UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Modelos híbridos de indicadores técnicos e aprendizado profundo para previsão de preços de ações: uma análise comparativa

Jhoe Nascimento Sá

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Jhoe Nascimento Sá

Modelos híbridos de indicadores técnicos e aprendizado profundo para previsão de preços de ações: uma análise comparativa

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Diego Raphael Amancio

Versão original

São Carlos

2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi, ICMC/USP, com os dados fornecidos pelo(a) autor(a)

Sá, Jhoe Nascimento
N244m Modelos híbridos de indicadores técnicos e aprendizado profundo para previsão de preços de ações: uma análise comparativa / Jhoe Nascimento Sá ; orientador Diego Raphael Amancio. – São Carlos, 2024.
114 p. : il. (algumas color.) ; 30 cm.

Monografia (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. PREVISÃO (ANÁLISE DE SÉRIES TEMPORAIS). 2. COMBINAÇÃO: MÉTODOS DE AVALIAÇÃO. 3. APRENDIZAGEM PROFUNDA. I. Amancio, Diego Raphael, orient.. II. Título.

Jhoe Nascimento Sá

Hybrid models of technical indicators and deep learning for stock price prediction: A comparative analysis

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Dr. Diego Raphael Amancio

Original version

São Carlos

2024

Este trabalho é dedicado aos alunos da USP, como uma contribuição das Bibliotecas do Campus USP de São Carlos para o desenvolvimento e disseminação da pesquisa científica da Universidade.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a todos que contribuíram para a realização deste trabalho.

Agradeço ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, pela oportunidade de realizar este MBA e ao meu orientador, Diego Raphael Amancio, pela orientação, paciência e ensinamentos valiosos durante o desenvolvimento deste trabalho.

À minha família, especialmente ao meu avô Davi, que infelizmente não está mais entre nós, mas que sempre me incentivou a estudar e a me especializar, sou profundamente grato. Seu entusiasmo em me incentivar e sua calma, constância e foco sempre foram uma grande inspiração para mim. Agradeço também aos meus amigos pelo suporte constante e pela compreensão nas horas de ausência.

E aos professores, cujo conhecimento e dedicação enriqueceram minha formação, expresso meu sincero agradecimento.

A todos, minha mais sincera gratidão.

"O estudo, a busca da verdade e da beleza são domínios em que nos é consentido sermos crianças por toda a vida."

Albert Einstein

RESUMO

Sá, J. **Modelos híbridos de indicadores técnicos e aprendizado profundo para previsão de preços de ações: uma análise comparativa**. 2024. 114 p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Este estudo aborda a aplicação de modelos híbridos na previsão de preços de ações, combinando indicadores técnicos com técnicas de aprendizado profundo, como redes *Long Short-Term Memory* (*LSTM*). O contexto desta pesquisa está relacionado ao crescente interesse no uso de técnicas modernas de aprendizado de máquina para superar limitações dos métodos tradicionais de análise técnica e fundamentalista, oferecendo novas abordagens mais eficazes.

O estudo utilizou indicadores técnicos como *Média Móvel Simples* (*SMA*), *Média Móvel Exponencial* (*EMA*) e *Índice de Força Relativa* (*RSI*), além do filtro de *Kalman* para suavizar as séries de preços de fechamento. A metodologia incluiu a coleta de dados históricos do índice S&P 500 e a construção de modelos *LSTM*, *Bidirectional LSTM* (*BiLSTM*) e *Stacked LSTM*, tanto com indicadores técnicos quanto sem eles, e com a utilização do filtro de *Kalman* para comparação de desempenho. A avaliação dos modelos utilizou métricas como *Root Mean Squared Error* (*RMSE*), *Mean Absolute Error* (*MAE*) e coeficiente de determinação (R^2).

Os resultados mostram que a inclusão de indicadores técnicos e do filtro de *Kalman* pode melhorar significativamente a precisão das previsões, sendo o modelo *Stacked LSTM* o que apresentou o melhor desempenho. A pesquisa conclui que a integração de técnicas tradicionais e modernas tem um grande potencial para aprimorar as análises financeiras, possibilitando avanços significativos na gestão de portfólios e na tomada de decisão. Assim, este estudo contribui para a literatura ao demonstrar a eficácia dessas abordagens híbridas e ao sugerir novas possibilidades de aplicação no campo da gestão financeira.

Palavras-chave: Aprendizado profundo. LSTM. Indicadores técnicos. Previsão de ações. Filtro de Kalman.

ABSTRACT

Sá, J. **Hybrid models of technical indicators and deep learning for stock price prediction: A comparative analysis**. 2024. 114 p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

This study explores the application of hybrid models in stock price prediction, combining technical indicators with deep learning techniques, such as Long Short-Term Memory (LSTM) networks. The research is set in the context of growing interest in using modern machine learning techniques to overcome limitations of traditional technical and fundamental analysis methods, offering new and more effective approaches.

The study utilized technical indicators such as Simple Moving Average (SMA), Exponential Moving Average (EMA), and Relative Strength Index (RSI), in addition to the Kalman filter to smooth the closing price series. The methodology included the collection of historical data from the S&P 500 index and the construction of LSTM, Bidirectional LSTM (BiLSTM), and Stacked LSTM models, both with and without technical indicators, and the use of the Kalman filter for performance comparison. Model evaluation was conducted using metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2).

The results show that the inclusion of technical indicators and the Kalman filter can significantly improve prediction accuracy, with the Stacked LSTM model demonstrating the best performance. The research concludes that integrating traditional and modern techniques has great potential to enhance financial analysis, enabling significant advances in portfolio management and decision-making. Thus, this study contributes to the literature by demonstrating the effectiveness of these hybrid approaches and suggesting new application possibilities in financial management.

Keywords: Deep learning. LSTM. Technical indicators. Stock prediction. Kalman filter.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
abnTeX	ABsurdas Normas para TeX
IBGE	Instituto Brasileiro de Geografia e Estatística
LaTeX	Lamport TeX
USP	Universidade de São Paulo
USPSC	Campus USP de São Carlos
LSTM	- Long Short-Term Memory
Bi-LSTM	- Bidirectional Long Short-Term Memory
SMA	- Simple Moving Average
EMA	- Exponential Moving Average
RSI	- Relative Strength Index
RMSE	- Root Mean Squared Error
MAE	- Mean Absolute Error
MSE	- Mean Squared Error
R^2	- Coeficiente de Determinação
DA	- Direcional Accuracy
ICMC	- Instituto de Ciências Matemáticas e de Computação
USP	- Universidade de São Paulo
Kalman	- Filtro de Kalman

SUMÁRIO

1 INTRODUÇÃO

A previsão de preços de ações é uma tarefa desafiadora e essencial para investidores e gestores de portfólio, tradicionalmente confiada a analistas financeiros que utilizam métodos de análise técnica e fundamentalista. Nos últimos anos, entretanto, o avanço do aprendizado profundo (*deep learning*), uma vertente de inteligência artificial, vem modificando essas abordagens ao demonstrar uma capacidade notável de aprender e interpretar padrões complexos em dados financeiros, muitas vezes ruidosos e não lineares (Bao; Yue; Rao, 2017; Selvin *et al.*, 2017).

Esse interesse crescente pelo aprendizado profundo no contexto financeiro tem destacado a relevância de técnicas como redes de memória de longo e curto prazo (LSTM) e *autoencoders empilhados*, conforme discutido em (**??**). Revisões abrangentes da literatura, como a de (Sezer; Gudelek; Ozbayoglu, 2020), reforçam o potencial dessas técnicas, especialmente na previsão de séries temporais financeiras entre 2005 e 2019, evidenciando sua capacidade de capturar relações complexas em dados financeiros.

Apesar dos avanços observados, há lacunas na literatura em relação à integração de indicadores técnicos tradicionais com modelos de aprendizado profundo para a previsão de preços de ações (Jiang, 2021), (Kumbure *et al.*, 2022). Esta intersecção entre técnicas clássicas e abordagens modernas de inteligência artificial ainda não foi explorada extensivamente, sugerindo um vasto campo para investigação e desenvolvimento.

A hipótese central deste estudo é que modelos híbridos, que combinam indicadores técnicos com redes LSTM de aprendizado profundo, podem melhorar a precisão das previsões de preços de ações. A escolha das redes LSTM é justificada por sua habilidade de capturar dependências temporais de longo prazo, uma característica crucial ao lidar com dados financeiros, notoriamente voláteis e não lineares. Segundo (Jiang, 2021), esses tipos de rede de aprendizagem se destacam na análise de séries temporais financeiras devido à sua capacidade de processar dados sequenciais.

O objetivo deste estudo é investigar a eficácia de diferentes tipos de LSTMs aplicados à previsão de preços de ações, conforme a literatura referenciada, buscando identificar sinergias e aprimoramentos ao realizar a integração de indicadores técnicos, mantendo o foco na precisão, acurácia e confiabilidade das previsões.

2 FUNDAMENTAÇÃO TEÓRICA

A integração de técnicas avançadas de análise de dados tem se mostrado um campo promissor no contexto da previsão financeira, conforme evidenciado por (Nogueira; Lima, 2021). Neste estudo, houve uma melhora no desempenho do modelo de previsão ao adicionar mais camadas de diferentes técnicas e mecanismos, além de incluir novos indicadores técnicos como entrada para os modelos.

O artigo (Nogueira; Lima, 2021), intitulado "Previsão dos Preços de Abertura, Mínima e Máxima de Índices de Mercados Financeiros Usando a Associação de Redes Neurais LSTM", explora como o uso de redes neurais LSTM (*Long Short-Term Memory*) contribui para melhorar a precisão na previsão de séries temporais financeiras. Esse modelo é capaz de capturar padrões temporais e dependências de longo prazo, aspectos fundamentais para variáveis voláteis como preços de índices de mercado. A pesquisa, assim, reforça a relevância de métodos avançados para previsões robustas e acuradas, fundamentando a adoção dessas técnicas para decisões financeiras informadas.

A integração de técnicas avançadas de análise de dados tem se mostrado um campo promissor no contexto da previsão financeira. Este trabalho explora e compara modelos híbridos que combinam indicadores técnicos com métodos de aprendizado profundo, visando aprimorar a previsão de preços de ações. A hipótese central propõe que a fusão dessas técnicas possa superar as limitações dos métodos tradicionais, oferecendo uma abordagem mais robusta e precisa para o complexo mercado de ações. De acordo com (Yang, 2021), a aplicação do aprendizado profundo na previsão do mercado de ações tem mostrado progressos significativos recentemente. Ademais, uma revisão da literatura por (Kumbure *et al.*, 2022) corrobora a eficácia de técnicas de *machine learning* em previsões de mercado. Além disso, o M4 Competition, descrito por (Makridakis; Spiliotis; Assimakopoulos, 2020), que envolveu 100,000 séries temporais e 61 métodos de previsão, destaca a importância e a eficácia de metodologias híbridas em contextos analíticos complexos.

2.1 Aprendizado Profundo em Previsão de Mercado

O aprendizado profundo, uma subcategoria do aprendizado de máquina (machine learning), é notável por sua capacidade de capturar padrões complexos em vastos volumes de dados através de modelos computacionais que simulam a maneira como o cérebro humano processa informações. Esses modelos são chamados de redes neurais profundas devido às suas múltiplas camadas de processamento.

Uma das classes mais eficazes de redes neurais em contextos de dados sequenciais são as Redes Neurais Recorrentes (RNNs). As RNNs são projetadas para processar sequências

de dados, memorizando informações anteriores e utilizando-as para influenciar a saída atual, o que é essencial para tarefas como a análise de séries temporais financeiras. A capacidade de uma RNN de preservar informação de entradas anteriores é representada matematicamente por:

$$h_t = f(W \cdot [h_{t-1}, x_t] + b)$$

onde h_t é o estado oculto no tempo t, x_t é a entrada no tempo t, W e b são parâmetros da rede que são aprendidos durante o treinamento, e f é uma função de ativação não-linear, como a tangente hiperbólica ou ReLU.

No entanto, RNNs tradicionais muitas vezes falham em capturar dependências de longo prazo devido ao problema de desvanecimento do gradiente. Para superar isso, foram introduzidas as redes *Long Short-Term Memory* (LSTMs). LSTMs são uma especialização das RNNs que incorporam células de memória capazes de manter informações por longos períodos, e controladores de portas que regulam o fluxo de informações para dentro e para fora da célula. A dinâmica de uma célula LSTM pode ser descrita pelas seguintes equações:

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

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

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

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

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

$$h_t = o_t * \tanh(C_t)$$

Aqui, σ denota a função sigmóide, que ajuda a regular as portas de esquecimento f_t, entrada i_t, e saída o_t. C_t é o estado da célula que acumula informações ao longo do tempo, e $*$ representa o produto de Hadamard (elemento a elemento).

Em seu estudo de 2021, "Applications of deep learning in stock market prediction: Recent progress", (Jiang, 2021) destaca como as técnicas de *deep learning*, especialmente as LSTMs, têm avançado na previsão do mercado de ações. Ele enfatiza a habilidade desses modelos em lidar com a volatilidade e a não-linearidade dos dados de mercado, características que são intrínsecas às séries temporais financeiras e desafiadoras para métodos mais tradicionais de análise.

2.2 Redes Neurais Recorrentes (RNN) e o Problema de Desvanecimento do Gradiente

As Redes Neurais Recorrentes (RNNs) são uma classe de redes neurais projetadas para processar dados sequenciais, onde a ordem das informações importa, como em séries

temporais ou em linguagens naturais. Ao contrário das redes *feedforward*, em que as entradas são processadas de maneira independente, as RNNs possuem conexões recorrentes que permitem que o modelo mantenha um estado oculto que preserva informações das etapas anteriores. Isso permite que a rede tenha uma espécie de memória, essencial para tarefas como previsão de séries temporais.

A atualização de um estado oculto em uma RNN tradicional é definida como:

$$h_t = f(W \cdot [h_{t-1}, x_t] + b)$$

onde:

- h_t é o estado oculto atual,
- x_t é a entrada no tempo t,
- h_{t-1} é o estado oculto do passo anterior,
- W são os pesos da rede, e
- b é o viés da camada.

Essa capacidade de as RNNs manterem um histórico de entradas as torna adequadas para modelar dados que dependem do tempo, como os dados financeiros. Contudo, as RNNs tradicionais têm dificuldades em lidar com **dependências de longo prazo**, o que é conhecido como o **problema de desvanecimento do gradiente**.

2.2.0.1 Problema de Desvanecimento do Gradiente

Durante o treinamento das RNNs, o aprendizado ocorre por meio de um processo chamado *retropropagação do erro no tempo* (*backpropagation through time, BPTT*), onde os gradientes são calculados e usados para ajustar os pesos da rede. No entanto, em sequências longas, os gradientes dos pesos podem se tornar extremamente pequenos à medida que a rede retropropaga para os primeiros elementos da sequência, o que impede a rede de aprender padrões que dependem de eventos passados.

Esse problema ocorre principalmente devido ao uso de funções de ativação como a tangente hiperbólica (tanh) ou a função sigmóide, cujos gradientes diminuem rapidamente em valores extremos, resultando na incapacidade de as RNNs tradicionais capturarem **dependências de longo prazo**. Como resultado, as RNNs tornam-se ineficazes para prever séries temporais complexas, onde eventos passados longínquos podem influenciar de forma significativa as previsões futuras.

2.2.1 Long Short-Term Memory (LSTM)

Para superar as limitações das RNNs tradicionais, foram introduzidas as **Redes de Memória de Longo e Curto Prazo (Long Short-Term Memory, LSTM)**. As LSTMs são uma arquitetura específica das RNNs, projetada para mitigar o problema de desvanecimento do gradiente e permitir que o modelo aprenda e retenha informações por períodos de tempo mais longos.

2.2.1.1 Estrutura e Funcionamento das LSTMs

A LSTM utiliza uma célula de memória capaz de armazenar informações ao longo do tempo e uma série de **portas** que controlam o fluxo de informações, permitindo que o modelo decida o que manter, o que atualizar e o que esquecer em cada etapa do processo de aprendizagem.

Abaixo está um diagrama que ilustra a estrutura interna de uma célula LSTM, onde as portas de esquecimento, entrada e saída controlam o que é mantido, atualizado e transmitido para a próxima célula.

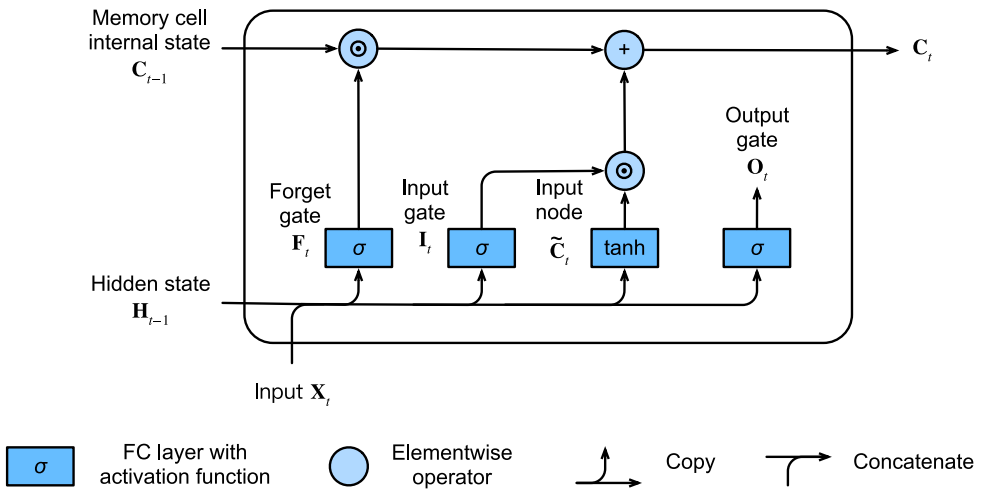

Figura 1 – Estrutura da célula LSTM. Adaptado de *Dive into Deep Learning*, https://d2l.ai/chapter_recurrent-modern/lstm.html

O **estado da célula**, que mantém a memória de longo prazo, é atualizado com base nas decisões tomadas pelas portas de esquecimento e de entrada. Ele é uma combinação da memória anterior, filtrada pela porta de esquecimento, e das novas informações, filtradas pela porta de entrada:

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

Esse mecanismo de controle permite que a LSTM mantenha informações relevantes por longos períodos e descarte informações desnecessárias, tornando-a mais robusta para lidar com séries temporais complexas e dados sequenciais de longo prazo.

Explicação dos Componentes da Célula LSTM

Cada componente mostrado na Figura 1 possui uma função específica:

- **Porta de Esquecimento (f_t)**: Esta porta controla a quantidade de informação a ser esquecida da célula de memória anterior. É calculada por uma função sigmoide que toma como entrada o estado oculto anterior h_{t-1} e a entrada atual x_t, ajustada pelos pesos e o viés da porta de esquecimento:

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

  O valor da sigmoide varia entre 0 e 1, permitindo um controle granular sobre a informação mantida.

- **Porta de Entrada (i_t)**: A porta de entrada decide quanto das informações da entrada atual será adicionado ao estado da célula. É também calculada com uma função sigmoide para produzir um valor de ativação que regula a entrada de novas informações:

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

- **Candidata à Memória ($\tilde{C}_t$)**: Este vetor armazena as informações novas que estão sendo consideradas para armazenamento na célula. Ele é calculado usando a função tanh, o que permite que os valores estejam no intervalo de -1 a 1:

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

- **Estado da Célula (C_t)**: O estado da célula acumula informações ao longo do tempo, combinando a retenção da célula anterior com as novas informações. Ele é atualizado pela fórmula:

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

  Esse cálculo permite que a célula armazene tanto informações antigas quanto as novas informações que passam pela porta de entrada.

- **Porta de Saída (o_t)**: A porta de saída determina a informação a ser enviada para o próximo estado oculto, baseando-se no estado atual da célula. É também regulada por uma função sigmoide:

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

- **Estado Oculto (h_t)**: O estado oculto é a saída da célula LSTM para a próxima unidade ou para a próxima camada da rede. Ele é obtido aplicando a função tanh ao estado da célula C_t e multiplicando-o pela ativação da porta de saída:

$$h_t = o_t \cdot \tanh(C_t)$$

A estrutura da LSTM é projetada para superar as limitações das redes recorrentes tradicionais, especialmente na retenção de informações importantes por longos períodos. Cada porta desempenha um papel fundamental, controlando a entrada, retenção e saída de informações em cada etapa do processamento sequencial. Essa arquitetura torna a LSTM adequada para tarefas em que o contexto e o histórico têm impacto significativo no resultado final.

2.3 LSTM Bidirecional e Stacked

Enquanto o LSTM básico processa informações em uma única direção temporal, os modelos LSTM bidirecionais (BiLSTM) e Stacked LSTM introduzem aprimoramentos para capturar padrões mais complexos e profundos em dados sequenciais.

2.3.1 LSTM Bidirecional

O LSTM bidirecional (BiLSTM) processa os dados em ambas as direções: do passado para o futuro e do futuro para o passado. Isso permite que o modelo capture informações contextuais que podem estar presentes tanto antes quanto depois de um ponto específico na sequência temporal.

Matematicamente, o BiLSTM pode ser descrito por duas cadeias de LSTMs independentes:

$$\overrightarrow{h_t} = f(\overrightarrow{W} \cdot [h_{t-1}, x_t] + \overrightarrow{b})$$
$$\overleftarrow{h_t} = f(\overleftarrow{W} \cdot [h_{t+1}, x_t] + \overleftarrow{b})$$

Aqui, $\overrightarrow{h_t}$ e $\overleftarrow{h_t}$ representam as saídas da LSTM processando a sequência em direções opostas. A saída final é geralmente uma concatenação dessas duas saídas, oferecendo uma representação mais rica do contexto:

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$$

Essa arquitetura é particularmente útil em tarefas onde o contexto de ambos os lados de um ponto na sequência é importante, como em tarefas de processamento de linguagem natural e reconhecimento de fala.

2.3.2 Stacked LSTM

O Stacked LSTM é uma variação onde múltiplas camadas de LSTM são empilhadas, uma sobre a outra. Cada camada adicional permite que o modelo aprenda representações em níveis mais altos de abstração.

A operação de um Stacked LSTM pode ser expressa como:

$$h_t^{(l)} = f(W^{(l)} \cdot [h_{t-1}^{(l)}, h_t^{(l-1)}] + b^{(l)})$$

onde l indica a camada atual, $h_t^{(l-1)}$ é a saída da camada anterior, e $h_t^{(l)}$ é a saída da camada l atual. O Stacked LSTM é eficaz para capturar relações temporais em séries temporais complexas, como as encontradas em dados financeiros de alta frequência.

Esses modelos avançados de LSTM, ao lado do LSTM básico, proporcionam ferramentas poderosas para a previsão de séries temporais, especialmente em domínios como o mercado financeiro, onde a identificação de padrões complexos e de longo prazo é crucial.

2.4 Indicadores Técnicos na Análise de Ações

Os indicadores técnicos são ferramentas amplamente empregadas na análise de ações e outros ativos financeiros. Baseados em dados históricos de preços e volumes, eles têm como objetivo identificar padrões e prever possíveis direções futuras dos preços. Diferentes estudos demonstram que esses indicadores, como a Média Móvel Simples (SMA), Média Móvel Exponencial (EMA) e Índice de Força Relativa (RSI), ajudam não apenas a visualizar tendências, mas também a avaliar a força do mercado e o momentum das ações. Dessa forma, é possível compreender as forças subjacentes que influenciam as flutuações de preços, sendo fundamentais na identificação de pontos de inflexão, como potenciais reversões de tendência ou consolidações de preço.

Pesquisas como a de Qiu e Song (Qiu; Song, 2016) mostram que modelos de redes neurais otimizados para prever movimentos de mercado alcançam maior precisão ao incorporar variáveis como indicadores técnicos. Os autores demonstram que o uso de dados históricos e técnicos, junto a algoritmos de otimização, pode aprimorar a capacidade do modelo em antecipar a direção dos índices de ações, ressaltando o valor da combinação entre dados de mercado e *machine learning* para previsões mais robustas.

Adicionalmente, Patel et al. (Patel *et al.*, 2015) exploraram diversas técnicas de *machine learning*, como redes neurais artificiais, máquinas de vetor de suporte e florestas aleatórias, utilizando indicadores técnicos como variáveis de entrada. Eles concluíram que, quando representados como dados determinísticos de tendência, esses indicadores aumentam o desempenho dos modelos, oferecendo uma compreensão quantitativa da volatilidade e comportamento do mercado. Tal combinação potencializa as estratégias preditivas e destaca o papel dos indicadores técnicos na identificação de padrões e tendências complexas.

Com o avanço das metodologias de *machine learning* e aprendizado profundo, a integração de indicadores técnicos com modelos como as LSTM tem mostrado resultados promissores na previsão de preços, ao combinar a interpretação de padrões históricos com

a capacidade desses modelos de aprender e capturar dependências temporais complexas nos dados financeiros. A seguir, são apresentadas as fórmulas e descrições dos indicadores técnicos mais utilizados, que também serão empregados neste trabalho.

- **Média Móvel Simples (SMA)**:

$$SMA = \frac{A_1 + A_2 + \ldots + A_n}{n}$$

  Onde $A_1, A_2, \ldots, A_n$ são os preços de fechamento do ativo durante n períodos.

- **Média Móvel Exponencial (EMA)**:

$$EMA_{\text{hoje}} = (Preço_{\text{hoje}} \times K) + (EMA_{\text{ontem}} \times (1 - K))$$

  Onde K é o fator de suavização, $K = \frac{2}{(n+1)}$, e n é o número de períodos.

- **Índice de Força Relativa (RSI)**:

$$RSI = 100 - \frac{100}{1 + RS}$$

$$RS = \frac{\text{Média de ganhos}}{\text{Média de perdas}}$$

  O RSI é calculado com base na média de ganhos e perdas ao longo de um período específico, geralmente 14 dias.

- **Convergência/Divergência de Média Móvel (MACD)**:

$$MACD = EMA_{12} - EMA_{26}$$

$$Sinal_{\text{MACD}} = EMA_9 \text{ do } MACD$$

  O MACD é a diferença entre a média móvel exponencial de 12 dias e a de 26 dias. A linha de sinal é a média móvel exponencial de 9 dias do MACD.

2.5 Filtro de Kalman para Suavização e Previsão de Séries Temporais

O filtro de Kalman é uma técnica amplamente utilizada para suavização e previsão de dados em séries temporais, especialmente em domínios onde as observações são sujeitas a ruídos e incertezas. Originalmente desenvolvido para resolver problemas de navegação e controle, ele foi adaptado para aplicações em diversos campos, incluindo finanças. No contexto financeiro, o filtro de Kalman permite uma estimativa otimizada de variáveis desconhecidas, considerando tanto os valores observados quanto o estado anterior da variável em questão. Esse processo ajuda a lidar com a volatilidade e o ruído característicos de dados de preços, proporcionando uma estimativa precisa mesmo em ambientes não-lineares e incertos (Wan; Merwe, 2000).

