UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE ECONOMIA
CURSO DE GRADUAÇÃO EM CIÊNCIAS ECONÔMICAS

Jorge Braga Sanches Filho

ARIMA *vs. Machine Learning*: uma aplicação para taxa de câmbio real no Brasil (2000-2023)

RIO DE JANEIRO
2024

Jorge Braga Sanches Filho

ARIMA vs. *Machine Learning*: uma aplicação para taxa de câmbio real no Brasil (2000-2023)

Trabalho de Conclusão de Curso apresentado ao Instituto de Economia da Universidade Federal do Rio de Janeiro, como parte dos requisitos para a obtenção do título de Bacharel em Ciências Econômicas.

Orientadora: Professora Dra. Susan Schommer

RIO DE JANEIRO

2024

CIP - Catalogação na Publicação

Sanches Filho, Jorge Braga
ARIMA vs. Machine Learning: uma aplicação para taxa de câmbio real no Brasil (2000-2023) / Jorge Braga Sanches Filho. -- Rio de Janeiro, 2024.
30 f.

Orientador: Susan Schommer.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Economia, Bacharel em Ciências Econômicas, 2024.

1. Modelos Preditivos. 2. Taxa de Câmbio Real. 3. ARIMA. 4. Machine Learning & Deep Learning. 5. TimeGPT-1. I. Schommer, Susan, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

JORGE BRAGA SANCHES FILHO

ARIMA VS. MACHINE LEARNING: UMA APLICAÇÃO PARA TAXA DE CÂMBIO REAL NO BRASIL (2000-2023)

Trabalho de conclusão de curso apresentado ao Instituto de Economia da Universidade Federal do Rio de Janeiro, como requisito para a obtenção do título de Bacharel em Ciências Econômicas.

Rio de Janeiro, 15/08/2024.

_____

SUSAN SCHOMMER - Presidente
Professora Dra. do Instituto de Economia da UFRJ

_____

GETÚLIO BORGES DA SILVEIRA FILHO
Professor Dr. do Instituto de Economia da UFRJ

_____

ANTONIO LUIS LICHA
Professor Dr. do Instituto de Economia da UFRJ

RESUMO

Nos últimos anos, houve grande avanço acerca da análise de dados, incluindo um novo instrumental a partir do *Machine Learning* ("aprendizado de máquina"). Essa nova abordagem, aplicada a séries temporais, possibilita uma modelagem e previsão mais eficiente[1]. Através do aprendizado supervisionado, via iteração de modelos e seus respectivos resultados, é obtido um modelo final mais robusto. O principal objetivo deste trabalho é comparar os resultados preditivos do modelo estatístico ARIMA, por meio da metodologia Box-Jenkins, frente às arquiteturas *naive*[2] de *Machine* e *Deep Learning*, em uma aplicação para taxa de câmbio real no Brasil (BRL/USD) entre 2000-2022, com previsões para 2023.

Palavras-chave: Modelos Preditivos; ARIMA; Machine Learning; Deep Learning; Taxa de Câmbio Real (BRL/USD); Support Vector Regressor (SVR); Random Forest (RF); Artificial Neural Network (ANN); Convolutional Neural Network (CNN); Recurrent Neural Network (RNN); TimeGPT-1.

[1] Diversos estudos empíricos mostram que a abordagem de ML supera os modelos de séries temporais na previsão de diversos ativos financeiros (Derbentsev, V., 2019; Hamid, S.A., 2014).

[2] *Naive*: ingênuo(a); "Naive Forecasting": uma previsão sem ajustes nos hiper parâmetros, buscando evidenciar o papel fundamental dos pesquisadores acerca da modelagem.

ABSTRACT

There has been significant progress in data analysis in recent years, including a new set of tools stemming from Machine Learning. This fresh approach, applied to time series, allows for more efficient[3] modeling and forecasting. A more robust final model is achieved through supervised learning, by iterating models and their respective outcomes. The main goal of this work is to compare the predictive results of the ARIMA econometric model, using the Box-Jenkins methodology, against naive[4] architectures of Machine and Deep Learning in an application for the real exchange rate in Brazil (BRL/USD) between 2000-2022, with forecasts for 2023.

Keywords: Predictive Models; ARIMA; Machine Learning; Deep Learning; Real Exchange Rate (BRL/USD); Support Vector Regressor (SVR); Random Forest (RF); Artificial Neural Network (ANN); Convolutional Neural Network (CNN); Recurrent Neural Network (RNN); TimeGPT-1.

[3] Various empirical studies show that the ML approach outperforms time series models in predicting various financial assets (Derbentsev, V., 2019; Hamid, S.A., 2014).

[4] Naive: naive; "Naive Forecasting": a forecast without adjustments in the hyperparameters, aiming to highlight the fundamental role of researchers in modeling.

SUMÁRIO

1 INTRODUÇÃO

O trabalho é uma comparação dos resultados preditivos do modelo estatístico ARIMA frente arquiteturas de *Machine Learning* e *Deep Learning*, tais como: *Support Vector Regressor* (SVR), *Random Forest* (RF), *Artificial Neural Network* (ANN), *Convolutional Neural Network* (CNN) e *Recurrent Neural Network* (RNN). Portanto, para avaliar os resultados das previsões *one-step* e *multi-step*[5] são utilizadas as seguintes métricas: Raiz do Erro Quadrático Médio (RMSE), Erro Médio Absoluto (MAE) e Erro Percentual Absoluto Médio (MAPE). Além dessas métricas, uma previsão via regressão linear é utilizada como base comparativa. Dada sua estrutura matemática mais simples, pressupõe-se que os modelos apresentados devem apresentar performances superiores.

Diante das constantes instabilidades financeiras internacionais, a taxa de câmbio e sua trajetória tornam-se objeto de estudo e atenção. Em um cenário de aprofundamento das cadeias globais de valor, marcado pela financeirização dos mercados, o capital passou a ser mais volátil. Além disso, os regimes cambiais flexíveis, adotados por grande parte dos países, tornaram a taxa de câmbio menos previsível e mais volátil. Consequentemente, países, principalmente em desenvolvimento, enfrentam dificuldades para manter a estabilidade cambial. Assim, fica evidente a importância da taxa de câmbio no cenário mundial, uma vez que suas variações impactam os balanços de pagamentos, o comércio internacional e o desempenho econômico, afetando os principais determinantes de atividade, como o investimento, o emprego e os fluxos de comércio.

Segundo Sicsú (2009), a taxa de câmbio é um elemento-chave para o desenvolvimento econômico. Portanto, sua administração deve compor uma estratégia de desenvolvimento. Em complemento, Bresser-Pereira (2012) argumenta que uma taxa de câmbio competitiva é basilar para o desenvolvimento econômico, funcionando como um "interruptor de luz" que "liga" ou "desliga" as empresas tecnologicamente e administrativamente competentes à demanda mundial.

A luz dos resultados empíricos apontados por Rocha, et al (2011), o estudo utiliza a taxa de câmbio real, por ser uma variável relevante para a compreensão do crescimento das economias emergentes e desenvolvidas. Ela possibilita uma melhor percepção acerca da evolução do poder de compra entre os países, ao considerar a inflação doméstica e externa.

