UNIVERSIDADE FEDERAL DO PAMPA

GEORGE PHELIPE AZEVEDO

ANÁLISE DE SÉRIES TEMPORAIS NO MERCADO FINANCEIRO COMO FERRAMENTA DE AUXÍLIO DE INVESTIMENTO

Bagé
2025

GEORGE PHELIPE AZEVEDO

ANÁLISE DE SÉRIES TEMPORAIS NO MERCADO FINANCEIRO COMO FERRAMENTA DE AUXÍLIO DE INVESTIMENTO

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Gerson Alberto Leiria Nunes

Bagé
2025

Ficha catalográfica elaborada automaticamente com os dados fornecidos pelo(a) autor(a) através do Módulo de Biblioteca do Sistema GURI (Gestão Unificada de Recursos Institucionais).

A994 Azevedo, George Phelipe

Análise de Séries temporais no mercado financeiro como ferramenta de auxílio de investimento / George Phelipe Azevedo.

95 f.: il.

Orientador: Gerson Alberto Leiria Nunes

Trabalho de Conclusão de Curso (Graduação) – Universidade Federal do Pampa, Engenharia de Computação, 2025.

I. Título.

GEORGE PHELIPE DURAES AZEVEDO

ANÁLISE DE SÉRIES TEMPORAIS NO MERCADO FINANCEIRO COMO FERRAMENTA DE AUXÍLIO DE INVESTIMENTO

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Dissertação defendida e aprovada em: 15 de dezembro de 2025.

Banca examinadora:

_____

Prof. Dr. Gerson Leiria Nunes

Orientador

(UNIPAMPA)

_____

Prof. Dr. Érico Marcelo Hoff Amaral

(UNIPAMPA)

_____

Prof. Dr. Milton Roberto Heinen

(UNIPAMPA)

Assinado eletronicamente por **GERSON ALBERTO LEIRIA NUNES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 16/12/2025, às 10:01, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **ERICO MARCELO HOFF DO AMARAL**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/12/2025, às 00:49, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **MILTON ROBERTO HEINEN**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/12/2025, às 18:02, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Dedico este trabalho a minha vó Maria da Glória, que sempre acreditou e investiu nos meus sonhos.

AGRADECIMENTO

Agradeço à Deus por me permitir chegar até aqui. Agradeço à minha família por todo apoio, incentivo e suporte incondicional ao longo de todos esses anos. À Geovanna por todo carinho, e por sempre estar do meu lado me dando apoio. Um agradecimento à meu orientador, Gerson Alberto Leiria Nunes pela dedicação, paciência e parceria. À todos os meus amigos que direta ou indiretamente estiveram comigo me ajudando e me acompanhando nessa jornada.

"It is only with the heart that one can see rightly; what is essential is invisible to the eye." — Antoine de Saint-Exupéry, *The Little Prince* (1943)

RESUMO

O mercado financeiro é extremamente instável e dinâmico, o que pode gerar dificuldades na tomada de decisão por parte de novos investidores. Este trabalho aborda conceitos de séries temporais e utiliza modelos de previsão, como Facebook's Prophet, LSTM e ARIMA, com o objetivo de auxiliar a análise de dados e o processo decisório. Ademais, foi desenvolvido um software capaz de realizar análises temporais de dados históricos de ativos financeiros. O objetivo do trabalho é desenvolver uma ferramenta que avalie o potencial de lucro de ativos financeiros nos próximos meses e auxilie investidores novatos na tomada de decisões, por meio da análise de dados históricos e da identificação de padrões como tendências e sazonalidades. Os resultados obtidos até o momento indicam que os modelos de previsão são capazes de capturar o comportamento temporal dos ativos financeiros, com destaque para o modelo LSTM, que apresentou melhor desempenho preditivo em relação ao ARIMA e ao Prophet, enquanto estes se mostraram eficientes na modelagem de tendências e componentes sazonais. Dessa forma, a ferramenta demonstra potencial para apoiar a elaboração de estratégias de investimento mais fundamentadas e eficazes no contexto do mercado financeiro.

Palavras-chave: Séries temporais. Facebook's Prophet. LSTM. ARIMA. Mercado Financeiro..

ABSTRACT

The financial market is extremely unstable and dynamic, which can make decision-making difficult for new investors. This work addresses concepts of time series analysis and employs forecasting models such as Facebook's Prophet, LSTM, and ARIMA to support data analysis and decision-making. In addition, a software tool was developed to perform temporal analyses of historical financial asset data. The objective of this study is to develop a tool that evaluates the profit potential of financial assets over the coming months and assists novice investors in their decision-making process through the analysis of historical data and the identification of patterns such as trends and seasonality. The results obtained so far indicate that the forecasting models are capable of capturing the temporal behavior of financial assets, with the LSTM model showing superior predictive performance compared to ARIMA and Prophet, while the latter proved effective in modeling trends and seasonal components. Therefore, the proposed tool demonstrates potential to support the development of more informed and effective investment strategies in the financial market.

Keywords: Time series. Facebook's Prophet. LSTM. ARIMA. Financial Market..

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADF	Augmented Dickey-Fuller
AIC	Akaike Information Criterion
API	Interfaces de Programação de Aplicações
ARIMA	Autoregressive Integrated Moving Average
BIC	Bayesian Information Criterion
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DFN	Deep Feedforward Networks
DNN	Deep Neural Networks
FP	Facebook Prophet
GPU	Graphics Processing Units
GRU	Gated Recurrent Unit
IA	Inteligência Artificial
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
KPSS	Kwiatkowski–Phillips–Schmidt–Shin
LSTM	Long Short Term Memory
MAE	Erro Absoluto Médio
MAPE	Erro Percentual Absoluto Médio
MSE	Mean Square Error
NN	Neural Networks
OHLCV	Open, High, Low, Close, Volume
ReLU	Unidade Linear Retificada
RMSE	Erro Quadrático Médio
RNA	Redes Neurais Artificiais
RNN	Recurrent Neural Networks

SUMÁRIO

1 INTRODUÇÃO

1.1 Introdução

O termo *analisar* tem por definição o ato de examinar e avaliar minuciosamente elementos com a finalidade de tirar conclusões baseadas nos dados fornecidos, permitindo a tomada de decisões embasadas. No cenário do mercado financeiro, essas análises se mostram ainda mais essenciais no auxílio das decisões relacionadas a novos investimentos, pois o ambiente econômico é caracterizado por sua volatilidade e complexidade. Com o aumento do volume de dados disponíveis e o avanço das tecnologias de processamento e modelagem, a análise de séries temporais se destaca como uma ferramenta fundamental para investidores, por permitir a observação do comportamento histórico do mercado, a identificação de padrões recorrentes e a previsão do desempenho futuro de ativos financeiros. Essa abordagem possibilita melhor compreensão das tendências e maior capacidade de antecipação de movimentos de mercado.

Uma série temporal é qualquer sequência de observações ordenadas ao longo do tempo. Quando os dados históricos podem ser usados para fazer previsões e auxiliar nas tomadas de decisão, a análise de uma série temporal torna-se ainda mais atraente. Esses dados podem representar valores diários de poluição de uma cidade, temperaturas mensais, registros de maré ou índices financeiros, sendo este último o foco deste trabalho (MORETTIN; TOLOI, 2005).

Segundo Nielsen (2019), há muito tempo os indicadores de produção e eficiência nos mercados oferecem informações valiosas para análise por meio de séries temporais. A necessidade mais urgente tem sido antecipar cenários econômicos futuros com base em eventos passados. Tais previsões não servem apenas para gerar lucros, mas também contribuem para a promoção da estabilidade econômica e prevenção de crises sociais.

Dada a natureza incerta do mercado financeiro, a habilidade de interpretar dados históricos de maneira eficaz e confiável representa uma vantagem competitiva significativa. Utilizando-se de métodos matemáticos e estatísticos, a análise de séries temporais permite realizar projeções observando o comportamento dos ativos ao longo do tempo, favorecendo decisões estratégicas, como identificar o momento adequado para comprar ou vender um ativo.

No contexto da análise quantitativa, o termo *previsão* refere-se ao ato de estimar

um valor futuro a partir de dados históricos, com objetivo específico de apoiar decisões. Prever não é um fim em si mesmo, mas sim uma ferramenta para subsidiar decisões objetivas, fundamentadas em métodos analíticos complexos e quantitativos que utilizam informações temporalmente estruturadas (MORETTIN, 2017).

De acordo com Mesquita (2019), diversos modelos de previsão foram desenvolvidos historicamente para compreender o comportamento de preços de ativos no mercado financeiro. O avanço da capacidade computacional permitiu o surgimento de modelos cada vez mais sofisticados, incluindo técnicas baseadas em aprendizado de máquina, que lidam com grandes volumes de dados. Contudo, mesmo com tais abordagens modernas, prever o mercado continua sendo um desafio devido à sua natureza dinâmica e imprevisível.

A economia dos países está profundamente ligada ao mercado financeiro. A bolsa de valores é um dos principais indicadores econômicos, responsável por movimentar grandes volumes financeiros e influenciar diretamente decisões de investidores, economistas e formuladores de políticas públicas. Diante da busca por compreender tanto o comportamento de curto quanto de longo prazo dos ativos, modelos computacionais de previsão de séries temporais tornam-se uma alternativa promissora.

As previsões aplicadas ao mercado financeiro estão em constante evolução, impulsionadas pela necessidade de obter resultados mais precisos em um ambiente altamente dinâmico. Entre os inúmeros desafios, destaca-se a influência de fatores externos — decisões políticas, comportamento coletivo dos investidores, crises internacionais, avanços tecnológicos, que tornam difícil prever movimentos de mercado. Por isso, observa-se um crescimento no uso do poder computacional aliado à inteligência artificial para tratar grandes volumes de dados, extrair padrões e modelar o comportamento dos ativos ao longo dos anos.

Nesse contexto, este trabalho desenvolveu um sistema completo de análise e previsão de séries temporais aplicado a ativos da bolsa de valores brasileira (B3), utilizando três modelos amplamente reconhecidos: *Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors* (SARIMAX), *Facebook Prophet* e *Long Short-Term Memory* (LSTM). O sistema foi implementado em *Python* e integra coleta automática de dados via *Yahoo Finance*, processamento das séries, geração de gráficos, comparação de desempenho entre modelos e criação de arquivos CSV contendo métricas e previsões. Além disso, foi desenvolvida uma interface gráfica em *PyQt5*, permitindo ao usuário escolher o ativo, definir o horizonte de previsão, executar os modelos, visualizar

resultados e acessar todas as métricas de forma intuitiva. O sistema também inclui um módulo de *ranking*, que compara os modelos com base em indicadores estatísticos, organiza os resultados e gera automaticamente gráficos e arquivos de saída.

Assim, este trabalho combina fundamentos teóricos de análise de séries temporais com uma implementação prática completa, visando oferecer uma ferramenta precisa e acessível para investidores e profissionais interessados em prever o comportamento futuro de ativos financeiros.

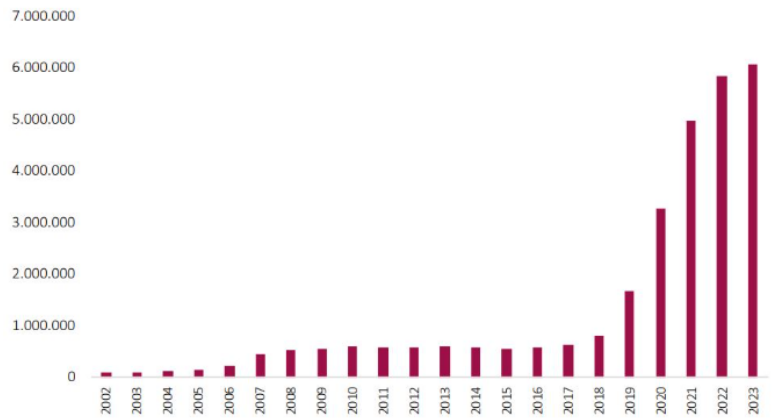
1.2 Problema de pesquisa

A pergunta de pesquisa elaborada: faz-se possível criar um software capaz de fazer análises temporais dos dados dos últimos anos de ativos financeiros com a finalidade de auxiliar novos investidores na tomada de decisão?

1.3 Importância e motivação da pesquisa

Atualmente, um número cada vez maior de pessoas estão investindo em ativos no mercado financeiro brasileiro, conforme o gráfico apresentado na Figura 1, impulsionado pelo aumento do acesso a plataformas de investimento e pelo interesse crescente de novos investidores , tornou-se essencial o desenvolvimento de ferramentas que auxiliem na análises e interpretações dados. Devido as constantes instabilidades desse mercado, busca-se cada vez mais métodos de previsão de mercado.

Figura 1 – Número de investidores no Brasil



Fonte: https://oespecialista.com.br/bolsa-de-valores-total-de-investidores/

Nesse contexto, surgiu a motivação para realizar esse trabalho, criar um recurso que ajude novos investidores a entender o mercado financeiro e antecipar seu comportamento. Com a crescente quantidade de investidores iniciantes no Brasil, muitos enfrentam desafios ao tentar compreender as informações do mercado e tomar decisões embasadas. A volatilidade e a complexidade do setor demandam análises detalhadas e táticas robustas, tornando essencial a utilização de modelos que possibilitem identificar padrões e tendências futuras.

Segundo Wong e Selvi (1998) o futuro das redes neurais na área financeira pode ver uma maior integração com outras tecnologias e técnicas estatísticas existentes ou em desenvolvimento. À medida que os avanços são feitos nas tecnologias de IA e nos sistemas relacionados baseados em computador, deve haver novas oportunidades para aplicar a tecnologia de rede neural para pesquisa financeira.

Usando analises de séries temporais que se apresenta como uma ferramenta essencial para reconhecer padrões históricos e antecipar tendências futuras, fornecendo dados úteis para a tomada de decisões, e ferramentas de séries temporais como Facebook Prophet [1], LTSM (Long Short Term Memory), Redes Neurais (RN) e ARIMA (BOX; JENKINS, 1970), esta pesquisa visa ajudar novos investidores a tomar decisões e escolher os melhores ativos para investir.

1.4 Objetivos

O objetivos deste trabalho é criar um software de analises das séries temporais de ativos financeiros com a finalidade de auxiliar investidores a aumentar seus potenciais de lucro, utilizando modelos numéricos de previsão de series temporais tais quais Facebook Prophet, LSTM e ARIMA, e a partir disto, auxiliar os investidores na tomada de decisão.

1.4.1 Objetivos específicos

- Realizar revisão bibliográfica sobre análise de séries temporais;
- Pesquisar artigos relacionados ao assunto;
- Investigar modelos matemáticos de análises de séries temporais;
- Adquiri dados de entradas para os modelos computacionais de previsão de série

[1] https://facebook.github.io/prophet/

temporal;

- Desenvolver os modelos computacionais de previsão de séries temporais;
- Realizar previsões de valores futuros da série, com auxilio de modelos computacionais;
- Descrever o comportamento da série através da construção de gráfico, verificar a existência de tendências, ciclos e variações sazonais, construção de histogramas e diagramas de dispersão;
- Parametrizar os modelos matemáticos que fazem previsões baseados em séries temporais;
- Analisar os resultados dos modelos testados;
- Validar os resultados dos modelos testados;
- Comparar as métricas dos resultados obtidos de cada modelo;

2 METODOLOGIA

A pesquisa científica pode ser definida como um estudo planejado, uma vez que esta tem como característica do aspecto cientifico da investigação o método de abordagem do problema. A pesquisa cientifica sempre parte de um problema que geralmente é dado por uma pergunta, e para que chegue-se à resposta dessa pergunta existem diversos métodos. A pesquisa exploratória tem como objetivo buscar mais informações a respeito do assunto que será investigado, auxiliando na definição e delimitação do tema da pesquisa. Em geral está relacionada com a forma de pesquisa bibliográfica e estudos de caso, proporcionando uma maior familiaridade com o problema(PRODANOV; FREITAS, 2013).

Segundo Prodanov e Freitas (2013) normalmente, essa pesquisa envolve a revisão da literatura e estudos de caso, fornecendo uma compreensão mais aprofundada do problema. Depois de delimitar o tema, o pesquisador realiza a pesquisa exploratória, com o objetivo de replicar as condições de um evento a ser estudado em um ambiente controlado, a fim de demonstrar como esse evento é produzido.

2.0.1 Material e métodos

O método adotado neste trabalho consiste em uma revisão bibliográfica, com o objetivo de reunir, sintetizar e avaliar o conhecimento existente, seguida pelo desenvolvimento de uma aplicação computacional em Python[1] destinada à análise e previsão de séries temporais financeiras. A condução da pesquisa seguiu duas etapas principais: (i) uma revisão sistemática da literatura, com foco nos fundamentos teóricos e técnicas modernas de modelagem temporal; e (ii) a implementação prática de modelos computacionais aplicados a dados reais de ativos da B3.

Na primeira etapa, foi realizada uma revisão sistemática de estudos, livros, artigos científicos e documentações técnicas relacionados à previsão de séries temporais, abordando conceitos fundamentais, limitações, aplicações e métodos estatísticos e de aprendizado de máquina. Foram considerados modelos amplamente utilizados na literatura, como ARIMA, SARIMAX, Facebook Prophet e redes neurais recorrentes, especialmente as arquiteturas do tipo *Long Short-Term Memory* (LSTM). A revisão permitiu identificar trabalhos correlatos, mapear abordagens existentes e compreender

[1] <https://www.python.org/>

avanços e desafios na área de previsão de ativos financeiros. Tal processo forneceu uma base teórica sólida para orientar as decisões de implementação e experimentação do estudo.

Na segunda etapa, foi desenvolvido um sistema completo de análise temporal utilizando a linguagem Python, integrando bibliotecas específicas para modelagem estatística e computação numérica. O código implementado inclui métodos de pré-processamento, divisão de dados em treino e teste, ajuste de modelos, geração de previsões, cálculo de métricas de desempenho (como *RMSE*, *MAPE* e R^2), além da exportação automática de resultados em arquivos CSV. Para tornar o sistema mais acessível ao usuário final, foi construída uma interface gráfica utilizando a biblioteca PyQt5, permitindo a seleção do ativo, definição do horizonte de previsão, execução dos modelos e visualização dos resultados de forma interativa.

Os modelos implementados, SARIMAX, Prophet e LSTM, foram aplicados a séries históricas obtidas automaticamente via a biblioteca `yfinance`, abrangendo ativos amplamente negociados na bolsa brasileira. O sistema gera gráficos de tendência, previsões futuras, componentes do Prophet (tendência, sazonalidade e pontos de mudança) e espectros de desempenho. Adicionalmente, foi implementado um módulo de *ranking* capaz de comparar os modelos entre si com base nas métricas calculadas, oferecendo uma avaliação quantitativa do desempenho para cada ativo analisado.

Por fim, as metodologias aplicadas foram organizadas seguindo princípios de desenvolvimento ágil, permitindo iteração contínua, validação frequente dos resultados e ajustes rápidos conforme novos testes eram realizados. Dessa forma, este trabalho integra rigor teórico e implementação prática, resultando em uma ferramenta robusta para análise e previsão de séries temporais no contexto financeiro.

2.0.2 Busca e seleção de trabalhos correlatos

Para garantir uma base teórica sólida e identificar estudos relevantes, a busca e seleção de trabalhos correlatos foram realizadas nas bases de dados Google Scholar[2], ResearchGate[3], IEEE Xplore[4], springerlink[5], ACM[6] e portal de teses e dissertações

[2]https://scholar.google.com/
[3]ttps://www.researchgate.net
[4]https://ieeexplore.ieee.org/
[5]https://link.springer.com/
[6]https://dl.acm.org/

da CAPES. Essas plataformas foram escolhidas por sua ampla cobertura de pesquisas acadêmicas e científicas na área de modelagem preditiva e análise de séries temporais no mercado financeiro.

A estratégia de busca foi cuidadosamente planejada para garantir a relevância dos trabalhos selecionados. Levando em conta a vasta quantidade de publicações pouco relacionadas ao assunto, utilizou-se strings de pesquisa específicas. Essas strings abrangem desde métodos tradicionais até as técnicas mais recentes de aprendizado de máquina aplicadas ao mercado financeiro. Para manter a atualidade da revisão, filtramos apenas estudos publicados a partir de 2020. As strings detalhadas podem ser consultadas na Figura 2:

Figura 2 – Strings

String em português	String em inglês
"previsão de preços de ações" OR "previsão de mercado financeiro"	"stock price prediction" OR "financial market forecasting"
"inteligência artificial no mercado financeiro" OR "IA previsão ações"	"artificial intelligence in financial market" OR "AI stock prediction"
"modelo ARIMA para análise de ações" OR "previsão de ações com ARIMA"	"ARIMA model for stock analysis" OR "stock forecasting using ARIMA"
"análise de séries temporais financeiras" OR "modelagem de séries temporais"	"financial time series analysis" OR "time series modeling"
"previsão de ações brasileiras" OR "análise de ações Petrobras"	"Brazilian stock forecasting" OR "Petrobras stock analysis"
"análise comparativa de modelos preditivos" OR "técnicas de previsão de ações"	"comparative analysis of predictive models" OR "stock forecasting techniques"
"modelo LSTM para previsão de mercado financeiro" OR "rede LSTM aplicada a finanças"	"LSTM model for financial market forecasting" OR "LSTM neural network in finance"
"previsão com Prophet" OR "modelos de suavização séries temporais"	"forecasting with Prophet" OR "time series smoothing models"

Fonte: Próprio Autor

2.1 Organização do texto

Este documento está organizado em capítulos. Sendo este primeiro o capítulo introdutório e responsável por apresentar o problema de pesquisa, objetivos e metodologia de pesquisa. Já o capítulo 3 apresenta definições básicas e conceitos utilizados pelos principais escritores do campo. O capítulo 4 é responsável por expor uma revisão dos trabalhos correlatos, tais como: artigos, monografias, dissertações e teses. No capítulo 5 são discutidas as hipóteses de implementações dos modelos computacionais tais como:

ARIMA (*AutoRegressive Integrated Moving Average*), Facebook Prophet, Redes Neurais e etc. Por fim, no capítulo 6 são discutidas as conclusões parciais e os trabalhos futuros.

3 REVISÃO BIBLIOGRÁFICA

Neste capítulo, é fornecido o suporte teórico essencial para a realização deste estudo, discutindo os conceitos e modelos mais relevantes para a previsão de séries temporais no cenário do mercado financeiro. O enfoque recai sobre métodos que utilizam redes neurais, particularmente as LSTM (Long Short-Term Memory), juntamente com modelos estatísticos convencionais como o ARIMA e estratégias contemporâneas como o Facebook Prophet.

3.1 Séries temporais

Atualmente, com o progresso tecnológico e a expansão acelerada da internet, a geração de dados alcançou um nível impressionante. Esta vasta quantidade de informações à disposição proporciona uma chance singular de investigar padrões e tendências, tornando a análise de séries temporais cada vez mais crucial. Devido a ela, há então, a possibilidade de reconhecimento dos comportamentos anteriores, de forma a antecipar acontecimentos futuros e fazer escolhas mais conscientes e estratégicas. Esta habilidade de converter informações históricas em percepções valiosas tem feito da análise de séries temporais um instrumento essencial na era digital. Morettin e Toloi (2005) citam que uma série temporal é qualquer conjunto de observações ordenadas no tempo.