2.5.1 Estrutura Matemática do Filtro de Kalman

O filtro de Kalman é descrito por duas etapas principais: *predição* e *atualização*, aplicadas recursivamente.

- **Etapa de Predição:**

$$\hat{x}_{t|t-1} = A\hat{x}_{t-1|t-1} + Bu_t,$$
$$P_{t|t-1} = AP_{t-1|t-1}A^T + Q.$$

- **Etapa de Atualização:**

$$K_t = \frac{P_{t|t-1}H^T}{HP_{t|t-1}H^T + R},$$
$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H\hat{x}_{t|t-1}),$$
$$P_{t|t} = (I - K_tH)P_{t|t-1}.$$

Nessas equações, $\hat{x}_{t|t-1}$ é a estimativa predita, $P_{t|t-1}$ representa a incerteza dessa predição, Q é a variância do processo, R é a variância do ruído, e K_t é o ganho de Kalman, que define o peso dado à nova observação.

2.5.2 Integração com Modelos de Aprendizado Profundo

O filtro de Kalman pode ser integrado com redes neurais LSTM para aprimorar a precisão e a robustez das previsões financeiras. Nessa abordagem híbrida, o filtro é aplicado aos dados de entrada ou às previsões, suavizando ruídos e melhorando a modelagem das dinâmicas de mercado. Essa combinação permite uma interpretação mais robusta das tendências em séries temporais complexas e ruidosas (Coskun *et al.*, 2017).

A escolha do filtro de Kalman para suavização de dados financeiros é motivada por sua capacidade de lidar com ruído e incerteza de forma adaptativa e eficiente. Comparado a outras técnicas de suavização, como médias móveis, o Kalman é mais dinâmico, ajustando-se continuamente conforme novas observações são recebidas. Segundo Wan e Van Der Merwe (2000), o filtro de Kalman destaca-se em cenários com alta incerteza devido ao seu processo iterativo de ajuste, permitindo uma atualização constante das previsões com base em novas informações (Wan; Merwe, 2000).

2.6 Avaliação de Modelo

Este capítulo explora metodologias recomendadas e eficazes para avaliar modelos LSTM, com base nas diretrizes sugeridas por artigos, enfatizando uma comparação entre a versão com mais variáveis do modelo e uma versão simplificada sem indicadores técnicos.

2.6.1 Métricas de Avaliação

As métricas específicas identificadas por (Makridakis; Spiliotis; Assimakopoulos, 2020) na competição M4 e os insights de (Kumbure *et al.*, 2022) são cruciais para avaliar o desempenho de modelos LSTM em dados financeiros. As principais métricas incluem:

Para embasar a escolha de métricas como RMSE, MAE e R^2 para a avaliação de modelos financeiros, a pesquisa de Makridakis, Spiliotis e Assimakopoulos (Makridakis; Spiliotis; Assimakopoulos, 2020) é particularmente relevante. O estudo destaca a importância dessas métricas para comparações de desempenho, especialmente no contexto da competição M4. Além disso, os insights de Kumbure et al. (Kumbure *et al.*, 2022) reforçam a necessidade dessas métricas ao avaliar o desempenho de modelos LSTM em dados financeiros.

As principais métricas utilizadas incluem:

- **RMSE (Root Mean Squared Error)**: mede a magnitude média dos erros das previsões em relação aos valores reais, penalizando erros maiores.
- **MAE (Mean Absolute Error)**: calcula a média dos erros absolutos, fornecendo uma métrica mais robusta frente a outliers.
- **R² (Coeficiente de Determinação)**: avalia a proporção da variância nos dados que é explicada pelo modelo, indicando a qualidade do ajuste.
- **Erro Quadrático Médio (MSE) e Raiz do Erro Quadrático Médio (RMSE)**: Avaliam a média dos erros ao quadrado, sendo o RMSE particularmente útil por estar na mesma unidade dos dados de previsão.
- **Erro Absoluto Médio (MAE)**: Mede a média dos valores absolutos dos erros, fornecendo uma compreensão direta das discrepâncias entre previsões e valores reais.
- **Coeficiente de Determinação (R^2)**: O R^2 mede a proporção da variância dos dados que é explicada pelo modelo. É definido como:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

  onde y_i são os valores observados, $\hat{y}_i$ são os valores previstos pelo modelo, e $\bar{y}$ é a média dos valores observados. Um R^2 de 1 indica um ajuste perfeito, enquanto um valor de 0 indica que o modelo não explica nenhuma variância dos dados.

- **Direção de Precisão (DA)**: Essencial para mercados financeiros, esta métrica verifica a capacidade do modelo de prever corretamente a direção da mudança de preço das ações.

2.6.2 Validação Cruzada em Série Temporal

A validação cruzada em séries temporais é recomendada para testar a eficácia dos modelos LSTM com dados sequenciais e não vistos anteriormente. Diferentemente da validação cruzada tradicional, onde os dados são aleatoriamente divididos, a validação cruzada em série temporal respeita a ordem cronológica dos dados, evitando o vazamento de informações futuras no treinamento. O conjunto de dados é dividido em várias partes consecutivas, onde cada conjunto de teste é precedido pelo correspondente conjunto de treinamento que inclui todos os dados até aquele ponto específico no tempo. Esta abordagem assegura que a avaliação do modelo seja tanto realista quanto rigorosa.

2.6.3 Comparação com Modelo sem Indicadores Técnicos

Ao avaliar a eficácia do modelo LSTM, uma estratégia fundamental é comparar seu desempenho com a versão simplificada do mesmo modelo que opera sem o uso de indicadores técnicos. Esta comparação fornece uma visão clara da contribuição real desses indicadores na precisão do modelo. Ao omitir indicadores técnicos, pode-se avaliar se a complexidade adicional do modelo completo é justificada pelo aumento na precisão das previsões ou se um modelo mais simples poderia atingir resultados semelhantes com menos variáveis e menor risco de overfitting.

2.6.4 Fluxo de Avaliação

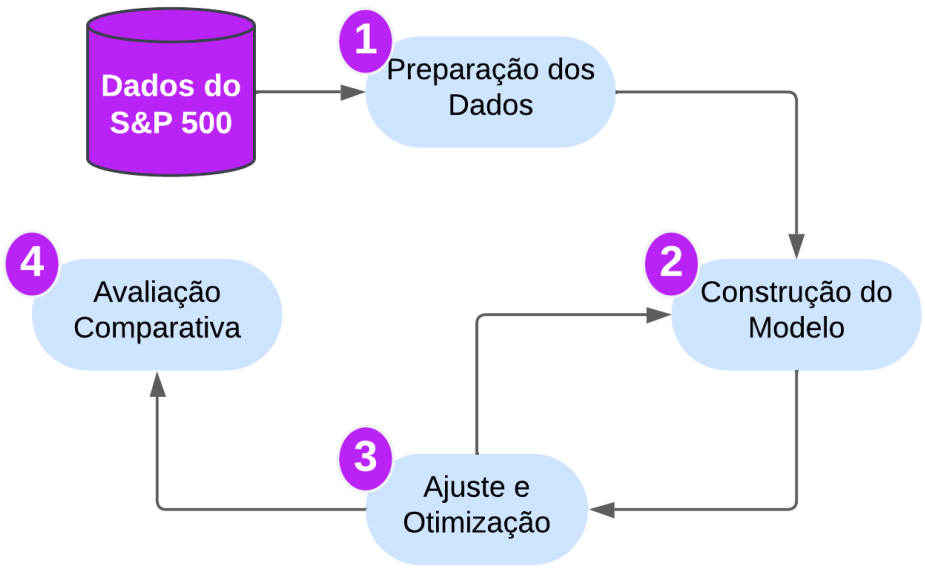

Figura 2 – Diagrama de fluxo para realização da avaliação.

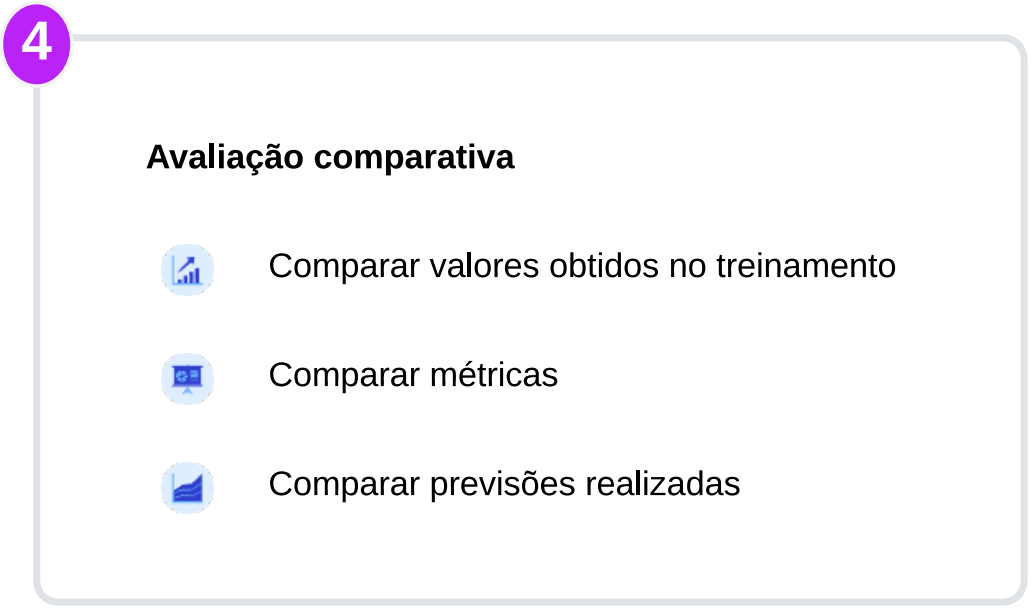

Figura 3 – Itens de avaliação comparativa.

O processo de avaliação deve seguir uma abordagem sistemática que inclui preparação de dados, construção e ajuste de modelo, seguida de uma fase rigorosa de avaliação:

- **Preparação dos dados:** envolve a coleta, filtragem, limpeza e normalização dos dados, além da seleção cuidadosa de variáveis explicativas para o modelo completo.
- **Construção do modelo:** escolha dos modelos e dos parâmetros fixos, como percentual da amostra para teste e validação, quantidade de épocas, paciência do modelo (quantidade de vezes em que ele pode não melhorar antes de interromper o treinamento), quantidade de dias para cálculo dos indicadores técnicos e intervalos para a recalibragem dos parâmetros autocalibrados.
- **Ajuste e otimização:** baseado nos resultados de perda e validação, o modelo calibra seus próprios parâmetros, como número de neurônios por camada e taxa de aprendizagem, conforme as faixas definidas anteriormente, sempre buscando um valor menor de perda nas validações.
- **Avaliação comparativa:** utilização das métricas de avaliação em ambos os modelos (completo e sem indicadores) para determinar a eficácia de cada abordagem.

3 ESTADO DA ARTE

Este capítulo examina o desenvolvimento e a eficácia de modelos híbridos que combinam técnicas de *deep learning*, como redes *Long Short-Term Memory (LSTM)*, com indicadores técnicos na previsão de preços de ações. A literatura recente sugere uma tendência crescente na adoção de abordagens que integram métodos tradicionais de análise técnica com avanços contemporâneos em aprendizado de máquina e *deep learning*. Essa integração visa melhorar a acurácia e a robustez dos modelos de previsão em séries temporais financeiras.

3.0.1 Fundamentos Teóricos e Aplicações de LSTM em Previsões Financeiras

A aplicação de redes *LSTM* no campo da previsão de séries temporais financeiras é bem documentada. (Fischer; Krauss, 2018) detalham como as redes *LSTM* podem capturar dependências de longo prazo em dados de séries temporais, o que é crucial para entender a dinâmica dos mercados financeiros. Por outro lado, (Kumar; Raj; Mahajan, 2020) exploram o uso de janelas deslizantes com *LSTM*, *RNN* e *CNN*, destacando a capacidade das *LSTMs* de lidar com a volatilidade e a não-linearidade dos dados de preços de ações.

3.0.2 Integração de Indicadores Técnicos com Modelos de Deep Learning

A pesquisa de (Bao; Yue; Rao, 2017) introduz um quadro de *deep learning* que utiliza *autoencoders* empilhados e *LSTM* para a análise de séries temporais financeiras, sugerindo que a adição de camadas de *autoencoders* pode ajudar na extração de características mais significativas dos dados brutos. Esta abordagem é um exemplo de como técnicas avançadas de *machine learning* podem ser aplicadas em conjunto com indicadores técnicos para melhorar a previsão de preços.

3.0.3 Avanços Recentes e Desafios

Ao discutir a literatura recente sobre o uso de aprendizado profundo em previsões financeiras, a obra de Sezer, Gudelek e Ozbayoglu (Sezer; Gudelek; Ozbayoglu, 2020) é uma referência fundamental. Em *"Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review: 2005–2019"*, os autores realizam uma revisão abrangente sobre as principais metodologias de deep learning aplicadas a séries temporais financeiras, incluindo LSTM e modelos híbridos. Eles destacam a eficácia dessas abordagens para captar padrões complexos e sutis nos dados financeiros, além de abordar a fusão entre métodos tradicionais e modernos. Essa revisão é essencial para compreender os avanços recentes na área e os desafios enfrentados, como a adaptação dos modelos a mudanças abruptas no mercado.

Em complemento, Jiang (Jiang, 2021) também discute esses progressos e ressalta a necessidade de modelos que respondam rapidamente às dinâmicas de mercado, alinhando-se à perspectiva de Sezer et al. sobre a complexidade crescente e a importância dos modelos híbridos na previsão de séries financeiras.

3.0.4 Revisão das Competições e Benchmarks

(Makridakis; Spiliotis; Assimakopoulos, 2020) relatam os resultados da Competição *M4*, que envolveu 100.000 séries temporais e 61 métodos de previsão. A competição oferece um *benchmark* significativo para avaliar a eficácia das abordagens híbridas, revelando que a combinação de diferentes métodos pode resultar em melhorias significativas na precisão das previsões.

3.0.5 Implicações para Futuras Pesquisas

A literatura revisada indica um consenso crescente sobre o potencial das abordagens híbridas em melhorar a previsão de preços de ações. No entanto, também identifica a necessidade de mais pesquisas para explorar como diferentes combinações de indicadores técnicos e modelos de *deep learning* podem ser otimizados para diferentes condições de mercado. (Kumbure *et al.*, 2022) ressaltam a importância de dados de qualidade e a seleção adequada de características como fatores críticos para o sucesso desses modelos.

3.0.6 Conclusão do Capítulo de Estado da Arte

Este capítulo revisou a evolução dos modelos híbridos que combinam *deep learning* e análise técnica, evidenciando a popularidade crescente do uso de redes *LSTM* integradas com indicadores técnicos na previsão de preços de ações. As abordagens híbridas destacadas na literatura mostram-se promissoras em captar a complexidade dos mercados financeiros, superando métodos puramente tradicionais ou de *deep learning* isolados. Os modelos *LSTM*, em particular, demonstram alta capacidade de lidar com a volatilidade e não-linearidade intrínsecas das séries temporais financeiras, proporcionando previsões mais robustas e precisas.

Contudo, os estudos também identificam desafios persistentes, como a necessidade de maior eficiência computacional, a adaptação a rápidas mudanças nos mercados e a escolha precisa de indicadores técnicos que maximizem o desempenho dos modelos. Em paralelo, competições como a *M4* revelam *benchmarks* importantes, sugerindo que abordagens combinadas podem oferecer uma precisão superior.

No estado da arte atual, observa-se um consenso sobre o potencial dos modelos híbridos, embora se reconheça a importância de mais estudos empíricos que explorem diferentes combinações de métodos e indicadores para otimizar o desempenho em condições variadas de mercado. Esta revisão reforça a necessidade contínua de inovação nos modelos

e estratégias de previsão, abrindo caminho para investigações futuras que possam superar as limitações atuais e explorar novas arquiteturas, como *Transformers* e redes neurais baseadas em grafos, que estão emergindo como alternativas promissoras no campo de previsão financeira.

4 METODOLOGIA

A metodologia proposta neste trabalho visa superar as limitações das abordagens tradicionais na previsão de preços de ações ao combinar indicadores técnicos com redes neurais *Long Short-Term Memory* (LSTM). Esse modelo híbrido é motivado pela necessidade de técnicas que capturem a complexidade das séries temporais financeiras, caracterizadas por sua volatilidade e ruído.

A utilização do filtro de *Kalman* desempenha um papel crucial na suavização dos dados de preços de fechamento, proporcionando uma análise mais precisa ao reduzir as flutuações e focar nas tendências subjacentes. Além disso, a inclusão de indicadores técnicos, como Média Móvel Simples (SMA), Média Móvel Exponencial (EMA) e Índice de Força Relativa (RSI), enriquece os dados com informações contextuais sobre o comportamento do mercado, permitindo que os modelos LSTM capturem padrões temporais com maior profundidade.

Essa abordagem busca preencher uma lacuna na literatura ao integrar esses indicadores tradicionais com modelos avançados de *deep learning*, o que não só aumenta a robustez das previsões, mas também abre novas perspectivas para aplicações na gestão de portfólios e tomada de decisão no mercado financeiro.

A Figura 1 apresenta o diagrama que fornece uma visão geral do fluxo dos processos, passando pelas etapas de coleta, processamento e treinamento, culminando na previsão de novos valores para o índice S&P 500 e a comparação dos resultados de cada modelo de previsão.

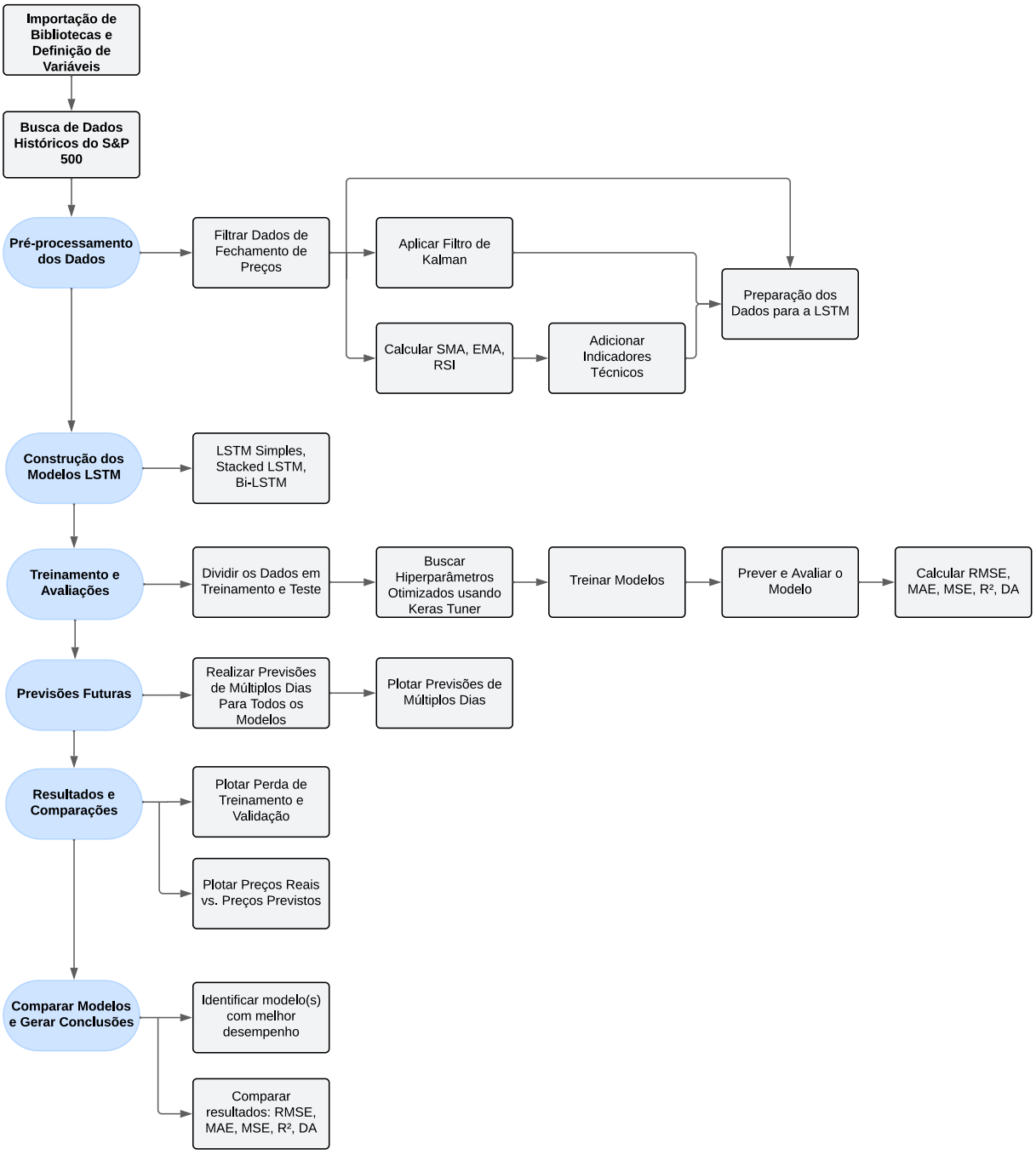

Figura 4 – Diagrama do Fluxo do Processo

A Figura 2 apresenta, nas caixas em cinza, cada modelo construído, e os diferencia conforme o fluxo.

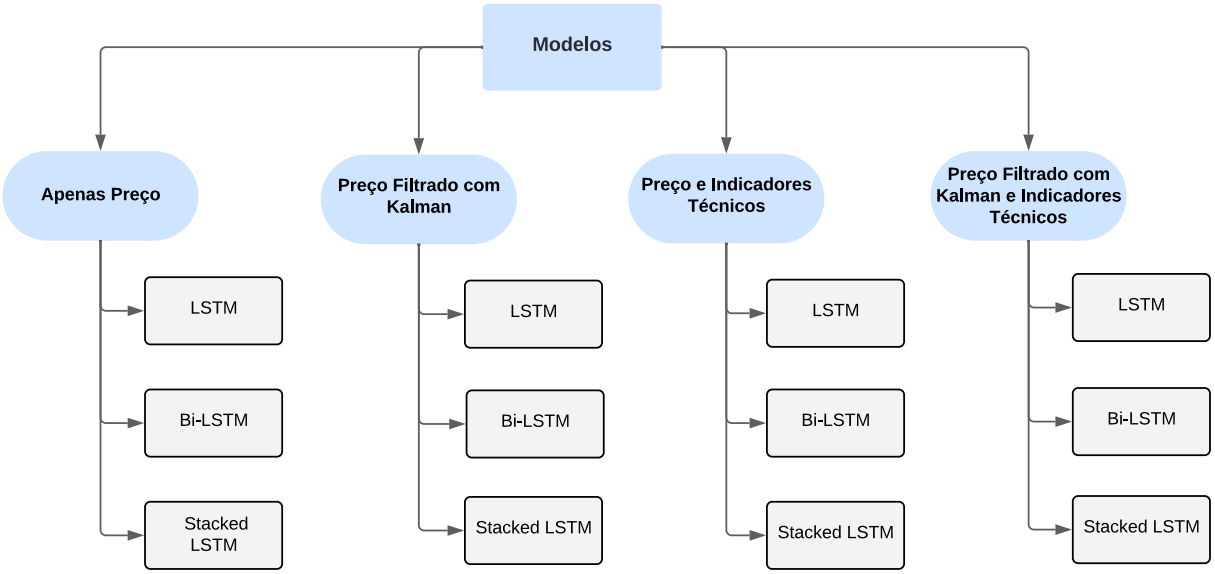

Figura 5 – Diagrama de Modelos Confeccionados

- **Importação de Bibliotecas e Definição de Variáveis:** O processo se inicia com a importação das bibliotecas necessárias, como Pandas, NumPy, TensorFlow e Keras, que são essenciais para a manipulação de dados e a construção de modelos de aprendizado profundo. Além disso, são definidas variáveis globais, como o número de dias da janela de previsão e os indicadores técnicos. A configuração dessas variáveis no início permite maior controle e flexibilidade no ajuste dos modelos, facilitando a experimentação com diferentes parâmetros e cenários.

- **Busca de Dados Históricos do S&P 500:** Utiliza-se a API `yfinance` para importar dados históricos do índice S&P 500, focando principalmente nos preços de fechamento ajustados. Esses dados são críticos para entender as tendências de mercado e servem como a base para a análise subsequente. A escolha de preços de fechamento ajustados ajuda a evitar distorções causadas por eventos corporativos, como dividendos e desdobramentos de ações.

- **Pré-processamento dos Dados:**

  - **Filtragem de Dados de Fechamento de Preços:** Seleciona-se apenas os preços de fechamento necessários para a análise, eliminando dados desnecessários e facilitando o foco nas informações relevantes para a previsão.

  - **Aplicação do Filtro de Kalman:** Para suavizar as séries temporais de preços e reduzir o ruído, aplica-se o filtro de Kalman. Esse filtro é especialmente útil para lidar com a volatilidade dos mercados financeiros, fornecendo uma estimativa mais precisa dos preços.

  - **Cálculo de Indicadores Técnicos:** Indicadores técnicos, como a Média Móvel Simples (SMA), Média Móvel Exponencial (EMA) e Índice de Força Relativa (RSI), são calculados. Esses indicadores ajudam a capturar diferentes aspectos do comportamento do mercado, como tendências de preços e momentos de sobrecompra ou sobrevenda.
  - **Preparação dos Dados para a LSTM:** Os dados são normalizados e transformados em sequências temporais, adequando-os para entrada nos modelos de LSTM. A normalização é crucial para garantir que os diferentes escalas de dados não influenciem indevidamente o treinamento do modelo.

- **Construção dos Modelos LSTM:**

  - **Modelos com Base nos Dados Filtrados e Indicadores Técnicos:** São definidos diversos modelos baseados em diferentes combinações de dados:
    * **Apenas Preço:** Utiliza LSTM, Bi-LSTM e Stacked LSTM sem indicadores adicionais.
    * **Preço Filtrado com Kalman:** Utiliza os modelos com os dados de preço filtrados pelo Kalman.
    * **Preço e Indicadores Técnicos:** Incorpora indicadores como SMA, EMA e RSI nos modelos.
    * **Preço Filtrado com Kalman e Indicadores Técnicos:** Combina o filtro de Kalman com indicadores técnicos.

    Cada um desses modelos busca capturar diferentes aspectos das tendências de mercado, oferecendo uma perspectiva ampla sobre o desempenho futuro do S&P 500.

