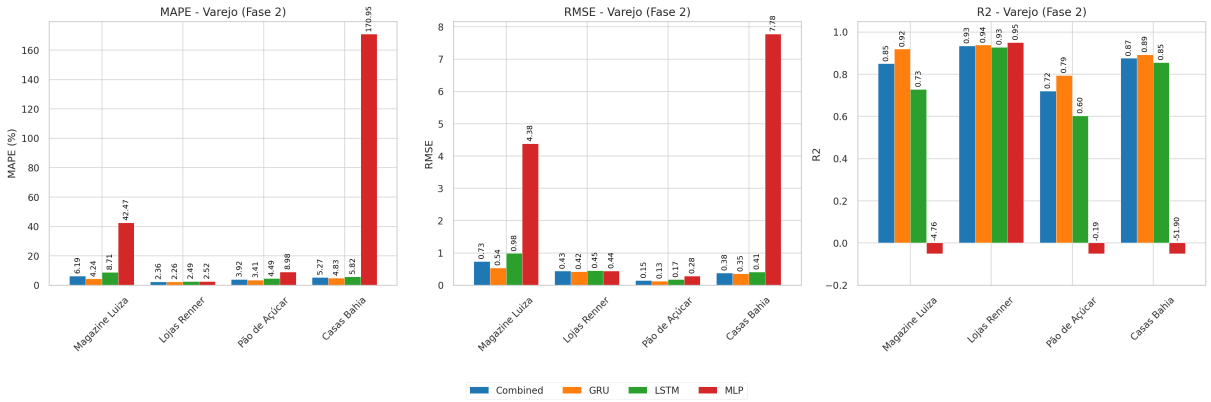
Figura 23 – Gráfico de Predições para o Setor de Varejo (Fase 2 - Otimizada)



Fonte: Autor (2025).

Após a otimização, observa-se na Figura 23 que as predições tornaram-se mais próximas da trajetória real, reduzindo os desvios observados na Fase 1 e aumentando a aderência em períodos de forte oscilação.

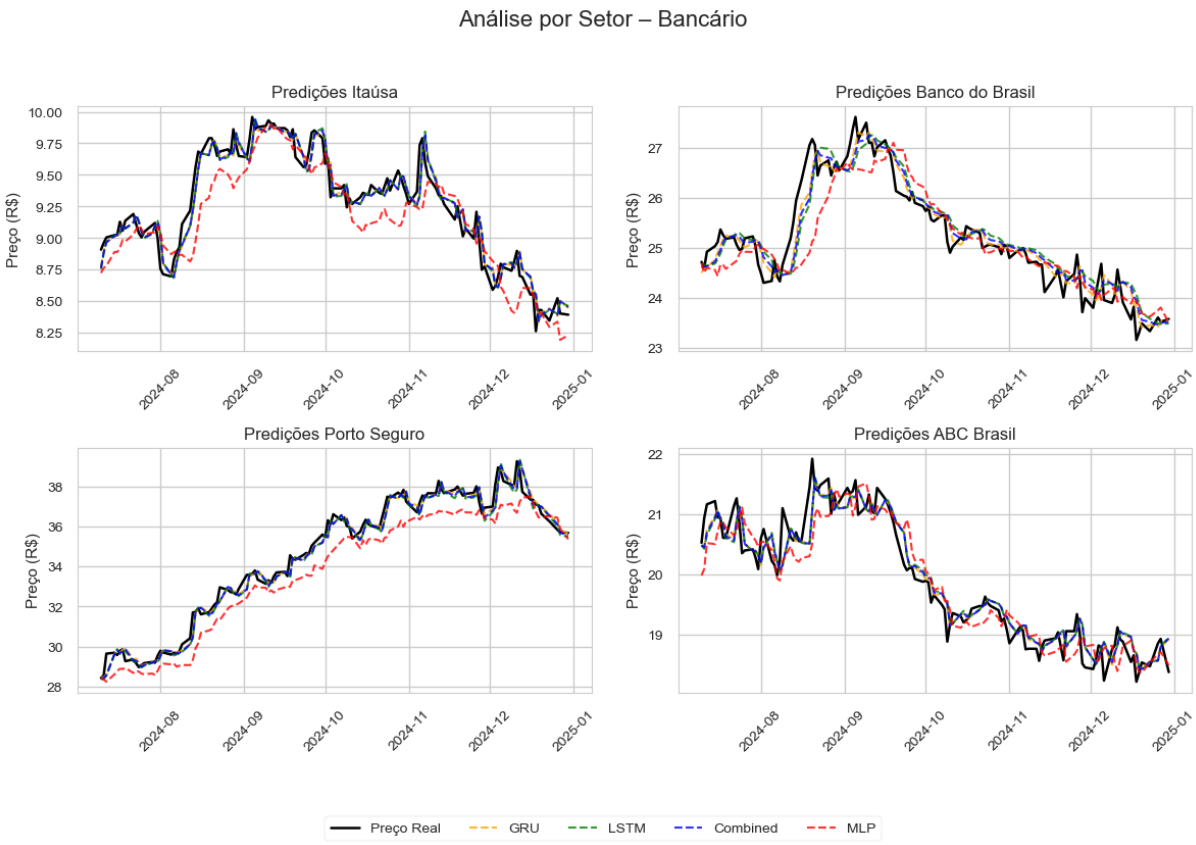
Figura 24 – Métricas de Erro para o Setor de Varejo (Fase 2 - Otimizada)



Fonte: Autor (2025).

Na Figura 24, destaca-se o melhor desempenho para o modelo GRU, que manteve o menor erro médio para este setor após a otimização, reforçando sua robustez mesmo em séries de alta instabilidade.

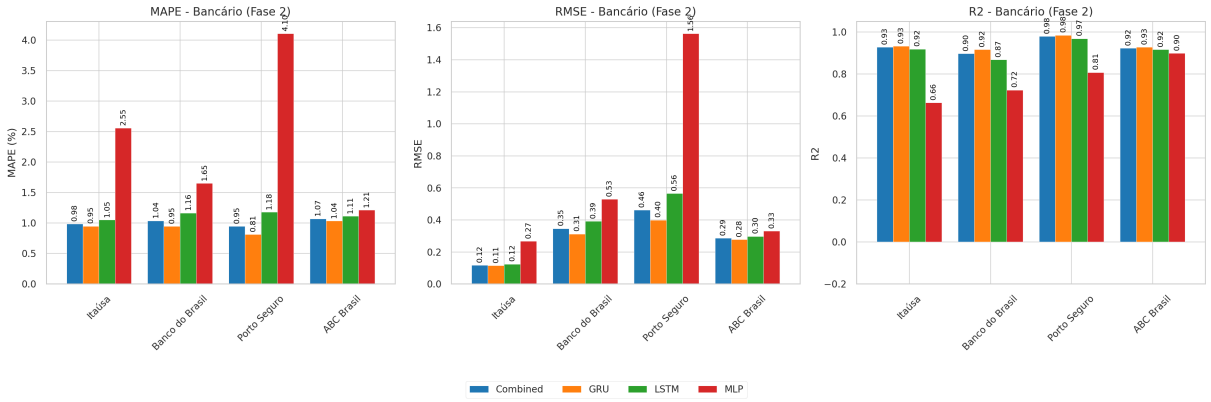
Figura 25 – Gráfico de Predições para o Setor Bancário (Fase 2 - Otimizada)



Fonte: Autor (2025).

Para o setor Bancário (Figura 25), pode-se verificar que a otimização reduziu o atraso em captar as mudanças, demonstrando um alto grau de precisão, praticamente sobrepondo-se às séries reais. Isso confirma o potencial das arquiteturas recorrentes em ambientes de menor risco.

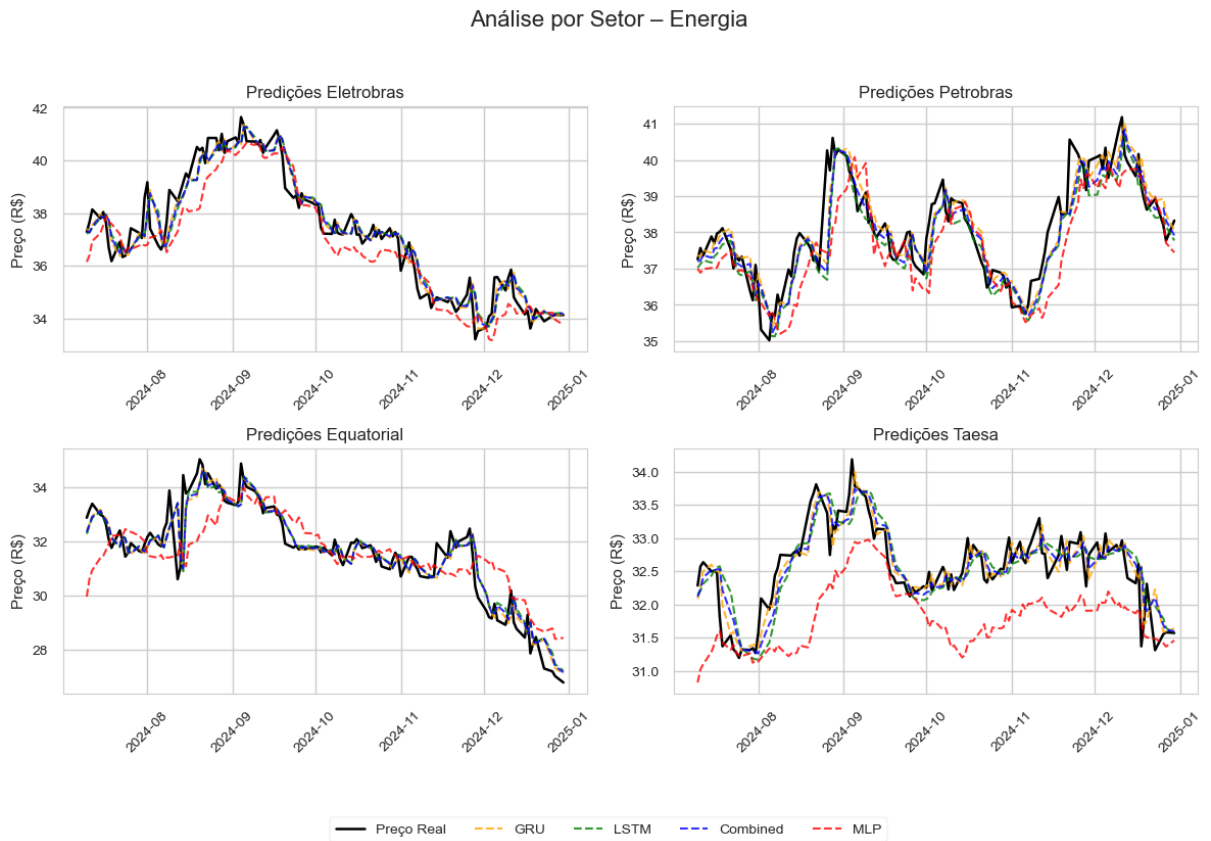
Figura 26 – Métricas de Erro para o Setor Bancário (Fase 2 - Otimizada)



Fonte: Autor (2025).