[5] *One-step* e *Multi-step Forecast*: Na previsão de um passo o modelo é atualizado a cada previsão, enquanto na previsão de vários passos, são previstos todos os pontos de uma vez, sem atualização entre os passos.

Outrossim, mudanças na taxa de câmbio real podem ser indicadores de mudanças nas condições econômicas futuras, como desequilíbrios comerciais, pressões inflacionárias ou mudanças na competitividade das empresas domésticas, sendo essencial na determinação de políticas econômicas. Dito isto, a análise da taxa de câmbio real pode ajudar na previsão de tendências econômicas.

2 METODOLOGIA

2.1 Modelagem ARIMA

Os modelos de previsão são capazes de utilizar observações passadas para prever o comportamento de uma série temporal, conforme apontado por Morettin e Toloi (2004). Entre esses modelos, destaca-se o ARIMA (p,d,q), conhecido como Modelo Autorregressivo, Integrado e de Médias Móveis. Esse modelo é composto por um componente autorregressivo (AR), um filtro de integração (I) e um componente de médias móveis (MA), sendo indicado para séries não-estacionárias (Santos, 2014).

A componente autorregressiva (AR) descreve como cada observação é uma função das observações anteriores, considerada a parte capaz de lidar com tendências. Sendo representada como: $Y_t = c + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \cdots + \varphi_p Y_{t-p} + e_t$.

Onde:

- Y_t é o valor observado em t,
- Y_{t-1} é o valor observado no tempo t−1,
- e_t é o erro aleatório no tempo t,
- c uma constante,
- $\varphi_1, \varphi_2, ..., \varphi_p$ são os coeficientes dos termos autorregressivos até a ordem p.

Já a componente de diferenciação (I) determina o número de diferenciações necessárias para tornar a série temporal estacionária. A componente de médias móveis (MA), por sua vez, é responsável por capturar a sazonalidade e padrões de autocorrelação, descrevendo como cada observação é uma função dos erros (ou resíduos) anteriores. Ela é representada como: $Y_t = c + \theta_1 e_{t-1} + e_t$.

Onde:

- θ_1 é o coeficiente do termo de médias móveis,
- e_{t-1} é o erro aleatório no temo t−1.

Portanto, a fórmula geral do modelo ARIMA pode ser descrita da seguinte forma: $(1 - \varphi_1 B - \varphi_2 B^2 - ... - \varphi_p B^p)(1 - B)^d Y_t = (1 + \theta_1 B + \theta_2 B^2 + ... + \theta_p B^p) e_t$. Sendo B o operador de defasagem (diferenciação) e d o número de diferenciações necessárias para tornar a série estacionária.

Foi utilizada a metodologia Box Jenkins, que possibilita um ciclo iterativo para escolha do melhor modelo com base nos próprios dados da série. O primeiro passo é a identificação das ordens do modelo, realizada a partir das funções de autocorrelação e autocorrelação parcial da

série estacionária. Em seguida, é efetuada a estimação dos parâmetros do modelo para teste de significância. A partir do diagnóstico, verifica-se se os resíduos do modelo correspondem a um ruído branco. Assim, torna-se possível a previsão para valores futuros da série.

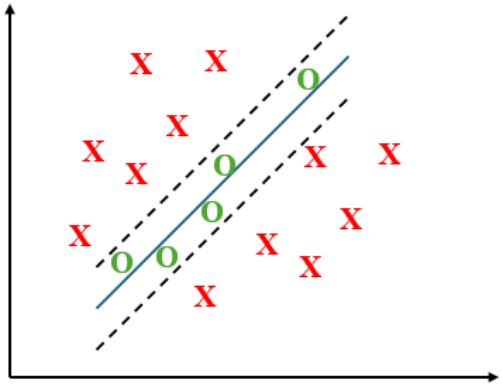
A seleção do modelo ARIMA mais adequado é feita utilizando os critérios de informação Bayesiano (BIC) e Akaike (AIC) dos modelos testados. Adicionalmente, para avaliar a precisão das previsões, são utilizadas métricas como a Raiz do Erro Quadrático Médio (RMSE), Erro Médio Absoluto (MAE) e Erro Percentual Absoluto Médio (MAPE). Conforme comentado por Hyndman (2001), os modelos ARIMA são um método popular de previsão devido à sua estrutura matemática bem desenvolvida, que permite quantificar a incerteza das previsões.

2.2 Machine Learning

2.2.1 Support Vector Regressor (SVR)

O *Support Vector Regressor* (SVR) é uma técnica de aprendizado supervisionado, que é uma extensão do *Support Vector Machine* (SVM), comumente usado para problemas de classificação, mas adaptado para resolver problemas de regressão, incluindo previsões em séries temporais. O artifício chave do SVR é a determinação de um hiperplano de regressão que melhor se ajusta aos dados, minimizando a função de perda e mantendo os erros dentro de uma margem definida. Essa margem é estabelecida pelos "vetores de suporte" (*support vectors*).

Figura 1: Ilustração Simplificada do Support Vector Regressor (SVR)



Fonte: Elaboração Própria

Para aplicar o SVR em séries temporais, é primeiro necessário entender a formulação do problema de otimização subjacente. À luz dos conceitos abordados por Borges (2020), a função de perda do SVR é dada por: $\min_{w,b,\xi i,\xi i^*} \frac{1}{2} ||w||^2 + C \sum (\xi_i + \xi_i^*)$.

Onde w representa o vetor de pesos, b é o termo de polarização (ou intercepto), ξ_i e ξ_i^* são as variáveis de folga que representam os erros de previsão e C é o parâmetro de regularização que controla o trade-off entre maximizar a margem e minimizar os erros.

As restrições associadas a essa função de perda são:

- $y_i - \langle w, \phi(x_i) \rangle - b \leq \epsilon + \xi_i$,
- $\langle w, \phi(x_i) \rangle + b - y_i \leq \epsilon + \xi_i^*$,
- $\xi_i, \xi_i^* \geq 0$.

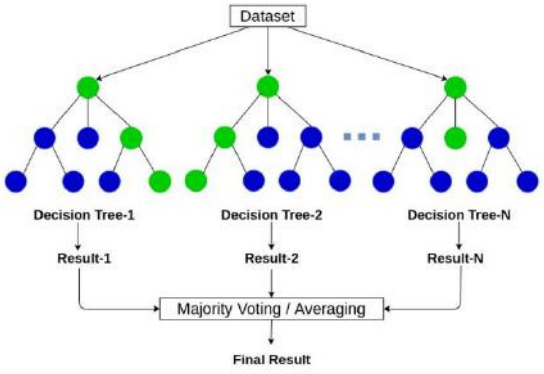
Considerando y_i o valor real de saída, $\phi(x_i)$ a função de kernel (função de identificação das características) e ϵ a margem. A função de kernel é basilar para o SVR, ela é responsável por mapear os dados de entrada para um espaço de características de maior dimensão, facilitando a linearização e/ou modelagem da relação entre os dados. É importante comentar que a escolha da função de kernel depende da natureza dos dados, podendo ser Linear, Polinomial, Gaussiana, Baseada em Função Radial (RBF), etc.