- **Treinamento e Avaliações:**

  - **Divisão dos Dados:** Os dados são separados em conjuntos de treinamento e teste, garantindo que a avaliação do modelo seja feita de forma independente dos dados usados para treinamento.
  - **Busca de Hiperparâmetros:** Utiliza-se o `Keras Tuner` para otimizar os hiperparâmetros dos modelos, buscando a configuração que proporcione o melhor desempenho preditivo.
  - **Treinamento dos Modelos:** Os modelos definidos são treinados usando os conjuntos de dados de treinamento. Durante o treinamento, são utilizados callbacks, como `EarlyStopping`, para prevenir overfitting.
  - **Previsão e Avaliação do Modelo:** Após o treinamento, são realizadas previsões utilizando os dados de teste. As previsões são avaliadas por meio de

métricas como RMSE (Erro Médio Quadrático da Raiz), MAE (Erro Médio Absoluto), MSE (Erro Quadrático Médio), R^2 (Coeficiente de Determinação) e DA (Precisão Direcional).

- **Previsões Futuras:** Usando os modelos treinados, realiza-se previsões de múltiplos dias para cada tipo de modelo, de acordo com a janela de previsão definida. As previsões são visualizadas por meio de gráficos, permitindo uma análise clara e comparativa das tendências futuras do mercado.

- **Resultados e Comparações:**

  - **Plotagem de Perdas de Treinamento e Validação:** Gráficos são gerados para mostrar a evolução das perdas durante o treinamento, ajudando a identificar possíveis problemas de overfitting ou underfitting.

  - **Plotagem de Preços Reais vs. Preços Previstos:** Os preços reais e os preços previstos pelos modelos são comparados visualmente. Esta comparação ajuda a avaliar a precisão dos modelos e identificar possíveis melhorias.

- **Comparação de Modelos e Conclusões:**

  - **Identificação dos Modelos com Melhor Desempenho:** Com base nas métricas calculadas, são identificados os modelos que apresentam melhor desempenho na previsão de preços.

  - **Comparação de Resultados:** Avalia-se os resultados das diferentes abordagens para determinar qual modelo, ou combinação de modelos e técnicas, oferece o melhor desempenho para as previsões de preços. Esta análise inclui a eficácia de indicadores técnicos, filtros aplicados e tipos de redes neurais utilizadas.

4.1 Escolha do Índice S&P 500

O índice S&P 500 é amplamente reconhecido como um dos principais indicadores de desempenho das ações de grandes empresas nos Estados Unidos. Ele inclui 500 das maiores empresas de capital aberto dos EUA, abrangendo uma variedade significativa de setores, o que o torna um dos índices mais representativos do mercado acionário norte-americano (Fischer; Krauss, 2018). A escolha do S&P 500 para este estudo se justifica pela sua representatividade do mercado de ações americano e pela disponibilidade de dados históricos confiáveis e detalhados (Jiang, 2021). Além disso, sua volatilidade e liquidez tornam o índice um cenário robusto para testes de modelos de previsão, especialmente em análises de séries temporais complexas, como mostrado em estudos de deep learning aplicados ao mercado financeiro (Bao; Yue; Rao, 2017; Kumbure *et al.*, 2022).

4.2 Coleta de Dados

Os dados históricos do índice S&P 500 foram coletados pela biblioteca yfinance em Python, abrangendo preços de fechamento diários de 1º de janeiro de 1990 a 1º de janeiro de 2024, sendo de 1º de janeiro de 2020 a 1º de janeiro de 2024 o intervalo utilizado no treinamento e teste dos modelos.

A decomposição temporal do S&P 500 no período de 1990 a 2024 mostra uma tendência de crescimento com interrupções em crises como a de 2008 e a pandemia de COVID-19, evidenciando uma sazonalidade clara e padrões cíclicos que o modelo LSTM pode capturar. No intervalo de 2020 a 2024, a decomposição destaca uma tendência de recuperação pós-pandemia e oscilações de curto prazo, favorecendo previsões mais precisas ao reduzir a influência de ciclos econômicos mais longos.

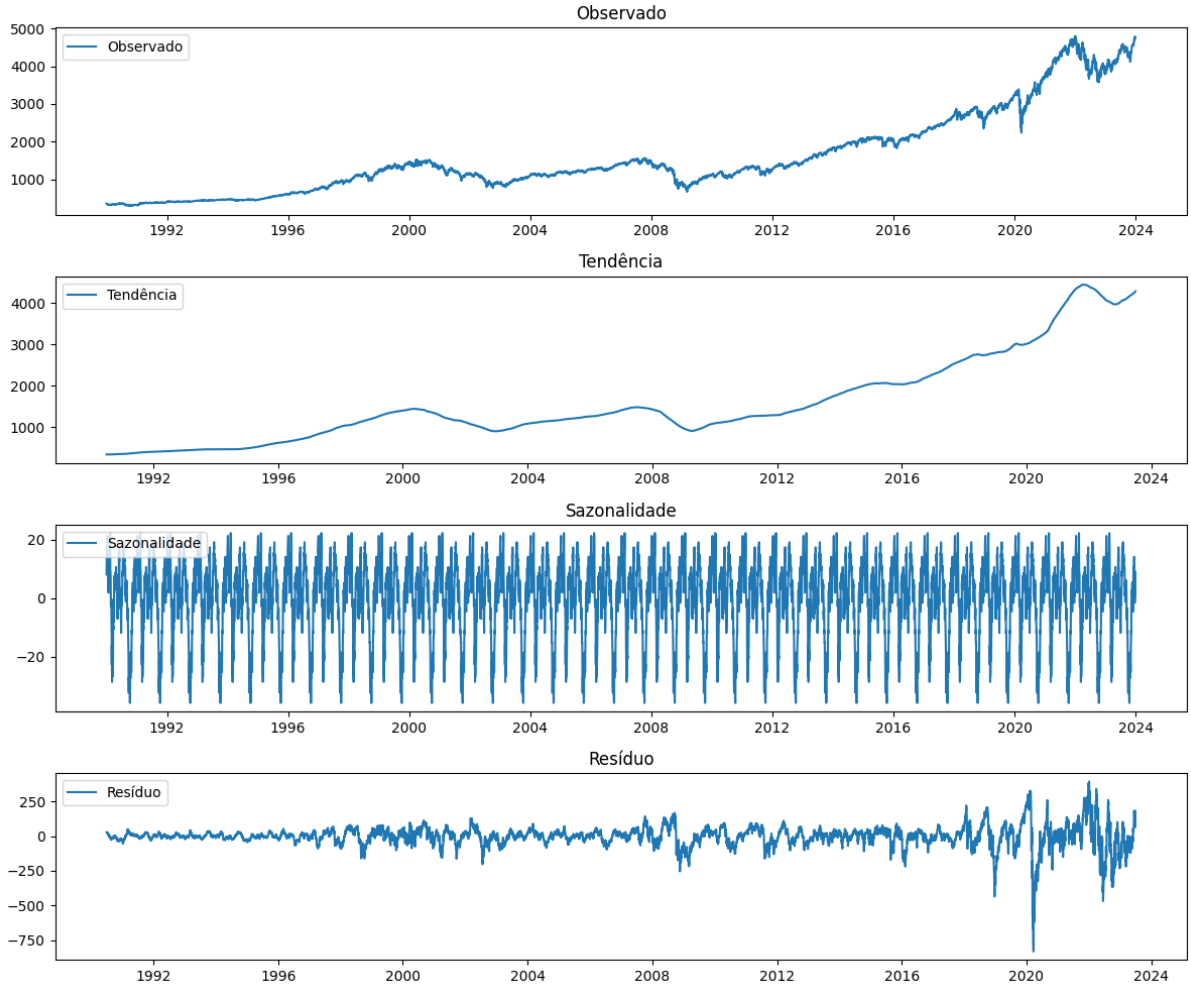

Figura 6 – Decomposição temporal do índice S&P 500 entre 1990 e 2024, destacando tendências, sazonalidade e eventos econômicos significativos.

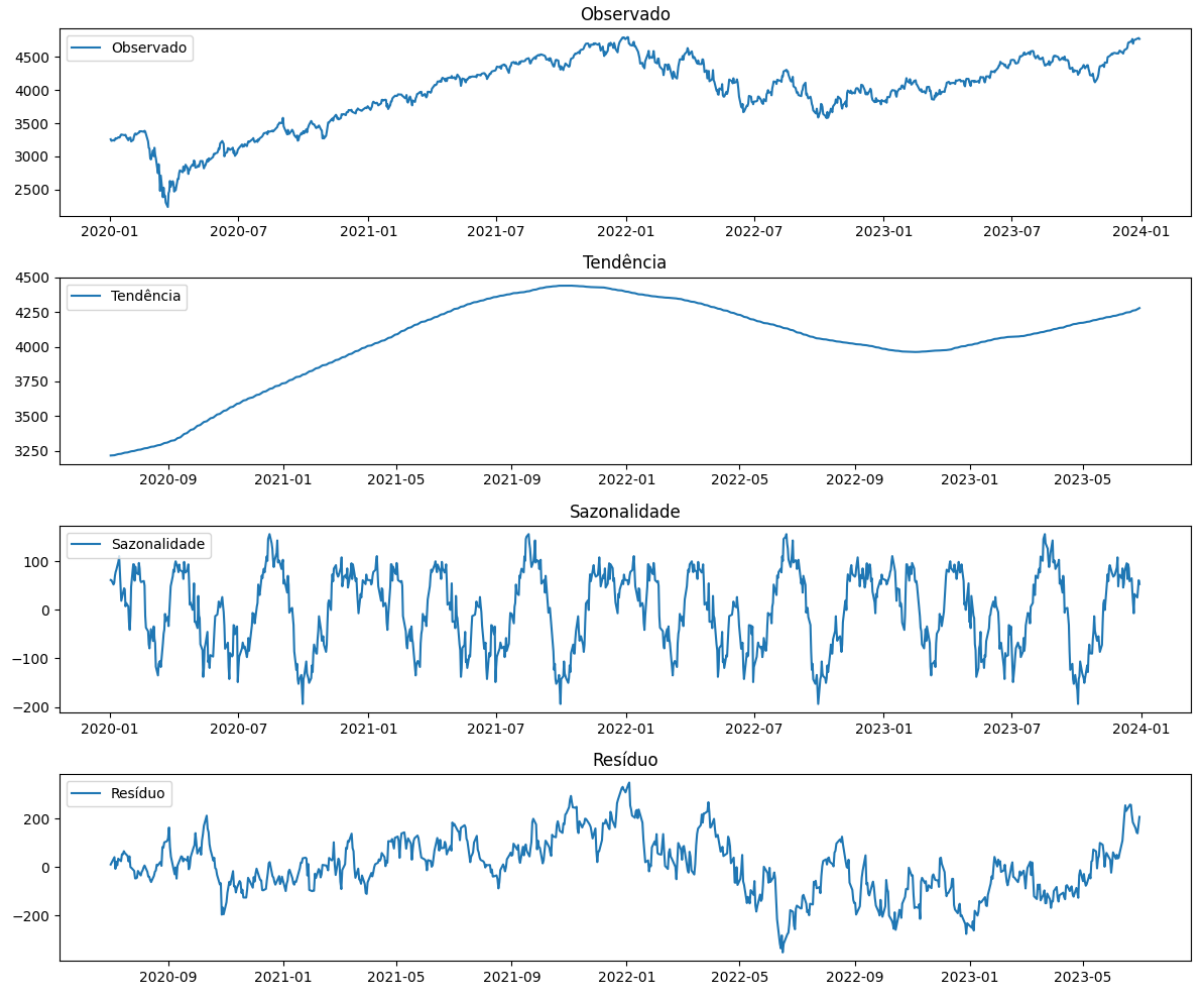

Figura 7 – Decomposição temporal do índice S&P 500 no período de 2020 a 2024, com foco nas flutuações de curto prazo e recuperação pós-pandemia.

A análise dos gráficos também nos revela que o componente de resíduo não se comporta como um ruído branco, apresentando padrões e picos em momentos específicos. Esses comportamentos indicam que a decomposição baseada apenas em preço não captura completamente todos os fatores que influenciam as variações do índice. Isso sugere que um modelo baseado exclusivamente nos preços não é suficiente para representar a dinâmica completa da série, já que eventos externos e choques econômicos introduzem variabilidade não explicada pelos componentes de tendência e sazonalidade.

A decomposição de séries temporais em componentes sazonais e de tendência permite distinguir padrões recorrentes e a direção geral dos dados ao longo do tempo. O componente sazonal mostra as flutuações mensais típicas em um ano, indicando aumentos e quedas regulares. Esse padrão é útil para prever impactos de eventos que ocorrem sazonalmente.

Nos gráficos a seguir, a sazonalidade mensal revela oscilações anuais, como o

aumento gradual de atividades nos últimos meses do ano. No gráfico de tendência, observa-se um crescimento anual consistente, indicando uma expansão de longo prazo influenciada por fatores macroeconômicos.

Especificamente, no primeiro conjunto de gráficos (1990-2024), há uma oscilação mensal mais leve, mas a tendência anual mostra um crescimento acentuado a partir de 2010, sugerindo uma aceleração econômica ou aumento de atividades. Já no segundo conjunto (2020-2024), a sazonalidade destaca mais volatilidade, com picos negativos e o principal pico positivo no mes 08, e a tendência anual mantém crescimento, com uma leve estabilização entre 2021 e 2023, antes de uma nova alta em 2024.

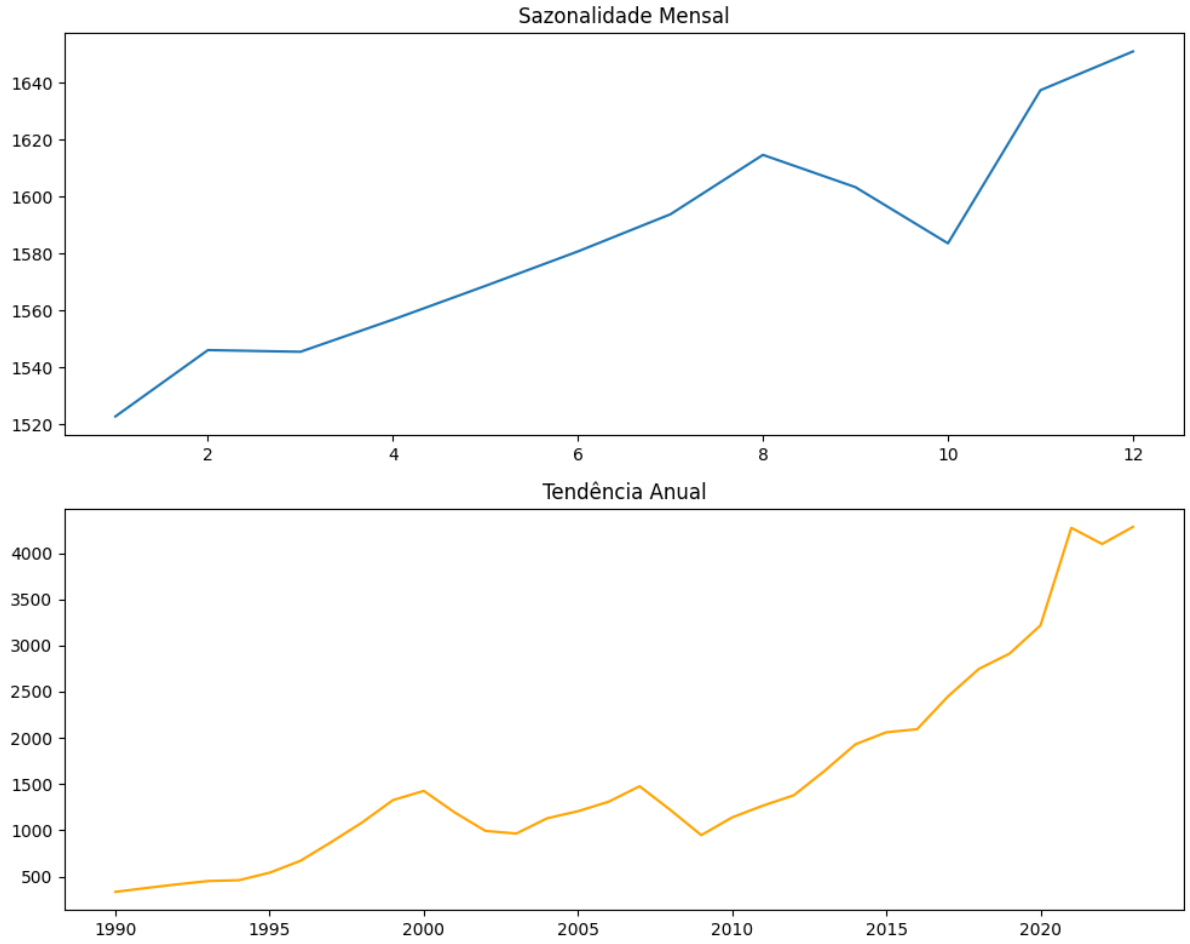

Figura 8 – Sazonalidade e tendência do índice S&P 500 de 1990 a 2024, evidenciando padrões cíclicos e oscilações de longo prazo.

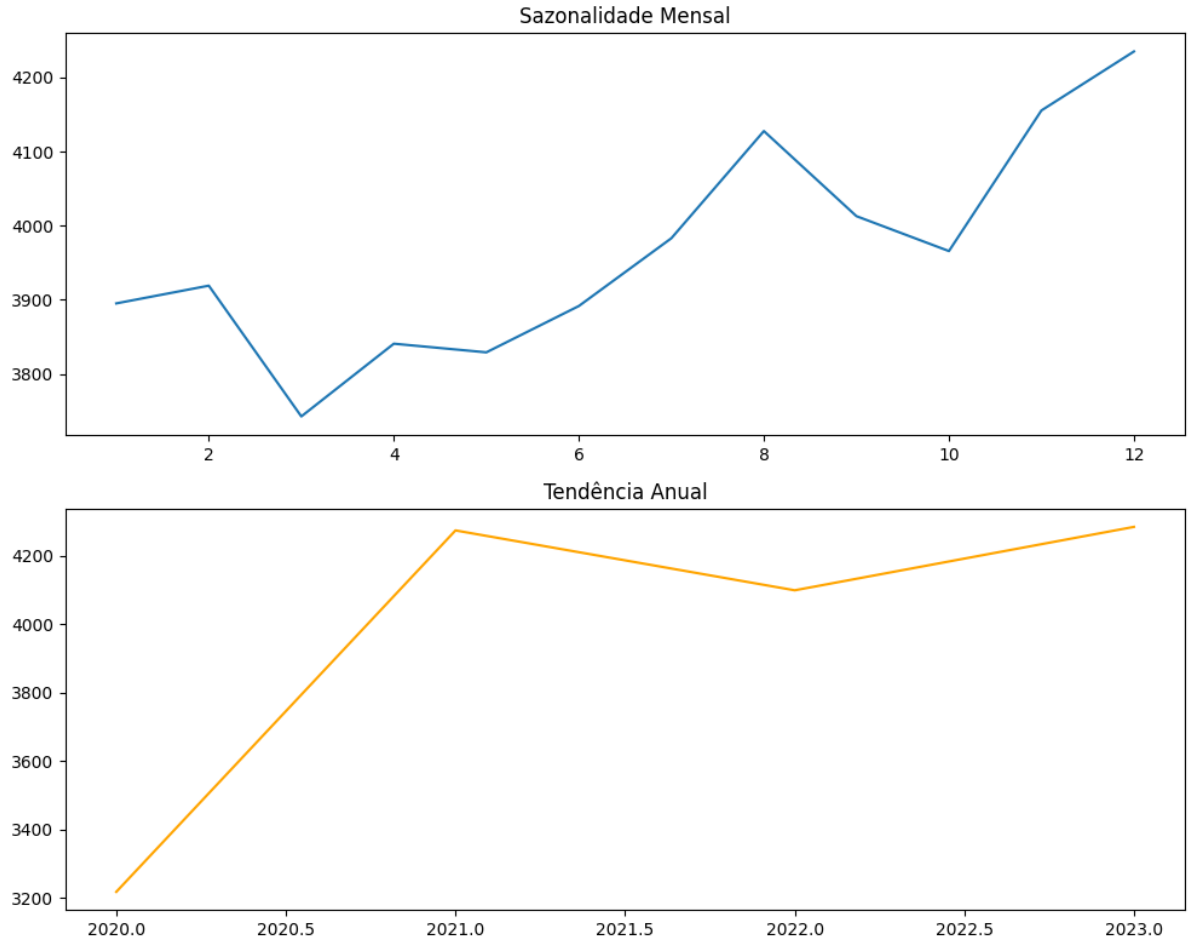

Figura 9 – Sazonalidade e tendência do índice S&P 500 no período de 2020 a 2024, destacando recuperação e variações de curto prazo.

Os gráficos a seguir mostram a função de autocorrelação (ACF) e a função de autocorrelação parcial (PACF) para a série temporal. A ACF (à esquerda) indica a persistência temporal nos dados, enquanto a PACF (à direita) ajuda a identificar correlações diretas entre valores defasados.

Essas análises orientam tanto modelos ARIMA, ajudando a definir ordens AR e MA, quanto redes LSTM, sugerindo a profundidade de memória temporal a ser capturada pela rede. A figura 10 cobre o período completo, enquanto a figura 11 foca nos anos mais recentes, de 2020 a 2024.

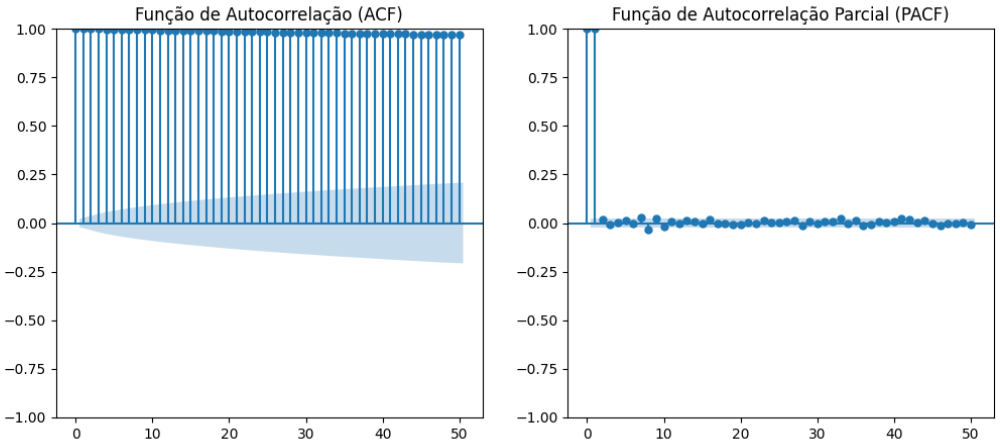

Figura 10 – Funções de Autocorrelação e Autocorrelação Parcial (PACF) do índice S&P 500 entre 1990 e 2024, mostrando a dependência temporal em longo prazo.

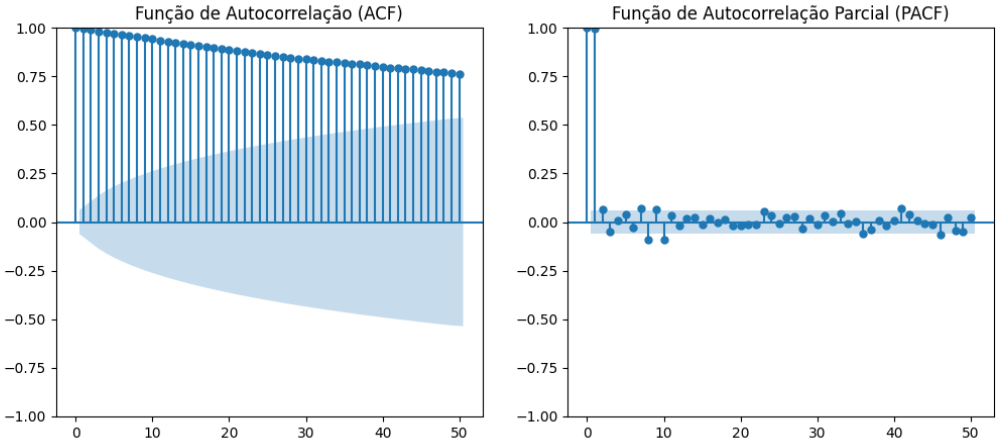

Figura 11 – Funções de Autocorrelação e Autocorrelação Parcial (PACF) do índice S&P 500 no período de 2020 a 2024, evidenciando dependências temporais em curto prazo.

O gráfico de retorno diário do S&P 500 permite observar a volatilidade diária do índice ao longo do tempo, destacando eventos de alta e baixa intensidade, como picos de retorno e quedas acentuadas em períodos de crise. Esse comportamento é importante para análise financeira, pois captura as oscilações de curto prazo do mercado, oferecendo uma visão detalhada da reação do índice a fatores econômicos e políticos.

A métrica de Direcional Accuracy (DA) é utilizada para avaliar a precisão de um modelo em prever a direção desses retornos diários, ou seja, se o índice irá subir ou cair em determinado dia. Um alto DA indica que o modelo é eficaz em capturar a direção

das mudanças, um aspecto crucial na análise de volatilidade e no desenvolvimento de estratégias de investimento baseadas em previsões.

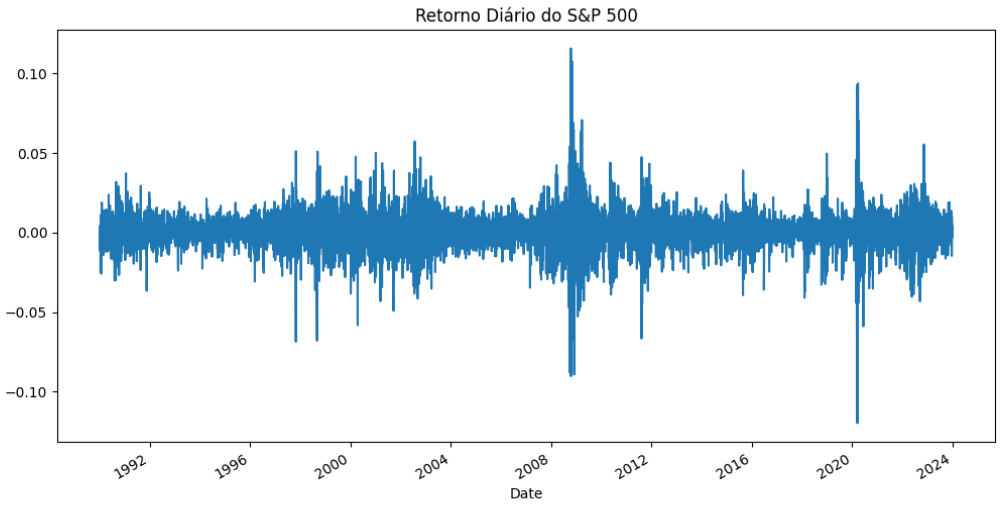

Figura 12 – Retorno diário do índice S&P 500 de 1990 a 2024, mostrando as variações percentuais diárias e sua volatilidade ao longo do período.