As métricas da Figura 26 mostram resultados consistentes, com reduções adicionais no MAPE e RMSE em comparação à Fase 1. A GRU consolidou-se como o modelo mais eficaz nesse setor, apresentando desempenho estável e previsões de alta qualidade.

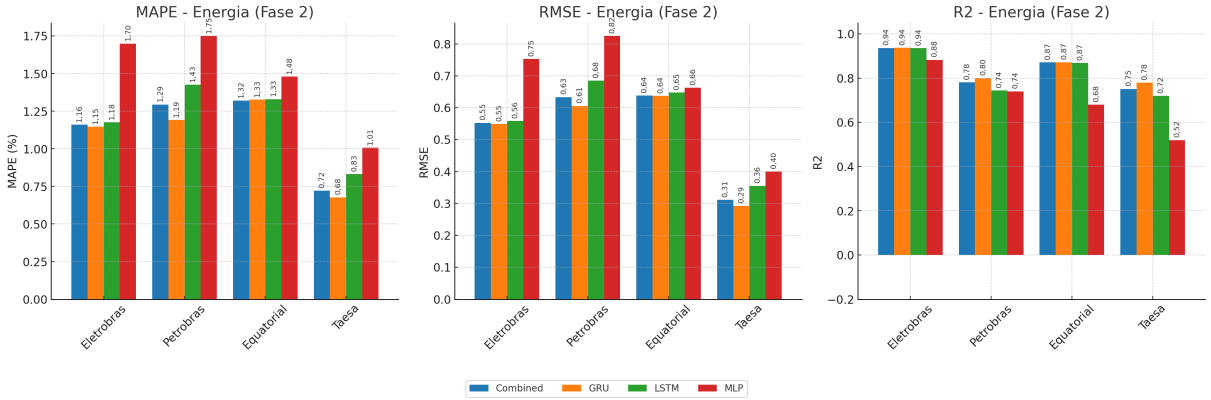
Figura 27 – Gráfico de Predições para o Setor de Energia (Fase 2 - Otimizada)



Fonte: Autor (2025).

Já o setor de Energia (Figura 27) apresentou ganhos visíveis após a otimização, com séries previstas que se alinham de forma mais estreita à trajetória real, reforçando a consistência dos modelos ajustados.

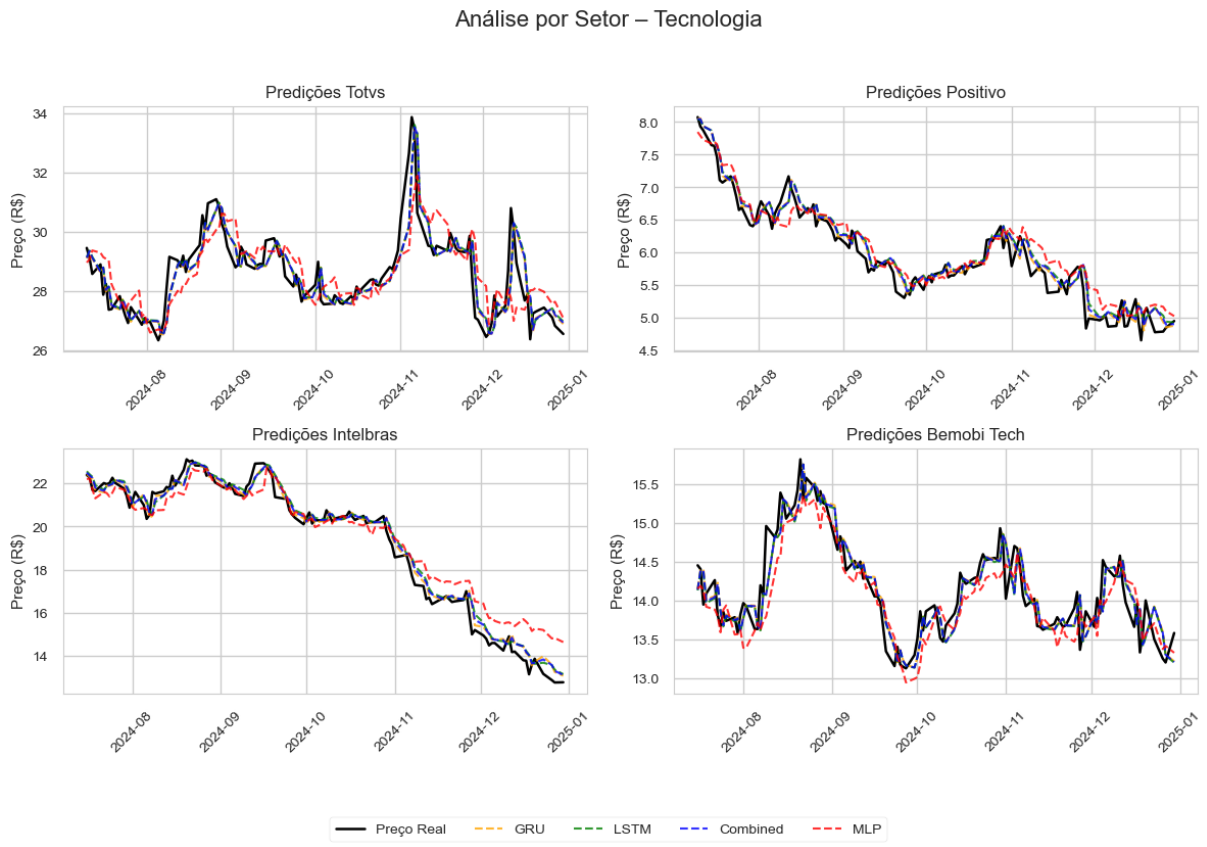
Figura 28 – Métricas de Erro para o Setor de Energia (Fase 2 - Otimizada)



Fonte: Autor (2025).

Nas métricas do setor representadas pela Figura 28, observa-se que a otimização contribuiu para ajustes mais finos neste setor, reduzindo os erros em todos os modelos recorrentes. Ainda que a diferença entre GRU e *Combined* seja pequena, o desempenho final manteve-se superior ao *baseline*.

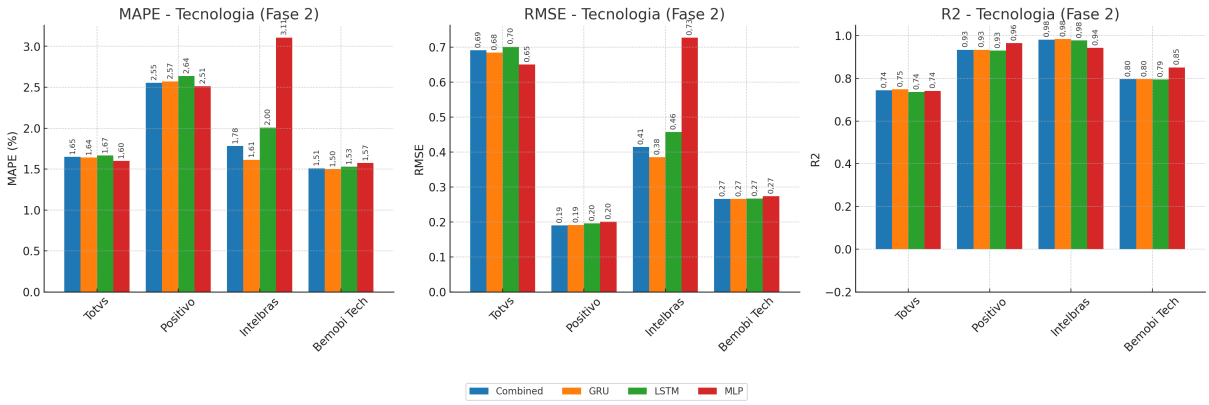
Figura 29 – Gráfico de Predições para o Setor de Tecnologia (Fase 2 - Otimizada)



Fonte: Autor (2025).

Por fim, no setor de Tecnologia (Figura 29), os modelos otimizados mostraram evolução significativa, capturando a tendência dos preços com menor dispersão e aumentando a confiabilidade das previsões.

Figura 30 – Métricas de Erro para o Setor de Tecnologia (Fase 2 - Otimizada)



Fonte: Autor (2025).

A análise das métricas do setor de Tecnologia (Figura 30) confirma o impacto positivo da otimização, com todos os modelos recorrentes apresentando reduções de erro. O GRU mais uma vez se destacou como a arquitetura mais consistente, validando sua aplicabilidade em setores de maior dinamismo.

5.3.3 Custo computacional

Em termos de custo computacional, a otimização de hiperparâmetros implementada na Fase 2 representou um aumento substancial no tempo de processamento. Enquanto a Fase 1, com hiperparâmetros fixos, demandou aproximadamente 6 horas de execução para todo o conjunto de empresas e setores, a Fase 2 consumiu cerca de 24 horas no mesmo ambiente computacional. Esse acréscimo de tempo decorre do processo de *Grid Search*, que envolve o treinamento repetido dos modelos para todas as combinações de parâmetros definidos. Apesar do custo quadruplicado em tempo de execução, o ganho obtido em termos de acurácia e estabilidade das previsões justifica o investimento computacional adicional, especialmente em aplicações nas quais a precisão preditiva é fator determinante para a tomada de decisão.