No trabalho, utilizou-se a função de *kernel* radial (RBF – Radial Basis Function). Ela pode ser descrita da seguinte forma: $K(x, x') = \exp(-\gamma||x - x'||^2)$, sendo x e x' vetores de características e γ um parâmetro de ajuste que controla a largura do kernel. A função de kernel RBF é amplamente utilizada devido à sua capacidade de capturar relações não lineares entre os dados e à sua eficácia em diversos cenários de modelagem, incluindo previsão em séries temporais.

2.2.2 Random Forest (RF)

A partir dos conceitos introduzidos por Breiman (2001), entende-se que o modelo *Random Forest* (RF) é uma técnica de *Machine Learning* que utiliza uma coleção de "árvores de decisão" para realizar tarefas de classificação ou regressão. Trata-se de uma abordagem de "aprendizado por agrupamento" (*ensemble learning*), onde múltiplos regressores são gerados e seus resultados são agregados para produzir uma predição final. É importante comentar que a metodologia de "árvores de decisão" é uma extensão do método de "*bagging*" (*bootstrap aggregating*).

Figura 2: Ilustração do Random Forest (RF)



Fonte: Anas Brital Website

O "*bagging*" (*bootstrap aggregating*) é uma técnica que visa reduzir a variância de modelos de ML, especialmente aqueles sensíveis a pequenas variações nos dados de treinamento. Nesse sentido, são treinados diversos subconjuntos aleatórios do conjunto de dados original. As "árvores de decisão" são modelos simples que dividem o espaço amostral em regiões retangulares. Cada árvore representa uma decisão baseada em um valor amostral e as folhas representam as classes ou valores de regressão previstos.

Múltiplas "árvores de decisão" são construídas de forma independente, garantindo que cada árvore tenha uma visão diferente dos dados e, assim, reduzindo a correlação entre elas. Além de usar subconjuntos aleatórios dos dados, a RF introduz mais randomização ao selecionar apenas um subconjunto aleatório de características (*features*) para cada árvore. Isso assegura que cada árvore tenha uma perspectiva diferente do espaço de características, reduzindo ainda mais a correlação entre elas.

Para fazer uma previsão para uma nova amostra, as previsões de todas as árvores na floresta são agregadas. Para problemas de classificação, isso pode ser feito por meio de votação majoritária, enquanto, para regressão, a média das previsões de todas as árvores é calculada. Essa abordagem de *ensemble* ajuda a reduzir o *overfitting* e a melhorar a generalização do modelo, tornando-o robusto e preciso mesmo em conjuntos de dados complexos.

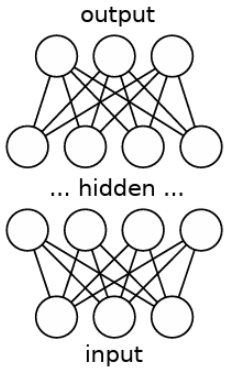
2.3 Deep Learning

2.3.1 Artificial Neural Network (ANN)

Segundo Gardner e Dorling (1998), o modelo *Artificial Neural Network* (ANN) segue a arquitetura de um Perceptron de Múltiplas Camadas (MLP, *Multi-Layer Perceptron*). A MLP consiste em várias camadas de neurônios, cada uma conectada à próxima por meio de conexões ponderadas. São consideradas três camadas principais: a Camada de Entrada (*Input Layer*), responsável por receber os dados amostrais; as Camadas Ocultas (*Hidden Layers*), que realizam

cálculos intermediários; e a Camada de Saída (*Output Layer*), responsável pelos resultados preditivos.

Figura 3: Ilustração ANN



Fonte: Gamboa, 2017: "Deep learning for time-series analysis."

Cada “nó” (neurônio), na camada oculta ou de saída, recebe entradas ponderadas das camadas anteriores. Essa agregação é normalmente a soma ponderada das entradas, representada pela equação: $z_j = \sum_i w_{ij}.x_i + b_j$. Sendo z_j a soma ponderada na camada j, w_{ij} o peso da conexão entre o neurônio i na camada anterior e o neurônio j na camada atual, x_i a entrada para o neurônio j na camada atual e b_j o viés do neurônio j.

O próximo passo é introduzir a não-linearidade na rede através da função de ativação aplicada no resultado da agregação acima. Exemplos comuns de funções de ativação incluem a função sigmóide logística, a tangente hiperbólica (tanh) e as unidades lineares retificadas (ReLU), sendo esta última a função de ativação utilizada no trabalho em questão. A função de ativação pode ser representada pela equação: $a_j = f(z_j)$. Onde a_j é a saída ativada do neurônio j na camada atual e f(.) a função de ativação.

Além disso, dois conceitos fundamentais para compreender a ANN são a Propagação (*Forward Propagation*) e a Retropropagação (*Backpropagation*). A Propagação refere-se às etapas de cálculo das saídas da rede neural, desde a camada de entrada até a camada de saída, passando pelas camadas ocultas. Já a Retropropagação refere-se ao algoritmo utilizado para treinar a rede, que calcula o gradiente da função de perda em relação aos pesos da rede, permitindo que os pesos sejam ajustados para minimizar a perda.

Conforme explicado por Derbentsev et al. (2020), o aprendizado em rede consiste em ajustar os pesos das conexões entre os neurônios de forma que a diferença entre a saída desejada e a saída predita seja minimizada. Isso é alcançado através da implementação de um algoritmo de otimização, geralmente o Gradiente Descendente, aplicado em conjunto a Retropropagação. No trabalho em questão, o algoritmo de otimização especificado é o "Adam", um otimizador baseado em Gradiente Descendente Estocástico com taxas de aprendizado adaptativas. O

"Adam" ajusta a taxa de aprendizado para cada parâmetro da rede neural com base nas estimativas do primeiro momento (média) e do segundo momento (variação) dos gradientes, permitindo uma convergência mais eficiente e rápida em comparação com o Gradiente Descendente tradicional com uma taxa de aprendizado fixa.

2.3.2 Convolutional Neural Network (CNN)

Uma *Convolutional Neural Network* (CNN) é um tipo de rede neural artificial especialmente projetada para processar dados de forma análoga ao processamento visual no cérebro humano. Originalmente desenvolvida para tarefas de processamento de imagens, as CNNs também têm encontrado aplicação em diversas áreas, incluindo o tratamento de séries temporais. No contexto das séries temporais, os dados são tratados como imagens unidimensionais, onde cada ponto na série é considerado como um pixel na imagem. Isso permite que a CNN aprenda padrões temporais em diferentes escalas de maneira eficaz.

Sob o prisma dos conceitos abordados por Lecun, Bengio e Hinton (2015), a estrutura básica de uma CNN consiste em camadas de convolução, camadas de *pooling* e camadas totalmente conectadas. As camadas de convolução são fundamentais, aplicando conjuntos de filtros (ou *kernels*) para extrair características cruciais dos dados de entrada. Cada filtro é convoluído com a entrada, gerando um "mapa de características" que destaca as informações aprendidas. Após a convolução, aplicam-se camadas de *pooling* para reduzir a dimensionalidade dos "mapas de características", aumentando a eficiência do processamento e ajudando a evitar o *overfitting*. A operação de *pooling*, como o *max pooling* aplicado no estudo, reduz a resolução espacial do "mapa de características", mantendo as informações mais relevantes. Por fim, após várias camadas de convolução e *pooling*, a saída é passada para uma ou mais camadas totalmente conectadas, que realizam a regressão final com base nas características extraídas anteriormente, funcionando de maneira similar às redes neurais tradicionais (ANNs).