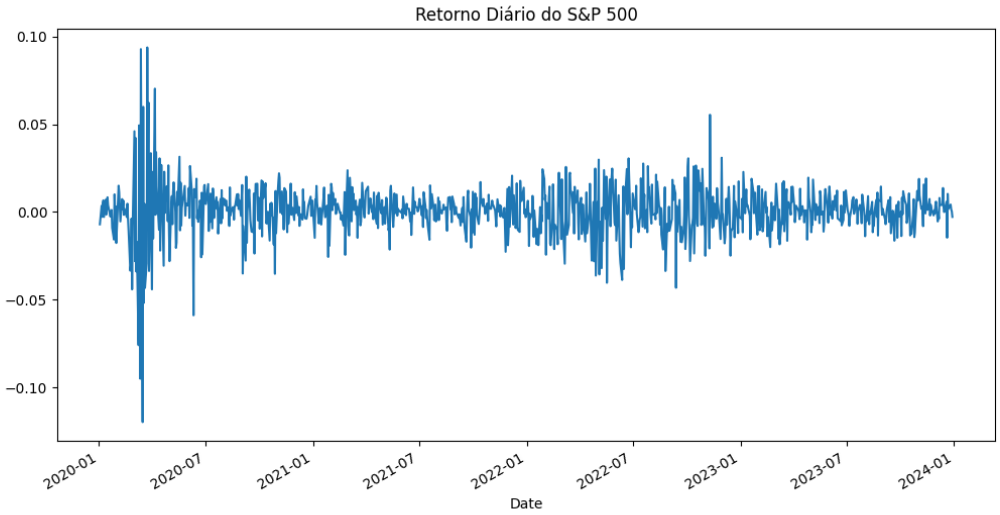

Figura 13 – Retorno diário do índice S&P 500 entre 2020 e 2024, destacando as variações diárias e a volatilidade recente pós-pandemia.

Estes gráficos, em conjunto, ilustram a tendência de crescimento do índice ao longo dos anos, com destaque para os períodos de maior volatilidade, como os mencionados

acima. Esta análise gráfica permite visualizar as tendências de longo prazo e identificar os principais eventos econômicos que afetaram o mercado.

A Figura 14 apresenta o histograma dos preços de fechamento diários do S&P 500 no período de 1990 a 2024, revelando uma distribuição multimodal dos preços, com picos em torno de 500, 1000, 1500, 2500 e entre 3500 e 4000.

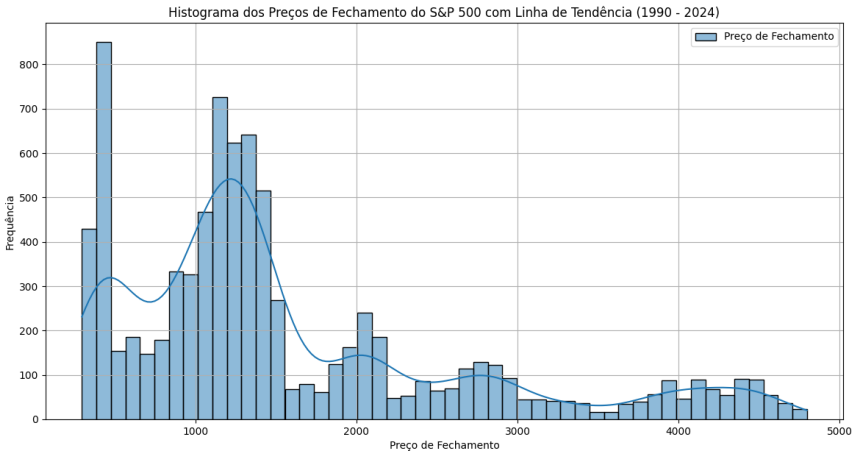

Figura 14 – Histograma dos preços de fechamento diários do S&P 500 (1990-2024).

Já a Figura 15 apresenta o histograma dos preços de fechamento para o período mais recente, de 2020 a 2024, destacando a concentração de preços em níveis mais elevados, refletindo a nova fase de valorização do índice.

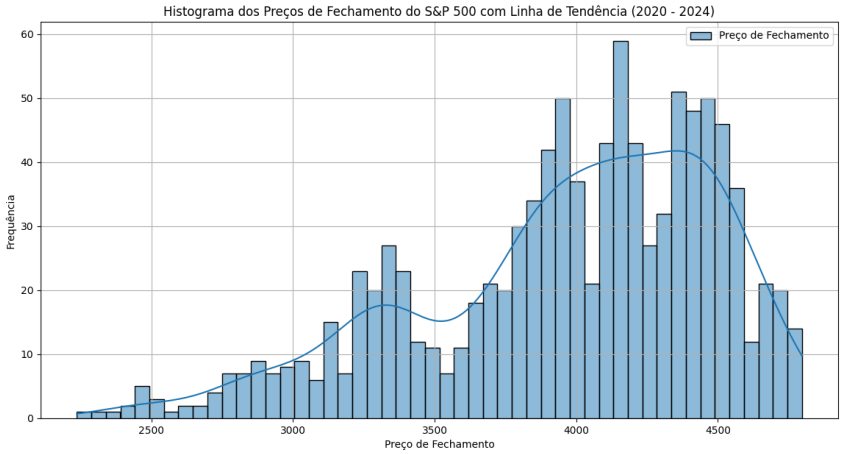

Figura 15 – Histograma dos preços de fechamento diários do S&P 500 (2020-2024).

Os gráficos mostram os preços de fechamento diários do S&P 500 de 1º de janeiro de 1990 a 1º de janeiro de 2024, e em um olhar ampliado de 1º de janeiro de 2020 a 1º

de janeiro de 2024. O primeiro histograma evidencia uma distribuição multimodal dos preços de fechamento, refletindo diferentes fases econômicas e eventos de mercado. A linha de tendência suavizada acompanha essas flutuações. Além disso, observa-se que o patamar de 4.000 pontos é um fenômeno ainda pouco frequente no longo prazo. Conforme o gráfico linear anterior, este nível de preços tem se tornado mais frequente a partir de 2021, mostrando um novo padrão de estabilização e indicando uma nova fase de valorização do índice.

4.3 Aplicação do Filtro de Kalman

Para suavizar os dados de fechamento e mitigar o ruído intrínseco dos preços diários de mercado, aplicamos o filtro de Kalman. Este filtro, desenvolvido inicialmente para sistemas de controle e estimativa de estados, é um algoritmo recursivo poderoso que permite a obtenção de estimativas precisas de estados desconhecidos em sistemas dinâmicos por meio de uma sequência de medições ruidosas (Kalman, 1960). No contexto da previsão financeira, o filtro de Kalman ajuda a isolar a tendência subjacente nos preços de fechamento, removendo flutuações aleatórias que podem mascarar padrões relevantes para análise (Chen; Liu, 2000). Dessa forma, o filtro é amplamente utilizado para capturar mudanças estruturais significativas, fornecendo uma visão mais clara das variações de preços ao longo do tempo (Evensen, 2009).

A Figura 16 ilustra a diferença entre os valores de fechamento originais e os valores suavizados obtidos pelo filtro de Kalman. Observa-se que os valores suavizados seguem de perto os movimentos gerais dos valores originais, mas com menor volatilidade, o que confirma a eficácia do filtro na redução de ruídos.

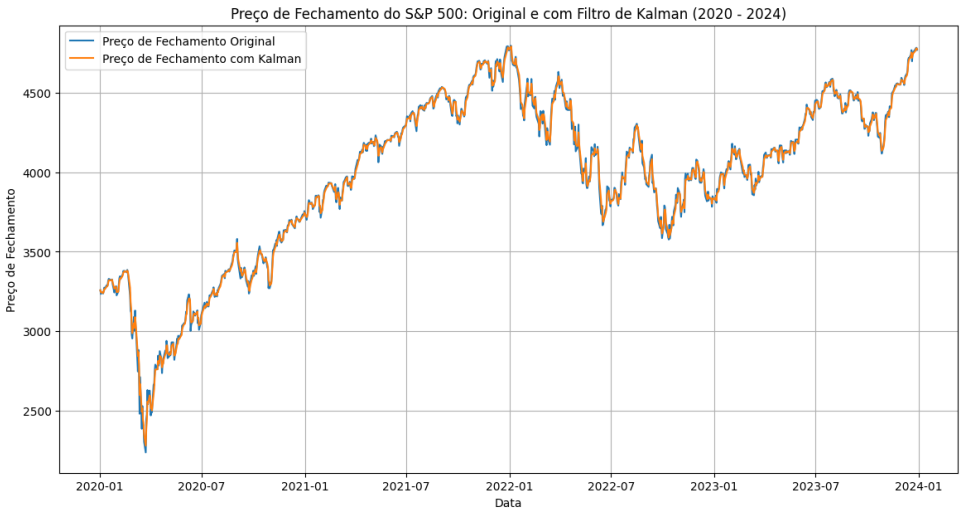

Figura 16 – Comparação entre valores de fechamento originais e suavizados pelo filtro de Kalman.

Além da suavização temporal, é importante analisar a distribuição dos valores

suavizados para entender melhor como o filtro de Kalman afeta os dados. A Figura 17 apresenta um histograma dos valores de fechamento suavizados em comparação com os valores originais. Este histograma permite observar a distribuição dos preços antes e depois da aplicação do filtro de Kalman.

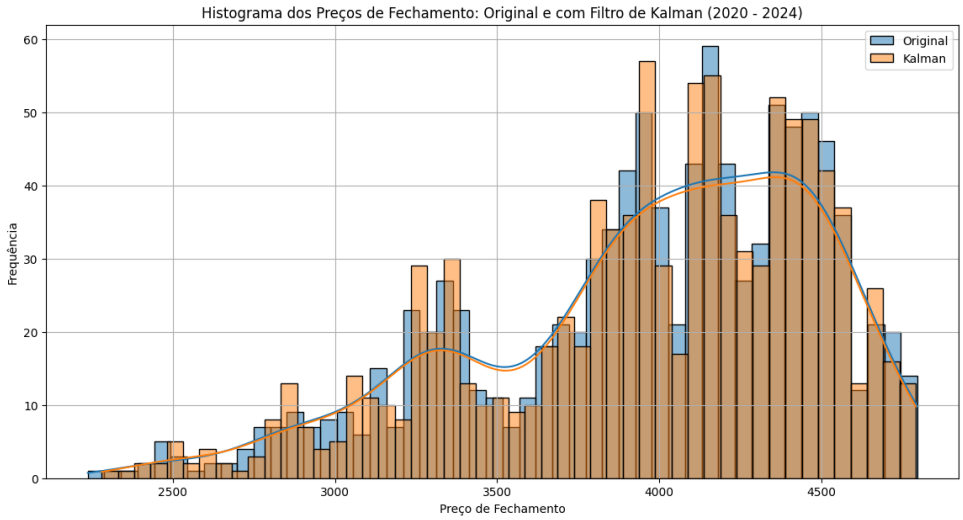

Figura 17 – Histograma comparativo da distribuição dos valores de fechamento originais e suavizados pelo filtro de Kalman.

A Figura 17 revela que, após a aplicação do filtro de Kalman, a distribuição dos valores de fechamento se torna mais concentrada em torno de uma média central, o que sugere uma redução das flutuações extremas observadas nos dados originais. Essa concentração indica que o filtro de Kalman é eficaz em suavizar os dados, eliminando ruídos e focando nas tendências subjacentes dos preços, facilitando a construção de modelos preditivos mais robustos.

4.3.1 Equações do Filtro de Kalman

A atualização do filtro de Kalman pode ser descrita pelas seguintes equações:

Inicialização

- `kf = KalmanFilter(dim_x=2, dim_z=1)`: Cria um objeto `KalmanFilter` com um estado de dimensão 2 (`dim_x=2`) e uma dimensão de medida de 1 (`dim_z=1`).
- `kf.x = np.array([data[0], 0.])`: Inicializa o estado com o primeiro valor dos dados e um valor de velocidade de 0.
- `kf.F = np.array([[1., 1.], [0., 1.]])`: Define a matriz de transição de estado $\mathbf{F}$.
- `kf.H = np.array([[1., 0.]])`: Define a matriz de observação $\mathbf{H}$.

- `kf.P *= 1000.`: Define a matriz de covariância inicial $\mathbf{P}$ com valores altos (incerteza inicial alta).
- `kf.R = 5`: Define a covariância do ruído de medição R.
- `kf.Q = 0.1`: Define a covariância do ruído do processo Q.

Predição e Atualização

Dentro do loop `for measurement in data`:

- `kf.predict()`: Executa a etapa de predição.
- `kf.update(measurement)`: Executa a etapa de atualização usando a medida atual.

Mapeamento com as Equações

Predição

`kf.predict()` executa as equações de predição:

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} \tag{4.1}$$

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q \tag{4.2}$$

Atualização

`kf.update(measurement)` executa as equações de atualização:

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \tag{4.3}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1}) \tag{4.4}$$

$$P_{k|k} = (I - K_kH)P_{k|k-1} \tag{4.5}$$

4.4 Cálculo do Indicador RSI

O Índice de Força Relativa (RSI) é um indicador de momentum que mede a velocidade e a mudança dos movimentos de preço. A fórmula para o cálculo do RSI é dada por:

$$RSI = 100 - \frac{100}{1 + RS} \tag{4.6}$$

onde RS é a razão entre a média dos ganhos ($\overline{Gain}$) e a média das perdas ($\overline{Loss}$) durante um período n:

$$RS = \frac{\overline{Gain}}{\overline{Loss}} \tag{4.7}$$

A função Python para calcular o RSI é a seguinte:

Listing 4.1 – Função para calcular o Índice de Força Relativa (RSI)

```
def calculate_rsi(data, window=14):
    delta = data.diff()
    gain = (delta.where(delta > 0, 0)).fillna(0)
    loss = (-delta.where(delta < 0, 0)).fillna(0)
    avg_gain = gain.rolling(window=window, min_periods=1).mean()
    avg_loss = loss.rolling(window=window, min_periods=1).mean()
    rs = avg_gain / avg_loss
    rsi = 100 - (100 / (1 + rs))
    rsi = rsi.fillna(50) # Substituir NaN por 50
    rsi = rsi.replace(0, 50) # Substituir 0 por 50
    return rsi
```

No código acima, os ganhos e perdas são calculados a partir das diferenças entre preços consecutivos. A média dos ganhos e perdas é calculada usando uma janela móvel, e o valor do RSI é ajustado para substituir NaN e zeros por 50, conforme necessário.

4.5 Cálculo da Média Móvel Simples (SMA)

A Média Móvel Simples (SMA) é uma média aritmética calculada sobre um número específico de pontos de dados, dada pela fórmula:

$$SMA = \frac{1}{n} \sum_{i=1}^{n} P_i \tag{4.8}$$

onde P_i são os preços de fechamento no período i e n é o número de períodos considerados.

A função Python para calcular a SMA é a seguinte:

Listing 4.2 – Função para calcular a Média Móvel Simples (SMA)

```
def calculate_sma(data, window=14):
    sma = data.rolling(window=window).mean()
```

```
3
4       # Preencher os primeiros valores NaN com a média dos
            primeiros 'window' valores
5       sma[:window] = data[:window].mean()
6
7       return sma
```

O código utiliza uma janela móvel para calcular a média dos preços de fechamento e preenche os valores iniciais NaN com a média dos primeiros valores da janela.

4.6 Cálculo da Média Móvel Exponencial (EMA)

A Média Móvel Exponencial (EMA) dá mais peso aos preços mais recentes, sendo calculada pela fórmula:

$$EMA = P_t \cdot \frac{2}{n+1} + EMA_{t-1} \cdot \left(1 - \frac{2}{n+1}\right) \tag{4.9}$$

onde P_t é o preço de fechamento no tempo t e n é o número de períodos.

A função Python para calcular a EMA é a seguinte:

Listing 4.3 – Função para calcular a Média Móvel Exponencial (EMA)

```
1 def calculate_ema(data, window=14):
2     ema = data.ewm(span=window, adjust=False).mean()
3     return ema
```

O código utiliza a função `ewm` do Pandas para calcular a média móvel exponencial, que aplica mais peso aos preços mais recentes de acordo com o período especificado.

4.7 Normalização dos Dados

Os dados foram normalizados usando a técnica de Min-Max Scaling, que mapeia os valores para um intervalo entre 0 e 1. Esta etapa é crucial para garantir que os dados estejam na mesma escala, facilitando o treinamento do modelo LSTM e ajudando na convergência do modelo. A fórmula para normalização Min-Max é:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

onde X_{norm} é o valor normalizado, X é o valor original, X_{min} e X_{max} são os valores mínimo e máximo da variável, respectivamente.

Tabela 1 – Valores de Fechamento do S&P 500 em 2020 e Normalização

Data	Fechamento	Fechamento Normalizado
02/01/2020	3257.85	0.287
01/02/2020	3225.52	0.266
01/03/2020	2584.59	0.000
01/04/2020	2912.43	0.200
01/05/2020	3044.31	0.278
01/06/2020	3100.29	0.300
01/07/2020	3271.12	0.308
01/08/2020	3500.31	0.470
01/09/2020	3363.00	0.382
01/10/2020	3269.96	0.307
01/11/2020	3621.63	0.543
01/12/2020	3756.07	0.622

Este é um exemplo de normalização dos dados utilizando os valores de fechamento do S&P 500 de 2020, aplicando a técnica de Min-Max Scaling para mapear os valores originais em um intervalo de 0 a 1.

4.8 Preparação dos Dados para a LSTM

A preparação adequada dos dados é crucial para o desempenho de um modelo de Long Short-Term Memory (LSTM). Para capturar os padrões temporais nos dados, é necessário organizar os dados de entrada e saída em sequências temporais.

Cada sequência de entrada consistia de um número fixo de dias anteriores, chamado de *time steps*. O valor de saída era o preço de fechamento do dia seguinte. Esta abordagem permite que a LSTM capture dependências temporais e padrões nos dados históricos. A organização dos dados é feita da seguinte forma:

$$X = \begin{bmatrix} P_1 & P_2 & \cdots & P_{T-1} \\ P_2 & P_3 & \cdots & P_T \\ \vdots & \vdots & \ddots & \vdots \\ P_{n-T+1} & P_{n-T+2} & \cdots & P_n \end{bmatrix}, \quad y = \begin{bmatrix} P_T \\ P_{T+1} \\ \vdots \\ P_{n+1} \end{bmatrix} \tag{4.10}$$

onde T é o número de *time steps* e n é o número total de dados.

A função `prepare_lstm_data`, implementada em Python, ilustra o processo de preparação dos dados para um modelo LSTM. O código é apresentado a seguir:

Listing 4.4 – Função para preparar os dados para a LSTM

```
# Função para preparar os dados para a LSTM
def prepare_lstm_data(data, indicators, time_step=TIME_STEP):
    X, y = [], []
```

```
4       for i in range(len(data)-time_step-1):
5           a = np.hstack([data[i:(i+time_step), 0]] + [ind[i:(i+
                time_step)] for ind in indicators])
6           X.append(a)
7           y.append(data[i + time_step, 0])
8       return np.array(X), np.array(y)
```

Nesta função, os parâmetros são:

- `data`: uma matriz contendo os preços históricos.
- `indicators`: uma lista de matrizes, cada uma contendo valores de indicadores técnicos.
- `time_step`: o número de *time steps* a serem considerados para cada sequência de entrada.

A função cria duas listas, `X` e `y`. Para cada sequência de *time steps* nos dados, a função concatena os preços históricos e os valores dos indicadores técnicos correspondentes, criando as entradas do modelo `X`. As saídas `y` são os preços de fechamento do dia seguinte.

Essa preparação dos dados garante que o modelo LSTM tenha a estrutura necessária para aprender as dependências temporais e realizar previsões precisas.

4.9 Construção dos Modelos LSTM

A construção de modelos LSTM utilizando o Keras Tuner envolve a configuração de hiperparâmetros dinâmicos, permitindo a seleção otimizada de parâmetros para cada modelo específico. Três tipos de modelos são considerados: LSTM simples, LSTM bidirecional (BiLSTM) e LSTM empilhado (Stacked LSTM). Cada um desses modelos ajusta a quantidade de unidades LSTM e a taxa de aprendizado através do uso de hiperparâmetros ajustáveis, proporcionando maior flexibilidade e potencial para capturar padrões complexos em dados de séries temporais.

- **LSTM Simples**: Este modelo inclui duas camadas LSTM, cada uma com um número de unidades ajustável. A primeira camada LSTM é configurada para retornar sequências, permitindo que a segunda camada LSTM processe essas sequências.
- **BiLSTM**: O modelo bidirecional incorpora duas camadas LSTM, mas cada uma delas é envolvida em uma camada bidirecional, permitindo que a rede aprenda dependências de longa e curta duração em ambas as direções.

- **Stacked LSTM**: Neste modelo, três camadas LSTM são empilhadas. As duas primeiras camadas são configuradas para retornar sequências, permitindo que a última camada LSTM processe a informação combinada.

A taxa de aprendizado do modelo é também ajustável, sendo determinada pelo Keras Tuner dentro de um intervalo definido, utilizando uma escala logarítmica para melhor exploração dos valores possíveis. Todos os modelos são compilados utilizando o otimizador Adam e a função de perda MSE (Mean Squared Error), que é adequada para problemas de regressão.

Para generalizar o processo de criação dos modelos e permitir uma busca eficiente de hiperparâmetros, foi desenvolvida uma função que cria tuners adaptáveis, utilizando o Keras Tuner. Essa função permite a seleção automática de diferentes configurações de modelos com base na minimização da perda de validação.

Listing 4.5 – Código para criação de modelos LSTM, BiLSTM e Stacked LSTM usando Keras Tuner

```
# Função para criar modelos LSTM ajustados com hiperparâmetros dinâmicos usando Keras Tuner
def create_lstm_model(hp, input_shape):
    model = Sequential()
    model.add(Input(shape=input_shape))

    # Unidades LSTM como hiperparâmetros ajustáveis
    units_1 = hp.Int('units_1', min_value=MIN_LSTM_UNITS, max_value=MAX_LSTM_UNITS, step=STEP_LSTM_UNITS)
    model.add(LSTM(units_1, return_sequences=True))

    units_2 = hp.Int('units_2', min_value=MIN_LSTM_UNITS, max_value=MAX_LSTM_UNITS, step=STEP_LSTM_UNITS)
    model.add(LSTM(units_2))

    model.add(Dense(1))

    # Taxa de aprendizado ajustável
    learning_rate = hp.Float('learning_rate', min_value=MIN_LEARNING_RATE, max_value=MAX_LEARNING_RATE, sampling='log')
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='mean_squared_error')

    return model
```

```
20
21 # Função para criar o modelo BiLSTM
22 def create_bilstm_model(hp, input_shape):
23     model = Sequential()
24     model.add(Input(shape=input_shape))
25
26     # Unidades LSTM como hiperparâmetros ajustáveis
27     units_1 = hp.Int('units_1', min_value=MIN_LSTM_UNITS,
          max_value=MAX_LSTM_UNITS, step=STEP_LSTM_UNITS)
28     model.add(Bidirectional(LSTM(units_1, return_sequences=True))
          )
29
30     units_2 = hp.Int('units_2', min_value=MIN_LSTM_UNITS,
          max_value=MAX_LSTM_UNITS, step=STEP_LSTM_UNITS)
31     model.add(Bidirectional(LSTM(units_2)))
32
33     model.add(Dense(1))
34
35     # Taxa de aprendizado ajustável
36     learning_rate = hp.Float('learning_rate', min_value=
          MIN_LEARNING_RATE, max_value=MAX_LEARNING_RATE, sampling='
          log')
37     model.compile(optimizer=Adam(learning_rate=learning_rate),
          loss='mean_squared_error')
38
39     return model
40
41 # Função para criar o modelo Stacked LSTM
42 def create_stacked_lstm_model(hp, input_shape):
43     model = Sequential()
44     model.add(Input(shape=input_shape))
45
46     # Unidades LSTM como hiperparâmetros ajustáveis
47     units_1 = hp.Int('units_1', min_value=MIN_LSTM_UNITS,
          max_value=MAX_LSTM_UNITS, step=STEP_LSTM_UNITS)
48     model.add(LSTM(units_1, return_sequences=True))
49
50     units_2 = hp.Int('units_2', min_value=MIN_LSTM_UNITS,
          max_value=MAX_LSTM_UNITS, step=STEP_LSTM_UNITS)
51     model.add(LSTM(units_2, return_sequences=True))
52
53     units_3 = hp.Int('units_3', min_value=MIN_LSTM_UNITS,
```

```
        max_value=MAX_LSTM_UNITS, step=STEP_LSTM_UNITS)
    model.add(LSTM(units_3))

    model.add(Dense(1))

    # Taxa de aprendizado ajustável
    learning_rate = hp.Float('learning_rate', min_value=
        MIN_LEARNING_RATE, max_value=MAX_LEARNING_RATE, sampling='
        log')
    model.compile(optimizer=Adam(learning_rate=learning_rate),
        loss='mean_squared_error')

    return model

# Função para criar um modelo com base na função de criação de
    modelo fornecida
def create_model(hp, model_fn, input_shape):
    return model_fn(hp, input_shape)

# Função para criar o tuner generalizado
def create_new_tuner(model_fn, input_shape, project_name):
    tuner = kt.RandomSearch(
        lambda hp: create_model(hp, model_fn, input_shape),
        objective='val_loss',
        max_trials=MAX_TRIALS,
        executions_per_trial=EXECUTIONS_PER_TRIAL,
        directory=f'my_dir_{project_name}',
        project_name=project_name
    )
    return tuner
```

Com essa configuração, o processo de ajuste dos hiperparâmetros dos modelos LSTM, BiLSTM e Stacked LSTM é automatizado, permitindo que a rede encontre a melhor arquitetura e taxa de aprendizado para minimizar a perda de validação. Esta abordagem aumenta a eficiência e precisão do modelo, especialmente em problemas complexos de séries temporais.

4.10 Treinamento dos Modelos LSTM

O treinamento dos modelos LSTM foi configurado para utilizar parâmetros específicos, definidos por meio de variáveis globais para facilitar ajustes e experimentos. Os parâmetros principais incluem o tamanho de lote, o número de épocas, e o uso de *Early*

Stopping para evitar *overfitting*. A *val_loss* é monitorada para interromper o treinamento se não houver melhora após um número predefinido de épocas consecutivas.