5.3.4 Validação estatística dos resultados

5.3.4.1 Comparação entre as fases

Para verificar o efeito global da otimização, comparou-se a distribuição dos erros (MAPE) entre as fases, agregando todos os modelos. O teste de Wilcoxon para duas amostras independentes indicou diferença estatisticamente significativa entre a Fase 1 e a Fase 2 ($W = 2531$, $p = 0{,}02148$). Como o valor-p é menor que 0,05, rejeita-se H0 de que as duas fases têm a mesma distribuição de MAPEs. Ou seja, há evidência de que os valores de MAPE não vieram da mesma distribuição, evidenciando redução do MAPE após a otimização de hiperparâmetros.

A Tabela 4 resume as estatísticas descritivas do MAPE por fase.

Tabela 4 – Estatísticas descritivas do MAPE por fase (conjunto de teste).

Fase	n	Média	dp	Mín	Q25	Mediana	Q75	Máx
1	64	6,692	22,169	0,824	1,537	2,002	3,672	170,947
2	64	5,495	21,674	0,678	1,163	1,607	2,558	170,947

Fonte: Autor (2025).
Nota: Q25 - Quantil 25; Q75 - Quantil 75; dp - desvio padrão

5.3.5 Comparação entre modelos na Fase 1

Na Fase 1, os modelos foram comparados pelo teste de Friedman, preservando o emparelhamento por ação ($N = 16$). O resultado foi significativo ($\chi^2_c = 17{,}325$, $p = 0{,}0006$), rejeitando H0 de desempenhos equivalentes. Isso indica que, pelo menos uma, ou um grupo, das arquiteturas apresentou diferença significativa em relação às demais.

Para identificar qual modelo ou modelos se diferenciam dos demais, aplicou-se o teste *post-hoc* de Nemenyi. Os resultados, juntamente com as estatístiscas descritivas, podem ser vistas na Tabela 5.

Tabela 5 – Estatísticas descritivas do MAPE por Modelo (Fase 1).

Modelo	n	Média	dp	Mín	Q25	Mediana	Q75	Máx	Teste[1]
LSTM	16	5,122	9,824	0,987	1,708	2,140	4,049	41,559	b
MLP	16	15,573	42,655	1,007	1,594	2,132	3,355	170,947	b
Combined	16	3,632	5,463	0,903	1,472	1,995	3,279	23,681	a
GRU	16	2,441	1,536	0,824	1,398	1,793	2,832	6,268	ab

Fonte: Autor (2025).
Nota: Modelos seguidos de mesma letra são estatisticamente iguais entre si, pelo teste de Nemenyi a 5%.

Com base na Tabela 5, pode-se extrair que há uma diferença significativa entre os modelos GRU e LSTM, e entre os modelos *Combined* e LSTM (valor-p < 0,05), não havendo diferenças estatisticamente significativas entre os demais modelos. Lembrando que quanto menor o valor de erro, aqui representado pelo MAPE, melhor é o modelo, pode-se concluir que GRU e o *Combined* foram superiores (menor Mediana do MAPE) em relação ao LSTM. Para

essa fase 1, já se notaram diferenças no MAPE dos modelos recorrentes e do modelo *baseline* MLP, ainda que não fossem estatisticamente significativas.

Na Tabela 6, observa-se os *valores-p* das comparações para a par dos modelos.

Tabela 6 – *Valores-p* das comparações par a par entre os modelos (MAPE no conjunto de teste) da Fase 1.

Comparação	GRU	LSTM	MLP
Combined	0,434	0,046	0,947
GRU	–	0,00023	0,168
LSTM	–	–	0,168

Fonte: Autor (2025).

5.3.6 Comparação entre modelos na Fase 2

Na Fase 2, após a otimização de hiperparâmetros utilizando *Grid Search*, foi realizado novamente o teste de Friedman, agora com as versões otimizadas dos modelos recorrentes. O resultado foi significativo ($\chi^2_c = 33{,}3$, $p < 0{,}0001$), indicando que, com a otimização, as diferenças de desempenho entre os modelos se tornaram mais evidentes, dado o menor valor-p.

Aplicou-se o teste *post-hoc* de Nemenyi para entender melhor quais modelos se destacaram. Os resultados, juntamente com as estatísticas descritivas, estão na Tabela 7.

Tabela 7 – Estatísticas descritivas do MAPE por modelo (Fase 2)

Modelo	n	Média	dp	Mín	Q25	Mediana	Q75	Máx	Teste[1]
MLP	16	15,573	42,655	1,007	1,594	2,132	3,355	170,947	a
LSTM	16	2,413	2,159	0,831	1,174	1,477	2,527	8,706	b
Combined	16	2,110	1,629	0,722	1,062	1,415	2,410	6,186	bc
GRU	16	1,884	1,260	0,678	1,013	1,413	2,335	4,828	c

Fonte: Autor (2025).
Nota: Modelos seguidos de mesma letra são estatisticamente iguais entre si, pelo teste de Nemenyi a 5%.

Segundo a Tabela 7, pode-se concluir que, após a otimização, os modelos recorrentes obtiveram desempenhos consistentemente melhores do que o *baseline* MLP, sendo todos esta-

tisticamente diferentes e, consolidando, em relação ao MAPE, o modelo GRU como o melhor, seguido pelo *Combined*, e, por fim, o LSTM sobre o MLP.

Por fim, na Tabela 8, observa-se os *valores-p* das comparações para a par dos modelos, demonstrando que os modelos são, de fato, estatisticamente diferentes (valor-p < 0,05).

Tabela 8 – *Valores-p* das comparações par a par entre os modelos (MAPE no conjunto de teste) da Fase 2.

Comparação	GRU	LSTM	MLP
Combined	0,354	0,065	0,002
GRU	–	0,001	1,2e-06
LSTM	–	–	0,007

Fonte: Autor (2025).

5.3.7 Análise por setor na Fase 2

A análise setorial buscou verificar se as diferenças globais observadas entre os modelos se mantêm em cada segmento da economia. O teste de Friedman foi aplicado separadamente em Varejo, Bancário, Energia e Tecnologia. Os resultados da Tabela 9) indicam diferenças estatísticas em três dos quatro setores.

Tabela 9 – Fase 2: Mediana do MAPE por Setor e Modelo.

Setor	χ^2_c	*valor-p*	Modelo	Mediana[1]	Teste[1]
Varejo	12,0	0,0074	GRU	4,237	b
			LSTM	8,706	ab
			Combined	6,186	ab
			MLP	42,473	a
Bancário	12,0	0,0074	GRU	2,257	b
			LSTM	4,075	ab
			Combined	3,032	ab
			MLP	10,594	a
Energia	11,1	0,0112	GRU	1,457	b
			LSTM	3,126	ab
			Combined	2,155	ab
			MLP	12,431	a
Tecnologia	3,9	0,2725	GRU	2,547	b
			LSTM	5,909	ab
			Combined	3,806	ab
			MLP	16,569	a

Fonte: Autor (2025).
Nota: [1]Modelos seguidos da mesma letra são estatisticamente iguais entre si, pelo teste de Nemenyi

Nos setores de Varejo e Bancário, a diferença entre MLP e GRU foi significativa, confirmando a fragilidade do *baseline* frente ao modelo recorrente. No setor de Energia, o padrão se repetiu, ainda que com menor intensidade. Já no setor de Tecnologia, o Teste de Friedman não rejeitou H0, indicando ausência de separação estatística entre os modelos. Esse resultado pode estar relacionado à boa performance do modelo MLP em três das quatro empresas avaliadas para o setor, superando até mesmo os modelos recorrentes, conforme pode-se observar na Figura 30.

O aumento do erro nos setores mais voláteis, como o de Varejo, evidencia uma limitação estrutural dos modelos recorrentes quando aplicados a séries financeiras com alta imprevisibilidade. As redes LSTM e GRU são eficientes em capturar dependências temporais e padrões recorrentes, mas assumem implicitamente que o comportamento passado contém informação

suficiente para estimar o futuro. Em mercados sujeitos a choques exógenos, mudanças abruptas de tendência ou eventos não recorrentes — comuns em setores sensíveis a política monetária, consumo e sazonalidade — essa premissa é enfraquecida, reduzindo a capacidade de generalização dos modelos (Cont, 2001). Além disso, a alta volatilidade aumenta a proporção de ruído em relação ao sinal, o que dificulta o aprendizado de relações significativas e favorece o sobreajuste (*overfitting*) a variações aleatórias. Esses resultados indicam que, embora modelos recorrentes capturem dinâmicas de curto prazo com eficiência, sua aplicabilidade em contextos de instabilidade elevada exige complementação com variáveis exógenas, técnicas de regularização mais fortes ou abordagens híbridas que incorporem informação contextual do mercado.

Em síntese, a análise setorial reforça a evidência de que os ganhos da Fase 2 se expressaram de forma clara em três setores (Varejo, Bancário e Energia), sempre com o MLP em desvantagem e GRU como melhor modelo. No setor de Tecnologia, apesar de não se observar diferença estatística, as médias de erro mantiveram a tendência de vantagem do GRU, tendo sido a MLP prejudicada por *outliers* para este setor.

6 CONSIDERAÇÕES FINAIS

O melhor modelo, à luz dos objetivos traçados, é a arquitetura GRU após a otimização da Fase 2. Ela se mantém como a escolha mais consistente e deve ser adotada como referência padrão para o problema estudado, com LSTM e *Combined* como alternativas secundárias quando houver restrições específicas. Neste caso, LSTM foi uma escolha menos interessante, e levou o desempenho do *Combined* para baixo também.

O desempenho superior do modelo GRU otimizado pode ser explicado por características estruturais da sua arquitetura. As *Gated Recurrent Units* compartilham princípios semelhantes às LSTM, mas apresentam uma estrutura mais simples, com apenas dois portões. Essa redução de complexidade implica menor número de parâmetros a serem ajustados e, consequentemente, menor risco de sobreajuste (*overfitting*), especialmente em séries temporais financeiras que possuem alta variabilidade e quantidade limitada de observações úteis para o aprendizado (Chung *et al.*, 2014).

Além disso, a GRU tende a reter informações de curto e médio prazo de forma mais eficiente, apresentando convergência mais rápida e estável durante o treinamento (Józefowicz; Zaremba; Sutskever, 2015). Em problemas de previsão de preços de ações, onde dependências temporais de longo prazo são mais fracas e ruído estocástico é predominante, essa capacidade de adaptação dinâmica é vantajosa. Assim, a combinação entre simplicidade estrutural, eficiência na atualização de estados e menor custo computacional confere à GRU maior robustez e melhor capacidade de generalização em comparação com as arquiteturas LSTM e MLP, mesmo após o ajuste fino de hiperparâmetros.