Se consideramos X a entrada para a camada de convolução, W como o conjunto de pesos do filtro (*kernel*) e b como o viés, a operação de convolução pode ser descrita como $Z = (W * X) + b$, onde * denota a operação de convolução. Para cada posição na entrada, o filtro é aplicado multiplicando seus pesos pela região correspondente na entrada e somando esses produtos, ao qual o viés é adicionado. O resultado é o valor de ativação para aquela posição no "mapa de características". Essa operação é repetida para toda a entrada, resultando em um "mapa de características" que destaca padrões importantes para a tarefa em questão.

2.3.3 Recurrent Neural Network (RNN)

O modelo de Rede Neural Recorrente (RNN) é uma arquitetura especializada de *Deep Learning* amplamente empregada na análise de dados sequenciais, como séries temporais, texto e áudio. Sua característica distintiva é a capacidade de processar sequências de dados de comprimento variável, mantendo um estado interno que encapsula informações sobre partes anteriores da sequência.

Uma RNN é caracterizada pela presença de loops em sua estrutura, o que lhe permite manter uma memória dos estados anteriores ao processar novas entradas. Essa capacidade é especialmente valiosa em domínios onde a ordem dos dados é crucial, como em séries temporais, onde o valor de uma observação pode depender do tempo ou de observações anteriores.

Segundo Goodfellow et al (2016), a arquitetura básica de uma RNN é composta por uma camada de entrada, uma ou mais camadas ocultas (também chamadas de unidades de memória) e uma camada de saída. O diferencial em relação às redes neurais *feedforward* convencionais é a presença de conexões de *feedback*, que permitem que a saída de uma camada seja retroalimentada como entrada para a próxima, possibilitando a modelagem de dependências temporais.

Para treinar uma RNN, emprega-se um algoritmo de retropropagação denominado *Backpropagation Through Time* (BPTT). Esse algoritmo é uma extensão do método de retropropagação utilizado em redes neurais *feedforward*, porém desenrola a rede ao longo do tempo, transformando-a em uma estrutura *feedforward* com múltiplas camadas. Durante o processo de treinamento, os gradientes dos pesos são calculados em relação a todos os passos temporais da sequência, e os pesos são ajustados para minimizar a função de perda.

Conforme exposto por Gamboa (2017), apesar da eficácia das RNNs na modelagem de sequências temporais, elas enfrentam o "problema do gradiente desvanecente", onde os gradientes tornam-se muito pequenos à medida que são retropropagados ao longo do tempo, dificultando o treinamento eficaz de redes profundas ou sequências longas.

Uma variação comum das RNNs é a Long Short-Term Memory (LSTM), desenvolvida para contornar o problema explicitado acima ao introduzir "células de memória", que retêm informações por longos períodos e regulam o fluxo de informações na rede. Cada LSTM é composta por três portões principais: o Portão do Esquecimento, o Portão de Entrada e o Portão de Saída. Esses portões, controlados por funções de ativação (no estudo, sigmoide e tangente hiperbólica), determinam quanta informação deve ser mantida ou descartada do estado interno, garantindo um fluxo eficiente de informações ao longo da sequência temporal. No estudo em

questão, foi implementada a arquitetura de uma RNN com uma camada LSTM para mitigar o problema do gradiente desvanecente.

As equações que regem o comportamento da LSTM não estão no escopo de abordagem do estudo. O importante é destacar a funcionalidade dessa estrutura para o modelo em questão. Para mais detalhes, consultar o trabalho "*Understanding LSTM: a tutorial into long short-term memory recurrent neural networks*" (Staudemeyer e Morris, 2019).

3 BASE DE DADOS E TRATAMENTOS

A base de dados considera informações mensais no período de janeiro de 2000 a dezembro de 2022, com o ano de 2023 utilizado para previsões. A série da taxa de câmbio real (BRL/USD) é fornecida pelo Banco Central do Brasil (BACEN) através do Sistema de Séries Temporais (SGS). A base de comparação é definida como 100 em janeiro de 2000 e é elaborada a partir do ajuste da taxa de câmbio nominal pela diferença entre a inflação brasileira (IPCA) e a inflação dos Estados Unidos (CPI), descrita da seguinte forma: Tx. de Câmbio Nominal * CPI/IPCA. Com esses dados, foi possível realizar uma análise detalhada, incluindo a construção de um gráfico de série em nível para verificar a possível existência de tendências e sazonalidade, além da aplicação de testes de estacionariedade.

Utilizando a série em log e em primeira diferença, ajustou-se o modelo ARIMA (p,d,q) com base nos critérios de informação. Em seguida, foram testados os modelos de *Machine Learning* e *Deep Learning*. É importante comentar que para todos os modelos foram realizados testes de especificação, como o Ljung-Box, para verificar a presença de autocorrelação nos resíduos e garantir que eles foram capazes de capturar completamente a estrutura temporal dos dados.

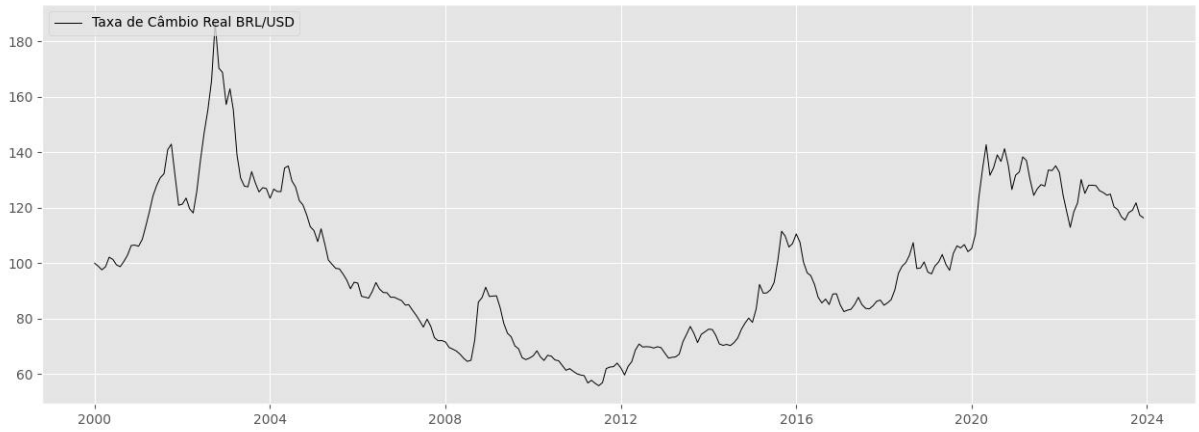
As previsões foram comparadas com base nas métricas de avaliação, como RMSE, MAE e MAPE, que permitem medir a diferença entre os dados reais e as predições obtidas. Além disso, foi elaborado um modelo de regressão linear como parâmetro base, que utiliza uma janela deslizante de 12 observações (tamanho da amostra de previsão). Na análise, foram utilizados os softwares *Python* e *Time Series Lab*. A biblioteca *Python* "*sklearn*" (*scikit-learn*) foi empregada para a elaboração dos modelos de regressão linear e *Machine Learning*. Enquanto para os modelos de *Deep Learning*, utilizou-se o *TensorFlow*.