De acordo com Nielsen (2019) A análise de séries temporais consiste em extrair dados estatísticos e resumidos relevantes de pontos dispostos em sequência cronológica. Isto é realizado para avaliar o comportamento anterior e para antecipar o comportamento futuro. Frequentemente, a avaliação de séries temporais se reduz à questão da causalidade, de como o passado pode afetar o futuro.

De acordo com Morettin (2017), no âmbito econômico e financeiro, as séries temporais apresentam algumas características peculiares. Uma das características mais notáveis é o fenômeno denominado agrupamento de volatilidade, no qual a variação dos dados não é constante, mas se altera com o passar do tempo, sendo influenciada pelo comportamento anterior da série. Dados como taxas de câmbio e preços de ações, ao serem registrados em intervalos variáveis ao longo do dia, originam o que denominamos de dados de alta frequência.

A maioria das técnicas de análise de séries temporais pode ser aplicada em

diversos campos. No entanto, uma característica marcante das séries de ativos financeiros é a volatilidade, que, apesar de identificável de várias formas, não é diretamente observável. Para capturar a presença de clusters de volatilidade, utilizam-se *modelos heterocedásticos condicionais (ENGLE, 1982)*. Nesses modelos, a volatilidade de um retorno em um dado instante depende de retornos passados e de outras informações disponíveis até aquele momento, exigindo a definição de uma variância condicional. Essa variância, por não ser constante, difere da variância global (incondicional) da série analisada. Além disso, a média e outros momentos da distribuição dos retornos também podem variar ao longo do tempo. (MORETTIN, 2017).

De acordo com Morettin e Toloi (2005) Uma das suposições mais comuns sobre uma série temporal é a de que ela é estacionária, isto é, evolui ao longo do tempo de forma aleatória em torno de uma média estável, indicando algum tipo de equilíbrio estável. No entanto, a maioria das séries que observa-se na realidade exibe algum tipo de não estacionariedade. Portanto, as séries econômicas e financeiras geralmente exibem tendências, sendo a mais básica a que oscila em torno de uma linha reta, com inclinação positiva ou negativa (tendência linear).

Nos anos 1970 e 1980, o avanço dos computadores e das técnicas econométricas permitiu um maior refinamento na análise de séries temporais, e foi nesse período que surgiram modelos clássicos, como o *"ARIMA"*, desenvolvido por Box e Jenkins (1970), que se tornou amplamente utilizado para prever preços de ativos financeiros. Esses modelos possibilitaram pela primeira vez uma análise mais detalhada de tendências, ciclos e volatilidade, oferecendo aos analistas uma ferramenta prática para prever movimentos de preços a curto e longo prazo.

Na última década, a popularização da inteligência artificial e do aprendizado de máquina revolucionou a análise de séries temporais. Modelos como LSTM (Long Short-Term Memory), introduzido por Hochreiter e Schmidhuber (1997), passaram a ser aplicados ao mercado financeiro para identificar padrões complexos em dados históricos. À medida que a computação foi avançando e o volume de dados foi crescendo, análises em tempo real passaram a ser possibilitadas por essas técnicas, sendo aprimoradas a previsão e a tomada de decisão.

A equação geral de uma série temporal e definida de acordo com a equação 1, sendo T a componente de nível ou tendência de longo prazo, S a componente sazonal, variações sistemáticas em períodos fixos, C é a componente cíclica, variações

de médio/longo prazo sem periodicidade fixa e E é o ruído ou erro.

$$Y_t = T_t + S_t + C_t + E_t \tag{1}$$

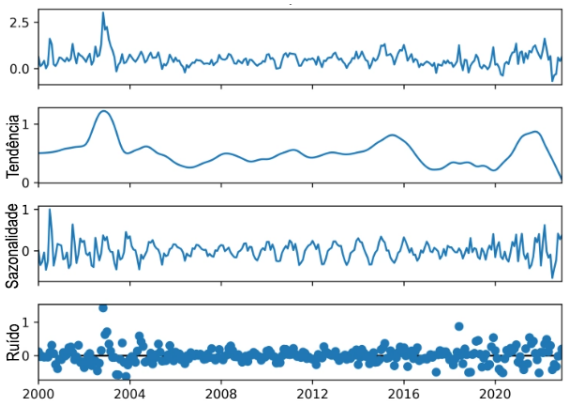
3.2 Decomposição de Séries temporais

Para analisar de forma eficiente os fatores que influenciam os dados ao longo do tempo e facilitar sua interpretação, podemos decompor uma série temporal em quatro componentes principais: tendência, sazonalidade, ciclo e nível. A decomposição de séries temporais (Y_t) são feitas considerando as observações ($Y_t, \quad t = 1,2,\ldots,n$), como uma soma de três componentes não-oberváveis, conforme, logo abaixo, apresenta a equação 2.

$$Y_t = T_t + S_t + \alpha_t \tag{2}$$

Sendo que T_t representam a tendência e S_t a sazonalidade, enquanto α_t é um ruído branco. Em uma série temporal um ruído branco é um tipo de processo estocástico que representa uma sequencia de variáveis aleatórias com média zero, variância constante e ausência correlação (MORETTIN; TOLOI, 2005). A Figura 3, adaptada de Análise Macro (2023), ilustra um gráfico de decomposição de série temporal, no qual a série observada é dividida em três componentes fundamentais: tendência (T_t), sazonalidade (S_t) e ruído aleatório (R_t). Essa decomposição parte do princípio de que uma série temporal é formada por diferentes padrões que, quando combinados, originam o comportamento observado ao longo do tempo.

Figura 3 – Gráfico de Decomposição série temporal



Modificado pelo Autor. Fonte: <https://analisemacro.com.br/\econometria-e-machine-learning/analise-exploratoria-de-series-temporais/>.

3.2.1 Tendências

A tendência indica a direção geral dos dados ao longo do tempo, mostrando se, de maneira ampla, a série está subindo, caindo ou se mantendo estável. Em um contexto financeiro, por exemplo, ela pode representar a valorização constante de um ativo ou de um índice de mercado, causada por mudanças econômicas de longo prazo. Segundo ALBUQUERQUE et al. (2018), uma série com tendência se caracteriza por apresentar, ao longo do tempo, um padrão que pode ser linear ou não linear, além de exibir um comportamento crescente, decrescente ou constante. Esses tipos de séries podem ser expressos como na equação (3), desconsiderando a sazonalidade, onde T_t representa o nível da série no tempo t e é obtido a partir do nível anterior somado à taxa de crescimento τ_{t-1}, ou seja, $T_t = T_{t-1} + \tau_{t-1}$. Já α_t corresponde ao termo de erro, considerado como um ruído branco, ou seja, uma sequência aleatória sem padrão definido.

$$Y_t = T_t + \alpha_t, \quad t = 1, 2, \ldots, n \tag{3}$$

De acordo com Morettin e Toloi (2005) há varios modelos para se estimar T_t. Os mais utilizados consistem em:

- Ajustar uma função do tempo, como um polinômio, uma exponencial ou outra função suave de t;
- Suavizar (ou filtrar) os valores da série ao redor de um ponto, para estimar atendência naquele ponto;
- Suavizar os valores da série através de sucessivos ajustes de retas de mínimos quadrados ponderador lowess (*Locally Weighted Scatterplot Smoothing*).

A abordagem LOWESS (Suavização por Regressão Ponderada Localmente) criada por Cleveland (1979) é a mais apropriada para informações financeiras, uma vez que gerencia bem as variações locais sem estabelecer uma tendência rígida. No âmbito financeiro, a suavização LOWESS é particularmente eficaz para identificar variações locais e minimizar o efeito de pontos divergentes, ao passo que os ajustes polinomiais ou exponenciais são mais apropriados para tendências claramente estabelecidas.

Segundo Franses, Dijk e Opschoor (2014), em diversas séries temporais, a tendência costuma ser a principal causa da variação dos dados, tornando-se essencial para fazer previsões além da amostra. Se essa tendência for mal ajustada no modelo, as previsões tendem a ser errôneas, especialmente quando se prevê para períodos mais

extensos.

3.2.2 Sazonalidades

A sazonalidade é o componente que identifica padrões recorrentes em intervalos específicos e regulares, como trimestres ou meses, onde é refletida na variância de uma série temporal. Segundo Nielsen (2019), a sazonalidade nos dados diz respeito a padrões que se repetem com uma frequência constante ao longo do tempo. Ela pode acontecer em diversas escalas ao mesmo tempo. Por exemplo, o comportamento humano apresenta sazonalidades diárias, como o hábito de almoçar no mesmo horário, semanais (as segundas-feiras se parecem com outras segundas) e anuais, como a diminuição do tráfego nas últimas horas da virada de ano Novo. Há também sazonalidade em sistemas físicos, como o período necessário para a Terra completar sua volta ao redor do Sol. No mercado financeiro, isso pode refletir comportamentos esperados em determinadas épocas do ano, como um aumento nas compras e no consumo durante o final do ano, que pode impactar ações de varejo.

De acordo com Franses, Dijk e Opschoor (2014), ao observar uma série temporal econômica em intervalos mensais ou trimestrais, é comum observar variações regulares entre as diferentes estações do ano. Em outras palavras, essas séries frequentemente exibem um padrão sazonal bem definido. Assim como ocorre com a tendência, cujo significado exato depende do modelo adotado, a sazonalidade não possui uma definição única e precisa.

3.2.3 Ciclos

Os ciclos são caracterizados pelas oscilações de subida e de queda nas séries, de forma suave e repetida, ao longo da componente de tendência. Por exemplo, ciclos relacionados à atividade econômica ou ciclos meteorológicos (BARROS, 2015). São flutuações de longo prazo que não são estritamente periódicas, que podem influenciar o comportamento da série ao longo do tempo.

Morettin e Toloi (2005) mostram uma função de modelo de erro clássico para a análise de séries econômicas, onde f_t que representa uma tendência polinomial determinística é composta da adição ou multiplicação de ambos os tipos de função:

$f_t = \mu_t + S_t$, onde o μ_t é a componente ciclo-tendência incluindo as flutuações cíclicas de longo período, que não podem ser detectadas com os dados disponíveis, enquanto S_t é a componente sazonal ou anual.

3.2.4 Nível

O nível de uma série temporal é essencialmente o valor médio da série em um período específico. Ela representa a tendência central dos dados, fornecendo um ponto de referência para analisar tendências e variações sazonais. Compreender o nível é crucial para fazer previsões e ajustes. Por exemplo, se o nível de uma série estiver aumentando, isso pode indicar uma tendência positiva, enquanto um nível estável não sugere nenhuma mudança significativa ao longo do tempo. Essas informações são vitais para a tomada de decisões nas áreas economia e de finanças. A estimativa do nível pode ser feita através de métodos como ajuste de funções do tempo, suavização ou tomada de diferenças, dependendo da natureza da série.

De acordo com Hyndman e Athanasopoulos (2018), o nível (ℓ_t) é como um ponto de partida para fazer previsões futuras, mostrando o valor que indica onde a série está "no momento" em relação à sua média. Ele é ajustado de tempos em tempos conforme surgem novos dados, para captar o comportamento mais atual da série. O nível é mostrado no modelo de suavização exponencial simples, sem tendência nem sazonalidade (equação 4) sendo y_t é o valor observado no tempo, α é o parâmetro de suavização e l_{t-1} é o nível no tempo anterior.

$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1} \tag{4}$$

3.3 Modelos de séries temporais

Os modelos que descrevem séries temporais são processos estocásticos, ou seja, regidos por leis probabilísticas. Independentemente da classificação adotada, existe uma ampla variedade de modelos para representar o comportamento de uma série específica. A escolha do modelo depende de diversos fatores, como a dinâmica do fenômeno, o conhecimento prévio sobre sua natureza, o objetivo da análise e a disponibilidade de *softwares* adequados (MORETTIN; TOLOI, 2005).

Nos últimos anos, os métodos de predição de séries temporais passaram por

uma evolução significativa, desde técnicas simples de regressão até algoritmos mais sofisticados na estatística e na inteligência artificial (IA). Cada modelo preditivo depende do conhecimento prévio sobre a distribuição dos dados, os quais podem ser classificados em abordagens paramétricas (como Suavização Exponencial e modelos baseados em ARIMA) e não paramétricas (Globais ou Locais) (CHATFIELD; XING, 2019).

3.3.1 ARIMA

O modelo ARIMA (AutoRegressive Integrated Moving Average) criado por Box e Jenkins (1970) no livro publicado Time Series: Forecasting and Control, também chamado de modelo Box-Jenkins, é uma das técnicas de modelo estatístico mais utilizadas para análise e previsão de séries temporais. Esse modelo resulta da combinação de três procedimentos estatísticos (BOX et al., 2015): AutoRegressivo (AR), Integrado (I) e Média Móvel (MA).

$$\left(1-\phi_1 B-\phi_2 B^2-\cdots-\phi_p B^p\right)(1-B)^d Y_t=\left(1+\theta_1 B+\theta_2 B^2+\cdots+\theta_q B^q\right)\varepsilon_t \quad (5)$$

O modelo autorregressivo (AR) baseia-se na intuição de que o passado prevê o futuro e, portanto, postula um processo de série temporal no qual o valor em um ponto no tempo t é uma função dos valores da série em pontos anteriores no tempo (NIELSEN, 2019). O valor presente da série é representado como uma combinação linear de seus valores anteriores. A posição do componente AR (indicada por **p**) no modelo ARIMA (p d q) determina a quantidade de valores recebidos no modelo. Segundo Morettin e Toloi (2005), os modelos autorregressivos são bastante populares em algumas áreas, como em Economia, onde é natural pensar o valor de alguma variável no instante t como função de valores defasados da mesma variável. O componente Integrado (I) refere-se à diferenciação da série temporal para torná-la estacionária. Uma série é estacionária quando sua média, variância e autocorrelação são constantes ao longo do tempo. A ordem de diferenciação (representada por **d**) indica quantas vezes a série foi diferenciada. Se uma ou mais raízes características da equação de diferenças são iguais ou maiores que um (1), a série temporal é considerada não estacionária. Nesse caso, a série precisa ser diferenciada para se tornar estacionária. O número de diferenciações necessárias é representado pelo parâmetro d no modelo ARIMA(ENDERS, 2004). A média móvel

(MA) captura a relação entre uma observação e um erro residual de um modelo de média móvel aplicado a observações atrasadas. É denotado como MA (**q**), sendo q representa o número de erros de previsão atrasados na equação de predição. O componente MA ajuda a suavizar as flutuações de curto prazo nos dados e a destacar tendências ou ciclos de longo prazo. Ao incorporar erros de previsão do passado, o modelo pode ajustar as previsões futuras com base nos erros cometidos nas previsões anteriores.

Morettin (2017) descreve que o modelo ARIMA abrange três categorias de processos: Processos lineares estacionários, que se destacam por suas características estatísticas consistentes ao longo do tempo. Processos lineares não estacionários homogêneos são uma extensão dos processos lineares estacionários, que presumem que o mecanismo que gera a série produz erros autocorrelacionados. Processos de longa memória, que são processos estáveis, têm uma função de autocorrelação hiperbólica (com um decaimento extremamente lento). Para sua análise, será necessária uma diferença fracionária ($0 < d < 0,5$). De acordo com Nielsen (2019), os modelos ARIMA continuam a proporcionar um desempenho comparável ao do estado da arte, particularmente em situações de conjuntos de dados pequenos onde o aprendizado de máquina mais avançado ou modelos de aprendizado profundo não estão em seu auge.

3.3.2 Suavizações exponenciais

O método de suavização surgiu em 1950 com o método de exponencial simples, que leva em conta apenas o nível da série temporal, sem levar em conta elementos de tendência ou sazonalidade. Holt (2004) aperfeiçoou a técnica em 1957 ao implementar a suavização exponencial dupla, possibilitando a representação da tendência. Em 1960, em parceria com Peter Winters, ampliou o método para abranger a sazonalidade, dando origem ao que é amplamente conhecido como suavização exponencial de Holt-Winters, método amplamente utilizado. (CHATFIELD, 1978)

Os modelos de Suavização Exponencial se caracterizam por decompor a série temporal em componentes como tendência, sazonalidade e nível, esses componentes são ponderados por pesos que decrescem exponencialmente à medida que as observações se tornam mais antigas. Em outras palavras, as observações mais recentes recebem maior peso na previsão, enquanto as mais antigas têm um peso relativamente menor. (PARMEZAN; SOUZA; BATISTA, 2019)

Segundo Morettin (2005), existe um grupo importante de métodos de previsão

chamado suavização, que busca lidar com as variações naturais das séries temporais. Essas técnicas partem do princípio de que picos ou quedas muito acentuadas nos dados representam flutuações aleatórias. Ao "suavizar" esses extremos, é possível destacar o comportamento real e os padrões essenciais da série, tornando mais fácil fazer previsões.

Embora esse método seja amplamente utilizada em séries temporais, apresenta algumas desvantagens. Ela pode ser lenta para se ajustar a mudanças rápidas nas tendências, resultando em previsões que não refletem adequadamente as condições atuais (BOX et al., 2015). Além disso, métodos de suavização exponencial, especialmente os mais simples, não capturam bem a sazonalidade ou complexidades de longo prazo, tornando-os menos eficazes para previsões em prazos mais distantes. Outro desafio é a necessidade de ajustar múltiplos hiperparâmetros, o que pode ser um processo desafiador e demorado especialmente em contextos com dados muito voláteis ou não-lineares como no caso do mercado financeiro, os modelos podem falhar em proporcionar em fornecer previsões precisas, prejudicando a tomada de decisão (HYNDMAN; ATHANASOPOULOS, 2018).

3.4 Modelos de séries temporais que usam IA

Esta seção introduz modelos de previsão de séries temporais fundamentados em métodos de Inteligência Artificial (IA), ressaltando estratégias que estão ganhando popularidade por sua habilidade em lidar com padrões complexos e não lineares encontrados nos dados. Estes modelos são particularmente valiosos quando se trata de métodos convencionais, como o ARIMA ou os modelos de suavização.

3.4.1 Redes Neurais

As redes neurais ou redes neurais artificiais (RNAs), são um conjunto de técnicas de aprendizado de máquina inspiradas no funcionamento do cérebro humano. Elas consistem em unidades de processamento interconectadas (neurônios artificiais) que aprendem a representar padrões complexos nos dados. As redes neurais têm suas raízes no trabalho de McCulloch e Pitts (1943), que propuseram um modelo matemático de neurônios artificiais. Nas décadas seguintes, avanços como o *Perceptron*, um algoritmo de aprendizado de máquina que classifica dados de entrada em classes específicas criado por

Rosenblatt (1958) e o algoritmo de *Backpropagation* (*Backward Propagation of Errors*) algoritmo para o treinamento de redes neurais artificiais de Rumelhart, Hinton e Williams (1986) impulsionaram o campo.

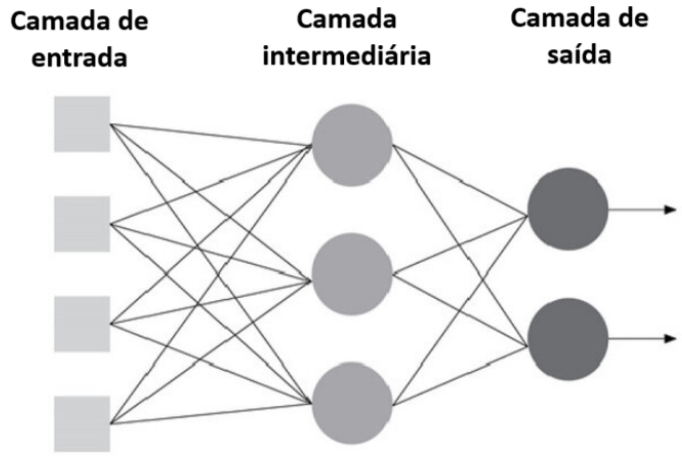
As redes neurais artificiais têm um papel crucial na representação dos processos cognitivos e de tomada de decisões humanas. Essa habilidade de simulação tem se mostrado especialmente útil na análise de dados financeiros. Em várias situações de tomada de decisões financeiras, esses modelos podem não só complementar, mas também superar as estratégias estatísticas tradicionais.(CHATTERJEE; AYADI; BOONE, 2000)

De acordo com Martins, Mette e Macedo (2008), as redes neurais artificiais consistem em neurônios conectados uns aos outros, simulando sinapses estruturadas artificiais, fundamentadas em modelos biológicos. Portanto, as RNAs apresentam características como a capacidade de aprender por meio de treinamento, a capacidade de generalizar e a tolerância a falhas.

Seguno Wong e Selvi (1998), as RNAs demostram-se como uma ferramenta indispensável para a organização, categorização e síntese de informações. Ademais, conseguem reconhecer padrões nos dados de entrada, demandando mínimas suposições e atingindo um elevado grau de acerto nas previsões. Essas qualidades tornam essa tecnologia uma opção promissora para uso no setor financeiro, destacando-se pela sua exatidão, flexibilidade, robustez, eficácia e eficiência na solução de problemas financeiros, tais como identificação de padrões, classificação e previsão de tendências.

As RNs são usadas em uma variedade de tarefas, como previsão de preços de ativos, gestão de risco, detecção de fraudes e mercado de câmbio. O sucesso recente deve-se à combinação de grandes volumes de dados, poder computacional avançado e técnicas de treinamento eficientes.

Figura 4 – Exemplo simplificado de uma rede neural multicamadas



Fonte: Haykin (2001)

Uma RNA, conforme apresenta a Figura 7, pode ser dividida em três níveis fundamentais: de entrada, de ocultação e de saída. A camada de entrada tem a função de captar os sinais de x_i (input signals), os neurônios artificiais que compõem as camadas ocultas ou intermediárias são os responsáveis pelo processamento interno da rede, e a camada de saída é formada pelos neurônios sintéticos encarregados de produzir o sinal y (output). (HAYKIN, 2001).