A otimização de hiperparâmetros gera ganhos práticos e replicáveis, suficientes para justificar o maior custo computacional e de tempo, que para as condições deste trabalho, aumentou o tempo de 6 horas de processamento na Fase 1 para 24 horas na Fase 2. Para as arquiteturas recorrentes, esse passo deve integrar o *pipeline* como procedimento rotineiro.

Em relação ao *baseline*, as arquiteturas recorrentes otimizadas superam o MLP de forma sistemática e generalizada entre ativos e setores, sustentando a substituição do *baseline* por um modelo recorrente otimizado — em particular, a GRU — como escolha preferencial.

Do ponto de vista prático, os resultados obtidos indicam que modelos recorrentes podem oferecer suporte relevante à tomada de decisão de investidores ao fornecer previsões consistentes de curto prazo e identificar padrões de comportamento nos preços. No entanto, é importante

destacar que tais modelos devem ser interpretados como ferramentas auxiliares, e não como mecanismos autônomos de decisão. A natureza não estacionária e altamente influenciada por eventos externos dos mercados financeiros impõe limites à capacidade de generalização dos modelos, que capturam essencialmente relações históricas e padrões de curto prazo (Houssein *et al.*, 2021). Assim, embora as redes recorrentes possam melhorar a acurácia preditiva em relação a abordagens tradicionais, seu uso prático exige integração com análises fundamentalistas, indicadores macroeconômicos e gestão de risco, de modo que o investidor real possa interpretar as previsões dentro de um contexto mais amplo de mercado e adotar estratégias baseadas em múltiplas fontes de informação.

7 TRABALHOS FUTUROS

Como diretrizes para a continuação e aperfeiçoamento do trabalho, pode-se citar alguns pontos, como a ampliação do número de ações por setor e o horizonte temporal, de modo a aumentar o poder estatístico das comparações estratificadas. Também pode-se explorar a adoção de avaliação por janelas deslizantes e validação cruzada bloqueada no tempo, reduzindo a dependência de um único corte amostral e permitindo estimar a variação temporal de desempenho.

Também se sugere a investigação de arquiteturas complementares às recorrentes, como *Temporal Convolutional Networks* e modelos baseados em atenção (por exemplo, *Temporal Fusion Transformer*), além de *ensembles* simples e empilhamento. Incorporar variáveis exógenas macroeconômicas, fatores setoriais e indicadores técnicos, com seleção de atrasos orientada por critérios de informação e testes de causalidade em sentido de *Granger*.

Pode-se também explorar a substituição do *Grid Search* por estratégias bayesianas (*TPE /SMBO*) e *early stopping* com validação temporal, avaliando também seleção adaptativa por ativo (meta-aprendizagem) e combinações ponderadas por desempenho recente.

Como forma de proporcionar reprodutibilidade e amplo uso por profissionais da área, realizar a padronização do *pipeline* de dados, detecção de *drift*,*retraining* periódico e monitoramento de métricas. Publicar artefatos de reprodutibilidade (código, configurações e *seeds*) e relatórios automatizados que consolidem desempenho, significância estatística e valor econômico ao longo do tempo.

REFERÊNCIAS

ANDERSON, D. R.; SWEENEY, D. J.; WILLIAMS, T. A. **Estatística aplicada à administração e economia**. 2. ed. São Paulo: Pioneira Thomson Learning, 2002.

BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. **IEEE Transactions on Neural Networks**, IEEE, v. 5, n. 2, p. 157–166, 1994.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **Journal of Machine Learning Research**, v. 13, p. 281–305, 2012. Disponível em: http://jmlr.org/papers/v13/bergstra12a.html. Acesso em: 25 out. 2024.

BISHOP, C. M. **Pattern recognition and machine learning**. New York, NY: Springer, 2006. Disponível em: https://www.springer.com/gp/book/9780387310732. Acesso em: 25 out. 2024.

CHANDRA, R.; HE, Y. Bayesian neural networks for stock price forecasting before and during COVID-19 pandemic. **Plos One**, Public Library of Science, v. 16, n. 7, p. e0253217, 2021.

CHUNG, J. *et al.* **Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling**. 2014. ArXiv preprint arXiv:1412.3555. Disponível em: https://arxiv.org/abs/1412.3555. Acesso em: 25 out. 2024.

COMISSÃO DE VALORES MOBILIÁRIOS (BRASIL). **Guia CVM do investidor: como funciona a bolsa de valores**. 1. ed. Rio de Janeiro: CVM, 2020. Disponível em: https://www.gov.br/cvm/pt-br/assuntos/educacional/publicacoes-educacionais/guia-cvm-investidor. Acesso em: 20 out. 2025.

CONT, R. Empirical properties of asset returns: stylized facts and statistical issues. **Quantitative Finance**, Taylor & Francis, v. 1, n. 2, p. 223–236, 2001.

DEMSAR, J. Statistical Comparisons of Classifiers over Multiple Data Sets. **Journal of Machine Learning Research**, v. 7, p. 1–30, 2006.

DEY, R.; SALEM, F. M. Gate-variants of gated recurrent unit (GRU) neural networks. *In*: IEEE 60TH INTERNATIONAL MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS (MWSCAS), 60., 2017, Boston, **Anais [...]**. Boston: IEEE, 2017. p. 1597-1600.

ENGLE, R. F. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. **Econometrica: Journal of the Econometric Society, JSTOR**, p. 987–1007, 1982.

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

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the American Statistical Association**, v. 32, n. 200, p. 675–701, 1937.

GARCÍA, S. *et al.* Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. **Information Sciences**, v. 180, n. 10, p. 2044–2064, 2010.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA: MIT Press, 2016. Disponível em: https://www.deeplearningbook.org/. Acesso em: 25 out. 2024.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HOLM, S. A simple sequentially rejective multiple test procedure. **Scandinavian Journal of Statistics, JSTOR**, v. 6, n. 2, p. 65–70, 1979.

HOUSSEIN, E. H. *et al.* Artificial neural networks for stock market prediction: a comprehensive review. **Metaheuristics in machine learning: theory and applications**, Springer, p. 409–444, 2021.

HULL, J. C. **Options, futures, and other derivatives**. 10. ed. [*S.l.*]: Pearson Education, 2018.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: principles and practice**. 2. ed. OTexts, 2018. Disponível em: https://otexts.com/fpp2/. Acesso em: 25 out. 2024.

HYNDMAN, R. J.; KOEHLER, A. B. Another look at measures of forecast accuracy. **International Journal of Forecasting**, v. 22, n. 4, p. 679–688, 2006.

JÓZEFOWICZ, R.; ZAREMBA, W.; SUTSKEVER, I. 2015. An empirical exploration of recurrent network architectures. *In*: INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML), 32., 2015, Lille. **Anais [...]**. Lille: PMLR, 2015. p. 2342-2350.

KAMALOV, F. Forecasting significant stock price changes using neural networks. **Neural Computing and Applications**, Springer, v. 32, n. 23, p. 17655–17667, 2020.

KAUFMAN, S. *et al.* Leakage in data mining: Formulation, detection, and avoidance. **ACM Transactions on Knowledge Discovery from Data (TKDD)**, ACM, v. 6, n. 4, p. 1–21, 2012.

KHANPURI, A.; DARAPANENI, N.; PADURI, A. R. Utilizing fundamental analysis to predict stock prices. **EAI Endorsed Transactions on AI and Robotics**, v. 3, 2024.

KOLARIK, T.; RUDORFER, G. **Time series forecasting using neural networks**. [*S.l.*: *s.n.*], 1997. 2–6 p.

KUTNER, M. H. *et al.* **Applied linear statistical models**. 5*th*. ed. [*S.l.*]: McGraw-Hill Education, 2005.

MARTINS, E.; ASSAF, A. **Valuation: Determinação do Valor das Empresas**. 2. ed. São

Paulo: Atlas, 2019. ISBN 978-8597019725.

MURPHY, J. J. **Technical analysis of the financial markets**. 2. ed. New York: New York Institute of Finance, 1999. ISBN 978-0735200661.

NEAL, L. **The Rise of Financial Capitalism: International Capital Markets in the Age of Reason**. Cambridge: Cambridge University Press, 2012. ISBN 9780521895176.

NELSON, D. M. R.; PEREIRA, A. C. M.; OLIVEIRA, R. A. R. de. [*S.l.*: *s.n.*]: [s.n.], 2017. 1419-1426 p. Stock market's price movement prediction with LSTM neural networks. *In*: International Joint Conference on Neural Networks (IJCNN), 2017, [*S.l.*]. **Anais** [...]. [*s.n.*], 2017. p. 1419-1426.

NEMENYI, P. B. **Distribution-free multiple comparisons**. 1963. 137 f. Tese (Doutorado em Estatística) - Princeton University, Princeton, NJ, 1963.

OLAH, C. **Understanding LSTM networks**. [*S.l.*: *s.n.*], 2015. Disponível em: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Acesso em: 22 jul. 2024.

OLIPHANT, T. E. Python for Scientific Computing. **Computing in Science & Engineering**, IEEE, v. 9, n. 3, p. 10–20, 2007.

PROBST, P.; WRIGHT, M. N.; BOULESTEIX, A. Hyperparameter Tuning: An Overview. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v. 9, n. 3, p. e1301, 2019.

RAHMADEYAN, A.; MUSTAKIM. Long short-term memory and gated recurrent unit for stock price prediction. **Procedia Computer Science**, Elsevier, v. 234, p. 204–212, 2024.

ROSSUM, G. V.; DRAKE JR., F. L. **Python 3 reference manual**. Scotts Valley, CA: CreateSpace, 2010. ISBN 9781441412690.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. 4. ed. Harlow: Pearson Education, 2022. ISBN 978-0134610993.

SELVIN, S. *et al.* 2017. Stock price prediction using LSTM, RNN and CNN-sliding window model. *In*: INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING, COMMUNICATIONS AND INFORMATICS (ICACCI), 2017, Udupi. **Anais** [...]. IEEE, 2017. p. 1643-1647.