4 RESULTADOS

4.1 Análise e Descrição da Série

O Gráfico 1 exibe a série da Taxa de Câmbio Real no Brasil em nível, coletada entre o período de janeiro de 2000 a dezembro de 2023.

Gráfico 1: Taxa de Câmbio Real (BRL/USD), entre 2000 a 2023, Jan2000=100

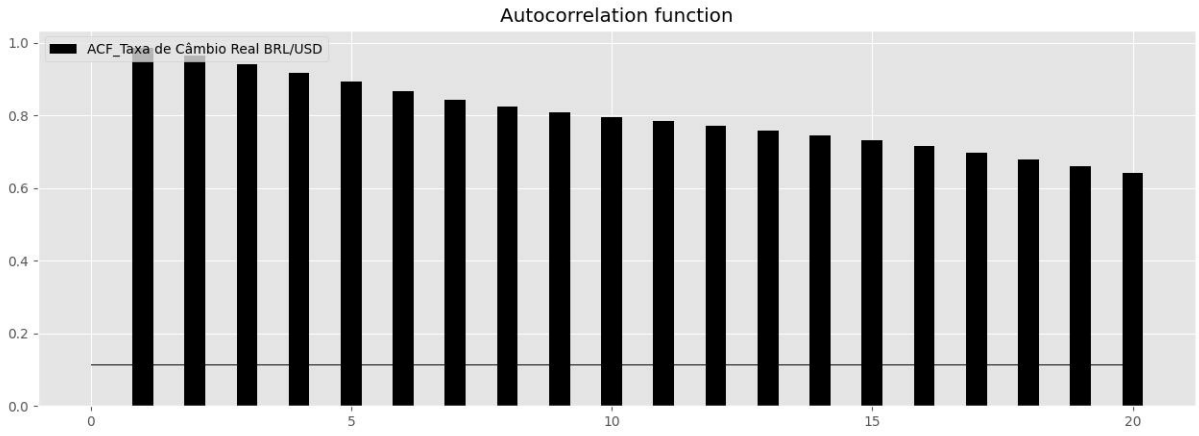


Fonte: Banco Central do Brasil (software Time Series Lab)

É possível observar que a série não apresenta um padrão de crescimento bem definido ao longo do tempo, sugerindo não estacionáriedade. Além disso, nota-se que a série não apresenta variações cíclicas, ou seja, não exibe um padrão de sazonalidade claro.

Nesse sentido, para validar a análise visual, foram realizados o exame do correlograma e os testes Dickey-Fuller e KPSS na série original, com o objetivo de confirmar a não estacionariedade da série. Observou-se que a função de autocorrelação, apresentada no Gráfico 2, não de decai rapidamente a zero, indicando que a série é não estacionária. Em complemento, na Tabela 1, a análise dos resultados dos testes confirma a não estacionariedade da série: não se rejeita a hipótese nula do teste Dickey-Fuller e se rejeita a hipótese nula do teste de KPSS.

Gráfico 2: Função de Autocorrelação (ACF) da Série em Nível



Fonte: Elaboração Própria (software Time Series Lab)

Tabela 1 – Testes de Dickey-Fuller e KPSS para Série em Nível

Augmented Dickey-Fuller	
Statistic	-1.40
P-value	0.8615
Lags	7
H0: unit root is present	
Test result: do not reject H0	

KPSS	
Statistic	0.5385
P-value <	0.0330
Lags	10
H0: series is stationary	
Test result: reject H0	

KPSS	
Statistic	0.5381
P-value <	0.0100
Lags	10
H0: series is trend stationary	
Test result: reject H0	

Fonte: Elaboração Própria (resultados obtidos no Time Series Lab)

4.2 Diferenciação e Modelagem da Série

Após a detecção de não estacionariedade na série em nível, foram aplicadas a transformação logarítmica e o operador de diferenças para tornar a série estacionária. Essas transformações foram suficientes para estacionarizar a série, conforme evidenciado pelos testes na Tabela 2.

Tabela 2 – Testes de Dickey-Fuller e KPSS para Série em Log e na Primeira Diferença

Augmented Dickey-Fuller	
Statistic	-12.11
P-value	2,54E-15
Lags	0
H0: unit root is present	
Test result: reject H0	

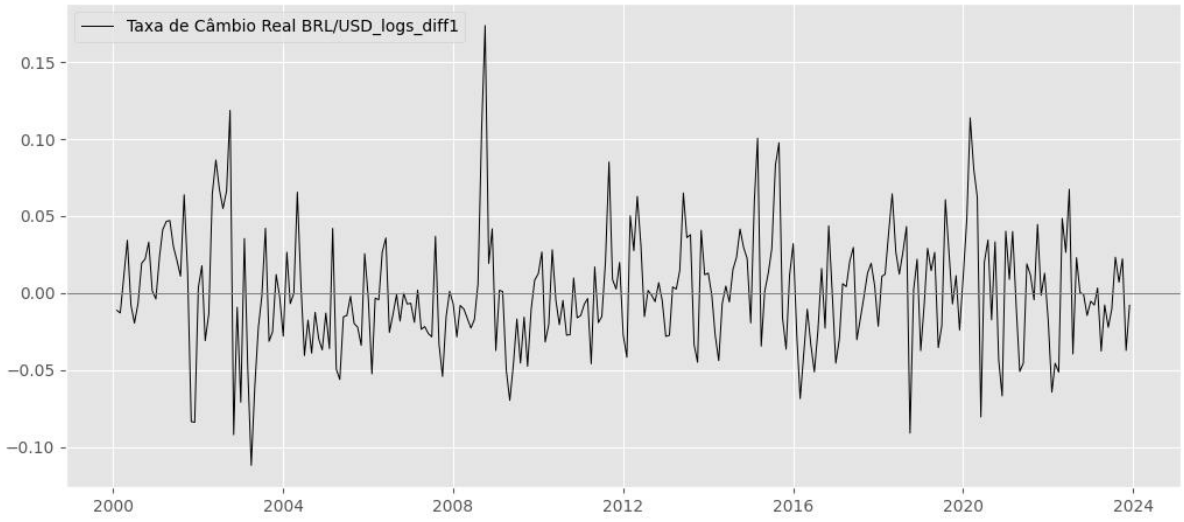
KPSS	
Statistic	0.1307
P-value >	0.0100
Lags	5
H0: series is stationary	
Test result: do not reject H0	

KPSS	
Statistic	0.0807
P-value>	0.1000
Lags	4
H0: series is trend stationary	
Test result: do not reject H0	

Fonte: Elaboração Própria (resultados obtidos no Time Series Lab)

Na Gráfico 3 é possível observar a série ajustada e constatar que ela está estacionarizada.