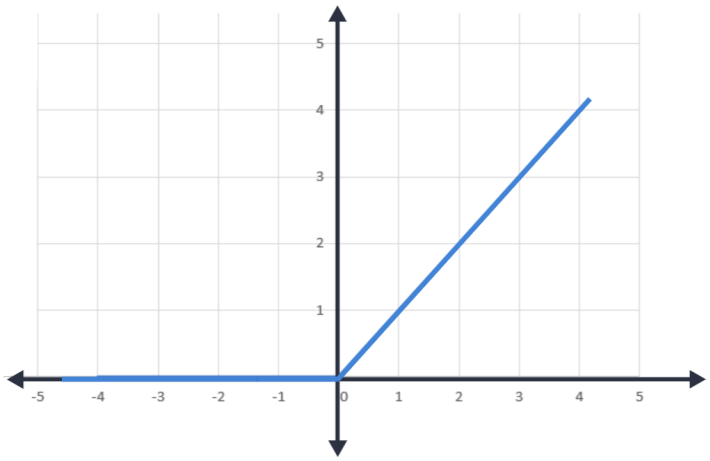
$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{6}$$

$$h = \phi(W^T x + b) \tag{7}$$

Segundo Goodfellow, Bengio e Courville (2016), cada unidade oculta em uma rede neural realiza transformações não lineares nos dados, começando pelo cálculo de uma combinação linear das entradas, seguida da aplicação de uma função de ativação não linear ϕ. A equação 6 ilustra esse prieiro processo, descrevendo o funcionamento básico de uma unidade: primeiro é realizada a soma ponderada das entradas x_i com seus respectivos pesos w_i, somada a um viés b; em seguida, aplica-se a função de ativação f, responsável por introduzir a não linearidade ao modelo. Na equação (7), reforça o mesmo princípio, onde x são as entradas, W os pesos, e b o viés e ϕ é a função de ativação aplicada. Cada elemento da RN efetua uma transformação linear ponderada das entradas, ajustando os pesos durante o treinamento para minimizar o erro. O viés (b) permite melhor adaptação aos dados ao ser somado antes da aplicação da função de ativação, que introduz

não linearidade ao modelo. A função ReLU (Unidade Linear Retificada), representada pela equação (8), é uma das mais utilizadas atualmente.

Figura 5 – Gráfico da ativação da ReLU



Fonte: Próprio Autor

$$\phi(z) = \max(0, z) \tag{8}$$

Redes neurais são modelos estruturados em camadas, cada nível converte os dados numa representação um pouco mais abstrata. O conceito central é que essas camadas têm a capacidade de aprender a retratar características cada vez mais complexas dos dados. As RN de multiplas camadas são denomidas DNNs (*Deep Neural Networks*) e são capazes de aprender representações hierárquicas, independentemente da arquitetura ser *feedforward*, recorrente ou convolucional (GOODFELLOW; BENGIO; COURVILLE, 2016).

A arquitetura das FNNs (*FeedForward Neural Networks*) é composta por camadas sequenciais, uma camada de entrada, uma ou mais camadas escondidas e uma camada de saída. Cada neurônio executa uma função de ativação nos dados que recebe, e as diferenças de peso entre os neurônios são ajustadas por meio de retropropagação para reduzir o erro. No trabalho de Jabin (2014) as FNNs são apresentadas como eficazes na previsão de movimentos de curto prazo no mercado de ações, particularmente em setores com tendências estáveis. Elas possuem uma arquitetura simples e um desempenho satisfatório em cenários de volatilidade reduzida. Contudo, o autor ressalta que há algumas restrições tais como: a sensibilidade a ruídos de dados, a complexidade de identificar padrões complexos, e a necessidade de ajustar minuciosamente os parâmetros. Essas restrições infelizmente, afetam a efetividade em períodos mais extensos ou em mercados mais voláteis.

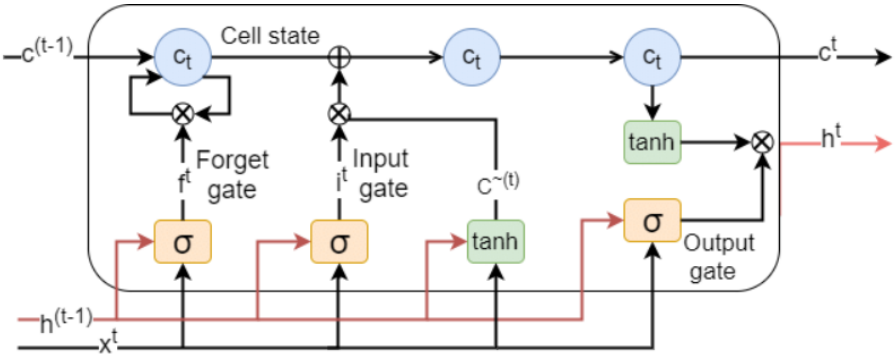
Ao tratar de séries temporais, as redes neurais convencionais do tipo *Feedforward Neural Networks* são intrinsecamente impróprias, uma vez que processam as entradas de maneira autônoma, desconsiderando as dependências temporais. Por outro lado, as redes neurais recorrentes foram concebidas para lidar com sequências, mantendo um estado oculto que atua como uma memória das entradas anteriores (YU et al., 2019).

3.4.2 LTSM

O modelo LSTM (*Long Short-Term Memory*), criado por Hochreiter e Schmidhuber (1997) para resolver um problema crítico das redes neurais recorrentes (RNNs), pois as RNNs convencionais têm dificuldade de aprender dependências de longo prazo em sequências temporais devido ao problema do gradiente desaparecendo quando estes se tornam muito pequenos durante a retropropagação e o problema do gradiente explodindo, quando crescem exponencialmente, tornando a rede instável. Isso fazia com que as RNNs comuns esquecessem rapidamente informações importantes de eventos passados, tornando-as ineficientes para modelar sequências longas, como séries temporais financeiras.

De acordo com Goodfellow, Bengio e Courville (2016), a LSTM soluciona a questão das dependências a longo prazo através de um auto-loop linear, possibilitando a fluidez dos gradientes por um extenso período. Sua principal inovação é o uso de portões que regulam dinamicamente o fluxo de informações. A porta de esquecimento, por exemplo, controla a influência do auto-loop, ajustando seu peso de 0 a 1 por meio de uma função sigmoide, isso permite modificar a escala de tempo de integração conforme a entrada. Cada célula LSTM possui uma recorrência interna, auto-loop e uma recorrência externa, como nas RNNs tradicionais. Ela mantém uma unidade de estado $S_{t,i}$ que preserva informações ao longo do tempo, regulada pela porta de esquecimento. Essa estrutura permite ao modelo decidir o que manter, esquecer ou atualizar, sendo eficaz na identificação de padrões temporais complexos.

Figura 6 – Os componentes fundamentais de uma célula LSTM



Fonte: Jenkins et al. (2018)

A Figura 6 mostra os elementos fundamentais de uma célula LSTM são o portão de esquecimento (*forget gate*), o portão de entrada (*input gate*), o portão de saída (*output gate*) e o estado da célula (*cell state*). No modelo sugerido por Jenkins et al. (2018), a célula LSTM tem um estado de memória encarregado de guardar informações, além de mecanismos que controlam a inserção e exclusão de dados durante o processo de aprendizado. Este estado de memória, simbolizado pelo *cell state*, percorre a célula LSTM e pode ser alterado através das operações básicas $\otimes$ e $\oplus$. Os diversos portões da célula controlam a quantidade e o tipo de informações que podem ser transmitidas.

Os três portões que compõem o LSTM são: O *forget gate* que decide quais informações devem ser descartadas da célula de estado; a *input gate* onde determina quais novas informações serão armazenadas e a *output gate* que decide quais informações serão passadas para a próxima célula. Esses portões empregam funções de ativação sigmoide e tangente hiperbólica (tanh) para controlar a atualização dos estados internos, possibilitando que o modelo armazene ou dispense informações de maneira adaptável, de acordo com a ordem de entrada (AL-SELWI et al., 2024).

Apesar de redes LSTM com uma única camada já se mostrarem eficazes em várias tarefas de modelagem sequencial, a utilização de múltiplas camadas empilhadas tem se destacado como uma estratégia valiosa para a captura de representações mais profundas e hierárquicas. Essa estrutura, que remete ao conceito de profundidade característico das DNNs, possibilita que as camadas inferiores identifiquem padrões locais e temporais de curto prazo, enquanto as camadas superiores extraem abstrações mais amplas e de longo alcance. Essa organização em níveis contribui para um desempenho superior em aplicações desafiadoras como tradução automática e geração de texto (SUTSKEVER; VINYALS; LE, 2014).

As LSTMs provaram ser bastante eficientes em aplicações financeiras,

especialmente quando empregadas em arquiteturas de múltiplas camadas. Essas redes são perfeitas para lidar com essas séries temporais complexas, que frequentemente apresentam ruídos, não linearidades e padrões de tempo sutis. Em modelos mais detalhados, as camadas inferiores tendem a aprender variações locais, como oscilações intradiárias, enquanto as camadas superiores são capazes de combinar essas informações para detectar tendências mais abrangentes, como variações sazonais ou alterações de regime no mercado. Esta estrutura de aprendizado hierárquica é crucial para distinguir sinais pertinentes do ruído existente nos dados. Contudo, a aplicação de LSTMs profundas requer precauções, pois necessitam de grandes quantidades de dados para um treinamento eficaz e são sensíveis à seleção de hiperparâmetros. Portanto, métodos como *dropout* e *batch* normalization são frequentemente empregados para diminuir o risco de *overfitting*, particularmente em contextos financeiros instáveis (OZBAYOGLU; GUDELEK; SEZER, 2020).

Ainda segundo Ozbayoglu, Gudelek e Sezer (2020), em atividades como a previsão de retornos e a identificação de regimes de mercado, LSTMs profundos se sobressaem sobre redes rasas e modelos estatísticos convencionais como o ARIMA, graças à sua habilidade de aprender não linearidades complexas e filtrar interferências do mercado. A efetividade de LSTMs profundas é fortemente influenciada pela qualidade e quantidade dos dados disponíveis. Os mercados financeiros apresentam instabilidade e ocorrências raras como crises, o que demanda métodos como a regularização (dropout, early stopping) e a elevação artificial de dados.

3.4.3 Facebook Prophet

O *Facebook Prophet* é um método de previsão de séries temporais baseado em um modelo aditivo, onde tendências não lineares são combinadas com componentes sazonais (anual, semanal e diário) e efeitos de feriados. Ele é especialmente eficaz para dados com: Forte sazonalidade, múltiplos ciclos históricos, dados ausentes ou mudanças abruptas de tendência, presença de *outliers* (valores atípicos). Desenvolvido por Taylor e Letham (2017) da equipe de ciência de dados do Facebook, que estava experimentando muito mais demanda por previsões de negócios de alta qualidade do que seus analistas eram capazes de fornecer, e disponibilizado como código aberto em 2017, o Prophet se destaca por sua facilidade de uso e robustez em cenários do mundo real. Sua interface flexível permite que tanto iniciantes quanto especialistas criem previsões relativamente precisas

com poucas linhas de código.

O Prophet é eficiente em lidar com dados ausentes e alterações de tendência, e muitas vezes lida com valores incomuns. O Prophet facilita a realização de previsões mais acuradas, que são extremamente ágeis em comparação com outras táticas de previsão de séries temporais. Este modelo demanda muito menos tempo de cálculo em relação a outros, sendo comparável aos modelos estatísticos para produzir previsões em apenas alguns segundos. Ele nos possibilita fazer previsões meteorológicas acuradas a partir de dados imprecisos sem qualquer trabalho manual. (KANINDE et al., 2022)

Em sua essência, o Prophet constitui um modelo de regressão aditiva. Conforme Rafferty (2021), esse modelo representa a soma combinada de diversos componentes (opcionais), a saber:

1. Componente de tendência
    - Curva de crescimento linear ou logístico
2. Componentes sazonais
    - Padrão sazonal anual
    - Variação sazonal semanal
    - Ciclo sazonal diário
    - Sazonalidades personalizáveis (ex.: horária, trimestral)
3. Efeitos externos
    - Impacto de feriados e eventos especiais

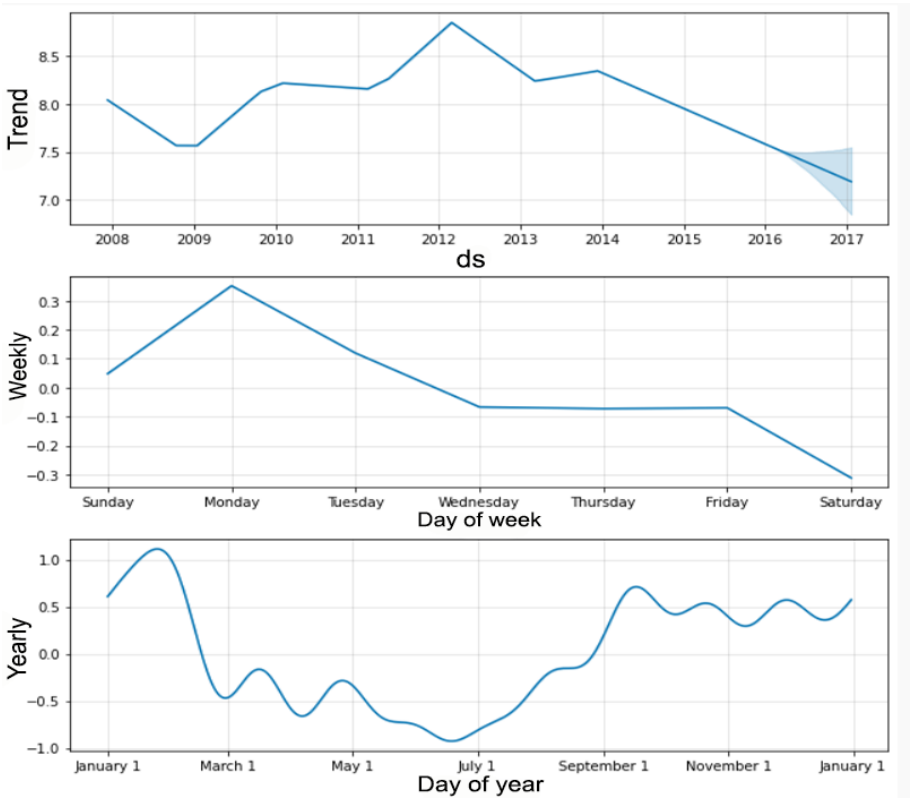
A arquitetura do Prophet foi construida em *Stan*[8], que utiliza modelagem estatística sofisticada usando inferência *Bayesiana* escrito em C++ que realiza a otimização dos modelos. O Stan permite que o Prophet seja flexível, ajustando-se automaticamente às características de sazonalidade, tendências e eventos que afetam os dados. Além disso, a modelagem bayesiana realizada pelo Stan fornece incertezas nas previsões, o que é particularmente útil para muitas aplicações em que a precisão absoluta não é tão importante quanto a compreensão do intervalo de incerteza.

O Facebook Prophet está disponível em duas linguagens populares para ciência de dados, *Python* e *R*[9], sendo que sua implementação mais robusta está na biblioteca Python fbprophet. Essas versões oferecem fácil integração com bibliotecas adicionais

[8]<https://mc-stan.org/>
[9]<https://https://www.r-project.org/>

como Pandas[10], Scikit-learn[11], Matplotlib[12] do python e da ggplot2[13] do R, ampliando as capacidades do Prophet para diversas análises e visualizações.

Figura 7 – Gráficos de previsões do Facebook Prophet



Fonte: https://facebook.github.io/prophet/docs/quick_start.html.

O Prophet é um *software* de código aberto, o que significa que seu código-fonte está totalmente disponível para inspeção, modificação e aprimoramento por qualquer usuário. Essa característica confere grande flexibilidade, permitindo a correção de erros e a adição de novos recursos conforme as necessidades específicas de cada aplicação.

No entanto, diferentemente de muitos *softwares* proprietários, que vêm acompanhados de instaladores independentes e interfaces gráficas intuitivas para facilitar a interação do usuário, o Prophet é acessado exclusivamente por meio das linguagens de programação Python ou R. Embora sua implementação utilize a linguagem Stan, isso não interfere na escolha da linguagem de entrada, já que a interação ocorre por meio de bibliotecas auxiliares em código aberto.

Essa abordagem torna o Prophet altamente personalizável e adaptável a diferentes

[10]<https://pandas.pydata.org>
[11]<https://scikit-learn.org>
[12]<https://matplotlib.org>
[13]<https://ggplot2.tidyverse.org>

cenários de modelagem, mas também pode representar um desafio para usuários menos familiarizados com programação, exigindo um nível maior de conhecimento técnico para sua utilização eficiente.

O Prophet utiliza um modelo de séries temporais decomponível (HARVEY; PETERS, 1990), com três componentes principais: tendência, sazonalidade e feriados. Eles são combinados conforme apresenta a equação 9:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \tag{9}$$

Neste contexto, $g(t)$ representa a função de tendência que representa as alterações não periódicas no valor da série temporal, enquanto $s(t)$ simboliza as alterações periódicas (como a sazonalidade semanal e anual), e $h(t)$ simboliza as consequências dos feriados, que podem ocorrer em horários potencialmente anômalos durante um ou mais dias. O conceito de erro ε_t refere-se a alterações idiossincráticas que o modelo não consegue acomodar, em seguida, faremos a suposição paramétrica de que ε_t é distribuído de forma normal (TAYLOR; LETHAM, 2017).

Segundo Taylor e Letham (2017), o desafio de encarar a previsão como atividade de ajuste de curvas é fundamentalmente distinto dos modelos convencionais de séries temporais que explicitamente levam em conta a estrutura de dependência temporal nos dados. Apesar de implicar renunciar a algumas importantes vantagens inferenciais de modelos geracionais como o ARIMA, essa abordagem proporciona várias vantagens práticas:

- Flexibilidade: O modelo possibilita a incorporação simples de padrões sazonais de vários períodos, permitindo que os analistas façam diversas suposições sobre tendências.
- Ao contrário dos modelos ARIMA, as observações não precisam estar espaçadas uniformemente no tempo, o que elimina a necessidade de interpolar valores ausentes, como ocorre na remoção de *outliers*.
- O ajuste do modelo é incrivelmente ágil, possibilitando aos analistas explorar diversas especificações de maneira interativa, como por exemplo, em aplicações Shiny (CHANG et al., 2015).
- Adicionalmente, os parâmetros do modelo de previsão podem ser facilmente compreendidos e os analistas podem modificá-los para incluir diversas premissas nas previsões. É importante salientar que os especialistas geralmente têm

experiência em modelos de regressão, o que simplifica a ampliação do modelo para incorporar novos elementos quando necessário.

O Prophet utiliza séries de Fourier para reconhecer padrões sazonais, enquanto trata separadamente de feriados e ocasiões especiais. O modelo identifica três tipos de eventos: datas comemorativas anuais fixas, eventos temporários e eventos sem uma regularidade definida. A identificação de diferenças regionais e calendários alternativos, como o lunar, representa uma inovação relevante. No aspecto técnico, o modelo assume a independência entre os eventos, tratando cada um como um fator único, utilizando uma metodologia similar à análise de sazonalidade, porém com parâmetros específicos. Esta estrutura combina modelagem matemática exata utilizando séries de Fourier com adaptabilidade para incorporar componentes culturais e sociais, mantendo ao mesmo tempo a precisão estatística e a habilidade de se adaptar a contextos reais. A tática permite reconhecer tanto padrões constantes quanto mudanças pontuais nos dados.

4 TRABALHOS CORRELATOS

Devido à complexidade crescente dos mercados financeiros e à natureza sequencial dos dados envolvidos, a modelagem de séries temporais tem ganhado destaque na ciência de dados aplicada à economia. Nota-se que várias estratégias têm sido testadas para a previsão de preços e tendências, com as redes neurais, particularmente as LSTMs. Ademais, modelos estatísticos estabelecidos, como o ARIMA, persistem em grande uso devido à sua fácil compreensão e desempenho constante em contextos estáveis. Simultaneamente, instrumentos modernos como o FP (*Facebook Prophet*) têm se destacado pela sua combinação de simplicidade de uso e desempenho satisfatório em séries temporais com sazonalidades e tendências evidentes. Hoje em dia, a maior parte dos estudos focados na previsão de mercados mescla diversos métodos e configurações, modificando hiperparâmetros, experimentando diversas janelas de tempo e utilizando validações sólidas, visando a criação de sistemas preditivos mais seguros e eficientes em contextos voláteis. Portanto, este estudo se enquadra neste cenário, com o objetivo de investigar e contrastar os principais modelos de previsão em diversos cenários do mercado financeiro, com o intuito de fundamentar decisões e aprimorar estratégias de investimento.

4.1 Avaliação de modelos de previsão dos valores das ações no mercado financeiro usando aprendizado de máquina

Machado e Corrêa (2022) conduziram uma pesquisa focada na previsão dos preços das ações no mercado financeiro do Brasil, utilizando técnicas de aprendizado de máquina. O propósito deste estudo foi criar, treinar e analisar o desempenho dos modelos de previsão de séries temporais, ARIMA, LSTM e *Facebook Prophet*, utilizados para prever o desempenho de ativos categorizados como *Blue Chips* (PETR4, ABEV3, CMIG4, ITUB4 e VALE3) e *Small Caps* (TOTS3, ENEV3, MOVI3, ARZZ3 e CAML3), negociados na bolsa brasileira B3. Para tal, usaram informações de séries históricas de preços diários no intervalo de 2017 a 2022, obtidas da plataforma *Yahoo Finance*.

No pré-processamento, foram aplicadas técnicas de interpolação linear para imputação de dados ausentes, normalização de dados via *StandardScaler* para o LSTM e a divisão dos dados em conjuntos de treinamento e validação na proporção de 80/20. Todo o desenvolvimento foi realizado utilizando a linguagem de programação Python e a ferramenta *Google Colab*. Para validar a acurácia das previsões, foram utilizados como

indicadores o Erro Absoluto Médio (MAE), o Erro Quadrático Médio (RMSE) e o Erro Percentual Absoluto Médio (MAPE).

A pesquisa analisou o desempenho de modelos estatísticos convencionais e modelos de aprendizado profundo, ressaltando que, contrariando as expectativas comumente presentes na literatura, o ARIMA alcançou os melhores resultados em termos de acurácia preditiva tanto para as ações Blue Chips quanto das Small Caps, superando até mesmo o LSTM, um modelo comumente recomendado na literatura para séries financeiras de comportamento não linear. Ademais, a pesquisa destacou o desempenho insatisfatório do Prophet na maioria das ações analisadas, sugerindo que, no cenário do mercado brasileiro, modelos estatísticos tradicionais bem parametrizados, como o ARIMA, podem superar métodos de aprendizado profundo, principalmente em séries de menor volatilidade.

Além da análise qualitativa, os autores também apresentaram uma tabela (Figura 8) com os valores numéricos das métricas de avaliação, aplicadas aos diferentes modelos e ativos. Os resultados mostraram que, de modo geral, o modelo ARIMA foi o que apresentou melhor desempenho preditivo para a maioria dos ativos avaliados, com menores valores de erro em relação aos demais. O LSTM também demonstrou bom desempenho, embora com maior variação entre os ativos. Já o Facebook Prophet obteve os piores resultados na maioria dos casos, apresentando valores de erro significativamente mais elevados.