SEWARD, L. E.; DOANE, D. P. **Estatística aplicada à administração e economia**. 4. ed. [*S.l.*]: AMGH Editora, 2014. ISBN 9788580553949. Disponível em: https://books.google.com.br/books?id=H7pTBAAAQBAJ. Acesso em: 25 out. 2024.

SHAH, D. V. *et al.* 2022. Stock price prediction using LSTM-ARIMA hybrid neural network model with sentiment analysis of news headlines. *In*: INTERNATIONAL CONFERENCE ON FUTURISTIC TECHNOLOGIES (INCOFT), 2022, Belgaum. **Anais** [...]. IEEE, 2022. p. 1-4.

SHEN, J.; SHAFIQ, M. O. **Short-term stock market price trend prediction using a comprehensive deep learning system**. **Journal of big Data**, Springer, v. 7, p. 1–33, 2020.

SRIVASTAVA, N. *et al.* Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 2014. Disponível em: http://jmlr.org/papers/v15/srivastava14a.html. Acesso em: 25 out. 2024.

TIWARI, R.; SRIVASTAVA, S.; GERA, R. Investigation of artificial intelligence techniques in finance and marketing. **Procedia Computer Science**, Elsevier, v. 173, p. 149–157, 2020.

WIKIMEDIA COMMONS. **Gated Recurrent Unit, base type**. [*S.l.*: *s.n.*], 2018. Disponível em: https://commons.wikimedia.org/wiki/File:Gated_Recurrent_Unit,_base_type.svg. Acesso em: 25 out. 2024.

WILCOXON, F. Individual comparisons by ranking methods. **Biometrics Bulletin**, **International Biometric Society**, v. 1, n. 6, p. 80–83, 1945.

WINARDI, S. *et al.* Robust Stock Price Prediction using Gated Recurrent Unit (GRU). **International Journal of Informatics and Computation**, v. 5, n. 1, p. 29–38, 2023.

YAHOO! **Yahoo Finance**. [*S.l.*]: Yahoo!, 2024. Disponível em: https://finance.yahoo.com/. Acesso em: 25 out. 2024.

YU, P.; YAN, X. Stock price prediction based on deep neural networks. **Neural Computing and Applications**, Springer, v. 32, n. 6, p. 1609–1628, 2020.

ZHANG, G. P.; PATUWO, B. E.; HU, M. Y. Time series forecasting using a hybrid ARIMA and neural network model. **Neurocomputing**, Elsevier, v. 50, p. 159–175, 2001.

APÊNDICE A - Tabela 10: Principais Bibliotecas e *Frameworks* Utilizados

Tabela 10 – Principais Bibliotecas e Frameworks Utilizados

Biblioteca / Framework	**Versão**	**Aplicação Principal no Projeto**
Pandas	2.2.2	Estruturação e manipulação de dados, como o processamento das séries temporais de preços e retornos.
NumPy	1.26.4	Base para todas as operações numéricas, cálculo de métricas e manipulação de arrays para as redes neurais.
yfinance	0.2.43	Ferramenta para download dos dados históricos de preços de fechamento dos ativos diretamente do Yahoo Finance.
Matplotlib	3.9.2	Biblioteca fundamental para a criação de todas as visualizações estáticas, como gráficos de linha e de barras.
Seaborn	0.13.2	Utilizada para aprimorar a estética dos gráficos, aplicando estilos visuais consistentes e paletas de cores.
Scikit-learn	1.5.1	Forneceu as ferramentas para pré-processamento de dados (MinMaxScaler) e para o cálculo das métricas de avaliação (MAPE, RMSE, R^2).
TensorFlow/Keras	2.17.0	Framework de *deep learning* utilizado para a construção, treinamento e avaliação de todos os modelos de redes neurais (MLP, GRU e LSTM).
Statsmodels	0.14.2	Empregada na análise exploratória para a decomposição de séries temporais.
SciPy	1.14.0	Utilizada para a realização de testes estatísticos, como o teste de Kruskal-Wallis na análise de sazonalidade.

Fonte: Autor (2025)