As variáveis globais usadas para o treinamento e ajuste de hiperparâmetros são definidas da seguinte maneira:

Listing 4.6 – Definição de variáveis globais para o treinamento de modelos LSTM

```
TEST_SIZE = 0.33
EPOCHS = 100  # Número de épocas para o treinamento
BATCH_SIZE = 32  # Tamanho do lote
PATIENCE = 10  # Número de épocas sem melhora para ativar o early stopping
SHOW_DETAILS = 1  # Verbosidade do treinamento

# Variáveis globais para ajuste de hiperparâmetros
MIN_LSTM_UNITS = 50
MAX_LSTM_UNITS = 400
STEP_LSTM_UNITS = 50
MIN_LEARNING_RATE = 1e-4
MAX_LEARNING_RATE = 1e-2
MAX_TRIALS = 3  # Número máximo de experimentos durante a busca
EXECUTIONS_PER_TRIAL = 2  # Quantas vezes executar o modelo para cada configuração
```

A seguir, apresentamos a função de treinamento e previsão dos modelos LSTM, que utiliza estas variáveis globais:

Listing 4.7 – Função para treinar e prever com modelos LSTM

```
# Função para treinar o modelo e prever
def train_and_predict_model(model_title, model, X_train, Y_train, X_test, Y_test):
    early_stopping = EarlyStopping(monitor='val_loss', patience=PATIENCE, restore_best_weights=True)
    history = model.fit(X_train, Y_train, epochs=EPOCHS,
                        batch_size=BATCH_SIZE,
                        validation_data=(X_test, Y_test),
                        callbacks=[early_stopping],
                        verbose=SHOW_DETAILS)
    plot_loss(model_title, history)
    return model.predict(X_test)
```

Nesta função, os seguintes componentes são considerados:

- `model_title`: uma string que representa o título do modelo, usada para fins de identificação ao plotar gráficos de perda.
- `model`: o modelo LSTM que será treinado e utilizado para fazer previsões.
- `X_train` e `Y_train`: conjuntos de dados de treino, onde `X_train` contém as entradas e `Y_train` contém as saídas correspondentes.
- `X_test` e `Y_test`: conjuntos de dados de teste, usados para validação do modelo durante o treinamento.
- `early_stopping`: um callback que monitora a métrica de validação e para o treinamento se não houver melhora após um número de épocas definido por `PATIENCE`.
- `history`: contém o histórico de perda do modelo durante o treinamento, que é utilizado para plotar a evolução da perda.
- `plot_loss`: uma função auxiliar que gera um gráfico mostrando a perda de treino e validação ao longo das épocas, facilitando a visualização do desempenho do modelo.
- `EPOCHS`, `BATCH_SIZE`, `PATIENCE`, e `SHOW_DETAILS`: constantes que definem, respectivamente, o número de épocas de treinamento, o tamanho do lote, a paciência para *early stopping*, e o nível de detalhamento das mensagens de treinamento.

Essa configuração permite flexibilidade e consistência na realização de experimentos, ajustando facilmente os parâmetros de treinamento e os hiperparâmetros dos modelos LSTM. O uso de variáveis globais facilita a modificação desses valores para diferentes cenários de treinamento e validação, proporcionando uma abordagem sistemática para encontrar as melhores configurações para previsões de séries temporais.

4.11 Resumo da Construção e Treinamento

A construção do modelo LSTM é realizada utilizando uma sequência de camadas com hiperparâmetros ajustáveis, definidos por meio de variáveis globais. As camadas LSTM são configuradas para ter um número de unidades variável, entre 50 e 400, ajustado em etapas de 50 unidades, o que permite a flexibilidade necessária para capturar diferentes padrões nos dados de séries temporais.

- **Modelo LSTM Simples**: Começa com uma camada LSTM que retorna sequências, seguida por uma segunda camada LSTM. As unidades de cada camada são determinadas dinamicamente pelo Keras Tuner, com base nos intervalos especificados.
- **Modelo BiLSTM**: Utiliza camadas LSTM bidirecionais, permitindo que o modelo aprenda dependências de longa e curta duração em ambas as direções.

- **Modelo Stacked LSTM**: Inclui três camadas LSTM empilhadas para capturar interações mais complexas nos dados de entrada.

Após as camadas LSTM, uma camada densa é adicionada para a previsão final. O modelo é compilado utilizando o otimizador Adam e a função de perda MSE (Mean Squared Error), que é apropriada para problemas de regressão.

Durante o treinamento, a função `EarlyStopping` é empregada para monitorar a perda de validação (`val_loss`) e interromper o treinamento se a perda não melhorar após um número de épocas consecutivas definido pela variável `PATIENCE` (neste caso, 10 épocas). O uso de *Early Stopping* ajuda a evitar *overfitting* restaurando os melhores pesos do modelo alcançados durante o treinamento.

A função `train_and_predict_model` treina o modelo usando os dados de treino e validação, enquanto ajusta os hiperparâmetros dinamicamente para otimizar o desempenho. O histórico de treinamento é registrado para análise posterior, permitindo a avaliação da evolução da perda ao longo do treinamento.

Esta abordagem, combinando a flexibilidade dos hiperparâmetros ajustáveis e o controle de *overfitting* por meio de *Early Stopping*, garante que os modelos LSTM sejam capazes de capturar padrões complexos nos dados de séries temporais, ao mesmo tempo em que generalizam bem para novos dados. As variáveis globais utilizadas fornecem um meio eficiente de ajustar e refinar o processo de construção e treinamento, permitindo experimentação contínua e melhoramento do modelo.

4.12 Previsão e Avaliação

Para avaliar o desempenho do modelo, é essencial dividir os dados em conjuntos de treino e teste. Essa divisão permite que o modelo seja treinado em um subconjunto dos dados e testado em outro, garantindo uma avaliação justa de sua capacidade de generalização.

A seguir, apresentamos a função Python utilizada para dividir os dados:

Listing 4.8 – Função para dividir dados em treinamento e teste

```
# Função para dividir dados em treinamento e teste
def split_data(X, Y, test_size=0.2):
    train_size = int(len(X) * (1 - test_size))
    X_train, X_test = X[:train_size], X[train_size:]
    Y_train, Y_test = Y[:train_size], Y[train_size:]
    return X_train, X_test, Y_train, Y_test
```

Nesta função, os parâmetros são:

- `X`: conjunto de dados de entrada.
- `Y`: conjunto de dados de saída.
- `test_size`: a proporção dos dados a ser utilizada como conjunto de teste.

A função calcula o tamanho do conjunto de treino com base na proporção especificada e divide os dados em `X_train`, `X_test`, `Y_train` e `Y_test`. Essa divisão é crucial para avaliar a performance do modelo de forma precisa e evitar *overfitting*, garantindo que o modelo possa generalizar bem para novos dados não vistos durante o treinamento.

4.12.1 Execução do Pipeline de Previsão e Avaliação

4.12.1.1 Preparação dos Dados

Primeiramente, os dados históricos de preços são buscados e escalonados. A seguir, os dados são preparados para serem usados por modelos que utilizam apenas o preço de fechamento e por modelos que utilizam indicadores técnicos.

Listing 4.9 – Preparação dos dados para previsão

```
# Buscando os dados de preços históricos do ativo
data = fetch_data(ticker=TICKER, start_date=START_DATE, end_date=
    END_DATE)

# Preparação dos dados para modelos sem indicadores técnicos (
    usando apenas o preço de fechamento)
scaler_close_only = MinMaxScaler(feature_range=(0, 1))
scaled_data_close_only = scaler_close_only.fit_transform(data)

# Criando conjuntos de dados de entrada (X) e saída (Y) usando
    uma janela de LOOK_BACK dias
X_basic, Y_basic = create_dataset(scaled_data_close_only,
    LOOK_BACK)
X_basic = np.reshape(X_basic, (X_basic.shape[0], X_basic.shape
    [1], 1))

# Dividindo os dados em conjuntos de treinamento e teste
X_train_basic, X_test_basic, Y_train_basic, Y_test_basic =
    split_data(X_basic, Y_basic, test_size=TEST_SIZE)
```

4.12.1.2 Preparação dos Dados com Indicadores Técnicos

Além do preço de fechamento, o uso de indicadores técnicos como SMA, EMA e RSI pode melhorar a capacidade preditiva do modelo. A preparação dos dados para incluir

esses indicadores é apresentada a seguir.

Listing 4.10 – Preparação dos dados com indicadores técnicos

```
# Adicionando indicadores técnicos aos dados
data['SMA'] = SMA(data['Close'], window=SMA_WINDOW)
data['EMA'] = EMA(data['Close'], window=EMA_WINDOW)
data['RSI'] = RSI(data['Close'], window=RSI_WINDOW)
data = data.dropna()  # Removendo valores nulos criados pela
    janela dos indicadores

# Escalando os dados com indicadores técnicos
scaler_tech = MinMaxScaler(feature_range=(0, 1))
scaled_data_tech = scaler_tech.fit_transform(data)

# Criando conjuntos de dados para modelos com indicadores té
    cnicos
X_tech, Y_tech = create_dataset(scaled_data_tech, LOOK_BACK)
X_tech = np.reshape(X_tech, (X_tech.shape[0], X_tech.shape[1],
    X_tech.shape[2]))

# Dividindo os dados em conjuntos de treinamento e teste para
    modelos com indicadores
X_train_tech, X_test_tech, Y_train_tech, Y_test_tech = split_data
    (X_tech, Y_tech, test_size=TEST_SIZE)
```

Nesta preparação, os dados são enriquecidos com indicadores técnicos e escalonados antes de serem organizados em janelas de tempo de tamanho especificado por `LOOK_BACK`. Posteriormente, os dados são divididos em conjuntos de treinamento e teste para serem utilizados nos modelos.

4.12.1.3 Avaliação dos Modelos

Após o treinamento, os modelos são avaliados utilizando as previsões realizadas. A função `evaluate_model` calcula várias métricas de erro e precisão, proporcionando uma visão abrangente do desempenho do modelo.

Listing 4.11 – Função para avaliar o modelo

```
# Função para avaliar o modelo
def evaluate_model(val_predict, y_val_actual):
    rmse = np.sqrt(mean_squared_error(y_val_actual, val_predict))
    mae = mean_absolute_error(y_val_actual, val_predict)
    mse = mean_squared_error(y_val_actual, val_predict)
    r2 = r2_score(y_val_actual, val_predict)
```

```
    da = directional_accuracy(val_predict, y_val_actual)

    # Criação do DataFrame para armazenar os resultados
    results = pd.DataFrame({
        'Métrica': ['RMSE', 'MAE', 'MSE', 'R²', 'DA'],
        'Valor': [rmse, mae, mse, r2, da]
    })

    # Plotando a tabela
    fig, ax = plt.subplots(figsize=(4, 1))
    ax.axis('off')
    table = ax.table(cellText=results.values, colLabels=results.
       columns, cellLoc='center', loc='center')
    table.scale(1, 1.5)
    table.auto_set_font_size(False)
    table.set_fontsize(12)
    table.auto_set_column_width(col=list(range(len(results.
       columns))))
    plt.show()

    return rmse, mae, mse, r2, da
```

Nesta função, os parâmetros são:

- `val_predict`: as previsões do modelo no conjunto de validação.
- `y_val_actual`: os valores reais do conjunto de validação.

4.12.2 Cálculo da Direcional Accuracy (DA)

O *Directional Accuracy* (DA) avalia se o modelo previu corretamente a direção do movimento dos preços (subida ou descida). A função para calcular o DA compara os sinais das diferenças entre previsões consecutivas e valores reais consecutivos, retornando a média dessas comparações.

Listing 4.12 – Função para calcular a Acurácia Direcional (DA)

```
# Função para calcular Directional Accuracy (DA)
def directional_accuracy(predictions, actuals):
    predictions = np.array(predictions)
    actuals = np.array(actuals)
    return np.mean(np.sign(predictions[1:] - predictions[:-1]) ==
        np.sign(actuals[1:] - actuals[:-1]))
```

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo apresenta os resultados dos experimentos realizados para validar a eficácia dos modelos de redes neurais recorrentes *LSTM* aplicados à previsão de preços do *S&P 500*. O objetivo é avaliar e comparar o desempenho de diferentes variantes de *LSTM*, incluindo modelos com e sem indicadores técnicos, bem como a integração do filtro de Kalman. A análise busca compreender a acurácia e a consistência desses modelos em prever preços de fechamento em diferentes cenários.

5.1 Ambiente Experimental

O ambiente de desenvolvimento foi configurado em *Python*, utilizando bibliotecas robustas para manipulação de dados, modelagem e visualização. Dentre as principais bibliotecas, destacam-se:

- *TensorFlow* e *Keras* para a construção e treinamento dos modelos *LSTM* e suas variantes, como *Bidirectional LSTM*.
- *filterpy* e *pykalman* para implementação do filtro de Kalman, que suaviza as previsões e aumenta a precisão dos modelos.
- *yfinance* para a obtenção de dados históricos do *S&P 500*, permitindo uma análise realista e temporal.
- *scikit-learn* para cálculos de métricas de avaliação, como *RMSE*, *MAE* e R^2, além de transformações com *MinMaxScaler*.
- *Matplotlib* e *Seaborn* para visualização dos dados e resultados, incluindo gráficos de séries temporais e histogramas.

5.2 Dinâmica Experimental e Tratamento dos Indicadores Técnicos

Os dados históricos de fechamento do *S&P 500* foram coletados desde 1990 até 2024, fornecendo uma base temporal extensa e robusta. Indicadores técnicos foram calculados para enriquecer as entradas dos modelos, melhorando sua capacidade de capturar padrões do mercado. Entre os indicadores utilizados, destacam-se:

- *SMA* (Média Móvel Simples), que fornece uma média simples dos preços de fechamento ao longo de uma janela específica de dias.
- *EMA* (Média Móvel Exponencial), que pondera mais os preços mais recentes, sendo particularmente útil para detectar tendências de curto prazo.

- *RSI* (Índice de Força Relativa), que indica a força e a velocidade das variações de preço, auxiliando na identificação de potenciais pontos de reversão.

Esses indicadores foram integrados aos modelos de duas formas: como entradas diretas nas camadas de *LSTM* e como variáveis processadas pelo filtro de Kalman, aprimorando a análise da dinâmica de preços e suavizando oscilações inesperadas. A configuração experimental incluiu o cálculo dessas métricas em janelas móveis, como *SMA* e *EMA*, que foram aplicados em períodos de sete dias para capturar tendências de curto prazo.

5.3 Treinamento e Avaliação dos Modelos

Diferentes arquiteturas de *LSTM* foram testadas com variações nos hiperparâmetros, como o número de unidades, taxa de aprendizado e configuração de camadas. O *Keras Tuner* foi utilizado para otimizar automaticamente esses parâmetros. Após o treinamento, os modelos foram avaliados por meio de métricas como *RMSE*, *MAE*, *MSE*, R^2 e acurácia direcional (*DA*). Foram gerados gráficos de perda (*loss*) e de previsão versus valores reais para visualização e análise, o que permitiu verificar a eficácia do uso de indicadores técnicos e do filtro de Kalman na melhoria das previsões.

5.3.1 Análise dos Modelos LSTM Apenas Preço

A figura 18 apresenta as curvas de perda para um modelo LSTM simples que utiliza apenas o preço de fechamento das ações como entrada. A linha azul representa a perda de treinamento e a linha laranja, a perda de validação. Ambas começam com valores elevados e diminuem rapidamente nas primeiras épocas, estabilizando-se por volta da quinta época. As perdas de treinamento e validação mantêm-se próximas, indicando que o modelo não sofre de overfitting significativo. Comparado aos outros modelos (Bi-LSTM e Stacked LSTM), o LSTM simples apresenta uma boa capacidade de aprender padrões dos dados sem superestimar o conjunto de treinamento. Embora não tenha a perda geral mais baixa, como o Bi-LSTM, ele é mais estável e não demonstra sinais de instabilidade significativos. Isso sugere que o LSTM consegue aprender padrões adequados do preço de fechamento, oferecendo uma solução confiável e consistente.

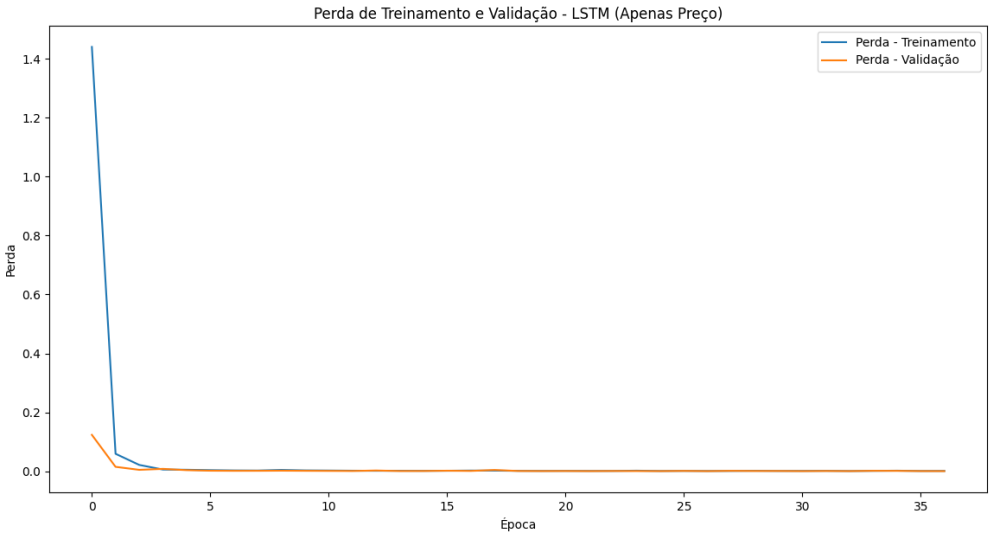

Figura 18 – Perda de Treinamento e Validação - LSTM (Apenas Preço)

A figura 19 mostra a perda de treinamento e validação para o modelo Bi-LSTM utilizando apenas o preço de fechamento das ações como entrada. Comparado com o modelo LSTM simples, o Bi-LSTM apresenta uma perda geral mais baixa e estabiliza-se rapidamente. No entanto, há uma leve oscilação na perda de validação ao longo das épocas, especialmente nas finais, indicando alguma instabilidade. O desempenho inicial do Bi-LSTM é superior ao do LSTM simples devido à sua perda menor. Contudo, a oscilação na curva de validação sugere uma instabilidade maior em relação ao Stacked LSTM 20. Apesar disso, o fato de as perdas de treinamento e validação permanecerem próximas mostra que o Bi-LSTM também não sofre de overfitting. Esse comportamento indica que, embora o Bi-LSTM possa capturar padrões complexos melhor do que o LSTM simples, ele apresenta uma certa variabilidade na validação que deve ser considerada.

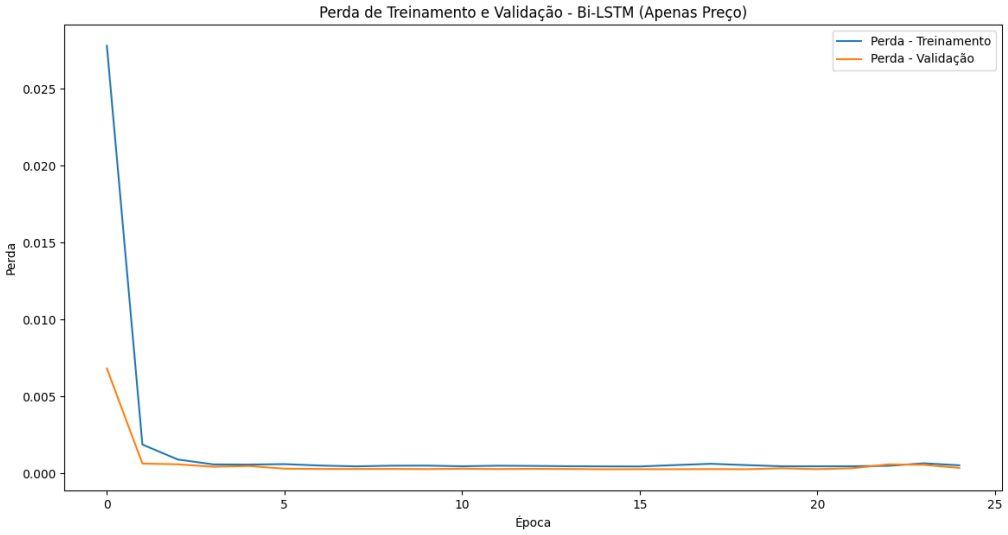

Figura 19 – Perda de Treinamento e Validação - Bi-LSTM (Apenas Preço)

A figura 20 exibe as curvas de perda para o modelo Stacked LSTM utilizando apenas o preço de fechamento das ações como entrada. Diferentemente do modelo Bi-LSTM, o Stacked LSTM apresenta uma perda geral mais alta, com uma oscilação mais pronunciada, especialmente na fase de validação (linha laranja). A perda não estabiliza de forma consistente, o que indica maior dificuldade em aprender os padrões dos dados e uma possível tendência a instabilidade durante o treinamento. Comparado aos modelos LSTM e Bi-LSTM, o Stacked LSTM tem um desempenho inferior, com uma perda geral mais elevada e uma oscilação mais significativa, principalmente na validação. Isso sugere que o Stacked LSTM pode estar enfrentando dificuldades em ajustar-se adequadamente aos dados, o que pode ser um indicativo de maior complexidade do modelo em relação à quantidade de dados de entrada. Portanto, apesar de sua arquitetura mais sofisticada, o Stacked LSTM não demonstra vantagens claras em relação aos outros modelos neste caso específico.

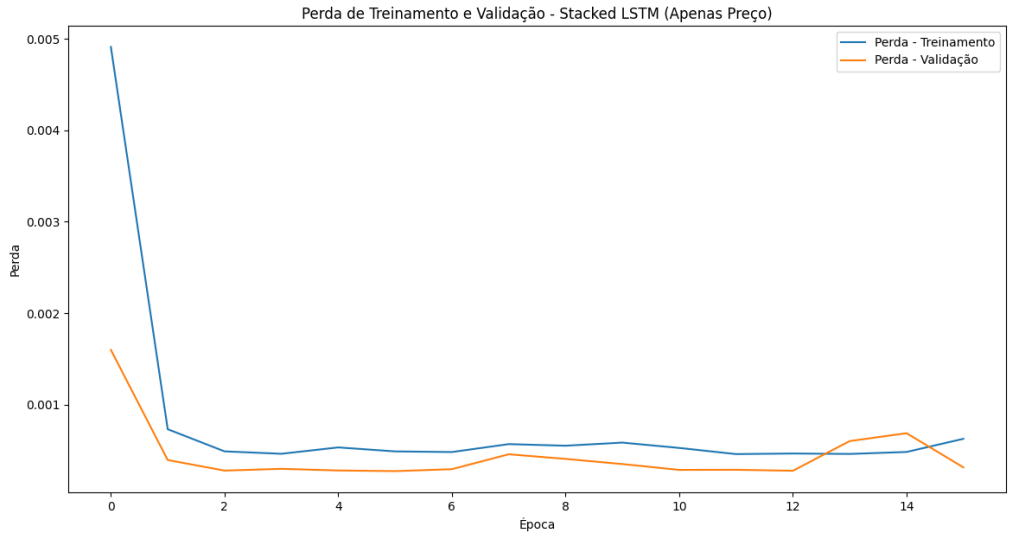

Figura 20 – Perda de Treinamento e Validação - Stacked LSTM (Apenas Preço)

A figura 21 apresenta a previsão de preço de fechamento durante o treinamento do modelo LSTM simples, comparando os valores previstos (linha vermelha) com os valores reais (linha azul). Nota-se que o modelo LSTM não consegue capturar bem os padrões dos dados reais, apresentando grandes divergências em vários pontos. Essa falta de alinhamento entre as previsões e os valores reais sugere que o modelo possui limitações na sua capacidade de aprender as variações do mercado durante o processo de treinamento.

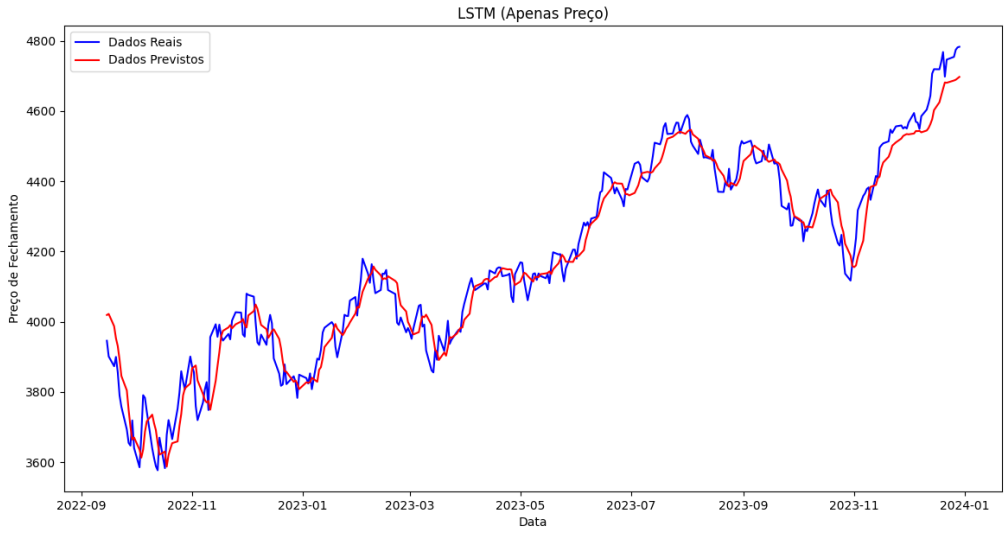

Figura 21 – Treinamento - LSTM (Apenas Preço)

Na figura 22, o desempenho do modelo Bi-LSTM durante o treinamento é exibido. As previsões estão significativamente mais alinhadas com os valores reais em comparação com o LSTM simples. A sobreposição é mais precisa, o que indica que neste caso o Bi-LSTM tem uma capacidade de aprendizado superior, conseguindo ajustar-se com maior eficácia aos padrões presentes nos dados históricos.

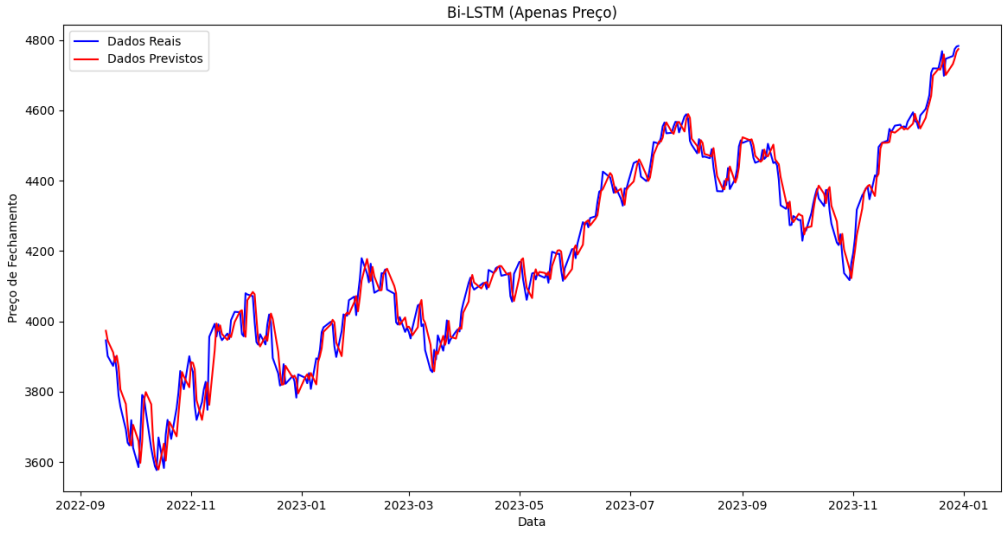

Figura 22 – Treinamento - Bi-LSTM (Apenas Preço)

A figura 23 mostra o resultado do treinamento do modelo Stacked LSTM. A previsão acompanha razoavelmente os valores reais, porém, há divergências mais frequentes e uma menor sobreposição em relação ao modelo Bi-LSTM. Esse comportamento indica que o Stacked LSTM, apesar de ser uma arquitetura mais complexa, não apresentou o melhor desempenho.