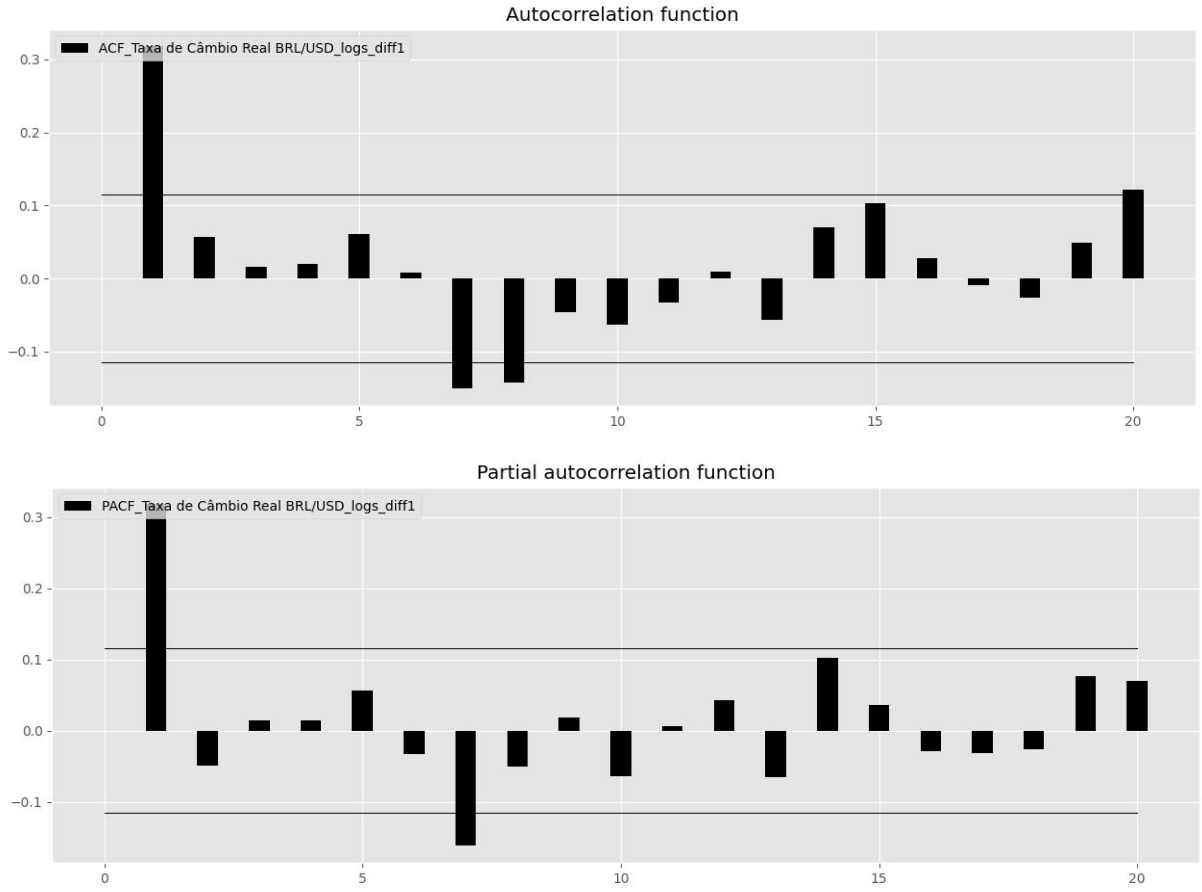
Gráfico 3: Série em Log e Primeira Diferença



Fonte: Elaboração Própria (software Time Series Lab)

Utilizando os gráficos das funções de autocorrelação e autocorrelação parcial (Gráficos 4 e 5), foi possível aplicar a metodologia Box-Jenkins e selecionar os possíveis parâmetros p e q do modelo ARIMA (p,d,q). Nesse sentido, foram selecionados os modelos (1,1,0), (1,1,1) e (0,1,1) para a comparação com base nos critérios de informação. Assim, com base nos critérios AIC e BIC (Tabela 3), o modelo ARIMA (0,1,1) foi o que melhor se adequou à série.

Gráficos 4 e 5: Função de Autocorrelação e Autocorrelação Parcial da Série em Log e Primeira Diferença



Fonte: Elaboração Própria (software Time Series Lab)

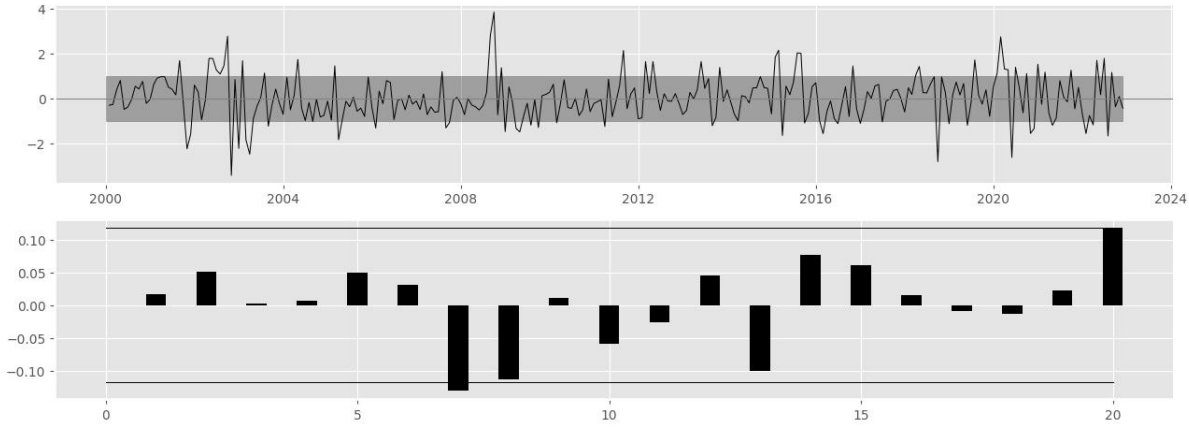
Tabela 3 – Estatísticas ARIMA

	ARIMA (p,d,q)		
	(1,1,0)	**(1,1,1)**	**(0,1,1)**
Log likelihood	519,3772	519,7923	519,3784
AIC	-1.032,7544	-1.031,5846	-1.034,7568
AICc	-1.032,6658	-1.031,4365	-1.034,7127
BIC	-1.021,9041	-1.017,1175	-1.027,5232
MSE	0,0013	0,0013	0,0013
RMSE	0,0366	0,0366	0,0367
MAE	0,0279	0,0279	0,0280
MAPE	155,8414	167,1289	174,9291

Fonte: Elaboração Própria (resultados obtidos no Time Series Lab)

Agora, é necessário verificar se os resíduos do modelo ajustado são realmente um ruído branco. Para isso, foram utilizados o gráfico dos resíduos padronizados e o correlograma dos resíduos padronizados (Gráficos 6 e 7). Em complemento, foi realizado o teste de Ljung-Box (Tabela 4).