Figura 8 – Métricas de erro

BLUE CHIPS				
Ativo	**Modelo**	**MAE**	**RSME**	**MAPE**
PETR4	PROPHET	2.641	3.128	8.786
ABEV3	PROPHET	3.439	04.06	22.871
CMIG4	PROPHET	0.92	1.144	8.998
ITUB4	PROPHET	5.642	6.62	22.931
VALE3	PROPHET	56.653	64.322	68.389
PETR4	LSTM	0.72	0.919	2.395
ABEV3	LSTM	0.286	0.376	1.838
CMIG4	LSTM	0.348	0.417	3.296
ITUB4	LSTM	0.498	0.686	1.975
VALE3	LSTM	2.763	3.667	3.232
PETR4	ARIMA	0.546	0.774	1.838
ABEV3	ARIMA	0.201	0.275	1.287
CMIG4	ARIMA	0.146	0.194	1.41
ITUB4	ARIMA	0.367	0.563	1.429
VALE3	ARIMA	1.592	2.16	1.853

Fonte: Machado e Corrêa (2022).

Este trabalho destaca a relevância de uma melhor avaliação na seleção de modelos preditivos, destacando que elementos como a natureza dos dados, a volatilidade dos ativos e as características específicas do mercado local têm um impacto direto no rendimento dos

modelos. O estudo, mostra que métodos tradicionais como o ARIMA, podem ultrapassar métodos mais modernos de aprendizado profundo em certos cenários, contribui de maneira significativa para o progresso das investigações em modelagem financeira.

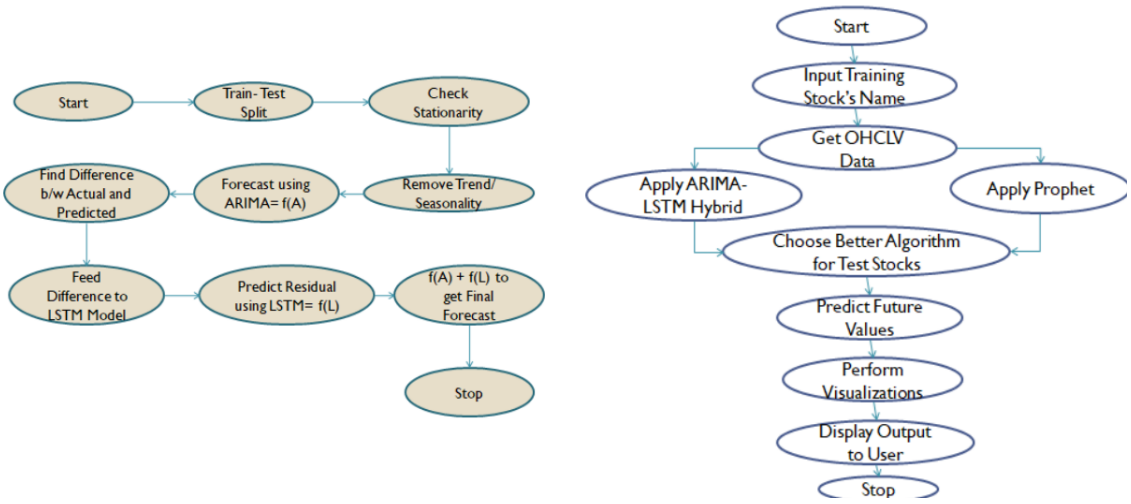
Os aspectos favoráveis do artigo são a sua estruturação clara e a definição precisa dos modelos avaliados, além de abordar ações de diversos perfis (Blue Chips e Small Caps), o que amplia a análise. A pesquisa emprega métricas amplamente reconhecidas na literatura, e a utilização de dados da B3 e de ações do Brasil torna a pesquisa extremamente pertinente para o cenário nacional. Os aspectos negativos incluem a descrição superficial da implementação dos modelos (especialmente o LSTM), sem detalhar a estrutura da rede, parâmetros ou funções de ativação. O artigo não incluiu testes de significância ou uma análise estatística sólida sobre as diferenças entre os modelos. Além disso, o foco do artigo está apenas em dados históricos e previsões de curto prazo, sem abordar cenários de longo prazo ou simulações.

4.2 *An ARIMA-LSTM Hybrid Model for Stock Market Prediction Using Live Data*

O trabalho de Kulshreshtha e Vijayalakshmi (2020), propôs um estudo voltado para a previsão dos preços do S&P 500, um índice com o desempenho das 500 maiores empresas listadas na bolsa dos Estados Unidos, utilizando dados do mercado de ações ao vivo capturados no formato OHLCV (*Open, High, Low, Close, Volume*), utilizando a linguagem Python suas bibliotecas, pmdarima, keras, Theano e o API (Interfaces de Programação de Aplicações) yfinance do *Yahoo* que fornece dados dos últimos 35 anos.

O objetivo principal do trabalho de Kulshreshtha e Vijayalakshmi foi desenvolver e comparar dois modelos para predição de séries temporais financeiras, os autores desenvolveram um modelo híbrido que combina ARIMA com LSTM (ARIMA-LSTM) essa combinação permite que este modelo capture tendências lineares e não lineares dos dados, este estudo propôs inicialmente a aplicação do modelo ARIMA para prever a parte linear da série temporal, cujos os valores não explicados pelo ARIMA foram então utilizados como entrada para o LSTM, responsável por modelar as componentes não lineares. O segundo modelo usado foi o Facebook Prophet, por sua vez, foi aplicado diretamente sobre os dados de fechamento das ações e também foi usado para lidar com dados ausentes e os valores discrepantes.

Figura 9 – Design do modelo híbrido ARIMA-LSTM proposto

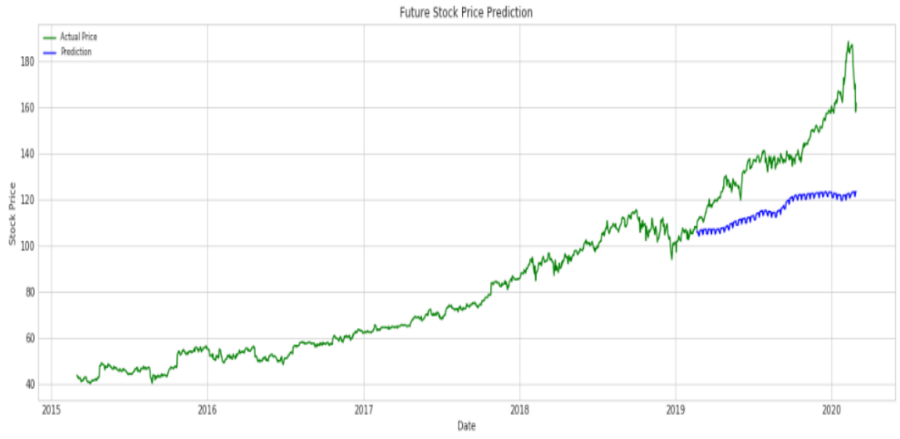


Fonte: Kulshreshtha e Vijayalakshmi (2020).

Para validar os modelos preditivos propostos, os autores utilizaram métricas de avaliação utilizadas em problemas de regressão, incluindo o RMSE, MSE (*Mean Square Error*), MAPE e o coeficiente de determinação (R^2), essas métricas permitiram calcular tanto a precisão quanto o ajuste dos modelos aos dados históricos de preços. Os resultados obtidos pelos autores indicaram uma significativa superioridade do modelo híbrido ARIMA-LSTM em comparação com o *Facebook Prophet*. O modelo híbrido alcançando um RMSE de 1.74, MSE de 3.03, MAPE de 0.009 e um ajuste de 99% (R^2) indicando um ótimo ajuste e uma taxa de erro reduzida, enquanto o FP, apesar de ser reconhecido pela sua robustez a dados ausentes e facilidade de implementação, teve um desempenho inferior nesse contexto, com um RMSE de 27.59 e MSE de 761.33, evidenciando uma maior discrepância entre os valores previstos e os observados.

Este trabalho tem como contribuição demostrar a eficiência da combinação entre métodos estatísticos tradicionais e técnicas de aprendizado profundo para previsão de séries financeiras, propondo um modelo híbrido capaz de lidar simultaneamente com padrões lineares e não lineares, algo essencial em mercados voláteis como o de ações. Além disso, destacou a importância do uso de dados em tempo real e APIs públicas como o yfinance, que viabilizam projetos acessíveis e atualizados para predição no mercado financeiro.

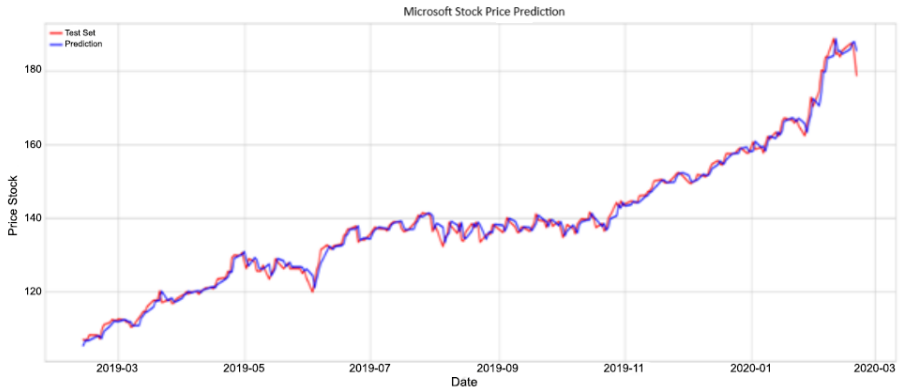
Figura 10 – Previsão usando Prophet



Mean Squared Error: 761.3267000824209
RMSE: 27.592149247248226
Mean Absolute Percentage Error: 0.16645848883781486

Fonte: Kulshreshtha e Vijayalakshmi (2020).

Figura 11 – Previsão usando ARIMA-LSTM



Mean Squared Error: 3.029949161998856
RMSE: 1.7406749156573884
R- squared score: 0.9904288759130871
Mean Absolute Percentage Error: 0.009200831915453274

Fonte: Kulshreshtha e Vijayalakshmi (2020).

O principal destaque do artigo é a sugestão de um modelo híbrido para a previsão de preços no mercado financeiro. O uso da API yfinance e a comparação direta com o modelo Prophet adicionam valor prático e metodológico à pesquisa. Ademais, os autores empregaram métricas robustas. Contudo, o artigo possui algumas restrições, tais como a demostração da aplicação limitada a um único ativo (ações da Microsoft), o que limita a generalização dos resultados, e a falta de informações técnicas mais detalhadas sobre a arquitetura da LSTM, tais como a quantidade de camadas, funções de ativação ou hiperparâmetros empregados.

4.3 Previsão de Preços de Ações Utilizando Inteligência Artificial

O estudo de Nascimento, Santos e Ferreira (2022) sugere uma estratégia prática para antecipar os preços das ações no mercado financeiro do Brasil, utilizando técnicas de Inteligência Artificial empregadas na avaliação de séries temporais. O propósito principal era avaliar e contrastar três modelos de aprendizado de máquina, ARIMA, *Facebook Prophet* e LSTM, em relação à sua habilidade de antecipar tendências de preços de ativos cotados na bolsa de valores B3, proporcionando uma solução que apoie investidores nas decisões de compra e venda.

Os escritores criaram um protótipo de uma aplicação web chamada The Seer, usando o framework Django (versão 4) e programação em Python para tal. A interface foi desenvolvida utilizando HTML, CSS e Bootstrap v5, enquanto os gráficos foram produzidos através da biblioteca Plotly, em linguagem JavaScript, o desenvolvimento web foi feito utilizando a ferramenta *Visual Studio Code* (VSCode). O aplicativo possibilita que o usuário insira o código do ativo que deseja e receba previsões automáticas de valores futuros.

As informações utilizadas na pesquisa foram obtidas através da plataforma *Yahoo Finance*, incluindo dados de fechamento diário de três ativos diferentes: duas ações, PETR4 (Petrobras) e ITUB4 (Itaú Unibanco), além de um ETF, BOVA11 (ETF que replica o índice Ibovespa). A base de dados abrangeu um período de cinco anos, segmentada em conjuntos de treinamento (80%) e de teste (20%). Em particular, o período de treinamento se estendeu de 19/01/2017 a 16/06/2021, enquanto o de teste se estendeu de 17/06/2021 a 19/01/2022. No total, foram registrados 1.243 registros, sendo 995 utilizados para treinamento e 248 para testes.

O modelo ARIMA foi automaticamente configurado com parâmetros padrão, ao passo que o FP foi implementado com configurações tradicionais. Por outro lado, o modelo LSTM teve uma configuração mais minuciosa: treinou com 100 ciclos, tamanho de lote de 100, 35 neurônios na camada de entrada, função de ativação padrão e técnica de *dropout* de 0,2, com o objetivo de prevenir *overfitting*.

A análise do desempenho dos modelos baseou-se em duas métricas: RMSE e MAPE. Os experimentos foram conduzidos com o objetivo de antecipar os valores dos ativos em períodos de 30, 60 e 90 dias. O artigo apresenta na Figura 12 os resultados alcançados, mostrando que o modelo LSTM apresentou os menores erros em todas as escalas de tempo e para todos os ativos examinados. Por exemplo, para o ativo PETR4,

o LSTM atingiu MAPE de 1,92% em 30 dias, 2,47% em 60 dias e 2,75% em 90 dias, com RMSEs correspondentes de 0,54, 0,69 e 0,77. Isso superou significativamente os modelos ARIMA e *Facebook Prophet*, principalmente em previsões de médio e longo prazo. Por outro lado, os modelos ARIMA e FP apresentaram desempenho adequado apenas para períodos curtos (até 30 dias), com uma queda na performance à medida que o período de previsão se estendia. O *Facebook Prophet* teve os piores desempenhos, especialmente nos primeiros 30 dias, não se ajustando corretamente à estrutura não estacionária das séries temporais empregadas. O desenvolvimento da aplicação The Seer, além dos excelentes resultados quantitativos, torna o trabalho ainda mais significativo ao incorporar um modelo prático e de fácil utilização, possibilitando que usuários comuns, como investidores em início de carreira, possam consultar previsões sem a exigência de conhecimento técnico.

Figura 12 – Valores obtidos durante a validação dos modelos

Modelo		MAPE (%)			RMSE (R$)		
Dias		30	60	90	30	60	90
PETR4	ARIMA	8,98	14,32	12,59	2,53	3,21	2,89
	PROPHET	9,59	12,52	9,35	6,93	8,85	6,10
	LSTM	1,92	2,47	2,75	0,54	0,69	0,77
ITUB4	ARIMA	12,30	12,33	12,26	3,49	3,42	3,18
	PROPHET	13,59	13,68	10,26	3,95	3,87	3,23
	LSTM	1,76	1,69	2,47	0,68	0,65	0,98
BOVA11	ARIMA	3,37	3,73	2,97	3,86	5,01	4,28
	PROPHET	7,60	8,07	5,67	9,43	10,02	8,21
	LSTM	1,85	1,96	2,93	2,52	2,80	4,07

Fonte: Nascimento, Santos e Ferreira (2022).

Os destaques do artigo incluem, a utilização de dados reais do mercado brasileiro, a apresentação nítida dos resultados numéricos (RMSE e MAPE) e a criação de uma ferramenta web eficaz. Contudo, o estudo apresenta algumas restrições, apesar do LSTM ter sido aprimorado com mais atenção, não há informações completas sobre a estrutura da rede, tais como a quantidade de camadas ocultas, funções de ativação intermediárias e o algoritmo de otimização. Não existem análises estatísticas sobre a relevância das variações entre os modelos, e a pesquisa se limita a apenas três ativos. No entanto, o artigo atinge com excelência sua meta principal, evidenciando que modelos baseados em redes neurais, particularmente o LSTM, têm excelente performance na previsão de preços de ações, superando opções estatísticas em séries temporais não estacionárias. O estudo oferece uma importante contribuição prática e teórica para a área de finanças quantitativas e a utilização de Inteligência Artificial no mercado de capitais.

4.4 *Stock Price Prediction using Facebook Prophet*

A pesquisa conduzida por Kaninde et al. (2022) tem como objetivo principal utilizar o modelo *Facebook Prophet* para antecipar os preços futuros de ações, baseando-se em séries temporais passadas. O texto inicia reconhecendo que o mercado de ações é extremamente volátil e afetado por uma variedade de fatores econômicos, políticos e comportamentais. Além disso, métodos convencionais, como ARIMA e LSTM, têm limitações ao tratar de dados ruidosos e tendências sazonais complexas.

Os dados históricos foram coletados do *Yahoo Finance*, abrangendo 20 anos de preços do *HDFC Bank* (para treinamento) e da *Apple Inc*. (para comparação de métricas). Os dados empregados seguem o formato OHLCV, sendo o preço de fechamento *(Close)* o parâmetro selecionado para a previsão. Os autores justificam esse parâmetro como o mais importante para a tomada de decisões por investidores e instituições.

A biblioteca *Facebook Prophet* foi empregada para a implementação do modelo, adicionalmente, o aplicativo interativo foi criado com o uso do *framework Streamlit*, que possibilita a criação de interfaces web simples e ágeis em Python. Os autores ressaltam que o *Facebook Prophet* apresenta benefícios significativos, tais como a habilidade de identificar tendências sazonais anuais, mensais e diárias, resistência à ausência de valores, facilidade de ajuste e tempo computacional reduzido para a criação das previsões. Ao longo da implementação, os dados foram reorganizados de acordo com as demandas do FP, sendo a coluna de datas alterada para ds e a variável-alvo (*Close*) alterada para y. Então, o modelo foi aprimorado com base na história e empregado para antecipar o comportamento da série nos próximos cinco anos. Os resultados foram apresentados através de gráficos comparativos entre os valores reais e as previsões.

A avaliação quantitativa dos resultados foi realizada através da métrica RMSE e do tempo de execução. Conforme a Figura 13, na comparação entre o *Facebook Prophet* e outros modelos (ARIMA, LSTM e Fast RNN), o FP exibiu um RMSE de 0,9355 e um tempo de execução de 0,66 segundos. Embora a rapidez seja o seu principal benefício, o modelo apresentou um desempenho inferior em termos de precisão quando comparado ao RNN rápido (RMSE: 0,2024) e ao LSTM (RMSE: 0,2287). Por outro lado, o ARIMA exibiu um rendimento intermediário, com um RMSE de 0,7961.

Figura 13 – Valores RMSE e do cálculo de tempo

Model Name	RMSE	Time (s)
ARIMA	0.796109	1.63
LSTM	0.228731	13.28157353
FB PROPHET	**0.935556**	**0.659962893**
FAST RNN	0.202456	3.337492943

Fonte: Kaninde et al. (2022).

Dentre as vantagens da pesquisa, destaca-se a utilização de um instrumento de fácil acesso e implementação, como o *Facebook Prophet*, que possibilita a manipulação de séries temporais de maneira intuitiva e ágil. A recolha de uma vasta base de dados (20 anos) e o desenvolvimento de uma aplicação web utilizando Streamlit realçam a natureza prática e aplicável do projeto, que pode ser empregado tanto por investigadores quanto por investidores em análises exploratórias. Contudo, o estudo apresenta algumas restrições significativas, o modelo não incorpora variáveis exógenas (ex.: notícias ou dados macroeconômicos), limitando sua capacidade de capturar fatores externos que impactam o mercado e sua capacidade. Finalmente, mesmo com essas restrições, o artigo mostra que o *Facebook Prophet* é uma opção válida para a previsão de séries temporais financeiras, particularmente quando se procura simplicidade, resistência a dados ausentes e previsões ágeis. A pesquisa oferece um exemplo concreto de uso e enfatiza a eficácia da ferramenta em cenários de mercado de capitais.

4.5 *Stock Price Prediction Using Facebook Prophet and Arima Models*

O trabalho realizado por Garlapati et al. (2021), sugere uma comparação entre dois modelos frequentemente empregados na previsão de séries temporais financeiras: ARIMA e *Facebook Prophet*. A principal meta foi antecipar os preços das ações futuras com base em informações históricas, analisando a efetividade de cada modelo através de testes estatísticos e métricas de erro.

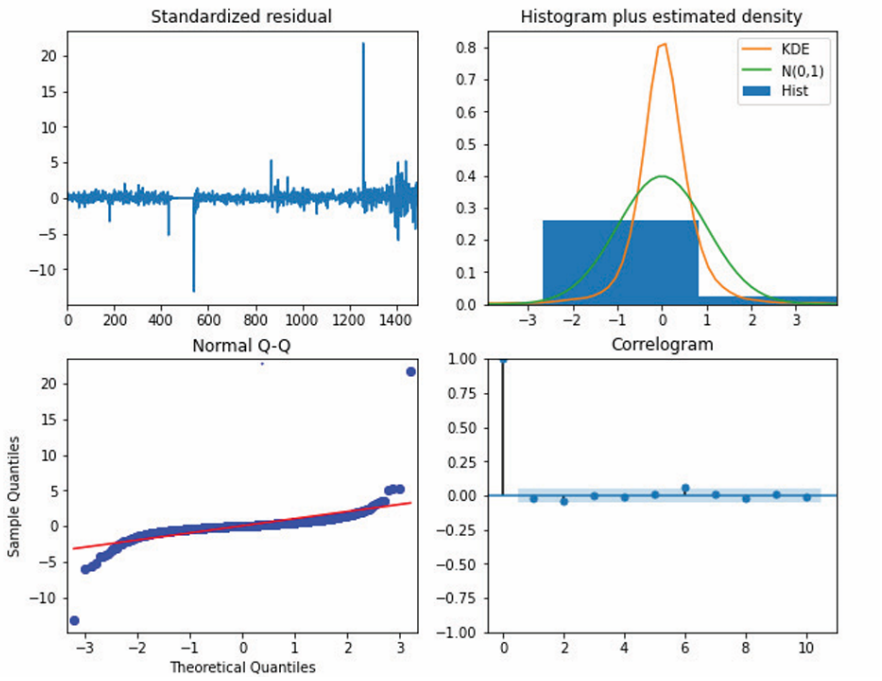
Os dados analisados abrangem um intervalo de oito anos (de 2012 a 2020) e foram estruturados de acordo com o formato convencional OHLCV (Aberto, Alto, Baixo, Fechado, Volume). A pesquisa emprega o preço de fechamento (*Close*) como variável

de referência, visto como mais representativo para decisões de investimento. A base de dados possui 1531 entradas, cada uma contendo seis colunas que representam as variáveis de preço e volume diárias.

Antes da aplicação dos modelos, o artigo dedica atenção significativa ao pré-processamento dos dados. Foram aplicadas técnicas de preenchimento de valores ausentes utilizando a média, normalização da coluna volume e análise exploratória de padrões com o auxílio de gráficos interativos gerados por meio da biblioteca Plotly Express. Além disso, os autores utilizaram mapas de calor (heatmaps) para investigar a correlação entre as variáveis envolvidas. Para verificar a estacionariedade da série temporal, foram aplicados os testes estatísticos KPSS (*Kwiatkowski–Phillips–Schmidt–Shin*) e ADF (*Augmented Dickey-Fuller*), que apontaram que os dados eram não estacionários, reforçando a necessidade de diferenciação e, consequentemente, a adoção do modelo ARIMA. Todo o desenvolvimento do projeto foi realizado utilizando a linguagem de programação Python.