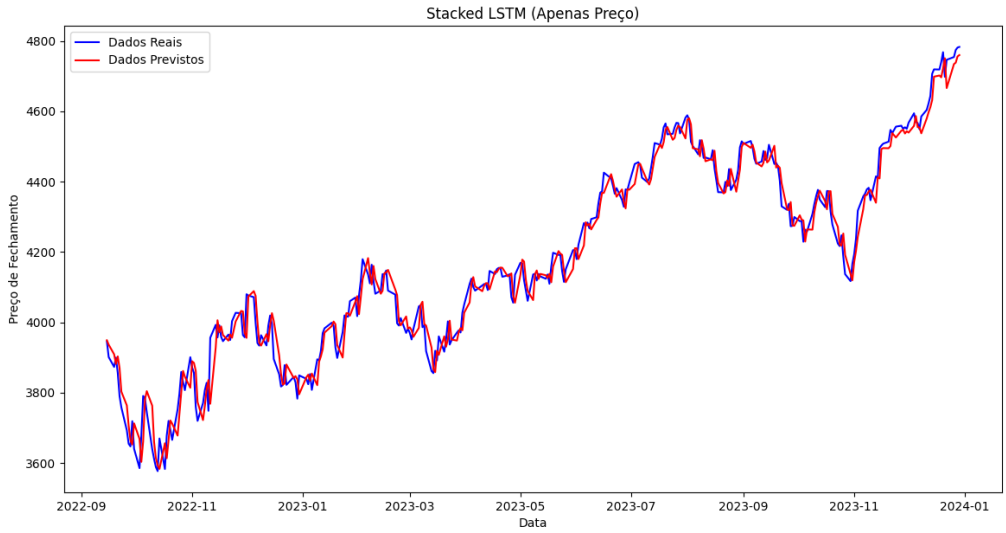

Figura 23 – Treinamento - Stacked LSTM (Apenas Preço)

As figuras 24, 25 e 26 apresentam as métricas de desempenho para os modelos LSTM, Bi-LSTM e Stacked LSTM, respectivamente, utilizando apenas o preço de fechamento como entrada.

O gráfico do LSTM com indicadores técnicos, representado na Figura 24, mostra uma queda rápida e significativa na perda durante as primeiras épocas, similar ao modelo sem indicadores. No entanto, observa-se que a perda se estabiliza em um valor significativamente menor, evidenciando a vantagem da incorporação de informações técnicas. Após essa estabilização, há oscilações mais perceptíveis na curva de perda de validação, mas em uma escala muito menor, sugerindo que, apesar dessas pequenas variações, o modelo com indicadores mantém uma precisão superior. Em resumo, os indicadores técnicos não apenas reduzem a perda geral, mas também melhoram a eficácia do modelo ao lidar com a complexidade dos dados.

Métrica	Valor
RMSE	56.84628656315973
MAE	45.55649639423078
MSE	3231.5002960208744
R^2	0.959270212508086
DA	0.49691358024691357

Figura 24 – Métricas - LSTM (Apenas Preço)

A figura 25 (Bi-LSTM) mostra o melhor desempenho entre os três modelos, com os menores valores para RMSE e MAE, indicando previsões mais precisas e menor erro médio. O MSE também é baixo, reforçando a precisão geral do modelo. Além disso, o R^2 está mais próximo de 1, sugerindo que o Bi-LSTM ajusta-se melhor aos padrões dos dados reais. Este desempenho superior corrobora com as análises anteriores, onde o Bi-LSTM demonstrou uma capacidade aprimorada em representar as variações dos preços de fechamento.

Métrica	Valor
RMSE	42.530680255893216
MAE	33.25710787259615
MSE	1808.858763029025
R^2	0.977201167794483
DA	0.4876543209876543

Figura 25 – Métricas - Bi-LSTM (Apenas Preço)

Já a figura 26 (Stacked LSTM) apresenta os piores resultados, com valores mais elevados de RMSE e MAE, indicando erros mais significativos nas previsões. O R^2 é inferior aos dos outros modelos, sugerindo um ajuste menos eficaz aos dados. Este comportamento é consistente com as análises de perda anteriores, onde o Stacked LSTM mostrou-se instável e incapaz de capturar adequadamente os padrões nos dados.

Métrica	Valor
RMSE	42.24421041555689
MAE	33.13268479567307
MSE	1784.5733136338454
R^2	0.9775072612812232
DA	0.4845679012345679

Figura 26 – Métricas - Stacked LSTM (Apenas Preço)

5.3.2 Análise dos Modelos LSTM com Indicadores Técnicos

Nesta subseção, são apresentados e analisados os resultados dos modelos LSTM, Bi-LSTM e Stacked LSTM com a inclusão de indicadores técnicos. O objetivo é verificar o impacto destes indicadores na precisão das previsões de preços de fechamento.

5.3.2.1 Perda de Treinamento e Validação

As figuras 27, 28 e 29 ilustram as curvas de perda para os modelos LSTM, Bi-LSTM e Stacked LSTM com indicadores técnicos.

Na Figura 27, o LSTM com indicadores técnicos mostra uma convergência suave, com uma rápida redução da perda nas primeiras épocas e subsequente estabilização. Essa suavidade indica que a inclusão dos indicadores ajudou o modelo a se adaptar melhor ao problema, resultando em um treinamento eficiente. A estabilidade da curva de perda sugere que o modelo aprende bem com os dados adicionais sem instabilidade significativa.

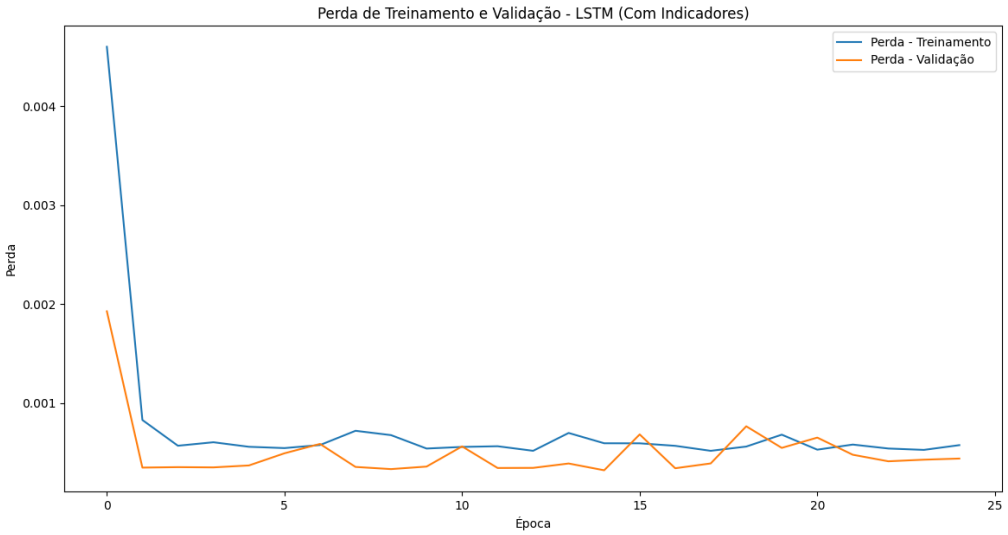

Figura 27 – Perda de Treinamento e Validação - LSTM (Com Indicadores)

A Figura 28 apresenta o Bi-LSTM com indicadores técnicos. Neste caso, observa-se uma leve oscilação nas curvas de perda, principalmente nas primeiras épocas. Apesar da instabilidade ser mais pronunciada do que no LSTM simples, o modelo ainda atinge uma convergência aceitável ao longo do treinamento. Isso sugere que, embora os indicadores técnicos forneçam informações adicionais, a maior complexidade do Bi-LSTM requer um ajuste mais cuidadoso.

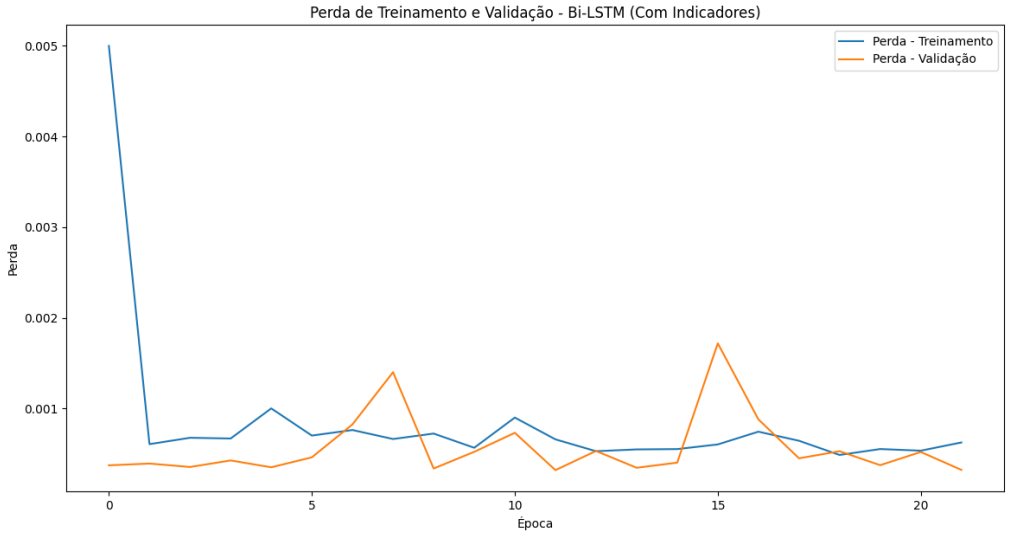

Figura 28 – Perda de Treinamento e Validação - Bi-LSTM (Com Indicadores)

Na Figura 29, o Stacked LSTM exibe uma estabilidade significativa durante todo o processo de treinamento e validação. Diferentemente da versão sem indicadores, a perda agora é mais controlada e estável, sugerindo que a complexidade do modelo se beneficia da inclusão dos indicadores. Essa estabilidade indica um melhor ajuste do modelo ao problema, demonstrando que a arquitetura do Stacked LSTM pode aproveitar as informações dos indicadores para melhorar a precisão.

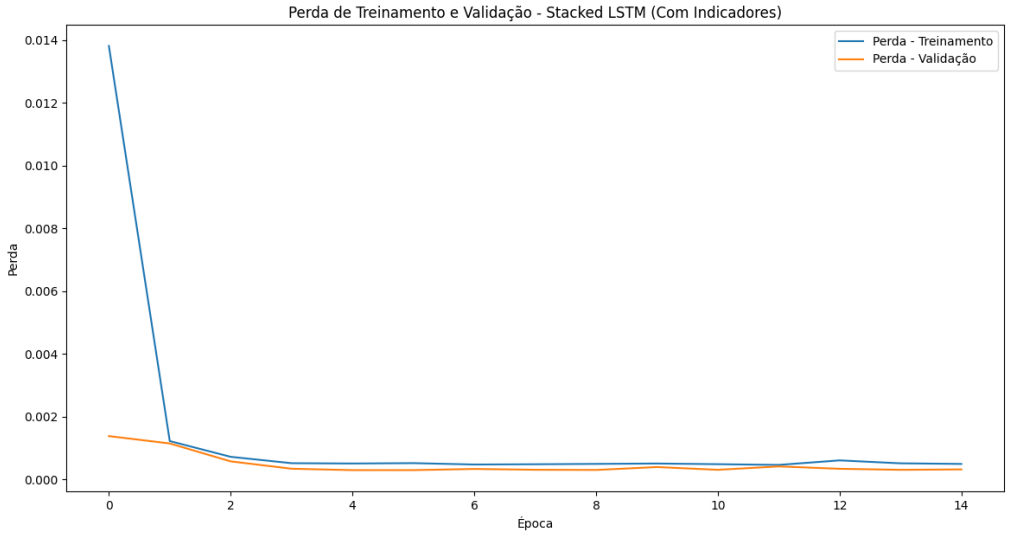

Figura 29 – Perda de Treinamento e Validação - Stacked LSTM (Com Indicadores)

5.3.2.2 Previsão de Preço

A figura 30 mostra uma sobreposição bastante próxima entre os dados reais e previstos, com uma correspondência quase perfeita em muitos pontos. A inclusão dos indicadores técnicos melhorou a capacidade do modelo de capturar as tendências, embora algumas pequenas discrepâncias ocorram em regiões de rápidas mudanças de preço, indicando limitações do LSTM para prever variações abruptas.

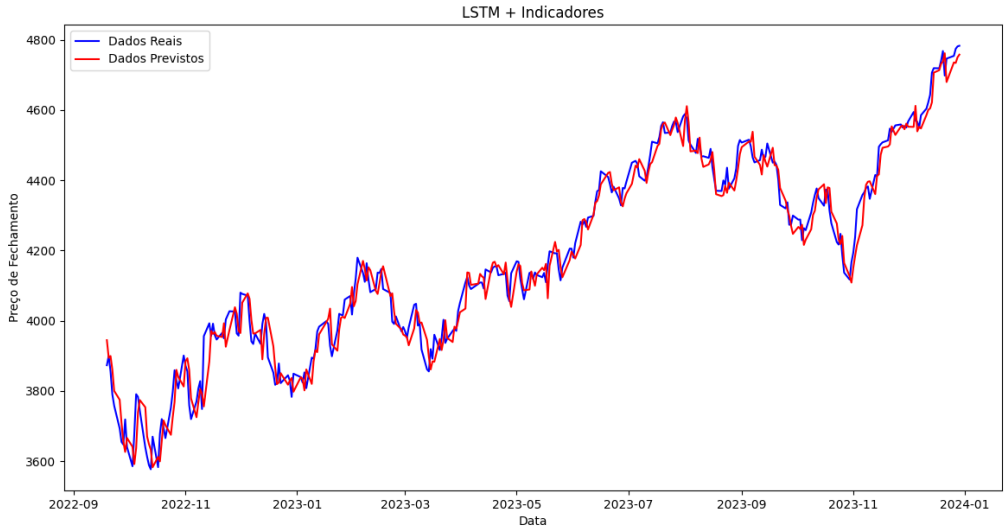

Figura 30 – Previsão de Preço - LSTM (Com Indicadores)

A figura 31 do Bi-LSTM apresenta uma sobreposição ainda mais precisa, com as previsões quase se fundindo aos dados reais na maior parte do gráfico. Isso evidencia a superioridade do Bi-LSTM em ajustar-se aos padrões complexos quando utiliza indicadores técnicos. As poucas discrepâncias aparecem em picos ou quedas acentuadas, mostrando que, apesar de seu desempenho, o modelo ainda tem dificuldades em capturar movimentos extremos.

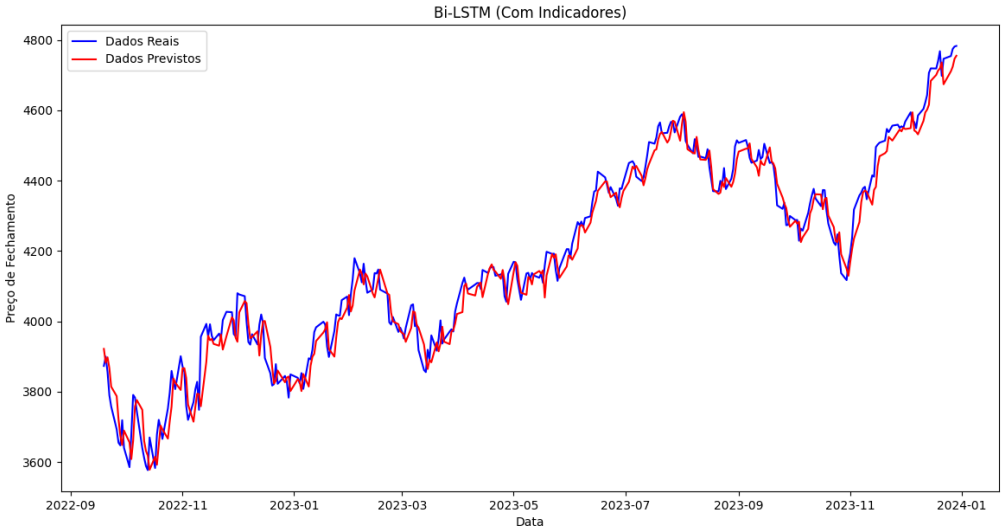

Figura 31 – Previsão de Preço - Bi-LSTM (Com Indicadores)

A figura 32 do Stacked LSTM demonstra uma excelente correspondência entre os dados reais e previstos, semelhante ao Bi-LSTM. No entanto, há uma tendência de suavizar movimentos bruscos do mercado, resultando em pequenas discrepâncias, especialmente em picos e vales acentuados. Isso sugere que o Stacked LSTM é preciso, mas pode ser mais conservador na resposta a rápidas mudanças.

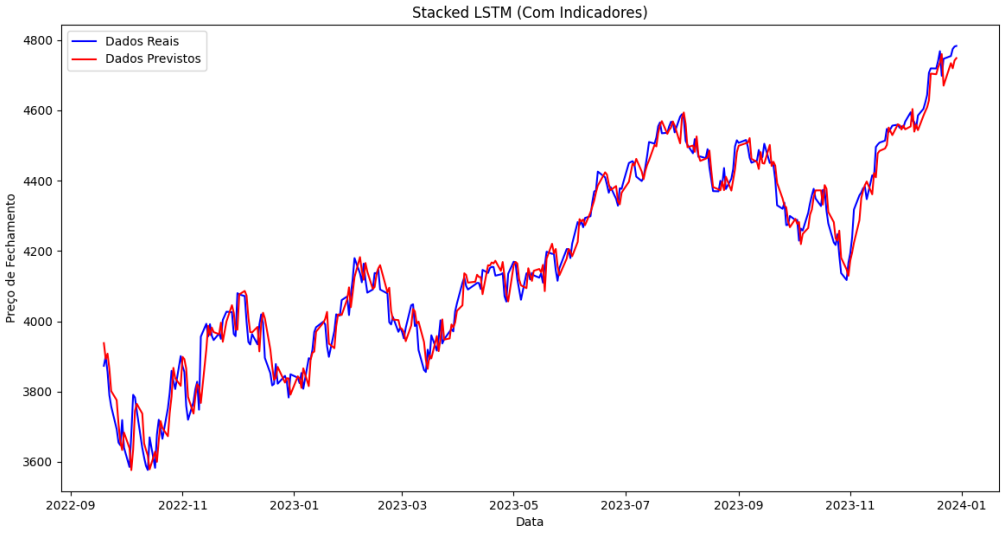

Figura 32 – Previsão de Preço - Stacked LSTM (Com Indicadores)

5.3.2.3 Comparação de Desempenho

A figura 33 do LSTM com indicadores mostra erros consideráveis nas previsões, embora o modelo consiga explicar uma boa parte da variação dos dados, conforme indicado pelo R^2. No entanto, o modelo apresenta dificuldades em prever corretamente a direção dos movimentos do mercado, com uma precisão direcional próxima do acaso.

Métrica	Valor
RMSE	45.845280434612555
MAE	35.966468503219346
MSE	2101.7897381282687
R^2	0.9735376475975555
DA	0.5008487326877821

Figura 33 – Métricas - LSTM (Com Indicadores)

A figura 34, o Bi-LSTM demonstra uma melhora significativa em relação ao LSTM, com menores erros nas previsões e um R^2 mais elevado, indicando uma capacidade superior de capturar os padrões dos dados quando são utilizados indicadores técnicos. A precisão direcional também apresenta uma leve melhora, embora ainda esteja próxima do desempenho aleatório.

Métrica	Valor
RMSE	30.3555861676487
MAE	23.885014936879923
MSE	921.461611581545
R^2	0.988642380488421
DA	0.5081015392924656

Figura 34 – Métricas - Bi-LSTM (Com Indicadores)

A figura 35 do Stacked LSTM mostra um desempenho muito semelhante ao do LSTM simples. A presença dos indicadores não trouxe vantagens significativas, resultando em erros comparáveis e um R^2 praticamente igual ao do LSTM. A precisão direcional permanece próxima do acaso, indicando que o modelo não se beneficiou da arquitetura mais complexa para prever a direção dos movimentos do mercado.

Métrica	Valor
RMSE	45.93273203716393
MAE	36.492577822459424
MSE	2109.815872397906
R^2	0.9734365954372849
DA	0.5006944176536399

Figura 35 – Métricas - Stacked LSTM (Com Indicadores)

5.3.3 Análise dos Modelos LSTM com Filtro de Kalman

Nesta subseção, são apresentados e analisados os resultados dos modelos LSTM, Bi-LSTM e Stacked LSTM com a aplicação do filtro de Kalman. O objetivo é verificar o impacto deste filtro na estabilidade dos modelos, bem como na precisão das previsões de preços de fechamento.

5.3.3.1 Perda de Treinamento e Validação

A figura 36 do LSTM com filtro de Kalman mostra uma rápida convergência nas primeiras épocas e uma estabilização. Comparado ao LSTM sem o filtro, observa-se uma redução geral da perda para níveis muito mais baixos, apesar de ainda haver algumas oscilações na validação, indicando que o filtro ajuda a controlar, mas não elimina completamente as flutuações.

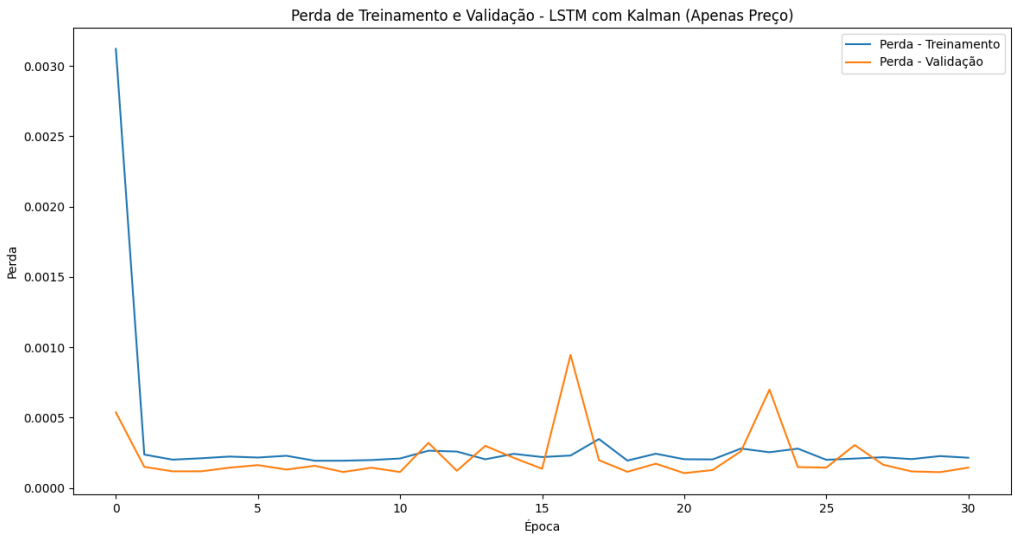

Figura 36 – Perda de Treinamento e Validação - LSTM com Kalman (Apenas Preço)

A figura 37 do Bi-LSTM com filtro de Kalman revela uma queda inicial acentuada e estabilização da perda, com oscilações moderadas ao longo do treinamento. Em comparação ao Bi-LSTM sem o filtro, a perda é significativamente menor, e as curvas de treinamento e validação mantêm-se mais próximas, destacando a eficácia do filtro de Kalman em reduzir os erros gerais do modelo e melhorar sua estabilidade.

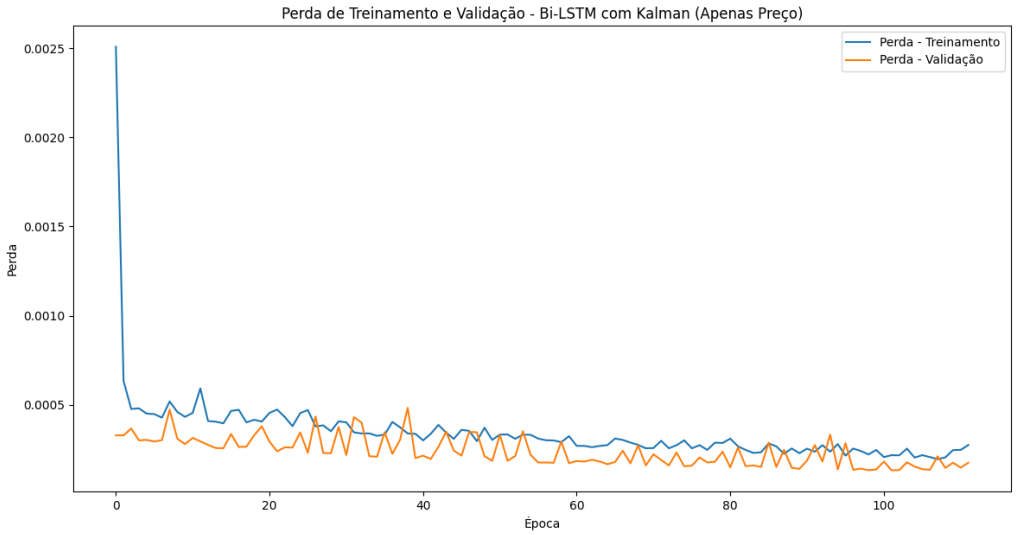

Figura 37 – Perda de Treinamento e Validação - Bi-LSTM com Kalman (Apenas Preço)

A figura 38 do Stacked LSTM com filtro de Kalman apresenta uma rápida redução na perda nas primeiras épocas e estabiliza com quase nenhuma oscilação nas curvas. A sobreposição das curvas de treinamento e validação sugere um ajuste mais preciso, indicando que o filtro de Kalman melhora significativamente a estabilidade do Stacked LSTM. Em comparação com o Stacked LSTM sem o filtro e sem indicadores, há uma clara redução nas flutuações, o que destaca a eficácia do filtro de Kalman em aumentar a precisão e robustez do modelo.