APÊNDICE B - Código Utilizado

Os códigos utilizados neste trabalho podem ser encontrados em anexo, bem como no seguinte repositório: LINK

```
#Fase 1 - Análise de Sazonalidade e Previsão de Ações sem Ajuste de
Hiperparâmetros

import os
import math
import logging
import json
import csv
from typing import List, Dict, Tuple, Union

import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_percentage_error, r2_score

import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model #type:
ignore
from tensorflow.keras.layers import Dense, GRU, LSTM, Dropout #type:
ignore
from tensorflow.keras.optimizers import Adam #type: ignore
from tensorflow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau #type: ignore
from tensorflow.keras.preprocessing.sequence import
TimeseriesGenerator #type: ignore

from pmdarima import auto_arima
from scipy.stats import kruskal

# APLICA ESTILO SEABORN AOS GRÁFICOS
sns.set_style("whitegrid")
plt.style.use("seaborn-v0_8-notebook")

# ====== 1. CONFIGURATION
=======================================================
START_DATE = "2021-01-01"
END_DATE = "2024-12-31"
MODEL_DIR = "trained_models"
SEQ_LENGTH = 30
BATCH_SIZE = 16
EPOCHS = 10 #era 100
LINE_WIDTH = 1.5
COMPANY_NAME_MAP = {
    "MGLU3.SA": "Magazine Luiza", "LREN3.SA": "Lojas Renner",
"PCAR3.SA": "Pão de Açúcar", "BHIA3.SA": "Casas Bahia",
    "ITSA3.SA": "Itaúsa", "BBAS3.SA": "Banco do Brasil", "PSSA3.SA":
"Porto Seguro", "ABCB4.SA": "ABC Brasil",
```

```
    "ELET3.SA": "Eletrobras", "PETR3.SA": "Petrobras", "EQTL3.SA":
"Equatorial", "TAEE11.SA": "Taesa",
    "TOTS3.SA": "Totvs", "POSI3.SA": "Positivo", "INTB3.SA":
"Intelbras", "BMOB3.SA": "Bemobi Tech"
}
SECTORS = {
    "Varejo": ["MGLU3.SA", "LREN3.SA", "PCAR3.SA", "BHIA3.SA"],
    "Bancário": ["ITSA3.SA", "BBAS3.SA", "PSSA3.SA", "ABCB4.SA"],
    "Energia": ["ELET3.SA", "PETR3.SA", "EQTL3.SA", "TAEE11.SA"],
    "Tecnologia": ["TOTS3.SA", "POSI3.SA", "INTB3.SA", "BMOB3.SA"]
}
COLOR_MAP = {"GRU": "orange", "LSTM": "green", "MLP": "red", "ARIMA":
"purple", "Combined": "blue"}
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %
(levelname)s - %(message)s')


# ====== 2. UTILITIES & PREPROCESSING
=======================================
def get_subplot_shape(n:int) -> Tuple[int,int]:
    cols = math.ceil(math.sqrt(n))
    return math.ceil(n/cols), cols

def download_stock_data(tickers:List[str], start, end) ->
Dict[str,pd.DataFrame]:
    data = {}
    for t in tickers:
        try:
            df = yf.download(t, start=start, end=end, progress=False,
auto_adjust=True)
            if not df.empty: data[t] = df
            else: logging.warning(f"Nenhum dado para {t} no período
especificado.")
        except Exception as e: logging.error(f"Erro baixando {t}:
{e}")
    return data

def preprocess(df:pd.DataFrame):
    close = df[['Close']].dropna()
    scaler = MinMaxScaler()
    scaled = scaler.fit_transform(close)
    n = len(scaled)
    tr_end = int(.7 * n)
    vl_end = int(.85 * n)
    return scaled[:tr_end], scaled[tr_end:vl_end], scaled[vl_end:],
scaler, (tr_end, vl_end)

def make_gens(tr, vl, te):
    return (TimeseriesGenerator(tr, tr, length=SEQ_LENGTH,
batch_size=BATCH_SIZE),
```

```
            TimeseriesGenerator(vl, vl, length=SEQ_LENGTH,
batch_size=BATCH_SIZE),
            TimeseriesGenerator(te, te, length=SEQ_LENGTH,
batch_size=BATCH_SIZE))

def mlp_datasets(tr, vl, te):
    def build(arr):
        X, y = [], []
        for i in range(SEQ_LENGTH, len(arr)):
            X.append(arr[i-SEQ_LENGTH:i, 0])
            y.append(arr[i, 0])
        return np.array(X), np.array(y)
    return (*build(tr), *build(vl), *build(te))

def empresa_tem_sazonalidade(ticker: str,
path='sazonalidade_empresas.csv') -> bool:
    # Esta função depende da execução prévia de
'analisar_sazonalidade_em_lote'
    try:
        df = pd.read_csv(path)
        linha = df[df["Ticker"] == ticker]
        if linha.empty: return False
        return bool(linha["Sazonalidade_Provável"].iloc[0])
    except:
        return False

# ====== 3. MODEL TRAINING & PREDICTION
=======================================
def model_path(ticker:str, kind:str, suffix="") -> str:
    return os.path.join(MODEL_DIR, f"{ticker}_{kind}{suffix}.keras")

def train_rnn_standard(kind: str, ticker: str, gtr, gvl, epochs: int)
-> Sequential:
    """Treina um modelo RNN com hiperparâmetros padrão."""
    path = model_path(ticker, kind, suffix="_standard")
    if os.path.isfile(path):
        logging.info(f"Carregando {kind} (Padrão) cacheado para
{ticker}")
        return load_model(path)

    logging.info(f"Treinando {kind} (Padrão) para {ticker}...")
    UNITS, DROPOUT, LEARNING_RATE = 50, 0.2, 1e-3
    net = Sequential()
    layer = GRU if kind == "GRU" else LSTM
    net.add(layer(UNITS, return_sequences=True,
input_shape=(SEQ_LENGTH, 1)))
    net.add(Dropout(DROPOUT))
    net.add(layer(UNITS))
    net.add(Dropout(DROPOUT))
    net.add(Dense(1))
```

```
    net.compile(optimizer=Adam(learning_rate=LEARNING_RATE),
loss="mse")
    cb = [EarlyStopping(patience=10, restore_best_weights=True)]
    net.fit(gtr, epochs=epochs, validation_data=gvl, verbose=0,
callbacks=cb)
    net.save(path)
    return net

def load_or_train_mlp(ticker:str, Xtr, ytr, Xvl, yvl, epochs:int) ->
Sequential:
    path = model_path(ticker, "MLP")
    if os.path.isfile(path):
        return load_model(path)
    net = Sequential([Dense(64, activation='relu',
input_shape=(SEQ_LENGTH,)), Dropout(0.1),
        Dense(32, activation='relu'), Dense(1)])
    net.compile(optimizer=Adam(1e-3), loss="mse")
    cb = [EarlyStopping(patience=6, restore_best_weights=True),
ReduceLROnPlateau(factor=0.2, patience=3)]
    net.fit(Xtr, ytr, epochs=epochs, batch_size=BATCH_SIZE,
validation_data=(Xvl, yvl), verbose=0, callbacks=cb)
    net.save(path)
    return net

# ====== 4. METRICS, PLOTTING & REPORTING
=====================================
def calc_metrics(y, ŷ):
    return dict(RMSE=np.sqrt(((y-ŷ)**2).mean()),
                MAPE=mean_absolute_percentage_error(y, ŷ)*100,
                R2=r2_score(y, ŷ))

def plot_preds(sector, stocks, actual, preds):
    r, c = get_subplot_shape(len(stocks))
    fig, axes = plt.subplots(r, c, figsize=(c * 6, r * 4),
squeeze=False)
    axes = np.atleast_1d(axes).ravel()

    real_style = {'lw': 1.8, 'alpha': 1.0, 'color': 'black'}
    pred_style = {'lw': 1.5, 'alpha': 0.8, 'linestyle': '--'}

    last_i = -1
    for i, tk in enumerate(stocks):
        ax = axes[i]
        last_i = i

        if not preds.get(tk) or not preds[tk]:
            ax.set_title(f"Dados indisponíveis para {tk}")
            continue

        act_df = actual[tk]
```

```
        max_len = max(len(p) for p in preds[tk].values())
        ax.plot(act_df.index[-max_len:], act_df['Close'].values[-
max_len:], label="Preço Real", **real_style)

        for kind, p in preds[tk].items():
            plot_values = p.values if isinstance(p, pd.Series) else p
            ax.plot(act_df.index[-len(plot_values):],
plot_values.reshape(-1), label=kind, color=COLOR_MAP.get(kind),
**pred_style)

        ax.set_title(f"Predições {COMPANY_NAME_MAP.get(tk, tk)}")
        ax.tick_params(axis='x', rotation=45)
        ax.set_ylabel("Preço (R$)")

    for j in range(last_i + 1, len(axes)):
        fig.delaxes(axes[j])

    handles, labels = axes[0].get_legend_handles_labels()
    if handles:
        fig.legend(handles, labels, loc='upper center',
bbox_to_anchor=(0.5, 0), ncol=len(handles))

    fig.suptitle(f"Análise por Setor — {sector}", fontsize=16, y=0.98)
    plt.tight_layout(rect=[0, 0.05, 1, 0.95])
    plt.show()

def plot_error(mdict, stocks, sector):
    valid_stocks = [s for s in stocks if s in mdict and mdict[s]]
    if not valid_stocks: return
    models = sorted({m for s in valid_stocks for m in mdict[s]})
    if not models: return

    fig, axes = plt.subplots(1, 3, figsize=(15, 5.5))

    for i, met in enumerate(["MAPE", "RMSE", "R2"]):
        ax = axes[i]

        all_vals = [mdict[s].get(m, {}).get(met, np.nan) for s in
valid_stocks for m in models]
        all_vals_clean = [v for v in all_vals if pd.notna(v) and
np.isfinite(v)]

        if all_vals_clean:
            q1, q3 = np.percentile(all_vals_clean, [25, 75])
            iqr = q3 - q1
            upper_bound = q3 + 1.5 * iqr
            if np.max(all_vals_clean) > upper_bound:
                ax.set_ylim(top=upper_bound * 1.1)
            lower_bound = q1 - 1.5 * iqr
            if np.min(all_vals_clean) < lower_bound:
```

```
                ax.set_ylim(bottom=lower_bound * 1.1)

        x = np.arange(len(valid_stocks))
        w = 0.8 / len(models)
        for j, m in enumerate(models):
            vals = [mdict[s].get(m, {}).get(met, np.nan) for s in
valid_stocks]

            # ALTERAÇÃO: Remove R2 negativo do plot
            if met == 'R2':
                vals = [v if v >= 0 else np.nan for v in vals]

            bars = ax.bar(x + j * w - w/2, vals, w, label=m,
color=COLOR_MAP.get(m))

            for bar in bars:
                height = bar.get_height()
                if pd.isna(height): continue
                bottom_lim, top_lim = ax.get_ylim()
                if bottom_lim <= height <= top_lim:
                    visible_range = top_lim - bottom_lim
                    if abs(height) / visible_range > 0.1:
                        y_pos = bottom_lim + visible_range * 0.03
                        va = 'bottom'
                        if height < 0:
                            y_pos = top_lim - visible_range * 0.03
                            va = 'top'
                        ax.annotate(f'{height:.2f}', xy=(bar.get_x() +
bar.get_width() / 2, y_pos),
                                    ha='center', va=va, fontsize=8,
color='white', rotation=90)

        ax.set_xticks(x)
        ax.set_xticklabels([COMPANY_NAME_MAP.get(s, s) for s in
valid_stocks], rotation=45, ha='right')
        ax.set_ylabel(met + (" (%)" if met == "MAPE" else ""))
        ax.set_title(met)

        if met == "R2":
            ax.set_ylim(0, 1.05) # Força o eixo de 0 a 1 (com uma
pequena margem no topo)

    handles, labels = axes[0].get_legend_handles_labels()
    fig.legend(handles, labels, loc='lower center',
bbox_to_anchor=(0.5, -0.12), ncol=len(labels))
    fig.suptitle(f"Métricas de Erro (Abordagem Padrão) — {sector}",
fontsize=16, y=1.0)
    plt.tight_layout(rect=[0, 0.05, 1, 0.95])
    plt.show()
```

```
def print_table(mdict, sector):
    rows = []
    for stk, mods in mdict.items():
        for mod, val in mods.items():
            rows.append(dict(Setor=sector,
Ação=COMPANY_NAME_MAP.get(stk, stk), Modelo=mod, **val))
    if not rows: return
    print(f"\nResumo das Métricas (Abordagem Padrão) — {sector}")
    print(pd.DataFrame(rows).to_string(index=False,