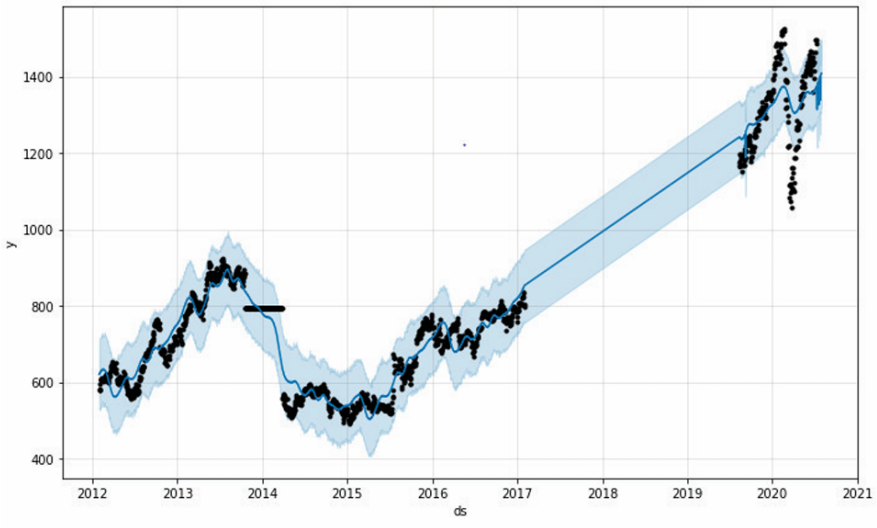
Diversas estratégias de médias móveis foram experimentadas, incluindo média simples, ponderada, exponencial e suavização exponencial, como método de análise inicial e suporte à modelagem de séries temporais. Os autores utilizaram a métrica RMSE (*Root Mean Squared Error*) como critério de avaliação para comparar essas técnicas. Eles descobriram que a suavização exponencial apresentou o menor erro (RMSE = 17,66), evidenciando uma melhor adaptação ao comportamento histórico da série histórica. Na modelagem estatística com ARIMA, optou-se pelo modelo ARIMA(2,1,2)(1,0,0), baseado nos critérios de seleção automatizados através dos indicadores AIC (*Akaike Information Criterion*) e BIC (*Bayesian Information Criterion*). Os valores de AIC foram 13207,962 e BIC foram 13239,961, respectivamente. A análise dos resíduos foi feita utilizando gráficos de autocorrelação, que possibilitaram confirmar a conformidade do modelo ajustado. O FP foi implementado na mesma base, com o objetivo de antecipar os valores futuros de encerramento da ação. As previsões foram feitas com uma margem de confiança de 95%, enfatizando os valores previstos ($\hat{y}$) e as bandas de cima e de baixo. O *Facebook Prophet* também foi apto a reconhecer pontos de inflexão (*changepoints*), que representam alterações abruptas na tendência da série, o que auxilia na solidez das previsões em dados não estacionários.

Figura 14 – Diagnoticos de saída do ARIMA



Fonte: Garlapati et al. (2021).

Figura 15 – Predições Futuras FB Prophet



Fonte: Garlapati et al. (2021).

Os autores usaram três métricas quantitativas para comparar o desempenho dos modelos: além do RMSE, também foram calculados o MAPE (Erro Absoluto Percentual Médio) e o AME (Erro Absoluto Médio), possibilitando uma avaliação mais completa da acurácia das previsões. Os achados indicaram que, mesmo com ambos os modelos

exibindo desempenho adequado, o ARIMA apresentou índices de precisão superiores, com valores de erro absoluto e percentual inferiores ao FP. Em contrapartida, o *Facebook Prophet* sobressaiu-se pela simplicidade de uso, tempo de execução reduzido e pela clara visualização das tendências e ciclos sazonais, o que pode ser benéfico em aplicações com menor exigência técnica e maior demanda por interpretação visual.

Os destaques do estudo incluem a comparação direta entre dois modelos de séries temporais estabelecidos, a utilização de uma vasta base de dados com oito anos de histórico, e a implementação de métodos de visualização contemporâneos, como gráficos interativos. Além disso, a notoriedade deste trabalho é percebida devido a execução meticulosa de testes de estacionariedade, pela descrição minuciosa dos parâmetros configurados no ARIMA, e pela utilização de diversas métricas de avaliação. Contudo, a pesquisa possui algumas restrições significativas, tais como a falta de identificação do ativo financeiro examinado, o que prejudica a replicabilidade, e a ausência de especificação das bibliotecas Python empregadas. Ademais, apesar da aplicação funcional do *Facebook Prophet*, sua análise quantitativa poderia ter sido mais detalhada, e a comparação entre os modelos não foi respaldada por testes estatísticos formais que confirmassem a relevância das diferenças identificadas.

4.6 *Stock values predictions using deep learning based hybrid models*

A pesquisa realizada por Yadav, Yadav e Saini (2022) propõe uma metodologia inovadora para a previsão de preços de ações por meio de modelos híbridos de aprendizagem profunda. O propósito principal do estudo foi criar modelos que pudessem fazer previsões em tempo real, combinando um tempo de execução reduzido com alta acurácia preditiva, particularmente em dados financeiros de alta frequência e volatilidade elevada. Os autores sugerem dois modelos: o primeiro é inteiramente fundamentado na arquitetura FastRNN, enquanto o segundo é um híbrido que combina FastRNN + CNN + Bi-LSTM, analisando as vantagens de cada técnica de forma individual.

A plataforma *Yahoo Finance* forneceu os dados utilizados no estudo, que incluem séries temporais com intervalos de um minuto de quatro empresas cotadas na Bolsa de Valores de Nova York (NYSE): Apple Inc., Facebook Inc., Nike Inc. e Uber Inc. Cada série incluía 430 registros de treinamento e 70 de teste, abrangendo um período de negociação de 1 a 3 dias. A ênfase recaiu na previsão do preço final (*Close*), visando antecipar os valores dos próximos 50 minutos com base em dados históricos

breves (7h10min de dados diários). O Google Colab foi o ambiente escolhido para o desenvolvimento, e as seguintes bibliotecas em Python foram empregadas: Keras, com o backend TensorFlow, para a criação e treinamento de redes neurais profundas, NumPy, Pandas e Matplotlib para a criação de representações gráficas. Os códigos estão disponíveis em um repositório aberto do GitHub.

Os autores empregaram a métrica RMSE para avaliar o rendimento dos modelos, juntamente com o tempo de execução em segundos. Os resultados foram comparados com nove modelos de referência, incluindo ARIMA, LSTM, *Facebook Prophet* e outros modelos híbridos de aprendizagem profunda como CNN-LSTM e BiLSTM. As tabelas de resultados Figura 16 mostram que, o modelo proposto FastRNN apresentou o RMSE mais baixo para as ações da Uber (0,0206) e Nike (0,0377), o modelo FastRNN + CNN + Bi-LSTM apresentou um desempenho similar ao primeiro, com uma ligeira melhoria no RMSE, porém com um aumento no tempo de execução; O *Facebook Prophet* foi o mais rápido com tempo inferior a 0,7s, mas com erro significativamente maior RMSE entre 0,55 e 1,59, o modelo ARIMA, embora clássico, teve desempenho inferior aos modelos baseados em redes neurais profundas.

Figura 16 – Resultados dos nove modelos comparativos.

Apple Inc.

Model name	RMSE	Time (in s)
ARIMA [41]	0.796109	1.63
BiLSTM_Attention_CNN_BiLSTM [42]	0.234644	25.72292113
CNN_LSTM_Attention_LSTM [43]	0.214821	16.00164294
FBProphet [44]	0.935556	**0.659962893**
LSTM [45]	0.228731	13.28157353
LSTM_Attention_CNN_BiLSTM [42]	0.263613	19.96186757
LSTM_Attention_CNN_LSTM [45]	0.27994	17.32732081
LSTM_Attention_LSTM [45]	0.299334	19.28274226
LSTM_CNN_BiLSTM [46]	0.23489	16.76800251
FastRNN (proposed)	**0.202456**	**3.337492943**
FASTRNN_CNN_BiLSTM (proposed)	**0.205647**	**13.49208355**

Facebook Inc.

Model name	RMSE	Time (in s)
ARIMA [41]	0.86664567	**1.60001**
BiLSTM_Attention_CNN_BiLSTM [42]	0.26834072	26.56901288
CNN_LSTM_Attention_LSTM [43]	0.15718991	16.61885238
FBProphet [44]	1.59050836	**0.405165672**
LSTM [45]	0.18714926	14.00746107
LSTM_Attention_CNN_BiLSTM [42]	0.22215487	20.65539312
LSTM_Attention_CNN_LSTM [45]	0.24017975	17.97418046
LSTM_Attention_LSTM [45]	0.29138532	19.86988878
LSTM_CNN_BiLSTM [46]	0.19898068	17.52065849
FastRNN (proposed)	**0.14932692**	**14.16171408**
FASTRNN_CNN_BiLSTM (proposed)	**0.15137204**	**3.447671652**

Nike inc.		
Model name	**RMSE**	**Time (in s)**
ARIMA [41]	0.465822234	**1.63001**
BiLSTM_Attention_CNN_BiLSTM [42]	0.106373725	26.56245756
CNN_LSTM_Attention_LSTM [43]	0.051930262	17.05163288
FBProphet [44]	0.550859082	**0.509964705**
LSTM [45]	0.052901	14.01486564
LSTM_Attention_CNN_BiLSTM [42]	0.062675555	20.4820962
LSTM_Attention_CNN_LSTM [45]	0.061914832	17.52905965
LSTM_Attention_LSTM [45]	0.102347907	19.73444676
LSTM_CNN_BiLSTM [46]	0.047966405	17.31208062
FastRNN (proposed)	**0.037727882**	**14.34483051**
FASTRNN_CNN_BiLSTM (proposed)	**0.039458341**	**3.807400703**

Fonte: Yadav, Yadav e Saini (2022).

Dentre as vantagens do artigo, ressalta-se a sugestão de uma metodologia sólida, contemporânea e bem organizada para a previsão de séries temporais financeiras. A arquitetura híbrida composta por FastRNN, CNN e Bi-LSTM demonstra ser eficiente ao combinar acurácia preditiva com um desempenho computacional eficiente. Os escritores também empregaram um ambiente de execução realístico e acessível, o que favorece a replicabilidade do experimento. Adicionalmente, o estudo faz uma comparação completa com vários modelos clássicos e avançados, destacando a superioridade quantitativa da solução sugerida. Um ponto importante é a divulgação pública do código, o que simplifica a confirmação dos resultados e a replicação da metodologia por outros estudiosos. Contudo, o estudo possui algumas restrições. Os dados empregados se referem a períodos de tempo extremamente breves, o que pode limitar a aplicação dos resultados para cenários de médio e longo prazo. O modelo também ignora variáveis externas, tais como indicadores econômicos, acontecimentos de mercado ou dados fundamentais, que poderiam aprimorar as previsões. Ademais, embora o artigo destaque a rapidez na execução, ele não aborda como o modelo se comportaria em cenários reais de produção ou sua integração com sistemas automatizados de tomada de decisões.

Tabela 1 – Análise comparativa entre os trabalhos correlatos

Autores	**Modelos**	**Métricas**	**Ferramentas**
Machado e Corrêa (2022)	ARIMA, LSTM, Prophet	MAE, RMSE, MAPE	Python, Yfinance
Kulshreshtha e Vijayalakshmi (2020)	Híbrido ARIMA-LSTM e Prophet	RMSE, MSE, MAPE, R	Python (pmdarima, Keras, Theano), Yfinance
Nascimento, Santos e Ferreira (2022)	ARIMA, LSTM, Prophet	RMSE, MAPE	Python, Django, HTML, CSS, Bootstrap, JS Plotly
Kaninde et al. (2022)	ARIMA, LSTM, Prophet, Fast RNN	RMSE, tempo de execução	Python, Streamlit
Garlapati et al. (2021)	ARIMA, FB Prophet	RMSE, MAPE, MAE	Python, Plotly Express
Yadav, Yadav e Saini (2022)	Fast RNN, Híbrido FastRNN-CNN-BiLSTM	RMSE, tempo de execução	Python, Keras, Matplotlib, Yfinance
Trabalho proposto	ARIMA, LSTM, Prophet	RMSE, MAPE	Python, Yfinance

Fonte: Próprio autor.

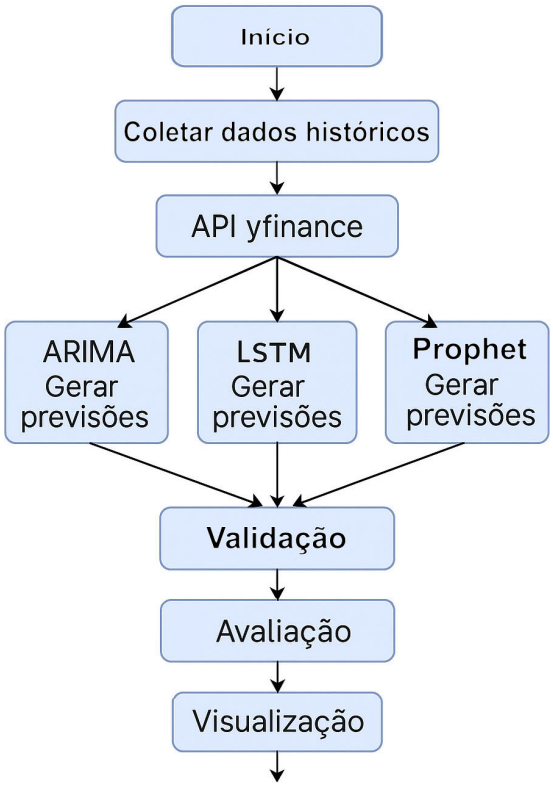
A Tabela 1 faz uma comparação entre os trabalhos discutidos nesta seção e o estudo proposto, destacando os modelos de previsão utilizados, as métricas adotadas para avaliação de desempenho e as ferramentas e linguagens empregadas em cada pesquisa. Essa síntese permite visualizar de forma clara as semelhanças e diferenças metodológicas entre os estudos analisados, além de contextualizar a contribuição da abordagem que será desenvolvida neste trabalho.

5 DESENVOLVIMENTO E RESULTADOS

Este capítulo apresenta o desenvolvimento da proposta do projeto, bem como os métodos empregados para sua implementação e a análise dos resultados obtidos. A solução desenvolvida fundamenta-se nos conceitos e abordagens discutidos nos Capítulos 3 e 4, os quais tiveram como objetivo sistematizar o conhecimento teórico acerca de modelos matemáticos e computacionais aplicados à previsão no mercado financeiro. Esse embasamento teórico foi essencial para a compreensão das metodologias mais recorrentes na literatura e para a definição das estratégias adotadas, garantindo uma base técnica consistente para o desenvolvimento e a avaliação da proposta.

A Figura 17 ilustra o fluxograma de funcionamento do sistema proposto para previsão de séries temporais aplicadas ao mercado financeiro. O processo inicia-se com a coleta dos dados históricos dos ativos por meio da API do yfinance, que permite acessar informações como preços de ações, ETFs e demais ativos financeiros com frequência diária. Esses dados são então processados no ambiente de desenvolvimento Python, onde serão aplicados os modelos preditivos selecionados: ARIMA, LSTM e Facebook Prophet. Cada modelo é executado de forma independente sobre a mesma base de dados, garantindo igualdade nas condições de comparação.

Figura 17 – Fluxograma do *software*



Fonte: Próprio Autor.

5.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer ao usuário final. A aplicação foi projetada para permitir a análise e previsão de séries temporais financeiras de forma interativa, por meio de uma interface gráfica intuitiva.

Os principais requisitos funcionais do sistema são:

- Permitir o download e a atualização automática dos dados históricos de preços de ativos financeiros;
- Possibilitar a seleção de diferentes ativos financeiros para análise;
- Executar modelos de previsão baseados em séries temporais, incluindo ARIMA/SARIMAX, Prophet e LSTM;
- Gerar previsões para horizontes futuros definidos pelo usuário;
- Exibir os resultados das previsões de forma gráfica, comparando valores reais e previstos;
- Apresentar métricas de desempenho dos modelos, possibilitando a comparação

entre abordagens;

- Exibir um ranking dos modelos de previsão com base nas métricas calculadas;
- Armazenar e consultar dados históricos e resultados no banco de dados local.

A Figura 18 apresenta o diagrama de casos de uso do sistema, ilustrando as interações entre o usuário e as funcionalidades disponibilizadas.

5.2 Requisitos do Sistema

Os requisitos do sistema descrevem os aspectos técnicos necessários para a implementação e execução da aplicação, incluindo linguagem de programação, bibliotecas e infraestrutura de hardware e software.

A aplicação foi desenvolvida em `Python 3.12`, integrando bibliotecas voltadas à análise de séries temporais, aprendizado de máquina, visualização de dados e construção de interfaces gráficas. Os principais pacotes utilizados são:

- **PyQt5**: responsável pela construção da interface gráfica (*GUI*), organização de abas, botões e tabelas, além do gerenciamento de execução assíncrona por meio das classes `QThread` e `QObject`;
- **pandas** e **numpy**: utilizadas para leitura, manipulação e análise dos dados históricos de preços;
- **matplotlib**: empregada na geração dos gráficos de previsão e comparação entre valores reais e previstos;
- **statsmodels**: utilizada na implementação do modelo estatístico SARIMAX;
- **prophet**: biblioteca desenvolvida pela Meta (Facebook) para modelagem temporal baseada em tendências e sazonalidades;
- **tensorflow.keras**: utilizada para construção, treinamento e validação da rede neural LSTM;
- **sqlite3**: módulo nativo do Python para gerenciamento do banco de dados;
- **time**, **datetime**, **os** e **sys**: utilizadas para controle de tempo de execução, manipulação de diretórios e integração com o sistema operacional.

Os requisitos mínimos para a execução da aplicação estão apresentados na Tabela 2.

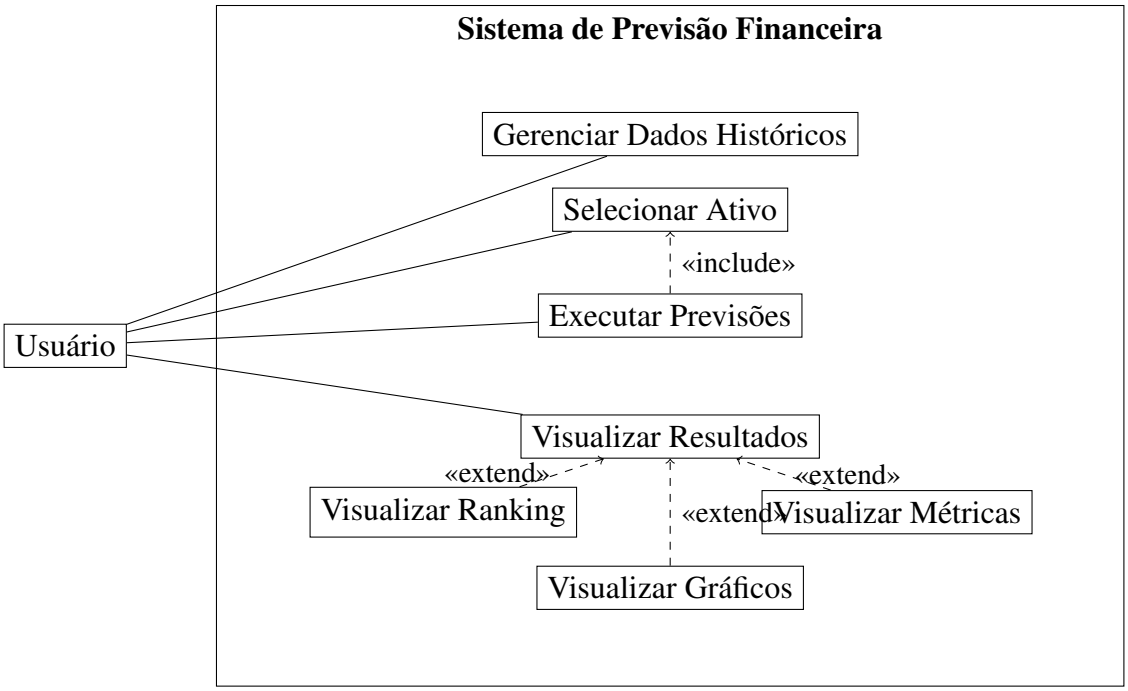
Tabela 2 – Requisitos mínimos para execução da interface final

Componente	**Requisito mínimo**
Sistema Operacional	Windows 10 / Linux Ubuntu 22.04
Versão do Python	Python 3.10 ou superior
Bibliotecas principais	PyQt5, pandas, numpy, matplotlib, statsmodels, prophet, tensorflow, sqlite3
Memória RAM	Mínimo de 8 GB (recomendável para execução da LSTM)
Processador	Dual-core 2.0 GHz ou superior (GPU recomendável para TensorFlow)
Armazenamento	500 MB livres para arquivos `.csv`, gráficos e banco de dados

5.2.1 Diagrama de Casos de Uso

O diagrama de casos de uso a seguir apresenta as principais funcionalidades do sistema sob a perspectiva do usuário, evidenciando o limite do sistema e as interações permitidas. O ator **Usuário** representa o investidor que utiliza a aplicação para realizar análises e previsões financeiras. Os casos de uso foram modelados de acordo com a notação UML, destacando funcionalidades centrais, como o gerenciamento de dados históricos, a seleção de ativos e a execução de previsões, bem como funcionalidades complementares, representadas como extensões, relacionadas à visualização dos resultados.

Figura 18 – Diagrama de Casos de Uso do Sistema de Previsão Financeira



Fonte: Próprio autor.

5.3 Requisitos Não Funcionais

Os requisitos não funcionais especificam critérios de qualidade, desempenho e restrições que afetam o funcionamento global do sistema, mas não estão diretamente relacionados às suas funcionalidades.

- **Desempenho**: o tempo de resposta para o processamento completo de uma previsão (incluindo treinamento e geração de gráficos) não deve ultrapassar 5 minutos para séries temporais de até 10 anos com frequência diária;
- **Usabilidade**: a interface gráfica deve ser intuitiva e de fácil navegação, permitindo o uso por usuários sem conhecimentos avançados em programação;
- **Confiabilidade**: o sistema deve registrar logs de execução e falhas, possibilitando auditoria, rastreabilidade de erros e análise de desempenho;
- **Escalabilidade**: a arquitetura do sistema deve permitir a inclusão de novos modelos de previsão e fontes de dados com impacto mínimo no código-fonte existente;
- **Segurança**: o sistema deve garantir o tratamento adequado dos dados do usuário, prevenindo acessos indevidos e vulnerabilidades como injeção de código;

- **Compatibilidade**: a aplicação deve ser compatível com sistemas operacionais Windows e Linux, utilizando bibliotecas amplamente documentadas e estáveis.

Como recomendação de infraestrutura para melhor desempenho, especialmente na execução de modelos baseados em redes neurais, sugere-se:

- Processador quad-core (mínimo: Intel i5 ou equivalente);
- 8 GB de memória RAM;
- 1 GB de espaço livre em disco;
- GPU com suporte à arquitetura CUDA, recomendada para o treinamento do modelo LSTM.