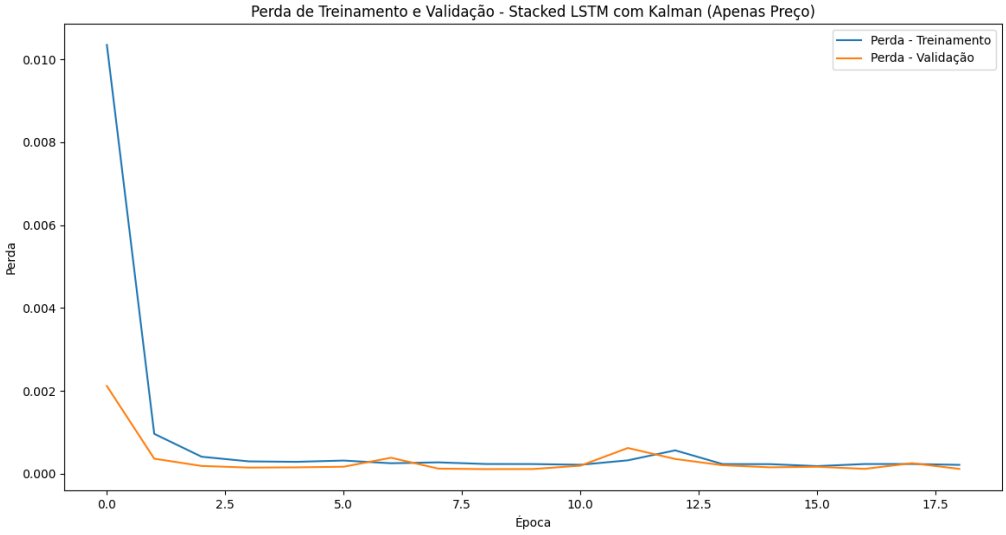

Figura 38 – Perda de Treinamento e Validação - Stacked LSTM com Kalman (Apenas Preço)

5.3.3.2 Previsão de Preço

A figura 39 do LSTM com Kalman mostra uma excelente correspondência entre os valores previstos e reais, indicando alta precisão nas previsões. Em comparação com o LSTM sem o filtro, as previsões são mais alinhadas aos dados reais, sugerindo que o filtro de Kalman melhorou significativamente o ajuste do modelo.

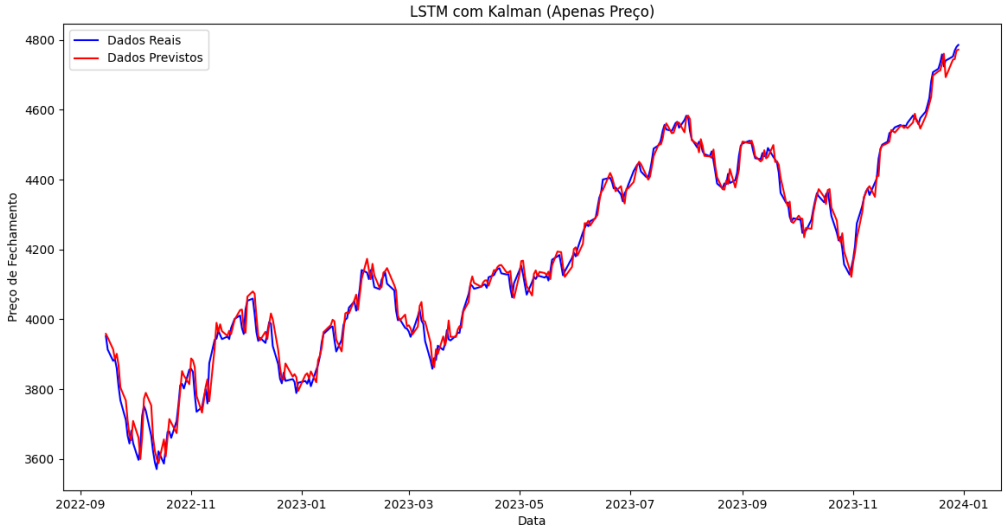

Figura 39 – Previsão de Preço - LSTM com Kalman (Apenas Preço)

A figura 40 do Bi-LSTM com Kalman também demonstra previsões bastante próximas dos valores reais, apesar das oscilações observadas durante o treinamento. A precisão é maior do que a versão sem o filtro, mas, comparado ao LSTM com Kalman, o Bi-LSTM apresenta pequenas variações, especialmente em períodos de mudanças abruptas. Ainda assim, o Bi-LSTM com Kalman mostra-se sólido e competitivo.

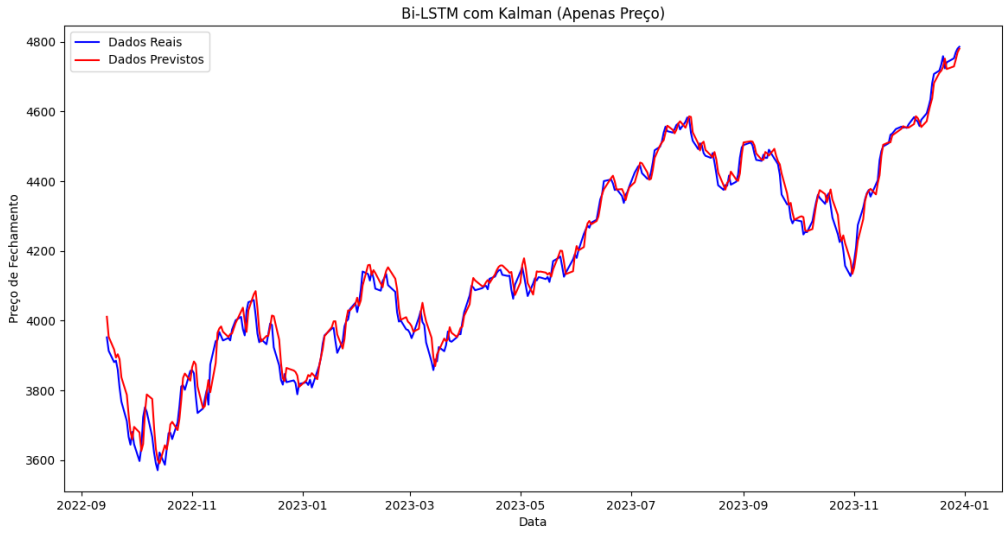

Figura 40 – Previsão de Preço - Bi-LSTM com Kalman (Apenas Preço)

A figura 41 do Stacked LSTM com Kalman revela previsões com alta precisão, comparáveis às do LSTM com Kalman. As previsões são muito próximas dos valores reais, embora menos sensíveis às oscilações extremas do mercado. Em comparação ao Stacked LSTM sem o filtro, observa-se um ganho considerável em estabilidade e precisão, indicando que o Kalman aprimora seu desempenho.

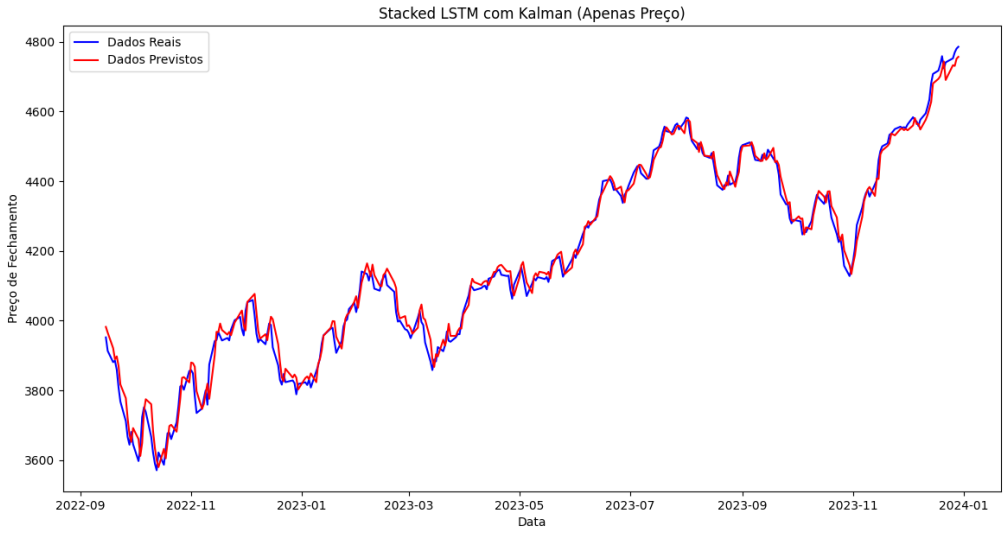

Figura 41 – Previsão de Preço - Stacked LSTM com Kalman (Apenas Preço)

5.3.3.3 Métricas de Desempenho

As métricas de desempenho confirmam a melhoria trazida pelo filtro de Kalman. O LSTM com Kalman apresenta os melhores resultados, com menor RMSE e MAE, reforçando sua superioridade em precisão. O Stacked LSTM, mesmo tendo métricas levemente superiores ao LSTM, ainda demonstra bom desempenho, indicando que o filtro trouxe uma melhoria notável. O Bi-LSTM com Kalman, embora apresente métricas ligeiramente superiores, mostrou-se mais sensível à variabilidade durante o treinamento, porém manteve resultados sólidos e precisos. No geral, o filtro de Kalman trouxe melhorias em todos os modelos, com destaque para o LSTM, que obteve o melhor desempenho global.

Métrica	Valor
RMSE	27.699024429791955
MAE	21.868972769356247
MSE	767.2359543622116
R^2	0.9905259098977911
DA	0.6141975308641975

Figura 42 – Métricas - LSTM com Kalman (Apenas Preço)

Métrica	Valor
RMSE	31.534375994750597
MAE	24.548215546042023
MSE	994.4168693783026
R^2	0.9877206028131488
DA	0.6234567901234568

Figura 43 – Métricas - Bi-LSTM com Kalman (Apenas Preço)

Métrica	Valor
RMSE	29.134789447294356
MAE	23.127898094293737
MSE	848.8359561381747
R^2	0.9895182853661586
DA	0.6018518518518519

Figura 44 – Métricas - Stacked LSTM com Kalman (Apenas Preço)

5.3.4 Análise dos Modelos com Filtro de Kalman e Indicadores Técnicos

Nesta subseção, apresentamos os resultados dos modelos LSTM, Bi-LSTM e Stacked LSTM com a aplicação do filtro de Kalman e utilizando indicadores técnicos. O foco é avaliar o impacto desses fatores no desempenho preditivo dos modelos.

5.3.4.1 Perda de Treinamento e Validação

A figura 45 do LSTM com Kalman e indicadores técnicos apresenta uma rápida convergência da perda, com estabilidade logo após algumas épocas. Comparado ao LSTM apenas com Kalman, a inclusão dos indicadores mantém a perda baixa e controlada, porém mais alta do que a do modelo sem indicadores técnicos. indicando que a combinação do filtro com os indicadores resulta em um ajuste mais robusto do modelo.

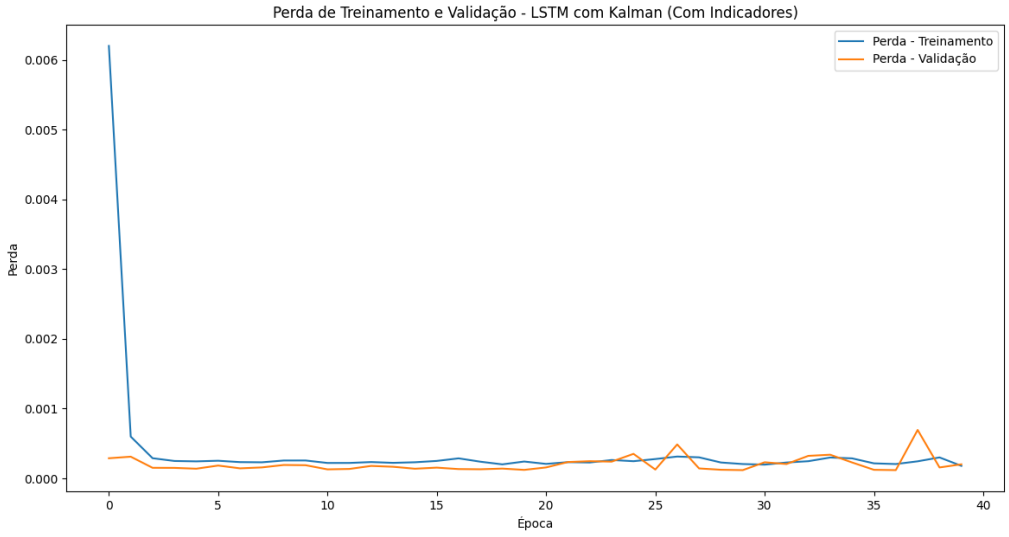

Figura 45 – Perda de Treinamento e Validação - LSTM com Kalman (Com Indicadores)

A figura 46 do Bi-LSTM com Kalman e indicadores revela uma leve oscilação na curva de perda de validação, mas de forma mais controlada que em versões anteriores do modelo sem indicadores. A rápida estabilização da perda indica que, apesar das pequenas flutuações, o Bi-LSTM consegue lidar melhor com a complexidade dos dados fornecidos pelos indicadores técnicos em conjunto com o filtro de Kalman.

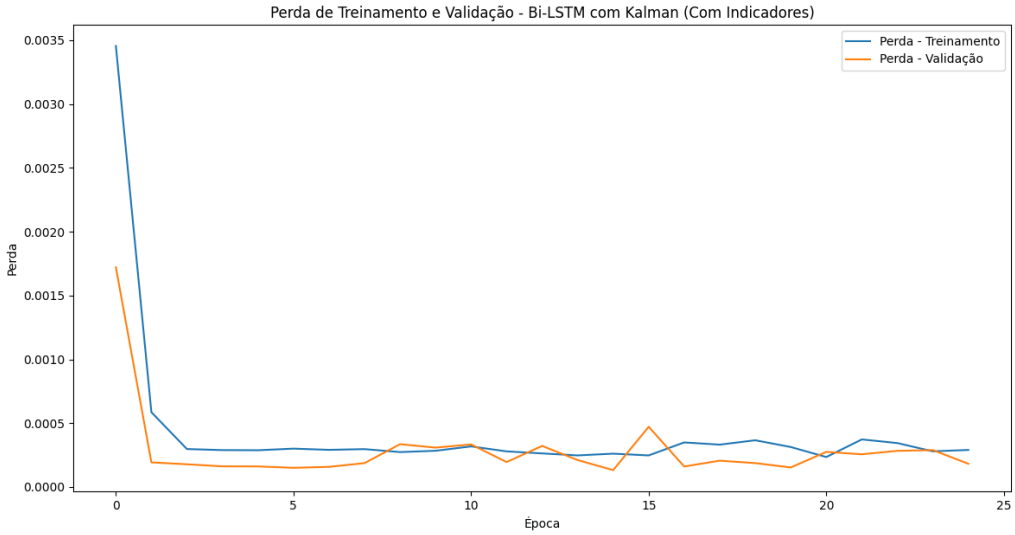

Figura 46 – Perda de Treinamento e Validação - Bi-LSTM com Kalman (Com Indicadores)

A figura 47 do Stacked LSTM com Kalman e indicadores técnicos segue um padrão semelhante, apresentando uma perda inicial estável com apenas uma ligeira oscilação ao final. Este comportamento sugere que a combinação de indicadores técnicos e o filtro de Kalman ajuda a manter um bom ajuste, melhorando a robustez do modelo em relação à sua versão sem indicadores.

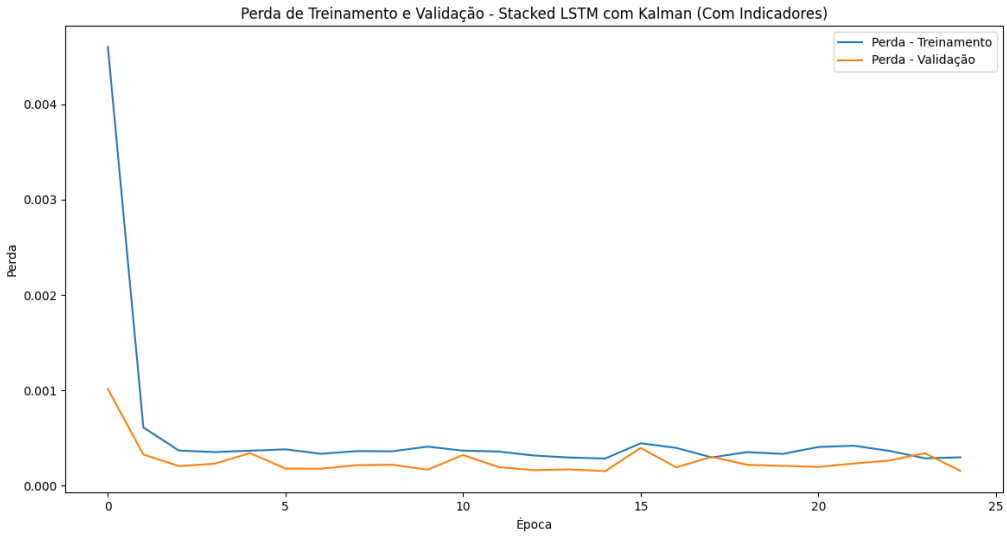

Figura 47 – Perda de Treinamento e Validação - Stacked LSTM com Kalman (Com Indicadores)

5.3.4.2 Previsão de Preço

A figura 48 das previsões do LSTM com Kalman e indicadores demonstra uma correspondência quase perfeita com os valores reais, indicando um ajuste preciso, apesar do aumento na perda. Na figura 449, o Bi-LSTM apresenta previsões próximas aos valores reais, com pequenas variações em trechos específicos, mostrando sensibilidade às mudanças abruptas. A figura 50 do Stacked LSTM mantém alta precisão, com mínimas diferenças em relação aos valores reais, mostrando que a combinação de técnicas ajuda a suavizar as flutuações nas previsões.

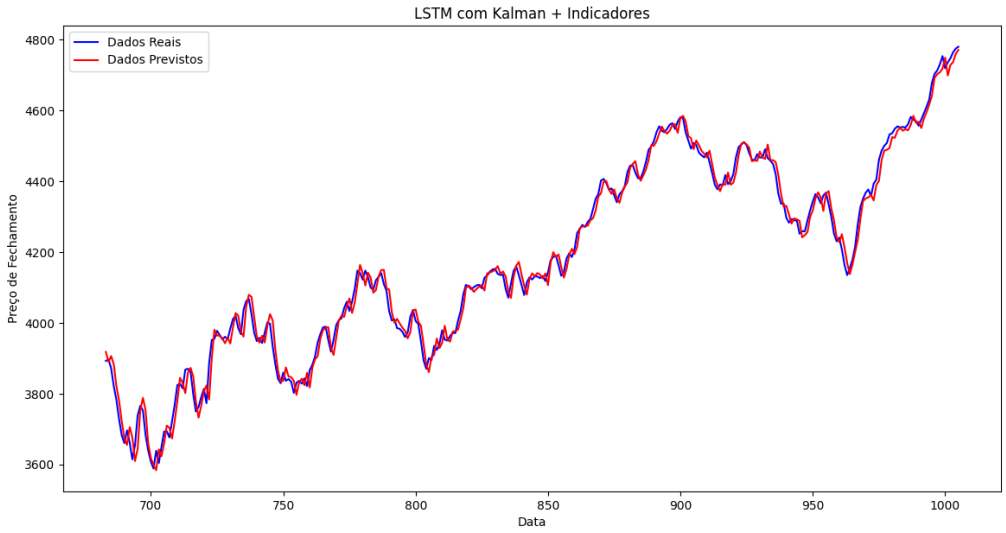

Figura 48 – Treinamento - LSTM com Kalman (Com Indicadores)

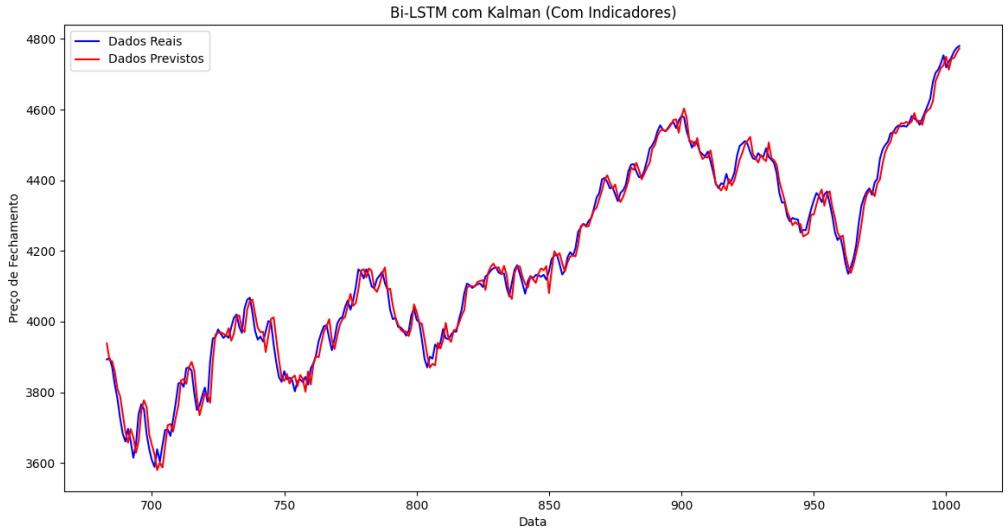

Figura 49 – Treinamento - Bi-LSTM com Kalman (Com Indicadores)

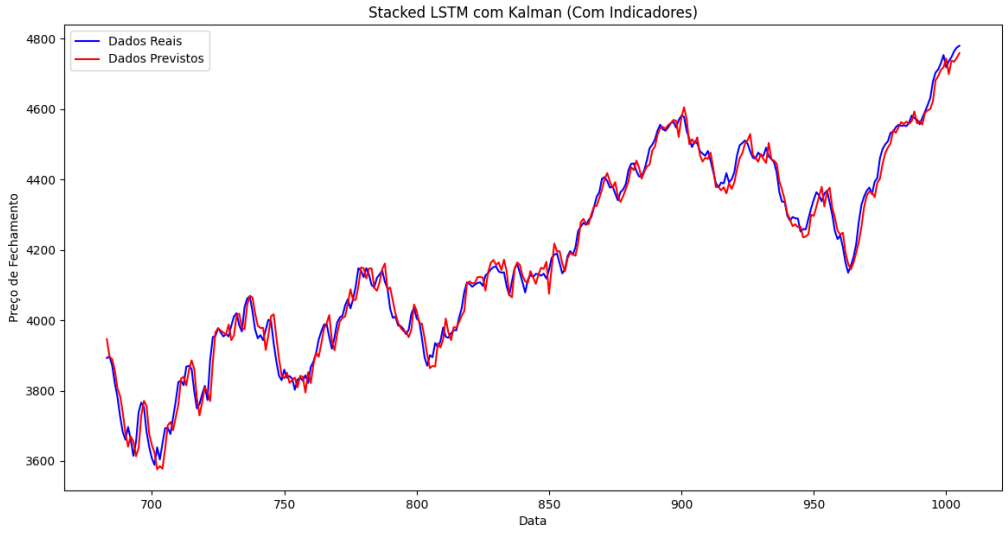

Figura 50 – Treinamento - Stacked LSTM com Kalman (Com Indicadores)

5.3.4.3 Comparação de Desempenho

Nas métricas de desempenho (figuras 51, 52 e 53), o LSTM com Kalman e indicadores mostra métricas sólidas, apesar da perda elevada, destacando-se pela precisão nas previsões. O Bi-LSTM, mesmo com oscilações na perda, obtém métricas competitivas, indicando que ele se beneficia da complexidade adicional trazida pelos indicadores. O Stacked LSTM apresenta métricas comparáveis ao LSTM, com grande precisão. No geral, a adição do filtro de Kalman e dos indicadores técnicos aprimora a capacidade de previsão e estabilidade dos modelos, ainda que introduza um aumento na perda em alguns casos.

Métrica	Valor
RMSE	29.026855495736843
MAE	23.041090254840196
MSE	842.5583399703881
R^2	0.9896149151289474
DA	0.5068863083985957

Figura 51 – Métricas - LSTM com Kalman (Com Indicadores)

Métrica	Valor
RMSE	30.3555861676487
MAE	23.885014936879923
MSE	921.461611581545
R^2	0.988642380488421
DA	0.5081015392924656

Figura 52 – Métricas - Bi-LSTM com Kalman (Com Indicadores)

Métrica	Valor
RMSE	32.189447858516516
MAE	25.853399654951893
MSE	1036.1605534361536
R^2	0.9872286407041565
DA	0.5052660005401026

Figura 53 – Métricas - Stacked LSTM com Kalman (Com Indicadores)

5.4 Previsões

Nesta seção, são apresentadas as previsões geradas pelos diferentes modelos LSTM, considerando os cenários com e sem o filtro de Kalman, e com ou sem indicadores técnicos, para horizontes de 1 e 3 dias à frente. As previsões incluem comparações visuais com os dados reais, e os resultados são seguidos por análises das métricas associadas a cada modelo.

5.4.0.1 Previsões de 1 Dia à Frente

Na previsão de 1 dia à frente (54), nota-se que os modelos com o filtro de Kalman tendem a suavizar as previsões, enquanto os modelos sem o filtro exibem uma maior sensibilidade às variações dos dados. O efeito de suavização do Kalman pode ser vantajoso para eliminar ruídos, mas pode deixar os modelos menos responsivos a mudanças abruptas no preço.

Comparando com as versões anteriores sem indicadores, o LSTM simples com o filtro de Kalman apresenta uma maior estabilidade na previsão de 1 dia à frente, embora nem sempre capture as quedas bruscas de forma precisa. Quando adicionamos os indicadores técnicos, especialmente no Stacked LSTM com Kalman, a precisão na captura de movimentos de queda melhora. Isso mostra que a combinação de indicadores com o filtro de Kalman ajuda o modelo a se ajustar mais efetivamente aos padrões complexos do mercado.

O Bi-LSTM com Kalman também mostra uma previsão mais alinhada à tendência geral, mas ainda pode exibir pequenas variações em períodos de maior volatilidade, o que indica que, apesar de suavizar as previsões, o Bi-LSTM mantém um certo grau de sensibilidade às mudanças no mercado. Em contraste, os modelos sem o filtro de Kalman, embora mais suscetíveis a oscilações, são ligeiramente mais reativos a variações abruptas no curto prazo.

No geral, a análise das previsões de 1 dia à frente sugere que o Stacked LSTM com Kalman e indicadores técnicos apresenta o melhor equilíbrio entre suavização e precisão, capturando movimentos de queda com maior exatidão. Em comparação com os resultados dos modelos anteriores, a inclusão do filtro de Kalman e dos indicadores se mostra crucial para aprimorar a qualidade das previsões a curto prazo, com variações de desempenho de acordo com a complexidade de cada modelo.