float_format="%.4f"))

# ====== 5. MAIN EXECUTION LOOP
================================================
def main(sectors:Dict[str,List[str]], start, end):
    # Roda a análise de sazonalidade primeiro, se necessário
    analisar_sazonalidade_em_lote(SECTORS, START_DATE, END_DATE)

    all_data = download_stock_data([tk for tks in sectors.values() for
tk in tks], start, end)
    for sector, tickers in sectors.items():
        logging.info(f"*** Processando Setor (Abordagem Padrão):
{sector} ***")
        sector_data = {t: all_data[t] for t in tickers if t in
all_data}
        if not sector_data: continue
        preds_sector, metrics_sector = {}, {}
        for tk, df in sector_data.items():
            logging.info(f">>> Processando ticker: {tk}")
            preds_sector[tk], metrics_sector[tk] = {}, {}
            tr, vl, te, sc, splits = preprocess(df)
            if len(te) < SEQ_LENGTH: continue
            gtr, gvl, gte = make_gens(tr, vl, te)
            y_true_test =
sc.inverse_transform(te[SEQ_LENGTH:]).ravel()

            # RNNs
            net_gru = train_rnn_standard("GRU", tk, gtr, gvl, EPOCHS)
            pred_gru = sc.inverse_transform(net_gru.predict(gte,
verbose=0)).ravel()
            preds_sector[tk]["GRU"] = pred_gru
            metrics_sector[tk]["GRU"] = calc_metrics(y_true_test,
pred_gru)

            net_lstm = train_rnn_standard("LSTM", tk, gtr, gvl,
EPOCHS)
            pred_lstm = sc.inverse_transform(net_lstm.predict(gte,
verbose=0)).ravel()
            preds_sector[tk]["LSTM"] = pred_lstm
            metrics_sector[tk]["LSTM"] = calc_metrics(y_true_test,
```

```
pred_lstm)

            pred_comb = (pred_gru + pred_lstm) / 2
            preds_sector[tk]["Combined"] = pred_comb
            metrics_sector[tk]["Combined"] = calc_metrics(y_true_test,
pred_comb)

            # Baselines
            Xtr, ytr, Xvl, yvl, Xte, yte = mlp_datasets(tr, vl, te)
            if len(Xte) > 0:
                mlp = load_or_train_mlp(tk, Xtr, ytr, Xvl, yvl,
EPOCHS)
                pred_mlp = sc.inverse_transform(mlp.predict(Xte,
verbose=0)).ravel()
                preds_sector[tk]['MLP'] = pred_mlp
                metrics_sector[tk]['MLP'] = calc_metrics(y_true_test,
pred_mlp)

            # ARIMA com transformação de log
            try:
                train_arima, test_arima =
df['Close'].iloc[:splits[1]], df['Close'].iloc[splits[1]:]
                train_arima_log = np.log(train_arima[train_arima > 0])

                sazonal = empresa_tem_sazonalidade(tk)
                arima_model = auto_arima(train_arima_log,
seasonal=sazonal, m=12 if sazonal else 1,
                                         stepwise=True,
suppress_warnings=True, error_action='ignore',
                                         max_p=3, max_q=3, max_P=2,
max_Q=2)

                fc_log =
arima_model.predict(n_periods=len(test_arima))
                fc = np.exp(fc_log)

                if len(fc) < len(y_true_test):
                    padding = np.full(len(y_true_test) - len(fc),
fc.iloc[-1] if not fc.empty else 0)
                    fc = pd.concat([fc, pd.Series(padding,
index=np.arange(len(fc), len(y_true_test)))]).values
                elif len(fc) > len(y_true_test):
                    fc = fc.iloc[-len(y_true_test):]

                preds_sector[tk]['ARIMA'] = fc
                metrics_sector[tk]['ARIMA'] =
calc_metrics(y_true_test, fc)
            except Exception as e:
                logging.error(f"ARIMA falhou para {tk}: {e}")
```

```
        plot_preds(sector, list(sector_data.keys()), sector_data,
preds_sector)
        plot_error(metrics_sector, list(sector_data.keys()), sector)
        print_table(metrics_sector, sector)

# ====== 6. SCRIPT ENTRYPOINT
================================================
if __name__ == "__main__":
    os.makedirs(MODEL_DIR, exist_ok=True)
    main(SECTORS, START_DATE, END_DATE)

#Fase 2 - Análise de Sazonalidade e Previsão de Ações com Ajuste de
Hiperparâmetros usando Grid Search

import os
import math
import logging
import json
import csv
from typing import List, Dict, Tuple, Union

import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_percentage_error, r2_score

import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense, GRU, LSTM, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau
from tensorflow.keras.preprocessing.sequence import
TimeseriesGenerator

from pmdarima import auto_arima
from scipy.stats import kruskal

# APLICA ESTILO SEABORN AOS GRÁFICOS
sns.set_style("whitegrid")
plt.style.use("seaborn-v0_8-notebook")

# ====== 1. CONFIGURATION
================================================
START_DATE = "2021-01-01"
END_DATE = "2024-12-31"
```

```
MODEL_DIR = "trained_models"
SEQ_LENGTH = 30
BATCH_SIZE = 16
EPOCHS = 100
LINE_WIDTH = 1.5
COMPANY_NAME_MAP = {
    "MGLU3.SA": "Magazine Luiza", "LREN3.SA": "Lojas Renner",
"PCAR3.SA": "Pão de Açúcar", "BHIA3.SA": "Casas Bahia",
    "ITSA3.SA": "Itaúsa", "BBAS3.SA": "Banco do Brasil", "PSSA3.SA":
"Porto Seguro", "ABCB4.SA": "ABC Brasil",
    "ELET3.SA": "Eletrobras", "PETR3.SA": "Petrobras", "EQTL3.SA":
"Equatorial", "TAEE11.SA": "Taesa",
    "TOTS3.SA": "Totvs", "POSI3.SA": "Positivo", "INTB3.SA":
"Intelbras", "BMOB3.SA": "Bemobi Tech"
}
SECTORS = {
    "Varejo": ["MGLU3.SA", "LREN3.SA", "PCAR3.SA", "BHIA3.SA"],
    "Bancário": ["ITSA3.SA", "BBAS3.SA", "PSSA3.SA", "ABCB4.SA"],
    "Energia": ["ELET3.SA", "PETR3.SA", "EQTL3.SA", "TAEE11.SA"],
    "Tecnologia": ["TOTS3.SA", "POSI3.SA", "INTB3.SA", "BMOB3.SA"]
}
COLOR_MAP = {"GRU": "orange", "LSTM": "green", "MLP": "red", "ARIMA":
"purple", "Combined": "blue"}
# Hiperparâmetros para o Grid Search
PARAM_GRID = {
    "units": [32, 50],
    "dropout": [0.1, 0.2],
    "learning_rate": [1e-3, 5e-4]
}
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %
(levelname)s - %(message)s')


# ====== 2. UTILITIES & PREPROCESSING
=======================================
# (Funções idênticas às da Fase 1)
def get_subplot_shape(n:int) -> Tuple[int,int]:
    cols = math.ceil(math.sqrt(n))
    return math.ceil(n/cols), cols

def download_stock_data(tickers:List[str], start, end) ->
Dict[str,pd.DataFrame]:
    data = {}
    for t in tickers:
        try:
            df = yf.download(t, start=start, end=end, progress=False,
auto_adjust=True)
            if not df.empty: data[t] = df
            else: logging.warning(f"Nenhum dado para {t} no período
especificado.")
```

```
        except Exception as e: logging.error(f"Erro baixando {t}:
{e}")
    return data

def preprocess(df:pd.DataFrame):
    close = df[['Close']].dropna()
    scaler = MinMaxScaler()
    scaled = scaler.fit_transform(close)
    n = len(scaled)
    tr_end = int(.7 * n)
    vl_end = int(.85 * n)
    return scaled[:tr_end], scaled[tr_end:vl_end], scaled[vl_end:],
scaler, (tr_end, vl_end)

def make_gens(tr, vl, te):
    return (TimeseriesGenerator(tr, tr, length=SEQ_LENGTH,
batch_size=BATCH_SIZE),
            TimeseriesGenerator(vl, vl, length=SEQ_LENGTH,
batch_size=BATCH_SIZE),
            TimeseriesGenerator(te, te, length=SEQ_LENGTH,
batch_size=BATCH_SIZE))

def mlp_datasets(tr, vl, te):
    def build(arr):
        X, y = [], []
        for i in range(SEQ_LENGTH, len(arr)):
            X.append(arr[i-SEQ_LENGTH:i, 0])
            y.append(arr[i, 0])
        return np.array(X), np.array(y)
    return (*build(tr), *build(vl), *build(te))

def empresa_tem_sazonalidade(ticker: str,
path='sazonalidade_empresas.csv') -> bool:
    try:
        df = pd.read_csv(path)
        linha = df[df["Ticker"] == ticker]
        if linha.empty: return False
        return bool(linha["Sazonalidade_Provável"].iloc[0])
    except:
        return False

# ====== 3. MODEL TRAINING & PREDICTION
=======================================
def model_path(ticker:str, kind:str, suffix="") -> str:
    return os.path.join(MODEL_DIR, f"{ticker}_{kind}{suffix}.keras")

# FUNÇÃO PARA A FASE 2: OTIMIZAÇÃO COM GRID SEARCH
def tune_rnn_model(kind: str, ticker: str, gtr, gvl, gte, scaler,
te_data, param_grid, epochs):
    model_file = model_path(ticker, kind, suffix="_tuned")
```

```
    param_file = os.path.join(MODEL_DIR,
f"{ticker}_{kind}_tuned_params.json")
    if os.path.exists(model_file) and os.path.exists(param_file):
        logging.info(f"Carregando modelo {kind} (Otimizado) e
parâmetros cacheados para {ticker}.")
        with open(param_file, 'r') as f:
            best_params = json.load(f)
        return load_model(model_file), best_params

    best_model, best_params, best_mape = None, None, float('inf')

    for units in param_grid['units']:
        for dropout in param_grid['dropout']:
            for lr in param_grid['learning_rate']:
                logging.info(f"Tuning {kind} for {ticker}:
units={units}, dropout={dropout}, lr={lr:.0e}")
                model = Sequential()
                layer = GRU if kind == "GRU" else LSTM
                model.add(layer(units, return_sequences=True,
input_shape=(SEQ_LENGTH, 1)))
                model.add(Dropout(dropout))
                model.add(layer(units))
                model.add(Dropout(dropout))
                model.add(Dense(1))
                model.compile(optimizer=Adam(learning_rate=lr),
loss="mse")
                cb = [EarlyStopping(patience=10,
restore_best_weights=True)]
                model.fit(gtr, epochs=epochs, validation_data=gvl,
verbose=0, callbacks=cb)

                y_pred_scaled = model.predict(gte, verbose=0).ravel()
                y_true_orig =
scaler.inverse_transform(te_data[SEQ_LENGTH:])
                y_pred_orig =
scaler.inverse_transform(y_pred_scaled.reshape(-1, 1))
                mape = mean_absolute_percentage_error(y_true_orig,
y_pred_orig) * 100

                if mape < best_mape:
                    best_mape = mape
                    best_params = {'units': units, 'dropout': dropout,
'learning_rate': lr}
                    best_model = model

    logging.info(f"Melhor MAPE para {ticker} ({kind}): {best_mape:.4f}
com params {best_params}")
    best_model.save(model_file)
    with open(param_file, 'w') as f:
        json.dump(best_params, f)
```

```
    return best_model, best_params

def load_or_train_mlp(ticker:str, Xtr, ytr, Xvl, yvl, epochs:int) ->
Sequential:
    path = model_path(ticker, "MLP")
    if os.path.isfile(path):
        return load_model(path)
    net = Sequential([Dense(64, activation='relu',
input_shape=(SEQ_LENGTH,)), Dropout(0.1),
        Dense(32, activation='relu'), Dense(1)])
    net.compile(optimizer=Adam(1e-3), loss="mse")
    cb = [EarlyStopping(patience=6, restore_best_weights=True),
ReduceLROnPlateau(factor=0.2, patience=3)]
    net.fit(Xtr, ytr, epochs=epochs, batch_size=BATCH_SIZE,
validation_data=(Xvl, yvl), verbose=0, callbacks=cb)
    net.save(path)
    return net