5.4 Ambiente Computacional

O desenvolvimento do sistema será realizado utilizando a linguagem de programação Python[14], por sua ampla adoção em projetos de ciência de dados. Serão utilizadas bibliotecas específicas para cada etapa do processo. Bibliotecas como TensorFlow e Keras possuem suporte ao uso de GPU (*Graphics Processing Units*), permitindo o processamento paralelo de grandes volumes de dados, o que otimiza o treinamento e a execução das previsões. O PyTorch também é uma alternativa viável para implementação de redes neurais, oferecendo bom desempenho em operações matriciais em GPUs.

- **yfinance**: coleta de dados financeiros históricos diretamente do Yahoo Finance;
- **pandas e numpy**: manipulação e estruturação dos dados em séries temporais;
- **matplotlib e seaborn**: visualização gráfica dos resultados e exploração dos dados;
- **statsmodels e pmdarima**: aplicação e automação do modelo ARIMA;
- **tensorflow e keras**: construção e treinamento do modelo LSTM, podendo utilizar GPU;
- **PyTorch**: alternativa para redes neurais, também compatível com aceleração por GPU;
- **prophet**: implementação do modelo Prophet, desenvolvido pelo Facebook;
- **scikit-learn**: pré-processamento de dados (normalização), cálculo de métricas (RMSE, MAPE) e validação cruzada temporal;

[14]https://www.python.org/

- **SciPy**: otimização de hiperparâmetros, testes estatísticos (Dickey-Fuller) e análise de resíduos;
- **pyqt5**: desenvolvimento de interfaces gráficas (GUI) interativas para visualização e controle do sistema;
- **time**: aferição do tempo de execução de cada modelo.

5.5 Descrição da Série Temporal Utilizada

Para a realização dos testes e comparações entre os modelos preditivos, foram utilizadas séries temporais de ativos financeiros negociados na B3 (Bolsa de Valores do Brasil). A lista de ativos foi inicialmente obtida por meio de um arquivo CSV contendo os códigos dos ativos, o qual foi gerado a partir do levantamento disponível no portal *Dados de Mercado*[15].

A partir dessa lista, os dados históricos de preços foram coletados automaticamente por meio da biblioteca *yfinance*, que permite acesso à base de dados do Yahoo Finance de forma programática. Esse processo garante a atualização periódica das informações e a padronização da coleta dos dados utilizadas nos experimentos.

A seleção dos ativos considerou critérios como representatividade no mercado brasileiro, volume de negociação, disponibilidade histórica e diferentes níveis de volatilidade, de modo a garantir diversidade estatística e robustez na avaliação dos modelos.

Os dados coletados possuem frequência diária e abrangem um período aproximado de dez anos. As informações seguem o padrão amplamente utilizado no mercado financeiro conhecido como formato **OHLCV** (*Open, High, Low, Close, Volume*), composto pelos seguintes atributos:

- **Open** – Preço de abertura do ativo;
- **High** – Maior preço negociado no período;
- **Low** – Menor preço negociado no período;
- **Close** – Preço de fechamento do ativo;
- **Volume** – Quantidade de ativos negociados.

Embora o conjunto de dados inclua todas as variáveis do padrão OHLCV, neste

[15] https://www.dadosdemercado.com.br/acoes

trabalho foi utilizado principalmente o atributo **Close** (preço de fechamento) como variável alvo para o treinamento, validação e geração das previsões pelos modelos SARIMAX, Prophet e LSTM, por se tratar do indicador mais representativo do valor consolidado do ativo no final de cada pregão.

Após a etapa de processamento e geração das previsões, os modelos serão submetidos ao processo de validação. Para isso, o conjunto de dados será dividido em duas partes: 80% dos dados serão utilizados para o treinamento, permitindo o ajuste de parâmetros e aprendizado dos padrões históricos, enquanto os 20% restantes serão reservados para validação, garantindo a análise de desempenho em dados não observados.

As previsões retroativas foram utilizadas exclusivamente para fins de validação dos modelos, por meio de um processo de *backtesting*, no qual uma parcela dos dados históricos é ocultada durante o treinamento e posteriormente utilizada para avaliar a capacidade preditiva dos algoritmos. Esse procedimento permitiu a mensuração objetiva do desempenho dos modelos por meio das métricas RMSE, MAPE e R^2.

Após a validação e seleção dos modelos mais robustos, foi realizada a etapa de previsão futura propriamente dita, na qual os algoritmos foram treinados com a totalidade da série histórica e utilizados para projetar o comportamento futuro dos ativos financeiros. Essas previsões futuras foram utilizadas na fase de análise final e na geração do ranking de ativos, com o objetivo de apoiar a tomada de decisão.

- **RMSE (Root Mean Squared Error)**: mede o erro quadrático médio, sendo sensível a grandes desvios;
- **MAPE (Mean Absolute Percentage Error)**: avalia o erro percentual médio em relação aos valores reais;
- **Tempo de execução**: mensura a eficiência computacional de cada modelo.

Baseando-se nesses critérios, será possível comparar os modelos e identificar aquele que apresentar melhor desempenho preditivo e maior eficiência de execução. Ao final, espera-se obter um sistema funcional que possa servir como ferramenta de apoio para investidores, permitindo a visualização e a projeção de valores futuros de ativos financeiros com base em dados históricos confiáveis e consolidados.

5.6 Banco de Dados

Para o armazenamento e manipulação dos dados históricos de preços de ações da B3, foi utilizado o sistema de gerenciamento de banco de dados relacional SQLite3[15], que permite fácil integração com Python e dispensa a instalação de um servidor externo.

5.6.1 Estrutura do Banco de Dados

O banco foi criado com uma tabela única denominada `precos_historicos`, projetada para armazenar informações essenciais de cada ativo financeiro. As colunas da tabela são descritas na Tabela 3.

Tabela 3 – Estrutura da tabela precos_historicos

Coluna	Tipo	Descrição
ticker	TEXT	Código do ativo (ex: PETR4.SA)
data	DATE	Data do registro do preço
abertura	REAL	Preço de abertura do ativo no dia
maxima	REAL	Maior preço atingido no dia
minima	REAL	Menor preço atingido no dia
fechamento	REAL	Preço de fechamento do dia
fech_ajustado	REAL	Preço ajustado considerando splits e dividendos
volume	INTEGER	Quantidade de ações negociadas no dia

5.6.2 Implementação em Python

A manipulação do banco de dados foi realizada com a biblioteca sqlite3 do Python, permitindo criar tabelas, inserir registros e realizar consultas SQL diretamente. O script inicial garante que a tabela exista, evitando conflitos de estrutura:

```
conn = sqlite3.connect(DB_NAME)
cursor = conn.cursor()
cursor.execute("""
   CREATE TABLE IF NOT EXISTS precos_historicos (
      ticker TEXT,
      data DATE,
```

[15] https://www.sqlite.org/

```
      abertura REAL,
      maxima REAL,
      minima REAL,
      fechamento REAL,
      fech_ajustado REAL,
      volume INTEGER
   );
""")
conn.commit()
conn.close()
```

Para obter os preços históricos, utilizou-se a biblioteca `yfinance`. A inserção no banco é feita evitando duplicações: ao atualizar os dados, o programa identifica a última data disponível e baixa apenas os registros ausentes.

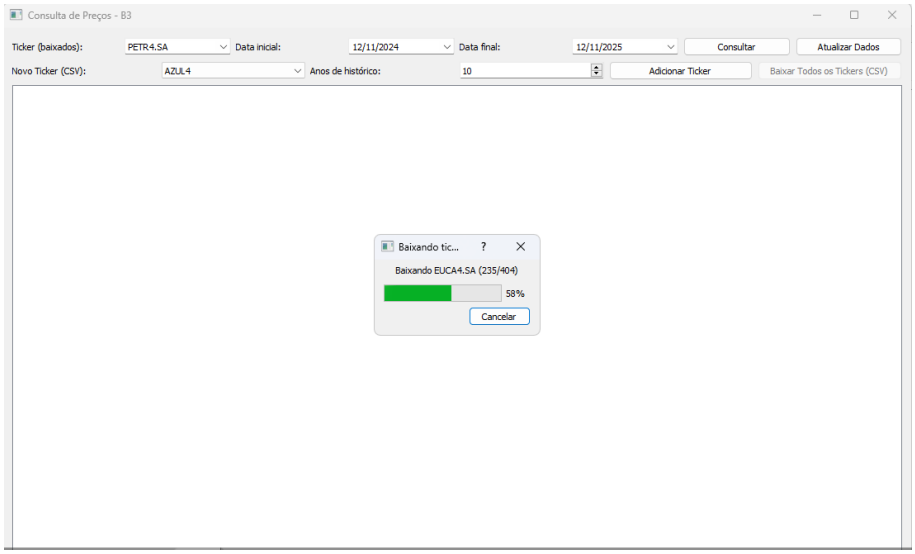
5.6.3 Integração com a Interface Gráfica

A interface foi implementada em **PyQt5**[16] Figura 19, permitindo que o usuário:

- Consulte o histórico de preços por **ticker** e intervalo de datas;
- Adicione novos tickers de um arquivo CSV contendo a lista de ações da B3;
- Atualize os dados existentes sem duplicar registros.

[16]https://www.riverbankcomputing.com/software/pyqt/intro

Figura 19 – Interface do Banco de dados



Fonte: Próprio autor.

O resultado das consultas é exibido em uma tabela dinâmica, construída a partir de um `QTableWidget`, que recebe os dados diretamente do `pandas`.

5.6.4 Integração com CSV de Tickers

Para facilitar a inclusão de novos ativos, o sistema lê um arquivo CSV com a lista de tickers disponíveis na B3, garantindo que apenas tickers válidos sejam adicionados ao banco:

```
def read_tickers_from_csv(path):
    df = pd.read_csv(path, sep=",", encoding="utf-8")
    tickers = df["Ticker"].dropna().astype(str).str.strip()
    .unique().tolist()
    return tickers
```

5.6.5 Consulta e Manipulação de Dados

A consulta ao banco utiliza SQL integrado ao Python, com filtros por ticker e intervalo de datas:

```
query = """
    SELECT ticker, data, abertura, maxima, minima, fechamento,
```

```
    fech_ajustado, volume
    FROM precos_historicos
    WHERE ticker = ? AND data BETWEEN ? AND ?
    ORDER BY data ASC;
"""
df = pd.read_sql_query(query, conn, params=(ticker, data_inicio,
data_fim))
```

Os resultados são convertidos para tabelas exibíveis na interface PyQt5, garantindo visualização clara e organizada dos preços históricos.

5.6.6 Considerações sobre desempenho e manutenção

O uso do SQLite3 é adequado para esta aplicação devido à sua simplicidade, portabilidade e facilidade de backup. A implementação garante que novas inserções não causem duplicações e que os dados possam ser atualizados de forma incremental, baixando apenas os registros ausentes.

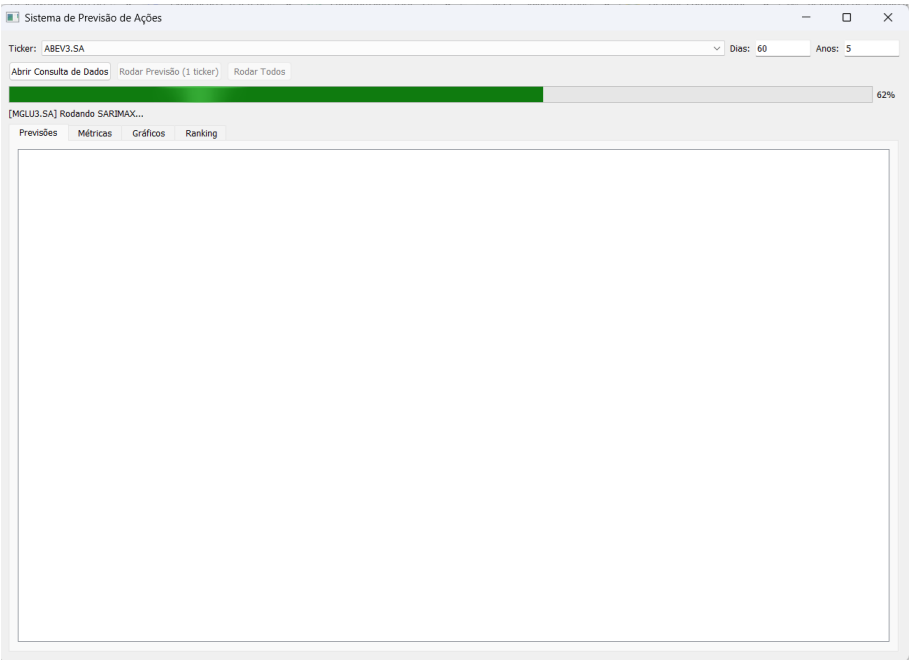
Além disso, o armazenamento local dos dados permite que o usuário trabalhe offline, sem necessidade de conexão contínua à internet, uma vez que todo o histórico de preços já está armazenado no banco de dados. Essa característica é especialmente útil para análises rápidas, testes e apresentações em ambientes sem acesso à internet.

5.7 Interface final

A interface final do sistema (Figura 20) foi desenvolvida em `Python`, utilizando o framework `PyQt5` para a criação da interface gráfica e integração com os módulos de previsão implementados anteriormente (SARIMAX, Prophet e LSTM). O objetivo principal dessa interface é proporcionar ao usuário uma forma unificada, intuitiva e automatizada de executar previsões de preços de ações, visualizar métricas de desempenho e analisar os gráficos comparativos gerados pelos três modelos.

Além disso, a interface inclui uma aba dedicada ao *ranking* dos ativos, que consolida automaticamente as métricas e previsões geradas pelo sistema, permitindo ao usuário interpretar os resultados de forma prática e apoiar a tomada de decisão.

Figura 20 – Interface Final



Fonte: Próprio autor.

5.7.1 Arquitetura e funcionamento geral

A aplicação é composta por um único arquivo principal, `interface.py`, responsável por orquestrar as interações entre o usuário, o banco de dados e os modelos de previsão. Os módulos `sarimax.py`, `fbprophet.py` e `lstm.py` são importados e executados de forma modular e independente, permitindo que cada modelo opere em threads separadas, evitando o travamento da interface durante o processamento.

O núcleo da interface é estruturado sobre duas classes de execução paralela:

- **SeqWorker:** executa os três modelos (SARIMAX, Prophet e LSTM) sequencialmente para um único ativo (ticker);
- **MassWorker:** executa os modelos para todos os tickers disponíveis no banco de dados, com barra de progresso atualizada em tempo real.

Antes da execução dos modelos, a interface realiza automaticamente a consulta ao banco de dados interno `b3_precos.db`. A função `carregar_serie_do_banco()` é responsável por obter os preços históricos do ativo selecionado pelo usuário, retornando uma série temporal organizada e pronta para ser enviada aos módulos SARIMAX, Prophet e LSTM. Dessa forma, a interface atua apenas como intermediária, acessando o banco e repassando os dados para os algoritmos

de previsão.

Os resultados são automaticamente armazenados em pastas predefinidas:

- `/Graficos` — contém os arquivos PNG gerados com as previsões e comparações;
- `/CSV` — contém as métricas e previsões em formato tabular;
- `b3_precos.db` — banco de dados `SQLite` com o histórico de preços dos ativos.

A execução é assíncrona: enquanto os modelos são processados, o usuário pode acompanhar o progresso por meio de uma barra dinâmica e mensagens de status em tempo real. Ao término, a interface atualiza automaticamente as abas de previsões, métricas, gráficos e ranking, este último responsável por consolidar os resultados dos modelos em uma análise comparativa final.

5.7.2 Layout e componentes da interface

A janela principal, representada pela classe `PrevisaoApp`, foi projetada com uma organização modular em quatro abas principais:

1. **Tabela de Previsões:** exibe os valores reais e as previsões combinadas dos três modelos;
2. **Métricas:** mostra os indicadores de desempenho (RMSE, MAPE, R^2, tempo e data de execução);
3. **Gráficos:** apresenta os gráficos salvos no diretório `/Graficos`;
4. **Ranking:** permite a comparação geral entre os ativos (descrita em seção posterior).

Além das abas, a interface possui menus de seleção (para o ticker e o número de dias de previsão), botões de ação (*Rodar Previsão* e *Rodar Todos os Tickers*), e uma barra de progresso integrada, conforme ilustrado na Figura **??**.

5.7.3 Trecho representativo do código-fonte

A seguir, é apresentado um trecho representativo da classe principal `PrevisaoApp`, responsável pela inicialização da interface e execução dos modelos em paralelo:

```
class PrevisaoApp(QWidget):
   def __init__(self):
```

```
        super().__init__()
        self.setWindowTitle("Sistema de Previso de Aes (3 Modelos)")
        self.resize(1250, 900)

        layout = QVBoxLayout(self)
        self.combo_acao = QComboBox()
        self.input_dias = QLineEdit("60")
        self.btn_rodar = QPushButton("Rodar Previso (3 modelos)")
        self.btn_todos = QPushButton("Rodar Todos os Tickers")
        self.progress = QProgressBar()
        self.tabs = QTabWidget()

        # Criao das abas principais
        self.tab_csv = QWidget()
        self.tab_metricas = QWidget()
        self.tab_plot = QWidget()
        self.tab_ranking = QWidget()

        self.tabs.addTab(self.tab_csv, "Tabela de Previses")
        self.tabs.addTab(self.tab_metricas, "Mtricas")
        self.tabs.addTab(self.tab_plot, "Grficos")
        self.tabs.addTab(self.tab_ranking, "Ranking")
        layout.addWidget(self.tabs)
```

Esse trecho ilustra a estrutura modular da aplicação, baseada em `QWidgets` e `Layouts` do `PyQt5`. A partir dessa classe, são instanciadas as threads de execução (`SeqWorker` e `MassWorker`) que conectam os cálculos de previsão à interface gráfica, garantindo a atualização automática dos elementos visuais sem bloquear o fluxo principal do programa.

5.8 Modelo SARIMAX

O modelo SARIMAX (*Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors*) é uma extensão do modelo ARIMA que incorpora efeitos sazonais e, opcionalmente, variáveis exógenas na modelagem de séries temporais. Ele é descrito como ARIMA$(p,d,q) \times (P,D,Q)_s$, onde p, d e q representam os parâmetros não sazonais de autorregressão, diferenciação e média móvel, respectivamente; P, D e Q são os parâmetros sazonais correspondentes; e s indica o período de sazonalidade. Tal estrutura

permite capturar padrões recorrentes que se repetem em intervalos regulares, como ocorre em séries financeiras (PAL; PRAKASH, 2017).

Neste trabalho, o modelo SARIMAX foi empregado para modelar e prever os preços de fechamento de ações da B3 obtidos via biblioteca `yfinance`. A escolha desse modelo deve-se à sua capacidade de lidar com séries temporais que apresentam tendências e sazonalidade mensal.

5.8.1 Configuração e ajuste do modelo

O modelo foi configurado com os parâmetros $(p,d,q) = (2,1,2)$ e $(P,D,Q,s) = (1,1,1,21)$, considerando $s = 21$ dias úteis como o período médio de um ciclo mensal de negociação. O ajuste foi realizado com a função `SARIMAX` da biblioteca `statsmodels`:

```
modelo = SARIMAX(
serie,
order=(2,1,2),
seasonal_order=(1,1,1,21),
trend='n',
enforce_stationarity=False,
enforce_invertibility=False
)
modelo_fit = modelo.fit(disp=False)
```

A definição dos parâmetros (p,d,q) e (P,D,Q,s) foi realizada com base na análise exploratória da série e na inspeção das funções de autocorrelação (ACF) e autocorrelação parcial (PACF), complementadas pelo teste de estacionariedade (ADF). O parâmetro $d = 1$ foi adotado por ser característico de séries financeiras de preços de ações, que apresentam tendência estocástica e se tornam estacionárias após uma diferenciação simples.

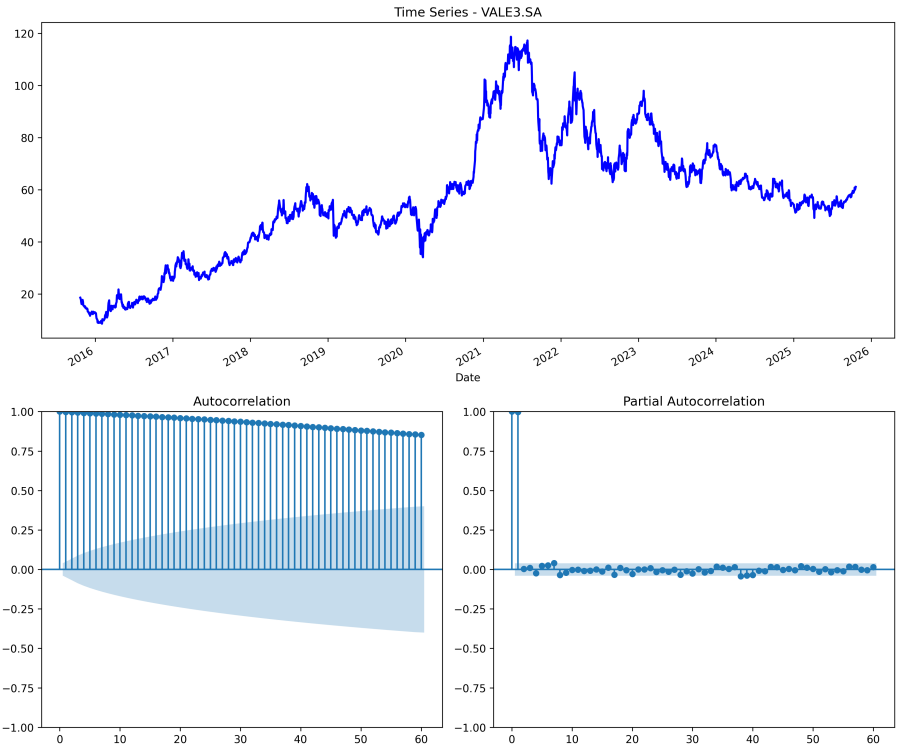
Os valores $p = 2$ e $q = 2$ foram definidos a partir da observação de dois picos significativos nos gráficos PACF e ACF (Figura 21), indicando dependência autorregressiva e de média móvel até a segunda defasagem. Essa configuração permite capturar oscilações de curto prazo sem sobreajustar o modelo.

No componente sazonal, o período $s = 21$ corresponde aproximadamente ao número de pregões mensais na B3. A diferenciação sazonal $D = 1$ remove padrões mensais recorrentes, enquanto $P = 1$ e $Q = 1$ modelam dependências sazonais de primeira

ordem nos termos autorregressivos e de média móvel.

A estrutura resultante $(2,1,2) \times (1,1,1,21)$ mostrou-se estável e apresentou bom equilíbrio entre simplicidade e desempenho, produzindo resíduos aproximadamente brancos e métricas satisfatórias. Essa configuração também está em consonância com estudos anteriores, como o de Kulshreshtha e Vijayalakshmi (2020), que identificaram o modelo ARIMA(2,1,2) como o de melhor ajuste entre as combinações testadas para séries de preços de ações.