Figuras 6 e 7: Resíduos Padronizados e Correlograma Residual ARIMA (0,1,1)



Fonte: Elaboração Própria (software Time Series Lab)

Tabela 4 – Teste de Ljung-Box Resíduos ARIMA (0,1,1)

Ljung-Box Test	**ARIMA (0,1,1)**	
Lags	Estatística	P-valor
1	0,8011	0,3708
2	0,8037	0,6691
3	0,8137	0,8462
4	1,4882	0,8287
5	1,7668	0,8804
6	6,5786	0,3616
7	10,1479	0,1803
8	10,1845	0,2523
9	11,1805	0,2635
10	11,3803	0,3287

Fonte: Elaboração Própria (resultados obtidos no Time Series Lab)

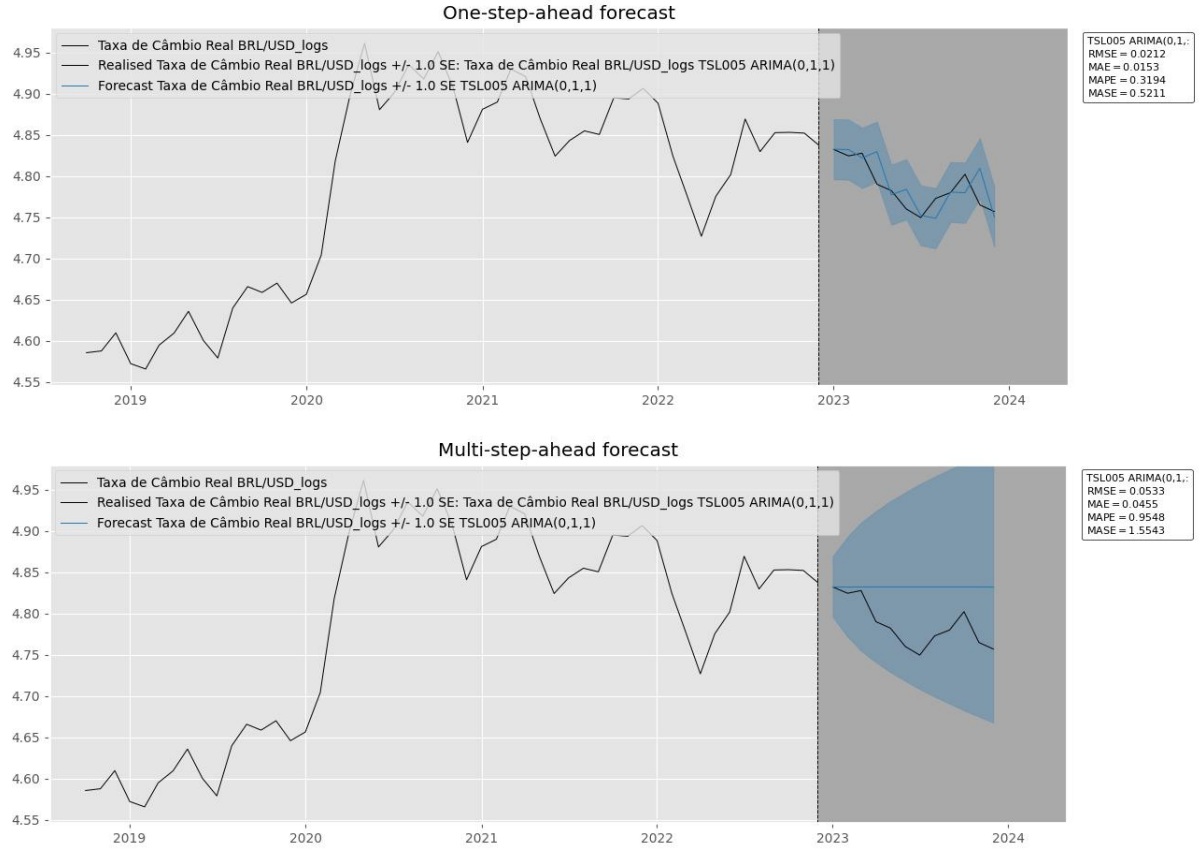
Com base nas informações expostas, há sinais de que os resíduos do modelo exibem características de ruído branco. Embora alguns pontos do resíduo estejam fora do intervalo de confiança, a função de autocorrelação dos resíduos padronizados mostra que a maioria das defasagens estão dentro desse intervalo. Isso indica que não há autocorrelação significativa nos resíduos após padronização. Além disso, os valores de probabilidade (p-valor) associados às estatísticas do teste de Ljung-Box são relativamente altos em todas as defasagens, o que sugere a ausência de autocorrelação nas defasagens dos resíduos. Portanto, é razoável concluir que os

resíduos do modelo se assemelham ao ruído branco, indicando que o modelo está capturando efetivamente toda a estrutura dos dados.

4.3 Outras Abordagens e Previsões

Para as abordagens de ML e DL foram aproveitados alguns dos resultados da modelagem ARIMA. Assim, a partir da série diferenciada, foram aplicadas as arquiteturas. De forma análoga, a série foi dividida entre a amostra de treinamento (2000-2022) e a amostra de teste (previsão, 2023). Além disso, a mesma metodologia foi aplicada ao modelo de regressão linear, representado da seguinte forma: $y'(w,x) = w_0 + w_1x_1 + ... + w_px_p$. Onde x representa os valores amostrais da jenela deslizante de 12 períodos, w_0 o intercepto (alfa)[6] e $w_{1...p}$ os coeficientes (betas)[7]. Destarte, foi possível obter os seguintes resultados:

Gráficos 8 e 9: Previsões ARIMA (0,1,1)

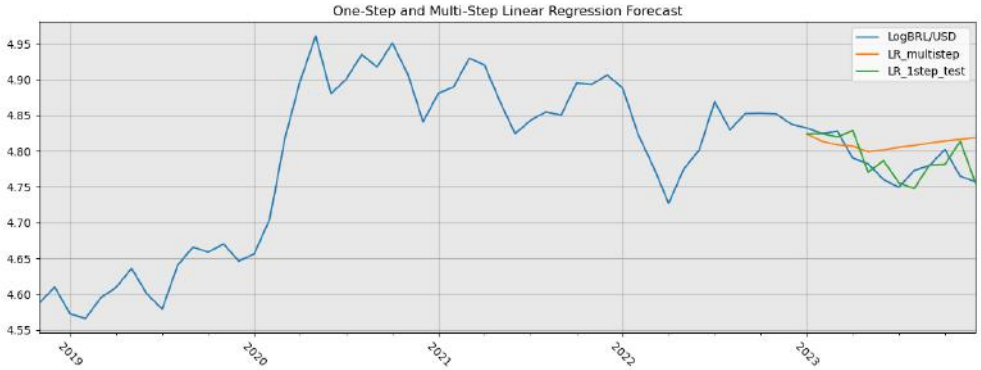


Fonte: Elaboração Própria (software Time Series Lab)

[6] Valor do Intercepto (alfa): 0,00045951128018561946

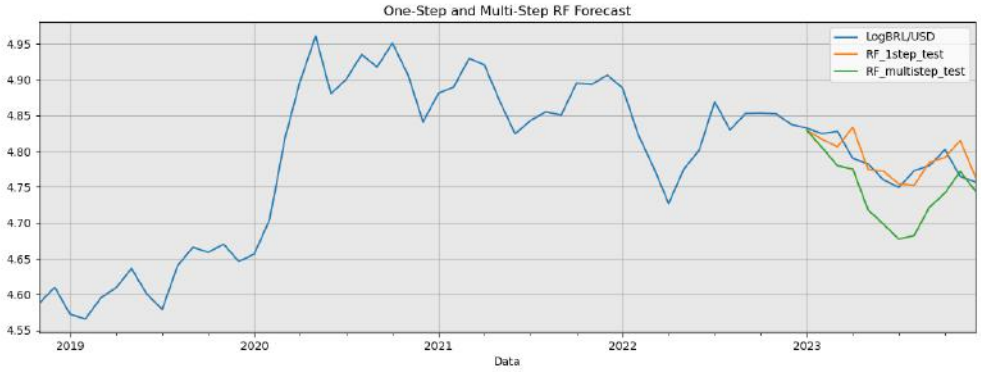
[7] Valor dos Coeficientes (betas): 0,04055387; 0,00336613; -0,07310354; 0,05445855; -0,06643635; -0,1461488; 0,01474138; 0,06694081; -0,00125706; 0,01003482; -0,04136847; 0,3287953