# ====== 4. METRICS, PLOTTING & REPORTING
=====================================
# (Funções idênticas às da Fase 1)
def calc_metrics(y, ŷ):
    return dict(RMSE=np.sqrt(((y-ŷ)**2).mean()),
                MAPE=mean_absolute_percentage_error(y, ŷ)*100,
                R2=r2_score(y, ŷ))

def plot_preds(sector, stocks, actual, preds):
    r, c = get_subplot_shape(len(stocks))
    fig, axes = plt.subplots(r, c, figsize=(c * 6, r * 4),
squeeze=False)
    axes = np.atleast_1d(axes).ravel()

    real_style = {'lw': 1.8, 'alpha': 1.0, 'color': 'black'}
    pred_style = {'lw': 1.5, 'alpha': 0.8, 'linestyle': '--'}

    last_i = -1
    for i, tk in enumerate(stocks):
        ax = axes[i]
        last_i = i

        if not preds.get(tk) or not preds[tk]:
            logging.warning(f"Nenhuma predição encontrada para {tk}.
Subplot ficará em branco.")
            ax.set_title(f"Dados indisponíveis para {tk}")
            continue

        act_df = actual[tk]
        max_len = max(len(p) for p in preds[tk].values())
        ax.plot(act_df.index[-max_len:], act_df['Close'].values[-
max_len:], label="Preço Real", **real_style)
```

```
        for kind, p in preds[tk].items():
            plot_values = p.values if isinstance(p, pd.Series) else p
            ax.plot(act_df.index[-len(plot_values):],
plot_values.reshape(-1), label=kind, color=COLOR_MAP.get(kind),
**pred_style)

        ax.set_title(f"Predições {COMPANY_NAME_MAP.get(tk, tk)}")
        ax.tick_params(axis='x', rotation=45)
        ax.set_ylabel("Preço (R$)")

    for j in range(last_i + 1, len(axes)):
        fig.delaxes(axes[j])

    handles, labels = axes[0].get_legend_handles_labels()
    if handles:
        fig.legend(handles, labels, loc='upper center',
bbox_to_anchor=(0.5, 0), ncol=len(handles))

    fig.suptitle(f"Análise por Setor — {sector}", fontsize=16, y=0.98)
    plt.tight_layout(rect=[0, 0.05, 1, 0.95])
    plt.show()

def plot_error(mdict, stocks, sector):
    valid_stocks = [s for s in stocks if s in mdict and mdict[s]]
    if not valid_stocks: return
    models = sorted({m for s in valid_stocks for m in mdict[s]})
    if not models: return

    fig, axes = plt.subplots(1, 3, figsize=(15, 5.5))

    for i, met in enumerate(["MAPE", "RMSE", "R2"]):
        ax = axes[i]

        all_vals = [mdict[s].get(m, {}).get(met, np.nan) for s in
valid_stocks for m in models]
        all_vals_clean = [v for v in all_vals if pd.notna(v) and
np.isfinite(v)]

        if all_vals_clean and met != 'R2':
            q1, q3 = np.percentile(all_vals_clean, [25, 75])
            iqr = q3 - q1
            upper_bound = q3 + 1.5 * iqr
            if np.max(all_vals_clean) > upper_bound:
                ax.set_ylim(top=upper_bound * 1.1)

        x = np.arange(len(valid_stocks))
        w = 0.8 / len(models)
        for j, m in enumerate(models):
            vals = [mdict[s].get(m, {}).get(met, np.nan) for s in
```

```
valid_stocks]
            if met == 'R2':
                vals = [v if v >= 0 else np.nan for v in vals]
            bars = ax.bar(x + j * w - w/2, vals, w, label=m,
color=COLOR_MAP.get(m))

            for bar in bars:
                height = bar.get_height()
                if pd.isna(height): continue
                bottom_lim, top_lim = ax.get_ylim()
                if bottom_lim <= height <= top_lim:
                    visible_range = top_lim - bottom_lim
                    if abs(height) / visible_range > 0.1:
                        y_pos = bottom_lim + visible_range * 0.03
                        va = 'bottom'
                        ax.annotate(f'{height:.2f}', xy=(bar.get_x() +
bar.get_width() / 2, y_pos),
                                    ha='center', va=va, fontsize=8,
color='white', rotation=90)

        ax.set_xticks(x)
        ax.set_xticklabels([COMPANY_NAME_MAP.get(s, s) for s in
valid_stocks], rotation=45, ha='right')
        ax.set_ylabel(met + (" (%)" if met == "MAPE" else ""))
        ax.set_title(met)

        if met == "R2":
            ax.set_ylim(0, 1.05)

    handles, labels = axes[0].get_legend_handles_labels()
    fig.legend(handles, labels, loc='lower center',
bbox_to_anchor=(0.5, -0.12), ncol=len(labels))
    fig.suptitle(f"Métricas de Erro (Otimizadas via Grid Search) —
{sector}", fontsize=16, y=1.0)
    plt.tight_layout(rect=[0, 0.05, 1, 0.95])
    plt.show()

def print_table(mdict, sector):
    rows = []
    for stk, mods in mdict.items():
        for mod, val in mods.items():
            rows.append(dict(Setor=sector,
Ação=COMPANY_NAME_MAP.get(stk, stk), Modelo=mod, **val))
    if not rows: return
    print(f"\nResumo das Métricas (Otimizadas via Grid Search) —
{sector}")
    print(pd.DataFrame(rows).to_string(index=False,
float_format="%.4f"))

# ====== 5. MAIN EXECUTION LOOP
```

```
================================================
def main(sectors:Dict[str,List[str]], start, end, param_grid):
    # A análise de sazonalidade não precisa ser chamada aqui, pois já
é chamada no entrypoint.
    all_data = download_stock_data([tk for tks in sectors.values() for
tk in tks], start, end)

    for sector, tickers in sectors.items():
        logging.info(f"*** Processando Setor (Otimizado): {sector}
***")
        sector_data = {t: all_data[t] for t in tickers if t in
all_data}
        if not sector_data: continue
        preds_sector, metrics_sector = {}, {}

        for tk, df in sector_data.items():
            logging.info(f">>> Otimizando para o ticker: {tk}")
            preds_sector[tk], metrics_sector[tk] = {}, {}
            tr, vl, te, sc, splits = preprocess(df)
            if len(te) < SEQ_LENGTH: continue
            gtr, gvl, gte = make_gens(tr, vl, te)
            y_true_test =
sc.inverse_transform(te[SEQ_LENGTH:]).ravel()

            # --- ALTERAÇÃO PARA FASE 2: Chama a função de otimização
---
            net_gru, _ = tune_rnn_model("GRU", tk, gtr, gvl, gte, sc,
te, param_grid, EPOCHS)
            pred_gru = sc.inverse_transform(net_gru.predict(gte,
verbose=0)).ravel()
            preds_sector[tk]["GRU"] = pred_gru
            metrics_sector[tk]["GRU"] = calc_metrics(y_true_test,
pred_gru)

            net_lstm, _ = tune_rnn_model("LSTM", tk, gtr, gvl, gte,
sc, te, param_grid, EPOCHS)
            pred_lstm = sc.inverse_transform(net_lstm.predict(gte,
verbose=0)).ravel()
            preds_sector[tk]["LSTM"] = pred_lstm
            metrics_sector[tk]["LSTM"] = calc_metrics(y_true_test,
pred_lstm)

            pred_comb = (pred_gru + pred_lstm) / 2
            preds_sector[tk]["Combined"] = pred_comb
            metrics_sector[tk]["Combined"] = calc_metrics(y_true_test,
pred_comb)

            # Baselines
            Xtr, ytr, Xvl, yvl, Xte, yte = mlp_datasets(tr, vl, te)
```

```
            if len(Xte) > 0:
                mlp = load_or_train_mlp(tk, Xtr, ytr, Xvl, yvl,
EPOCHS)
                pred_mlp = sc.inverse_transform(mlp.predict(Xte,
verbose=0)).ravel()
                preds_sector[tk]['MLP'] = pred_mlp
                metrics_sector[tk]['MLP'] = calc_metrics(y_true_test,
pred_mlp)

            # ARIMA
            try:
                train_arima, test_arima =
df['Close'].iloc[:splits[1]], df['Close'].iloc[splits[1]:]
                train_arima_log = np.log(train_arima[train_arima > 0])
                sazonal = empresa_tem_sazonalidade(tk)
                arima_model = auto_arima(train_arima_log,
seasonal=sazonal, m=12 if sazonal else 1,
                                         stepwise=True,
suppress_warnings=True, error_action='ignore',
                                         max_p=3, max_q=3, max_P=2,
max_Q=2)
                fc_log =
arima_model.predict(n_periods=len(test_arima))
                fc = np.exp(fc_log)

                if len(fc) < len(y_true_test):
                    padding = np.full(len(y_true_test) - len(fc),
fc.iloc[-1] if not fc.empty else 0)
                    fc = pd.concat([fc, pd.Series(padding,
index=np.arange(len(fc), len(y_true_test)))]).values
                elif len(fc) > len(y_true_test):
                    fc = fc.iloc[-len(y_true_test):]

                preds_sector[tk]['ARIMA'] = fc
                metrics_sector[tk]['ARIMA'] =
calc_metrics(y_true_test, fc)
            except Exception as e:
                logging.error(f"ARIMA falhou para {tk}: {e}")

        plot_preds(sector, list(sector_data.keys()), sector_data,
preds_sector)
        plot_error(metrics_sector, list(sector_data.keys()), sector)
        print_table(metrics_sector, sector)

# ====== 6. SCRIPT ENTRYPOINT
================================================
def analisar_sazonalidade_wrapper():
    # Wrapper para chamar a análise de sazonalidade apenas uma vez
    path = "sazonalidade_empresas.csv"
```

```
    if os.path.exists(path):
        logging.info(f"Arquivo de sazonalidade '{path}' já existe.
Pulando a análise.")
        return

    def verificar_sazonalidade(df: pd.DataFrame, ticker: str,
coluna='Close', periodo=30) -> dict:
        resultado = {"Estatística_KW": None, "p_valor": None,
"Sazonalidade_Provável": None}
        serie = df[coluna].dropna()
        if len(serie) < periodo * 2: return resultado
        df_aux = df.copy()
        df_aux['Mês'] = df_aux.index.month
        grupos = [g[coluna].dropna().values for _, g in
df_aux.groupby('Mês')]
        grupos_validos = [g for g in grupos if len(g) > 0]
        if len(grupos_validos) < 2: return resultado
        stat, p = kruskal(*grupos_validos)
        resultado.update({"Estatística_KW": float(stat), "p_valor":
float(p), "Sazonalidade_Provável": bool(p < 0.05)})
        return resultado

    resultados = []
    logging.info("--- Iniciando Análise de Sazonalidade ---")
    for setor, tickers in SECTORS.items():
        for ticker in tickers:
            nome = COMPANY_NAME_MAP.get(ticker, ticker)
            try:
                df = yf.download(ticker, start=START_DATE,
end=END_DATE, progress=False, auto_adjust=True)
                if df.empty: continue
                res = verificar_sazonalidade(df, ticker)
                res.update({"Empresa": nome, "Ticker": ticker,
"Setor": setor})
                resultados.append(res)
            except Exception as e:
                logging.error(f"Falha na análise de sazonalidade para
{ticker}: {e}")
    df_result = pd.DataFrame(resultados).reindex(columns=["Setor",
"Empresa", "Ticker", "Estatística_KW", "p_valor",
"Sazonalidade_Provável"])
    df_result.to_csv(path, index=False)
    logging.info(f"Resultados de sazonalidade salvos em '{path}'")
    print("\n Tabela de Sazonalidade Detectada:\n")
    print(df_result.to_string(index=False, float_format="%.4f"))


if __name__ == "__main__":
    os.makedirs(MODEL_DIR, exist_ok=True)
```

```
    analisar_sazonalidade_wrapper()
    main(SECTORS, START_DATE, END_DATE, PARAM_GRID)
```