Figura 21 – Funções de autocorrelação (ACF) e autocorrelação parcial (PACF)

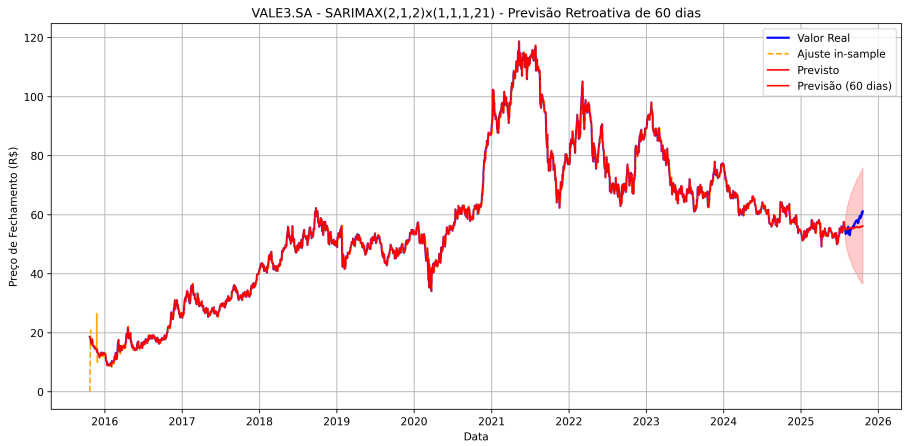


Fonte: Próprio autor.

5.8.2 Previsão e avaliação dos resultados

Após o ajuste do modelo, foi realizada a previsão retroativa dos últimos 60 dias da série VALE3.SA, comparando os valores previstos com os observados. A Figura 22 apresenta o gráfico geral de previsão gerado pelo modelo SARIMAX.

Figura 22 – Gráfico geral de previsões



Fonte: Próprio autor.

O modelo foi capaz de acompanhar as variações gerais da série, capturando oscilações de curto prazo e tendências locais. Pequenas divergências entre valores reais e previstos ocorreram principalmente em períodos de alta volatilidade, o que é esperado em séries financeiras não lineares.

A Tabela 4 apresenta as métricas de desempenho obtidas no conjunto de teste.

Tabela 4 – Métricas de desempenho do modelo SARIMAX

RMSE	**MAPE (%)**	$\mathbf{R^2}$
2.1676	3.06%	0.7291

Fonte: Próprio autor.

Os resultados e gráficos foram salvos automaticamente em diretórios dedicados para análise posterior. O arquivo de imagem é exportado em `Graficos/`, e os valores previstos, juntamente com as faixas de confiança, são armazenados em `CSV/`, conforme mostrado abaixo:

```
path_grafico = os.path.join("Graficos", f"{ticker}_sarimax_{dias}dias.png")
path_csv = os.path.join("CSV", f"{ticker}_sarimax_previsao_retroativa.csv")
```

Em conjunto, as métricas obtidas demonstram que o modelo SARIMAX apresentou desempenho satisfatório para previsões de curto prazo, equilibrando simplicidade, estabilidade e boa capacidade de generalização.

5.8.3 Síntese

O desenvolvimento do modelo SARIMAX envolveu as seguintes etapas:

1. Coleta e preparação da série de preços via `yfinance`;
2. Análise de ACF, PACF e ADF para definição dos parâmetros;
3. Ajuste do modelo $(2,1,2) \times (1,1,1,21)$ e geração de previsões retroativas;
4. Avaliação de desempenho por RMSE, MAPE e R^2;
5. Armazenamento automatizado dos gráficos e resultados em arquivos `.png` e `.csv`.

Essa abordagem permitiu avaliar a eficiência do modelo SARIMAX na previsão de séries financeiras reais, evidenciando sua aplicabilidade para análises de curto prazo em ativos negociados na B3.

5.9 Modelo Facebook Prophet

O modelo Facebook Prophet, proposto por Taylor e Letham (2017), é um método de decomposição aditiva que modela séries temporais como a soma de componentes de tendência, sazonalidade e ruído. Ele é especialmente eficaz em dados que apresentam tendências não lineares e pontos de mudança (*changepoints*) (RAFFERTY, 2021).

5.9.1 Coleta e preparação dos dados

A função `run_from_series()` foi desenvolvida para integrar o Prophet à interface do sistema. A série é transformada para o formato exigido pelo Facebook Prophet:

```
df = pd.DataFrame({
    "ds": pd.to_datetime(serie.index),
    "y": pd.to_numeric(serie.values, errors="coerce")
}).dropna()
```

O conjunto de dados é dividido em 80% para treino e 20% para teste, garantindo que o modelo seja avaliado com dados não vistos durante o treinamento.

5.9.2 Configuração e treinamento

O Facebook Prophet foi configurado para capturar as principais dinâmicas das séries financeiras de preços de fechamento, considerando tanto oscilações de curto prazo (semanais) quanto padrões de longo prazo (anuais). Os parâmetros utilizados foram:

```
from prophet import Prophet

modelo = Prophet(
   daily_seasonality=False,
   weekly_seasonality=True,
   yearly_seasonality=True,
   interval_width=0.80
)
modelo.fit(treino)
```

A justificativa para essa configuração está diretamente relacionada às características das séries de preços de ações:

- **daily_seasonality = False**: a sazonalidade diária foi desativada porque as séries utilizadas são compostas por preços de *fechamento diário* (um valor por dia útil). Como não há múltiplas observações dentro de um mesmo dia, não existem padrões intradiários a serem modelados. Além disso, o Prophet interpreta a sazonalidade diária como variações dentro de um único dia, o que não se aplica a dados de granularidade diária.
- **weekly_seasonality = True**: A sazonalidade semanal foi mantida porque o mercado financeiro apresenta padrões recorrentes ao longo dos dias da semana. Esses ciclos decorrem do próprio funcionamento do mercado como abertura após o fim de semana, ajustes de carteira realizados em determinados dias e comportamentos operacionais típicos. Assim, a sazonalidade semanal permite que o modelo capture essas oscilações curtas, porém frequentes, refletindo melhor a dinâmica real dos pregões.
- **yearly_seasonality = True**: a sazonalidade anual é essencial para capturar efeitos macroeconômicos e corporativos que se repetem a cada ano, como balanços trimestrais, distribuição de dividendos, datas de resultados e eventos sazonais do mercado (por exemplo, maiores volumes de negociação em determinados meses). Essa sazonalidade ajuda o modelo a ajustar tendências cíclicas de longo prazo nos preços dos ativos.

O intervalo de confiança foi definido em 80% (`interval_width=0.80`) para equilibrar a amplitude do "cone de incerteza" e evitar previsões excessivamente conservadoras, comuns quando se utiliza o padrão de 95%. Essa configuração mostrou-se adequada para representar a incerteza inerente às séries financeiras, sem comprometer a interpretabilidade visual dos resultados.

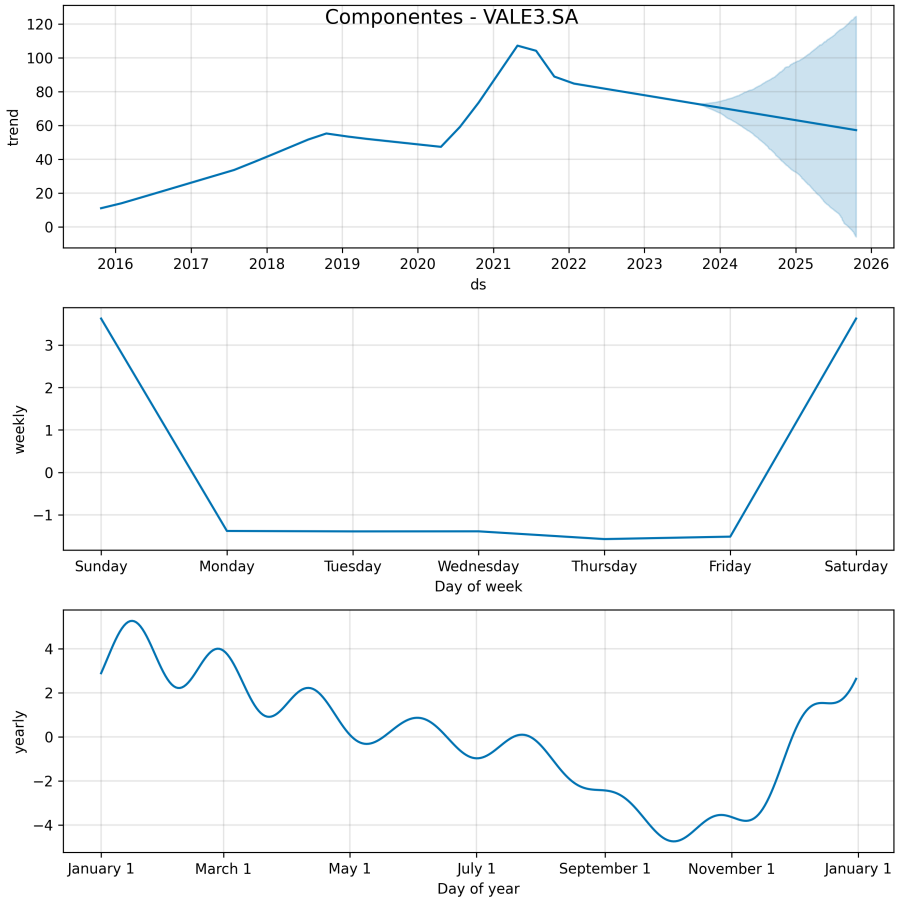
Após a configuração, o modelo é ajustado sobre o conjunto de treino (80% da série), aprendendo a decompor a série em tendência, sazonalidade semanal, sazonalidade anual e ruído aleatório. Em seguida, são realizadas previsões contínuas sobre todo o período analisado, mantendo um gráfico único com o cone de incerteza contínuo.

5.9.3 Geração de previsões e componentes

As previsões são realizadas para toda a série, produzindo uma projeção contínua com o "cone" de incerteza característico do Facebook Prophet. A partir das previsões geradas, são construídos os gráficos de componentes, que decompõem o valor previsto em suas principais partes: tendência, sazonalidade semanal e sazonalidade anual, conforme apresentado na Figura 23.

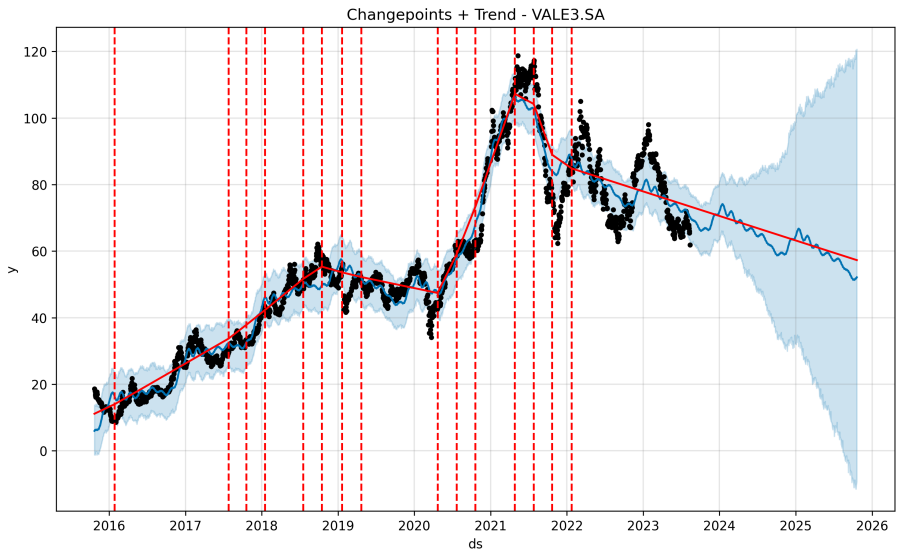
Além disso, é gerado o gráfico de tendência com pontos de mudança (*changepoints*), que ilustra visualmente as variações estruturais detectadas pelo modelo ao longo do tempo, conforme a Figura 24.

Figura 23 – Decomposição em tendência, sazonalidade semanal e anual.



Fonte: Próprio Autor.

Figura 24 – Gráfico de tendência e pontos de mudança (*changepoints*).

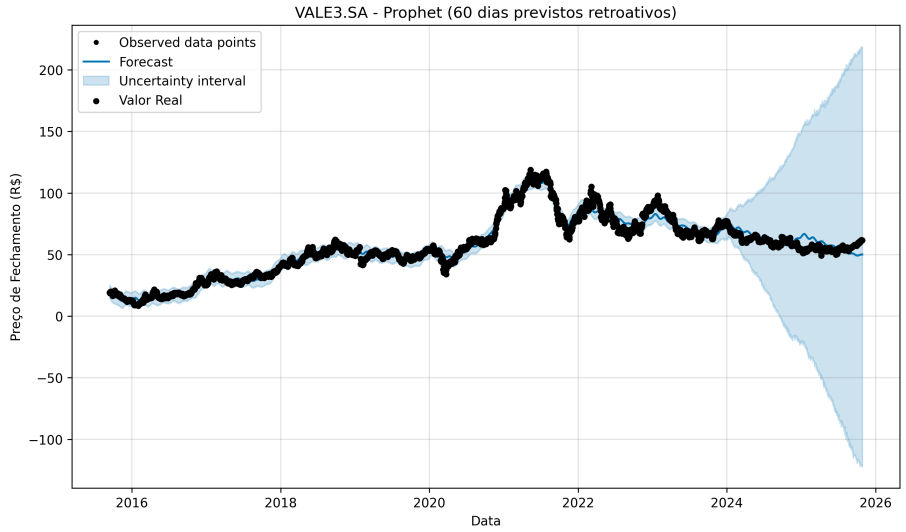


Fonte: Próprio Autor.

O gráfico de previsões (Figura 25) apresenta a série histórica completa juntamente com a projeção gerada pelo modelo Facebook Prophet. A linha azul representa os valores previstos (*yhat*), enquanto a faixa em azul claro indica o intervalo de confiança de 80%, que expressa a incerteza associada às estimativas futuras. Já os pontos em preto correspondem aos valores reais da série de preços.

Esse gráfico permite observar a capacidade do modelo em ajustar-se ao comportamento histórico dos dados e projetar tendências futuras de forma contínua. O cone de incerteza tende a se alargar à medida que o horizonte de previsão aumenta, refletindo o crescimento natural da incerteza em previsões de longo prazo. Dessa forma, o gráfico sintetiza visualmente o desempenho e a confiabilidade do Facebook Prophet na modelagem das séries financeiras analisadas.

Figura 25 – Gráfico geral de previsões



Fonte: Próprio Autor.

5.9.4 Avaliação e métricas

As principais métricas de desempenho empregadas foram:

- **RMSE (Root Mean Squared Error)**: mede a raiz do erro quadrático médio, indicando o desvio médio das previsões em relação aos valores reais. Quanto menor o RMSE, maior a precisão do modelo.
- **MAPE (Mean Absolute Percentage Error)**: expressa o erro percentual médio, permitindo comparar o desempenho entre diferentes séries ou ativos.

- **R (Coeficiente de Determinação)**: representa a proporção da variabilidade dos dados explicada pelo modelo. Valores próximos de 1 indicam bom ajuste.

O código a seguir apresenta o cálculo das métricas:

```
rmse = np.sqrt(mean_squared_error(y_true, y_pred))
mape = mean_absolute_percentage_error(y_true, y_pred) * 100
r2 = r2_score(y_true, y_pred)
```

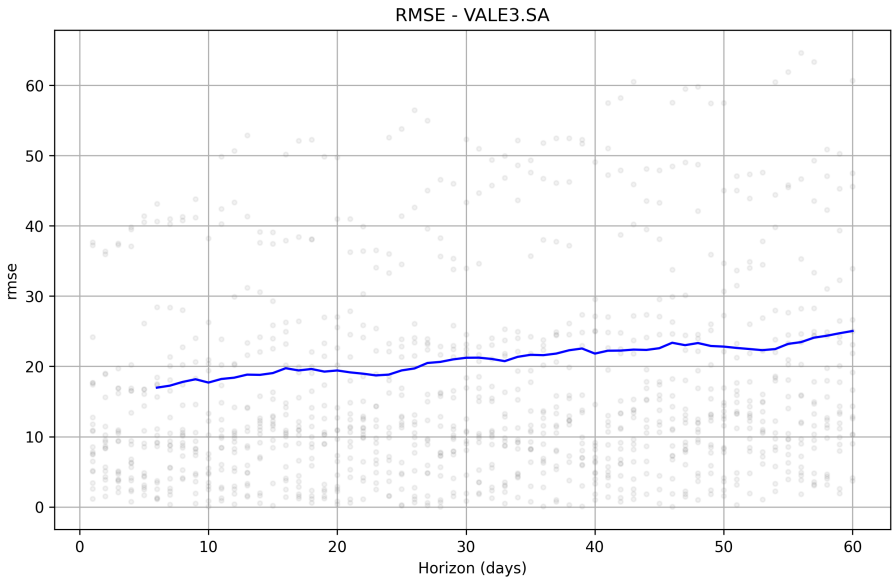
Tabela 5 – Métricas de desempenho do modelo Facebook Prophet

RMSE	MAPE	R^2
5.081	6.87%	0.385

Fonte: Próprio Autor.

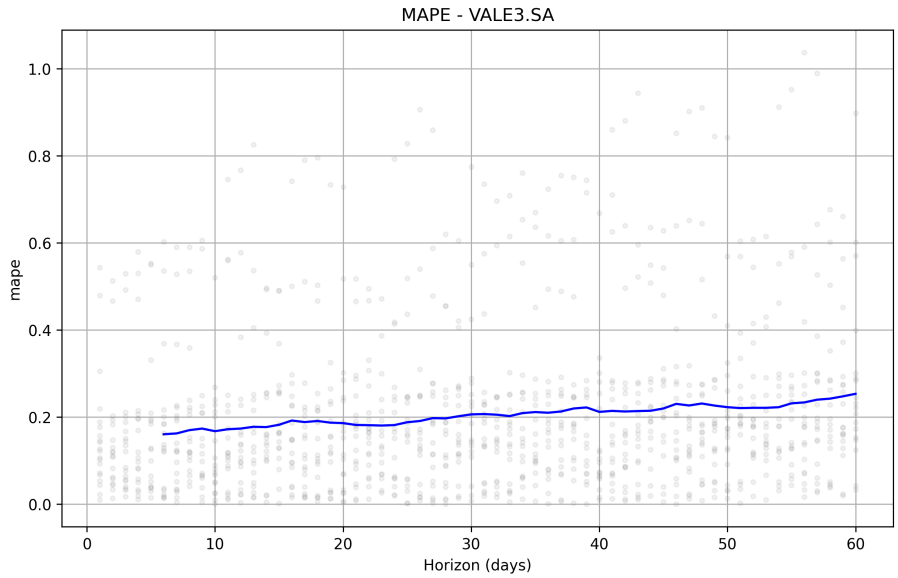
Além das métricas numéricas, foram gerados gráficos (Figura 26 e Figura 27) de desempenho que ilustram o erro de previsão e a qualidade do ajuste, com base na validação cruzada (`cross_validation`) e nas funções de diagnóstico da biblioteca `prophet.diagnostics`.

Figura 26 – Evolução do erro quadrático médio (RMSE) nas previsões do Facebook Prophet.



Fonte: Próprio Autor.

Figura 27 – Erro percentual médio absoluto (MAPE) em diferentes janelas de previsão.



Fonte: Próprio Autor.

Esses gráficos permitem avaliar a estabilidade preditiva do modelo em múltiplos horizontes temporais, identificando possíveis períodos de sobreajuste ou perda de acurácia.

Essas análises visuais complementam os resultados numéricos, permitindo compreender de forma mais intuitiva a robustez e a confiabilidade do Facebook Prophet na previsão de preços de ações.

5.9.5 Síntese

O desenvolvimento do modelo Facebook Prophet contemplou:

1. Preparação da série no formato Prophet;
2. Divisão treino-teste (80/20);
3. Treinamento com sazonalidades anuais e semanais;
4. Geração de previsões contínuas e gráficos de componentes;
5. Cálculo de métricas (RMSE, MAPE, R) e salvamento automatizado.

5.10 Modelo LSTM

As redes neurais do tipo LSTM (*Long Short-Term Memory*) são projetadas para lidar com dependências de longo prazo em séries temporais, superando limitações das redes recorrentes tradicionais (*RNNs*) no problema do desvanecimento ou explosão do gradiente (HOCHREITER; SCHMIDHUBER, 1997). No presente trabalho, foi desenvolvida uma arquitetura LSTM para previsão retroativa de preços de fechamento de ações, baseada em janelas deslizantes de 90 dias, tomando como referência a abordagem de Mukherjee, Singh e Vardhan (2023) para séries financeiras.

5.10.1 Pré-processamento e divisão dos dados

Os dados históricos foram obtidos pelo banco de dados desenvolvido, normalizados no intervalo [0,1] com `MinMaxScaler` e divididos em 80% para treino e 20% para teste. A criação das sequências temporais foi feita por janelas deslizantes de 90 observações, conforme a função abaixo:

```
def criar_sequencias(data, look_back=90):
   X, y = [], []
   for i in range(len(data) - look_back):F
      X.append(data[i:i + look_back])
      y.append(data[i + look_back])
   return np.array(X), np.array(y)
```

5.10.2 Arquitetura e justificativa do modelo

A arquitetura final da LSTM manteve duas camadas recorrentes empilhadas e uma camada densa intermediária, com regularização adaptativa (*dropout* dinâmico) e otimização baseada no algoritmo Adam:

```
model = Sequential([
   LSTM(128, return_sequences=True, input_shape=(look_back, 1)),
   Dropout(dropout_rate),
   LSTM(64, return_sequences=False),
   Dense(16, activation='relu'),
   Dense(1)
])
```

```
optimizer = Adam(learning_rate=0.001, amsgrad=True)
model.compile(optimizer=optimizer, loss='mse')
```

O valor do `dropout` é ajustado de forma dinâmica conforme o tamanho da amostra de treino, variando entre 0,1 e 0,3, o que evita sobreajuste em bases menores sem comprometer a capacidade de generalização em séries extensas. Essa arquitetura foi selecionada por equilibrar profundidade e custo computacional, permitindo capturar tanto padrões de curto quanto de longo prazo.

5.10.3 Treinamento e validação cruzada

O processo de treinamento utilizou os *callbacks* `EarlyStopping` (paciência = 10) e `ReduceLROnPlateau` (fator = 0,3; paciência = 5), prevenindo sobreajuste e otimizando o tempo de convergência.

Além disso, foi empregada uma validação cruzada temporal com dois *folds* (*Time Series Split 2-fold*), garantindo avaliação consistente da estabilidade do modelo em diferentes períodos da série:

```
tscv = TimeSeriesSplit(n_splits=2)
```

Essa estratégia reduz o viés temporal, avaliando o desempenho do modelo em dados históricos distintos sem aleatorização, o que é essencial em contextos financeiros.