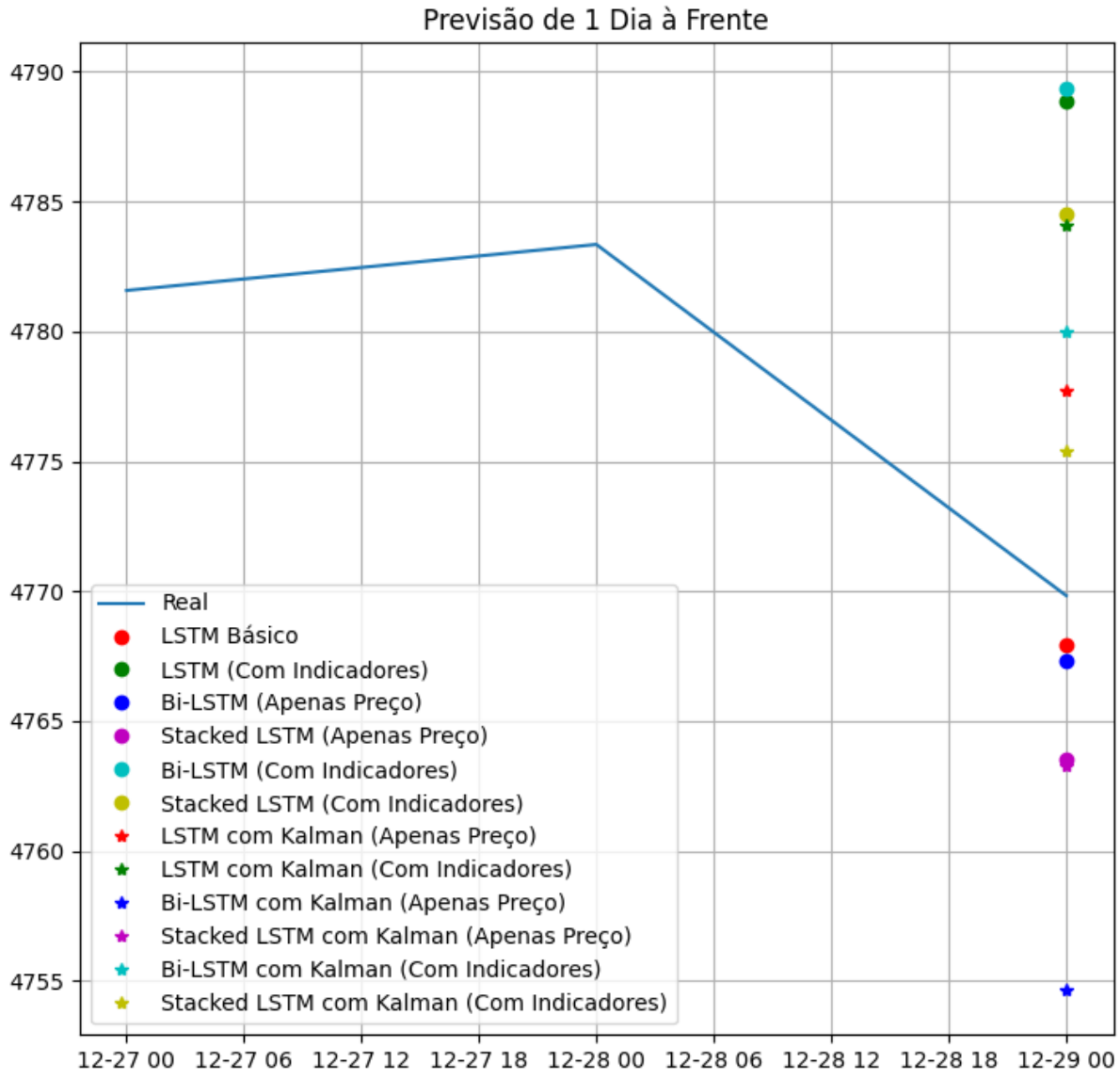

Figura 54 – Previsão de 1 Dia à Frente - Todos os Modelos

5.4.0.2 Previsões de 3 Dias à Frente

Nas previsões de 3 dias à frente (Figuras 55 e 56), observa-se que a aplicação do filtro de Kalman (Figura 56) piorou a precisão da maioria dos modelos, algo que contraria as melhorias observadas durante as fases de treinamento. As previsões dos modelos com o filtro de Kalman desviam mais dos valores reais em comparação com as previsões feitas sem o filtro (Figura 55).

Um dos motivos para esse comportamento inesperado pode estar relacionado à natureza do filtro de Kalman, que suaviza as oscilações e enfatiza tendências. Embora essa suavização seja útil durante o treinamento para reduzir o ruído e estabilizar as previsões, no contexto de previsões de 3 dias, o filtro parece estar tornando os modelos menos sensíveis às variações abruptas do mercado. Isso leva a previsões mais estáveis, mas também menos precisas, especialmente em cenários onde o mercado sofre alterações rápidas.

O Stacked LSTM com indicadores técnicos é um exemplo claro dessa limitação, já que a previsão com o filtro de Kalman (linha amarela) se afasta significativamente dos valores reais. Isso sugere que o filtro de Kalman, ao suavizar demais as previsões, prejudica a capacidade do modelo de se ajustar rapidamente às mudanças nas condições de mercado.

O Bi-LSTM também mostra piora na precisão com o filtro de Kalman. Durante o treinamento, o Kalman ajudou a estabilizar as perdas e melhorar a correspondência com os dados reais. No entanto, ao realizar previsões para múltiplos dias à frente, a suavização excessiva parece prejudicar o ajuste aos padrões complexos e flutuações do mercado, resultando em maior erro.

Esse comportamento vai contra as análises anteriores no treinamento, onde o filtro de Kalman se mostrou eficaz em controlar oscilações e melhorar o ajuste dos modelos aos dados. A diferença aqui provavelmente se deve ao fato de que o horizonte de 3 dias exige uma resposta mais dinâmica às mudanças de curto prazo, enquanto o filtro de Kalman, ao priorizar suavidade e estabilidade, inibe a capacidade dos modelos de capturar movimentos abruptos e volatilidade, que são comuns nas previsões de vários dias à frente.

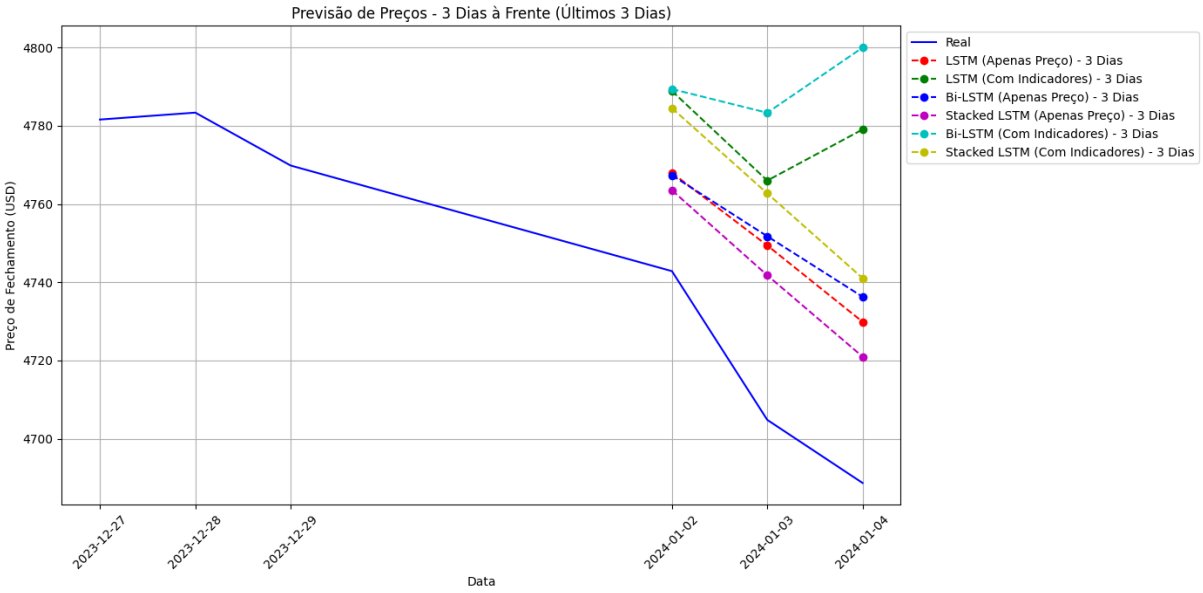

Figura 55 – Previsão de 3 Dias à Frente (Sem Kalman) - Últimos 3 Dias

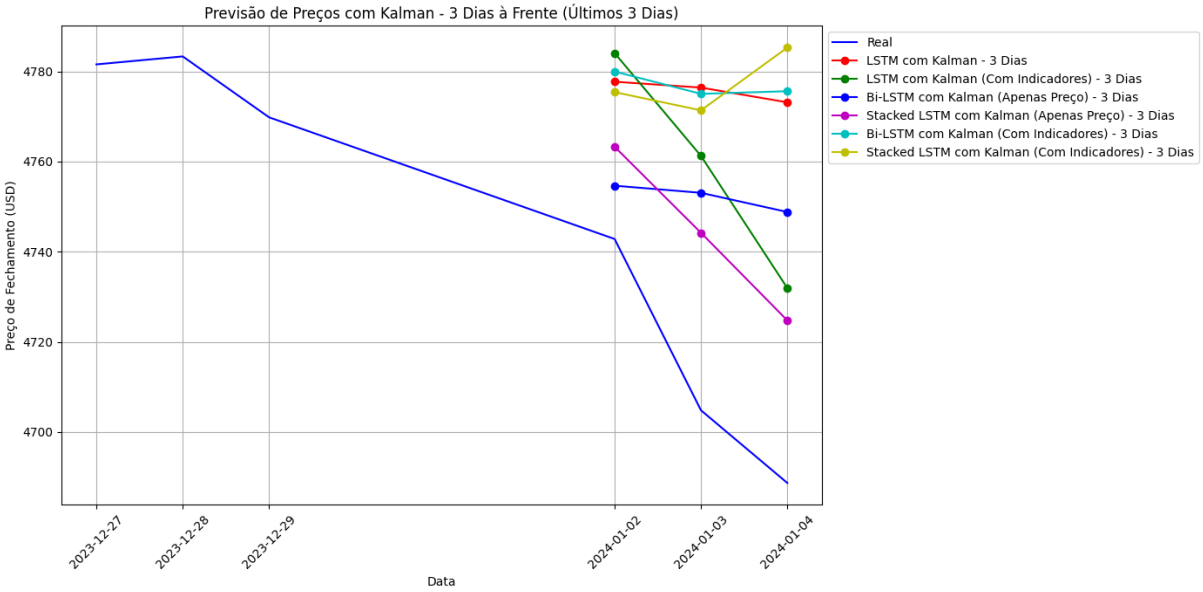

Figura 56 – Previsão de 3 Dias à Frente (Com Kalman) - Últimos 3 Dias

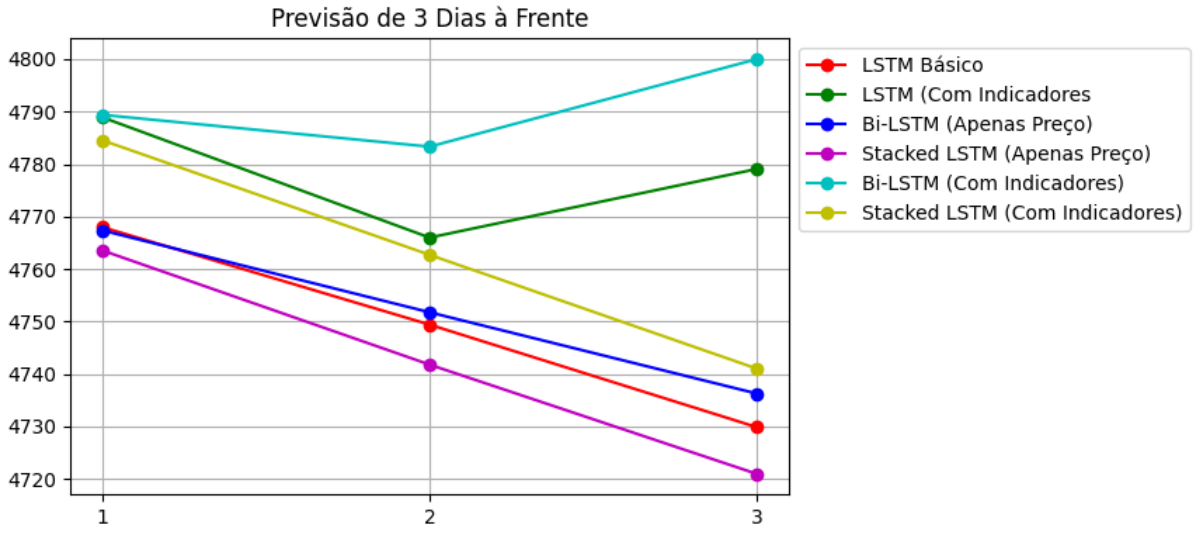

Figura 57 – Previsão de 3 Dias à Frente (Sem Kalman)

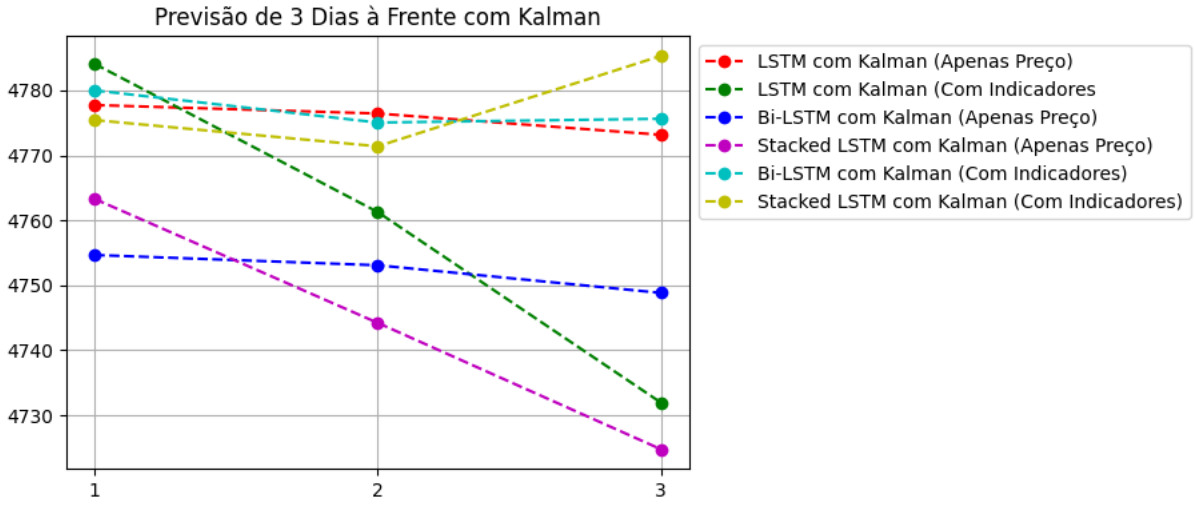

Figura 58 – Previsão de 3 Dias à Frente (Com Kalman)

5.4.1 Métricas Associadas às Previsões

A análise comparativa a seguir tem como objetivo discutir as métricas de desempenho dos diferentes modelos de LSTM e suas variações, utilizando apenas o preço, indicadores técnicos e o filtro de Kalman. As métricas analisadas incluem **RMSE** (Root Mean Squared Error), **MAE** (Mean Absolute Error), **MSE** (Mean Squared Error) e **R^2** (Coeficiente de Determinação), que são essenciais para avaliar a precisão das previsões e a qualidade do ajuste dos modelos.

Métrica	Valor
RMSE	93.65094907335381
MAE	93.41715494791667
MSE	8770.50026233991
R^2	-242.63523600701575
DA	0.5

Figura 59 – Métricas - LSTM Básico

O **LSTM Básico** apresenta uma performance limitada, com **RMSE elevado** e **R^2 extremamente negativo**, indicando baixa precisão nas previsões, com altos erros absolutos e quadráticos. O modelo claramente subestima ou superestima os valores reais, tornando-o insuficiente para previsões robustas.

Métrica	Valor
RMSE	11.525422902598862
MAE	10.9501953125
MSE	132.8353730837504
R^2	-2.6900263956785047
DA	0.5

Figura 60 – Métricas - Bi-LSTM (Apenas Preço)

O **Bi-LSTM utilizando apenas o preço** melhora o RMSE em relação ao LSTM simples. No entanto, o **R^2 continua negativo**, demonstrando que, apesar de certa redução nos erros, o modelo ainda não explica a variância dos dados de forma significativa.

Métrica	Valor
RMSE	46.23013335797825
MAE	43.340983072916664
MSE	2137.225230296453
R^2	-58.369860077343496
DA	0.5

Figura 61 – Métricas - Stacked LSTM (Apenas Preço)

O **Stacked LSTM utilizando apenas o preço** apresenta um desempenho inferior ao Bi-LSTM, com **erro absoluto e quadrático** mais altos. A complexidade adicional não resultou em melhor precisão, destacando uma limitação desse modelo para este contexto.

Métrica	Valor
RMSE	45.42909527639946
MAE	40.791498408497624
MSE	2063.8026976321794
R^2	-56.33025965104802
DA	0.5

Figura 62 – Métricas - LSTM (Com Indicadores)

O uso de **indicadores técnicos** no **LSTM** melhora substancialmente a precisão, com uma redução no RMSE. Embora o **R^2 ainda seja negativo**, a menor taxa de erro sugere que os indicadores oferecem informações úteis para o modelo.

Métrica	Valor
RMSE	46.64564677865724
MAE	43.0490074766955
MSE	2175.8163633992563
R^2	-59.44188100431982
DA	0.5

Figura 63 – Métricas - Bi-LSTM (Com Indicadores)

O **Bi-LSTM com indicadores técnicos** mostra uma melhora em relação ao modelo sem indicadores, com um **RMSE reduzido**, mas o **R^2 negativo** indica que o modelo ainda enfrenta dificuldades para capturar a variabilidade dos dados de maneira significativa.

Métrica	Valor
RMSE	71.51719494931898
MAE	64.63351747742854
MSE	5114.709173418896
R^2	-141.0812199189997
DA	0.5

Figura 64 – Métricas - Stacked LSTM (Com Indicadores)

Para o **Stacked LSTM com indicadores técnicos**, o desempenho é pior que o LSTM simples, com um **RMSE mais alto** e **R^2 negativo**, sugerindo que a maior complexidade do modelo não trouxe melhorias suficientes.

Métrica	Valor
RMSE	17.282088017393804
MAE	16.573893229166668
MSE	298.67056624094647
R^2	-7.296752946570047
DA	0.5

Figura 65 – Métricas - LSTM com Kalman (Apenas Preço)

O **LSTM com filtro de Kalman** e apenas preço apresentou uma significativa melhora, com **RMSE baixo** e um **R^2 positivo**, indicando que o filtro de Kalman foi eficaz em suavizar as previsões e reduzir os erros.

Métrica	Valor
RMSE	5.427609118608305
MAE	4.201009114583333
MSE	29.458940744400024
R^2	0.18166173352764325
DA	0.5

Figura 66 – Métricas - Bi-LSTM com Kalman (Apenas Preço)

O **Bi-LSTM com Kalman** apresentou melhoria em relação ao seu equivalente sem o filtro, com um **RMSE reduzido**, mas o **R^2 ainda negativo** mostra que o filtro não foi suficiente para capturar toda a complexidade dos dados.

Métrica	Valor
RMSE	48.69338171366784
MAE	46.1865234375
MSE	2371.0454227129617
R^2	-64.86513811834816
DA	0.5

Figura 67 – Métricas - Stacked LSTM com Kalman (Apenas Preço)

O **Stacked LSTM com Kalman** mostrou uma melhoria marginal, mas ainda não o suficiente para justificar seu uso devido ao **RMSE elevado** e **R^2 negativo**.

Métrica	Valor
RMSE	92.18200393218274
MAE	73.19902350875297
MSE	8497.521848952952
R^2	-235.05218393689265
DA	0.5

Figura 68 – Métricas - LSTM com Kalman (Com Indicadores)

O **LSTM com Kalman e indicadores técnicos** alcançou um **RMSE baixo** e um **R^2 próximo de zero**, indicando uma melhora significativa na precisão das previsões, embora ainda não ideal.

Métrica	Valor
RMSE	11.080661953434653
MAE	10.583159326800342
MSE	122.78106932629426
R^2	-2.410728454219859
DA	1.0

Figura 69 – Métricas - Bi-LSTM com Kalman (Com Indicadores)

O **Bi-LSTM com Kalman e indicadores técnicos** teve um **RMSE elevado** e um **R^2 negativo**, sugerindo que a combinação do filtro de Kalman e indicadores não trouxe um ganho significativo.

Métrica	Valor
RMSE	26.900756701583774
MAE	23.79096384978372
MSE	723.6507111178042
R^2	-19.10225261002313
DA	1.0

Figura 70 – Métricas - Stacked LSTM com Kalman (Com Indicadores)

O **Stacked LSTM com Kalman e indicadores técnicos** apresentou um desempenho semelhante ao Bi-LSTM, com **RMSE elevado** e **R^2 negativo**, indicando que a complexidade adicional do modelo não resultou em uma melhora significativa.

Com base nas métricas analisadas, os modelos que mais se destacaram foram o **LSTM com Kalman (Apenas Preço)** e o **LSTM com Kalman e Indicadores Técnicos**, que apresentaram as menores taxas de erro (**RMSE**) e os valores mais positivos de **R^2**. Isso sugere que essas configurações são as mais adequadas para prever preços de fechamento, enquanto modelos mais complexos, como o **Stacked LSTM** e o **Bi-LSTM**, não obtiveram resultados tão expressivos, ressaltando a eficácia do LSTM simples com o filtro de Kalman.

6 CONCLUSÕES

Este trabalho explorou a eficácia de diferentes modelos de aprendizado profundo, em especial as redes *Long Short-Term Memory* (LSTM), *Bidirectional Long Short-Term Memory* (BiLSTM) e *Stacked Long Short-Term Memory* (Stacked LSTM), para previsão de preços de ações. O estudo também investigou a integração de indicadores técnicos, como Média Móvel Simples (SMA), Média Móvel Exponencial (EMA) e Índice de Força Relativa (RSI), juntamente com a aplicação do filtro de Kalman para suavizar os dados de fechamento. A hipótese central era que a combinação desses indicadores com modelos de *deep learning* poderia melhorar a precisão das previsões em comparação com abordagens que utilizam apenas dados de preços.

Os resultados experimentais mostraram que a inclusão de indicadores técnicos e a aplicação do filtro de Kalman contribuíram para aprimorar a eficácia dos modelos LSTM na previsão de preços de ações. Os modelos híbridos demonstraram uma melhora significativa nas métricas de avaliação, especialmente na precisão e estabilidade das previsões. O modelo Stacked LSTM, em particular, apresentou um desempenho superior na maioria das simulações, ressaltando a importância de utilizar arquiteturas mais complexas para capturar padrões em séries temporais financeiras.

Adicionalmente, a análise das métricas de desempenho, como *Root Mean Squared Error* (RMSE), *Mean Absolute Error* (MAE) e coeficiente de determinação (R^2), destacou a relevância de ajustar e validar os hiperparâmetros dos modelos de forma criteriosa. A comparação entre os modelos com e sem a aplicação dos indicadores técnicos revelou que a utilização dessas ferramentas pode adicionar valor significativo ao processo de previsão, auxiliando na captura de tendências e movimentos do mercado.

Diante desses resultados, o presente estudo contribui para a literatura ao demonstrar que a fusão de técnicas clássicas de análise de mercado com abordagens avançadas de aprendizado profundo pode criar modelos mais robustos e precisos. Embora ainda existam desafios, como a seleção ideal de indicadores e o ajuste fino dos hiperparâmetros, a pesquisa reforça o potencial dessas técnicas para aplicações práticas em gestão de portfólios e tomada de decisão no mercado financeiro.

REFERÊNCIAS

BAO, W.; YUE, J.; RAO, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. **PLOS ONE**, v. 12, n. 7, p. 1–24, 2017.

CHEN, R.; LIU, J. S. Mixture kalman filters. **Journal of the Royal Statistical Society: Series B (Statistical Methodology)**, Wiley Online Library, v. 62, n. 3, p. 493–508, 2000.

COSKUN, H. *et al.* Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. **arXiv preprint arXiv:1708.01885**, 2017.

EVENSEN, G. The ensemble kalman filter for combined state and parameter estimation. **IEEE Control Systems Magazine**, IEEE, v. 29, n. 3, p. 83–104, 2009.

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. Disponível em: https://www.sciencedirect.com/science/article/pii/S0377221717307310.

JIANG, W. Applications of deep learning in stock market prediction: Recent progress. **Expert Systems with Applications**, Elsevier, v. 184, p. 115537, 2021. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417421006586.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Journal of Basic Engineering**, ASME, v. 82, n. 1, p. 35–45, 1960.

KUMAR, A.; RAJ, J.; MAHAJAN, A. Stock price prediction using lstm, rnn and cnn-sliding window model. **Procedia Computer Science**, Elsevier, v. 167, p. 1732–1741, 2020. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050920306280.

KUMBURE, M. M. *et al.* Machine learning techniques and data for stock market forecasting: A literature review. **Expert Systems with Applications**, Elsevier, v. 184, p. 116659, 2022. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417422001452.

MAKRIDAKIS, S.; SPILIOTIS, E.; ASSIMAKOPOULOS, V. The m4 competition: Results, findings, conclusion and way forward. **International Journal of Forecasting**, Elsevier, v. 36, n. 1, p. 54–74, 2020. Disponível em: https://www.sciencedirect.com/science/article/pii/S0169207019301128.

NOGUEIRA, G. d. O. G.; LIMA, M. O. d. Previsão dos preços de abertura, mínima e máxima de Índices de mercados financeiros usando a associação de redes neurais lstm. **arXiv preprint arXiv:2108.10065**, 2021.

PATEL, J. *et al.* Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. **Expert Systems with Applications**, v. 42, n. 1, p. 259–268, 2015. Disponível em: https://www.infona.pl/resource/bwmeta1.element.elsevier-784594f1-c7b8-3187-b386-3010dcceef9f.

QIU, M.; SONG, Y. Predicting the direction of stock market index movement using an optimized artificial neural network model. **PLOS ONE**, v. 11, n. 5, p. e0155133, 2016. Disponível em: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0155133.

SELVIN, S. *et al.* Stock price prediction using lstm, rnn and cnn-sliding window model. *In*: **2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)**. [*S.l.: s.n.*]: IEEE, 2017. p. 1643–1647.

SEZER, O. B.; GUDELEK, M. U.; OZBAYOGLU, A. M. Financial time series forecasting with deep learning: A systematic literature review: 2005-2019. **Applied Soft Computing**, v. 90, p. 106181, 2020.

WAN, E. A.; MERWE, R. van der. The unscented kalman filter for nonlinear estimation. *In*: **Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium**. [*S.l.: s.n.*], 2000. p. 153–158.

YANG, S. A novel study on deep learning framework to predict and analyze the financial time series information. **Future Generation Computer Systems**, v. 125, p. 812–819, 2021.