5.10.4 Previsão retroativa e avaliação

Após o treinamento, o modelo gera previsões retroativas dos últimos n dias da série utilizando abordagem recursiva, ou seja, o valor previsto em cada passo é reinserido como entrada para a próxima previsão. As métricas de avaliação incluem RMSE, MAPE e R^2, calculadas sobre o conjunto de teste:

```
rmse = np.sqrt(mean_squared_error(y_true, y_pred))
mape = mean_absolute_percentage_error(y_true, y_pred) * 100
r2 = r2_score(y_true, y_pred)
```

Tabela 6 – Métricas de desempenho do modelo LSTM

RMSE	MAPE	R^2
1.074	1.60%	0.912

Fonte: Próprio Autor.

5.10.5 Visualização e armazenamento dos resultados

O gráfico final apresenta os valores reais, o ajuste do modelo (treino + teste) e as previsões retroativas com intervalo de confiança de ±RMSE (Figura 28). Todos os resultados são exportados automaticamente para as pastas `Graficos` e `CSV`, garantindo reprodutibilidade e rastreabilidade do experimento:

```
figpath = os.path.join("Graficos", f"ticker}_LSTM_prev_{dias}dias.png")
csv_prev_path = os.path.join("CSV", f"{ticker}_previsao_LSTM_{dias}dias.csv")
```

Figura 28 – Gráfico geral de previsão

Fonte: Próprio Autor.

5.10.6 Síntese

O desenvolvimento do modelo LSTM envolveu as seguintes etapas principais:

1. Normalização e segmentação dos dados em janelas de 90 observações;
2. Construção de uma arquitetura LSTM profunda com regularização dinâmica;
3. Aplicação de *EarlyStopping*, *ReduceLROnPlateau* e validação cruzada 2-fold;
4. Avaliação quantitativa por RMSE, MAPE e R^2;
5. Armazenamento automático de previsões e gráficos.

O modelo demonstrou robustez e boa capacidade de generalização em séries financeiras com horizontes de curto prazo, apresentando estabilidade estatística entre os dois *folds* avaliados e convergência consistente ao longo do treinamento.

5.11 Algoritmo de Rankeamento

O módulo de *ranking* foi desenvolvido com o objetivo de consolidar os resultados dos modelos de previsão e transformá-los em um mecanismo objetivo de apoio à tomada de decisão. Diferentemente de abordagens puramente estatísticas, o algoritmo implementado combina a qualidade preditiva dos modelos com a tendência futura estimada, priorizando ativos com maior potencial de valorização.

A implementação foi desenvolvida em `Python` e opera diretamente sobre dois arquivos gerados automaticamente pela interface do sistema:

- `metricas_modelos.csv`, contendo as métricas RMSE, MAPE e R^2;
- `previsoes.csv`, contendo os valores reais e previstos para cada ativo.

Inicialmente, os dados são carregados e validados:

```
dfm = pd.read_csv(METRICAS_PATH)
dfp = pd.read_csv(PREVISOES_PATH)

if dfm.empty or dfp.empty:
   raise ValueError("Os arquivos esto vazios.")
```

A versão final do algoritmo utiliza exclusivamente os resultados do modelo **LSTM**, por apresentar maior estabilidade e melhor desempenho em séries financeiras:

```
dfm_lstm = dfm[dfm["Modelo"] == "LSTM"]
```

As métricas são agregadas por ativo, calculando-se a média dos valores de RMSE, MAPE e R^2 para cada *ticker*.

5.11.1 Normalização das métricas

Cada métrica é normalizada para o intervalo $[0, 1]$, com tratamento de valores ausentes e constantes. As métricas de erro (RMSE e MAPE) são invertidas, pois valores menores indicam melhor desempenho:

```
def normalizar(coluna, invertido=False):
    base = (coluna - coluna.min()) / (coluna.max() - coluna.min())
    return 1 - base if invertido else base
```

Com os valores normalizados, é calculado o **Score de Qualidade**, que representa a confiabilidade estatística do modelo:

$$Score_{Qualidade} = 0{,}4 \cdot RMSE_{norm} + 0{,}4 \cdot MAPE_{norm} + 0{,}2 \cdot R2_{norm}$$

5.11.2 Cálculo da variação prevista

A tendência futura é obtida a partir da diferença entre o último valor real e a última previsão realizada pelo modelo LSTM:

$$Variaçao(\%) = \frac{Previso_{final} - Valor_{real}}{Valor_{real}} \times 100$$

A variação é normalizada e ajustada para reforçar tendências positivas e penalizar cenários negativos:

```
if variacao > 0:
    score = min(1.0, variacao_norm * 1.35)
elif variacao < 0:
    score = max(0.0, variacao_norm * 0.6)
```

5.11.3 Cálculo do Score Final

O **Score Final** combina a confiabilidade do modelo com a tendência estimada:

$$Score_{Final} = 0{,}6 \cdot Score_{Qualidade} + 0{,}4 \cdot Variao_{Ajustada}$$

Esse valor representa diretamente o potencial do ativo para valorização.

5.11.4 Regras de recomendação

Com base nos quantis do Score Final, são geradas recomendações automáticas:

- **Comprar**: Score elevado e variação positiva;
- **Vender**: Score baixo ou variação inferior a -5%;
- **Neutra**: cenários intermediários.

```
if score_final >= q_compra and variacao > 0:
   recomendacao = "Comprar"
elif score_final <= q_venda or variacao < -5:
   recomendacao = "Vender"
else:
   recomendacao = "Neutra"
```

5.11.5 Geração do ranking

Os ativos são ordenados de forma decrescente pelo Score Final, de modo que os primeiros representam as melhores oportunidades de compra:

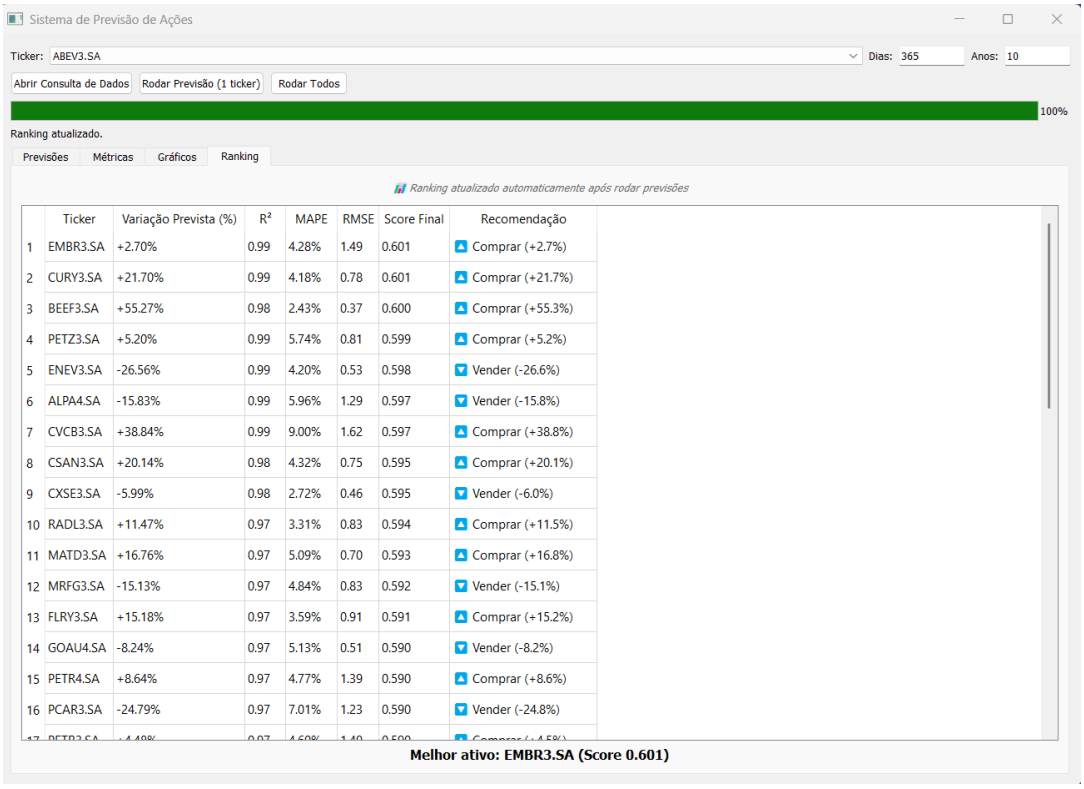
```
df_final = df_final.sort_values("Score_Final", ascending=False)
```

O resultado é exportado automaticamente para o arquivo: `ranking_acoes.csv`

5.11.6 Resultado visual do ranking

A Figura 29 apresenta o resultado real do ranking gerado pelo sistema, exibindo os ativos ordenados, as variações previstas e as recomendações automáticas.

Figura 29 – Resultado do ranking gerado pelo sistema



	Ticker	Variação Prevista (%)	R^2	MAPE	RMSE	Score Final	Recomendação
1	EMBR3.SA	+2.70%	0.99	4.28%	1.49	0.601	Comprar (+2.7%)
2	CURY3.SA	+21.70%	0.99	4.18%	0.78	0.601	Comprar (+21.7%)
3	BEEF3.SA	+55.27%	0.98	2.43%	0.37	0.600	Comprar (+55.3%)
4	PETZ3.SA	+5.20%	0.99	5.74%	0.81	0.599	Comprar (+5.2%)
5	ENEV3.SA	-26.56%	0.99	4.20%	0.53	0.598	Vender (-26.6%)
6	ALPA4.SA	-15.83%	0.99	5.96%	1.29	0.597	Vender (-15.8%)
7	CVCB3.SA	+38.84%	0.99	9.00%	1.62	0.597	Comprar (+38.8%)
8	CSAN3.SA	+20.14%	0.98	4.32%	0.75	0.595	Comprar (+20.1%)
9	CXSE3.SA	-5.99%	0.98	2.72%	0.46	0.595	Vender (-6.0%)
10	RADL3.SA	+11.47%	0.97	3.31%	0.83	0.594	Comprar (+11.5%)
11	MATD3.SA	+16.76%	0.97	5.09%	0.70	0.593	Comprar (+16.8%)
12	MRFG3.SA	-15.13%	0.97	4.84%	0.83	0.592	Vender (-15.1%)
13	FLRY3.SA	+15.18%	0.97	3.59%	0.91	0.591	Comprar (+15.2%)
14	GOAU4.SA	-8.24%	0.97	5.13%	0.51	0.590	Vender (-8.2%)
15	PETR4.SA	+8.64%	0.97	4.77%	1.39	0.590	Comprar (+8.6%)
16	PCAR3.SA	-24.79%	0.97	7.01%	1.23	0.590	Vender (-24.8%)

Fonte: Próprio autor.

Esse módulo transforma métricas técnicas e previsões numéricas em um instrumento prático de apoio à decisão, permitindo ao usuário identificar de forma rápida os ativos com maior potencial de valorização ou risco de queda.

6 CONSIDERAÇÕES FINAIS

Neste trabalho, foram implementadas todas as etapas previstas na metodologia, incluindo a revisão sistemática da literatura, a análise de trabalhos correlatos e o desenvolvimento de uma aplicação completa para previsão de séries temporais no mercado financeiro. O sistema desenvolvido integrou modelos estatísticos e de aprendizado de máquina, banco de dados relacional, interface gráfica interativa e um algoritmo próprio de ranking dos ativos.

A implementação prática permitiu observar, de forma empírica, o comportamento dos principais modelos utilizados na previsão de séries financeiras. Os experimentos confirmaram que modelos baseados em redes neurais recorrentes, especialmente o LSTM, apresentaram maior capacidade de capturar padrões complexos e não lineares, alcançando melhores métricas de desempenho quando comparados aos modelos SARIMAX e Facebook Prophet. Em contrapartida, constatou-se que esse ganho de desempenho está associado a um maior custo computacional, o que pode limitar seu uso em cenários que exigem respostas em tempo real.

O modelo Facebook Prophet demonstrou maior facilidade de uso e tempo reduzido de treinamento, porém apresentou desempenho inferior em relação aos demais modelos, reforçando a necessidade de avaliar cuidadosamente a escolha do algoritmo conforme o objetivo da aplicação. O SARIMAX mostrou-se uma alternativa intermediária, combinando interpretaribilidade estatística com desempenho satisfatório para determinados tipos de séries.

Um dos principais diferenciais deste trabalho foi o desenvolvimento de um módulo de ranking, capaz de transformar métricas técnicas e previsões em recomendações automáticas de compra, venda e neutralidade. Esse algoritmo combinou a qualidade estatística dos modelos com a tendência futura estimada, oferecendo uma visão mais prática e aplicável ao processo de tomada de decisão.

Dessa forma, conclui-se que a utilização integrada de modelos de previsão, banco de dados estruturado, interface gráfica e algoritmos de ranqueamento representa uma abordagem eficiente para o apoio à decisão em investimentos financeiros. O sistema desenvolvido demonstrou ser uma ferramenta funcional e expansível, podendo ser utilizado como base para futuras pesquisas e aplicações no contexto de análise de séries temporais financeiras.

6.1 Trabalhos Futuros

Com a finalização deste estudo, diversas possibilidades de continuidade e aprofundamento surgem como caminhos promissores para a evolução do projeto. As sugestões de trabalhos futuros contemplam tanto a ampliação da base científica quanto o aperfeiçoamento técnico do sistema desenvolvido.

Primeiramente, destaca-se a intenção de transformar os resultados obtidos neste trabalho em uma publicação científica formal. A Sociedade Brasileira de Computação (SBC), especialmente na trilha temática de Computação Aplicada, constitui um ambiente adequado para divulgação de pesquisas envolvendo previsão de séries temporais, aprendizado de máquina e aplicações financeiras. A robustez do sistema implementado e os resultados obtidos fornecem uma base sólida para elaboração de um artigo técnico que consolide a proposta metodológica e os experimentos realizados.

Outra direção relevante envolve a exploração de novos modelos de redes neurais artificiais e técnicas avançadas de *Deep Learning*. Uma linha de evolução especialmente promissora consiste no desenvolvimento de **modelos híbridos**, combinando abordagens estatísticas (como ARIMA, SARIMAX e ETS) com modelos neurais, de forma a integrar as vantagens de ambos os paradigmas. A criação de um módulo experimental para testar combinações entre modelos estatísticos e arquiteturas profundas pode ampliar significativamente o desempenho geral do sistema.

Por fim, uma contribuição importante a ser desenvolvida consiste no aperfeiçoamento e disponibilização aberta (*open-source*) das ferramentas criadas, incluindo o banco de dados, o módulo de ranking de ativos financeiros e o código completo da aplicação. A construção de um repositório público contendo scripts de coleta, processamento e análise de dados permitirá que estudantes, pesquisadores e investidores independentes utilizem, modifiquem e ampliem a plataforma, promovendo evolução contínua e colaborativa. Futuras versões também podem incorporar interfaces web, dashboards interativos e APIs de consulta.

Em síntese, as linhas de continuidade contemplam avanços científicos, metodológicos e tecnológicos, buscando ampliar o impacto acadêmico e prático da ferramenta desenvolvida neste trabalho.

REFERÊNCIAS

AL-SELWI, S. M. et al. Rnn-lstm: From applications to modeling techniques and beyond—systematic review. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, p. 102068, 2024.

ALBUQUERQUE, R. C. d. et al. **Modelagem em séries temporais: aplicaćão em dados de precipitaćão na região do sertão de Pernambuco-Brasil**. [S.l.]: Universidade Federal Rural de Pernambuco, 2018.

BARROS, M. F. de. **Análise e Previsão de Séries Temporais Utilizando Amortecimento Exponencial com Múltiplos Ciclos e Técnicas de Simulação na Produção de Energia Eólica**. Tese (Doutorado) — Dissertaçao de mestrado, Pontificia Universidade Católica do Rio de Janeiro . . . , 2015.

BOX, G.; JENKINS, G. **Time Series Analysis: Forecasting and Control**. Holden-Day, 1970. (Holden-Day series in time series analysis and digital processing). ISBN 9780816210947. Disponível em: https://books.google.com.br/books?id=5BVfnXaq03oC.

BOX, G. E. P. et al. **Time Series Analysis: Forecasting and Control**. 5th. ed. [S.l.]: John Wiley & Sons, 2015. ISBN 978-1-118-67502-1.

CHANG, W. et al. Shiny: Web application framework for r. **R Journal**, v. 7, n. 1, p. 1–15, 2015. Disponível em: https://journal.r-project.org/archive/2015-1/RJ-2015-1.pdf.

CHATFIELD, C. **The Analysis of Time Series: Theory and Practice**. London: Chapman and Hall, 1978.

CHATFIELD, C.; XING, H. **The Analysis of Time Series: An Introduction with R**. 7th. ed. Boca Raton, FL: CRC Press, 2019. ISBN 978-1-138-60965-1.

CHATTERJEE, A.; AYADI, O. F.; BOONE, B. E. Artificial neural network and the financial markets: A survey. **Managerial Finance**, MCB UP Ltd, v. 26, n. 12, p. 32–45, 2000.

CLEVELAND, W. S. Robust locally weighted regression and smoothing scatterplots. **Journal of the American Statistical Association**, Taylor & Francis, v. 74, n. 368, p. 829–836, 1979.

ENDERS, W. **Applied Econometric Time Series**. 2nd. ed. Hoboken, NJ: John Wiley & Sons, 2004.

ENGLE, R. F. Autoregressive conditional heteroskedasticity with estimates of the variance of united kingdom inflation. **Econometrica**, Wiley for the Econometric Society, v. 50, n. 4, p. 987–1007, 1982.

FRANSES, P. H.; DIJK, D. van; OPSCHOOR, A. **Time series models for business and economic forecasting**. 2. ed. Cambridge, England: Cambridge University Press, 2014.

GARLAPATI, A. et al. Stock price prediction using facebook prophet and arima models. In: IEEE. **2021 6th International Conference for Convergence in Technology (I2CT)**. [S.l.], 2021. p. 1–7.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. Disponível em: https://www.deeplearningbook.org/.

HARVEY, A. C.; PETERS, S. Estimation procedures for structural time series models. **Journal of Forecasting**, Wiley Online Library, v. 9, p. 89–108, 1990.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. 2nd. ed. [S.l.]: Prentice Hall, 2001. ISBN 9780132733502.

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

HOLT, C. C. Forecasting seasonals and trends by exponentially weighted moving averages. **International Journal of Forecasting**, Elsevier, v. 20, n. 1, p. 5–10, 2004.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: Principles and Practice**. 3rd. ed. OTexts, 2018. ISBN 978-0-9875071-1-6. Disponível em: https://otexts.com/fpp3/.

JABIN, S. Stock market prediction using feed-forward artificial neural network. **International Journal of Computer Applications**, v. 99, n. 9, p. 4–8, 2014.

JENKINS, I. R. et al. Accident scenario generation with recurrent neural networks. In: **2018 21st International Conference on Intelligent Transportation Systems (ITSC)**. [s.n.], 2018. p. 3340–3345. Disponível em: https://ieeexplore.ieee.org/document/8569661.

KANINDE, S. et al. Stock price prediction using facebook prophet. In: EDP SCIENCES. **ITM web of conferences**. [S.l.], 2022. v. 44, p. 03060.

KULSHRESHTHA, S.; VIJAYALAKSHMI, A. An arima-lstm hybrid model for stock market prediction using live data. **Journal of Engineering Science & Technology Review**, v. 13, n. 4, 2020.

MACHADO, A. A. de A.; CORRÊA, G. N. Avaliação de modelos de previsão dos valores das ações no mercado financeiro usando aprendizado de máquina. **RETEC-Revista de Tecnologias**, v. 15, n. 2, p. 36–46, 2022.

MARTINS, M. A. d. S.; METTE, F.; MACEDO, G. R. d. A utilização de redes neurais artificiais para a estimação dos preços da petrobrás pn na bovespa. **ConTexto**, v. 8, n. 14, p. 1–16, 2008. Disponível em: https://seer.ufrgs.br/ConTexto/article/view/11090.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MESQUITA, C. M. H. S. d. R. **Ciência de dados e aprendizado de máquina para predição em séries temporais financeiras**. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, 2019. Disponível em: http://hdl.handle.net/1843/30444.

MORETTIN, P. **Econometria financeira: um curso em séries temporais financeiras**. Editora Blucher, 2017. ISBN 9788521215851. Disponível em: https://books.google.com.br/books?id=I3i5DwAAQBAJ.

MORETTIN, P. A.; TOLOI, C. M. C. **Análise de Séries Temporais**. São Paulo, Brasil: Edgard Blucher, 2005.

MUKHERJEE, A.; SINGH, A.; VARDHAN, S. **Stock Market Prediction Using LSTM**. S.R.M. Nagar, Kattankulathur, Chengalpattu District: [s.n.], 2023. Mini Project Report, SRM Institute of Science and Technology. Guided by Dr. M. Ferni Ukrit, Department of Computer Science and Engineering.

NASCIMENTO, O. S.; SANTOS, F. G.; FERREIRA, K. H. A. Previsão de preços de ações utilizando inteligência artificial. In: SBC. **Brazilian Workshop on Artificial Intelligence in Finance (BWAIF)**. [S.l.], 2022. p. 37–47.

NIELSEN, A. **Practical time series analysis: Prediction with statistics and machine learning**. [S.l.]: O'Reilly Media, 2019.

OZBAYOGLU, A. M.; GUDELEK, M. U.; SEZER, O. B. Deep learning for financial applications: A survey. **Applied soft computing**, Elsevier, v. 93, p. 106384, 2020.

PAL, A.; PRAKASH, P. **Practical time series analysis: master time series data processing, visualization, and modeling using python**. [S.l.]: Packt Publishing Ltd, 2017.

PARMEZAN, A. R. S.; SOUZA, V. M.; BATISTA, G. E. A. P. A. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. **Information Sciences**, Elsevier, v. 484, p. 302–337, 2019.

PRODANOV, C. C.; FREITAS, E. C. d. **Metodologia do trabalho científico: Métodos e técnicas da pesquisa e do trabalho acadêmico**. 2. ed. [S.l.]: Novo Hamburgo: Feevale, 2013.

RAFFERTY, G. **Forecasting Time Series Data with Facebook Prophet**. S.l.: Packt Publishing, 2021.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, American Psychological Association, v. 65, n. 6, p. 386–408, 1958.

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

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. **Advances in neural information processing systems**, v. 27, 2014.

TAYLOR, S. J.; LETHAM, B. Forecasting at scale. **The American Statistician**, Taylor & Francis, v. 72, n. 1, p. 37–45, 2017.

WONG, B. K.; SELVI, Y. Neural network applications in finance: A review and analysis of literature (1990–1996). **Information & management**, Elsevier, v. 34, n. 3, p. 129–139, 1998.

YADAV, K.; YADAV, M.; SAINI, S. Stock values predictions using deep learning based hybrid models. **CAAI Transactions on Intelligence Technology**, Wiley Online Library, v. 7, n. 1, p. 107–116, 2022.

YU, Y. et al. A review of recurrent neural networks: Lstm cells and network architectures. **Neural Computation**, v. 31, n. 7, p. 1235–1270, 2019.