Gráfico 10: Previsões Regressão Linear



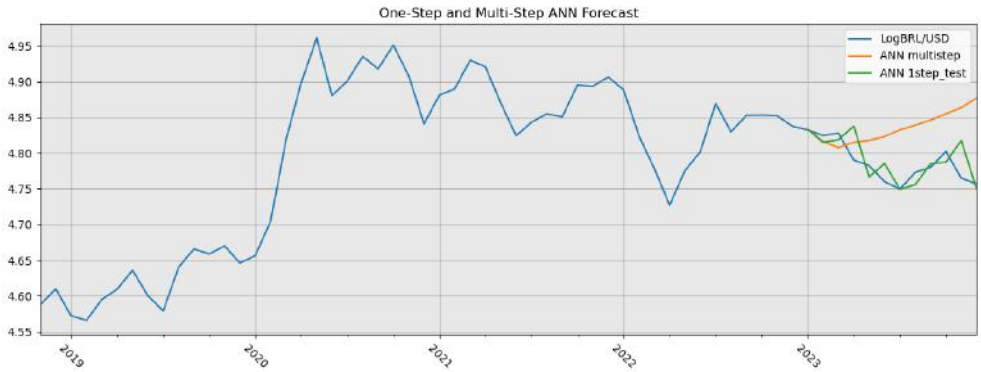
Fonte: Elaboração Própria (software Python)

Gráfico 11: Previsões Random Forest



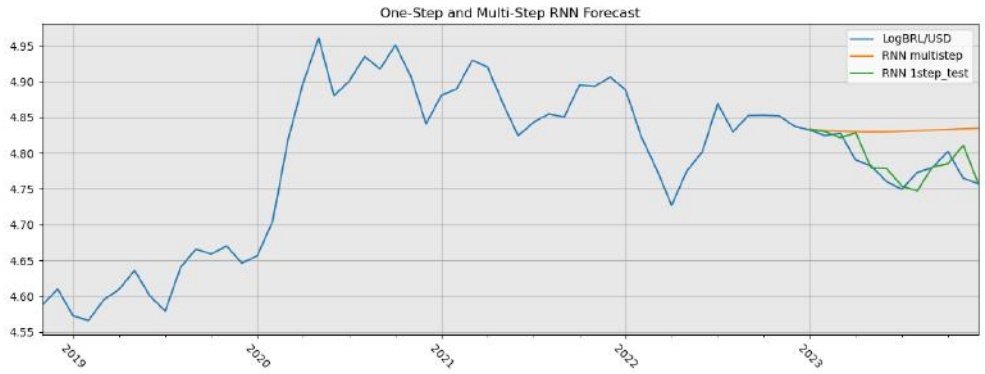
Fonte: Elaboração Própria (software Python)

Gráfico 12: Previsões ANN



Fonte: Elaboração Própria (software Python)

Gráfico 13: Previsões RNN



Fonte: Elaboração Própria (software Python)

Assim como no ARIMA, é necessário examinar a autocorrelação residual. Dito isto, na Tabela 5 constam os resultados do teste de Ljung-Box para dez defasagens dos demais modelos. Seus resultados apontam que a maioria dos modelos apresentam p-valores acima do limite de significância de 5%, o que sugera que não há autocorrelação significativa nos resíduos e que os modelos estão capturando adequadamente a estrutura de correlação nos dados. No entanto, os modelos SVR e CNN apresentaram exceções e, portanto, foram removidos da análise principal e transferidos para o Apêndice II, devido a problemas de especificação. O tratamento das autocorrelações é primordial para garantir a precisão nas previsões dos modelos. Ou seja, é necessário melhoria nesses modelos, o que pode ser alcançado através de ajustes nos hiperparâmetros.

Tabela 5: Teste de Ljung-Box nos Resíduos dos Demais Modelos

Ljung-Box Test	**Random Forest**		**ANN**		**RNN**		**Linear Regression**	
Lags	Estatística	P-valor	Estatística	P-valor	Estatística	P-valor	Estatística	P-valor
1	0,1467	0,7017	0,1591	0,6900	0,5052	0,4772	0,0025	0,9602
2	0,3219	0,8513	0,2423	0,8859	2,8541	0,2400	0,0041	0,9980
3	0,4433	0,9312	0,6795	0,8780	2,9227	0,4037	0,0225	0,9991
4	0,5130	0,9722	0,6900	0,9526	2,9821	0,5608	0,0349	0,9998
5	1,0378	0,9595	0,9190	0,9688	5,2309	0,3884	0,0524	1,0000
6	1,0405	0,9840	0,9542	0,9873	6,3086	0,3895	0,0906	1,0000
7	2,3248	0,9397	1,1475	0,9921	10,4620	0,1639	0,1013	1,0000
8	2,4027	0,9661	1,1834	0,9968	12,7339	0,1213	0,1093	1,0000
9	2,5167	0,9804	1,3812	0,9979	13,3861	0,1459	0,1113	1,0000
10	4,0894	0,9432	1,3820	0,9993	13,7493	0,1847	0,1590	1,0000

Fonte: Elaboração Própria (resultados obtidos no Python)

Outros dados que corroboram com os pontos supracitados são os coeficientes de determinação do modelo de regressão linear (Tabela 6) e as funções de perda da arquitetura RNN (Gráfico 14). As estatísticas do modelo de regressão linear indicam que ele consegue explicar apenas uma parte da variabilidade dos dados de treinamento e não está generalizando bem para novos dados de teste. Isso pode ser atribuído à simplicidade da arquitetura do modelo.

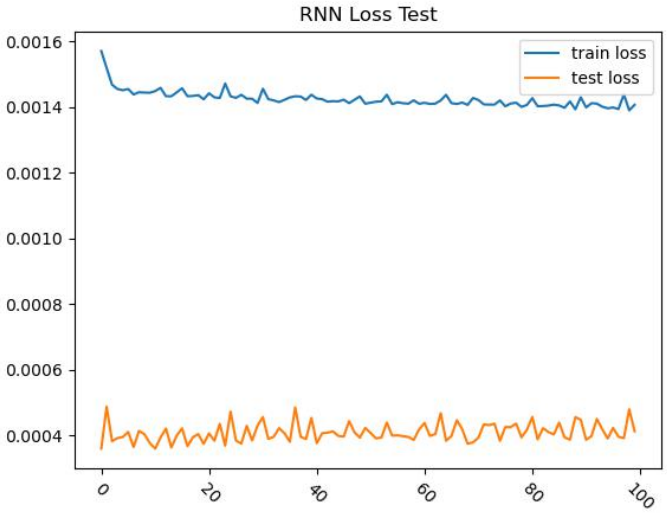
Entretanto, a RNN também apresenta sinais similares, com funções de perda que não convergem, indicando que o modelo não está bem ajustado aos dados e exibe sinais de instabilidade e inconsistência, além de um desequilíbrio entre viés e variância. Por outro lado, a arquitetura de ANN não apresenta o mesmo resultado insatisfatório observado na RNN, conforme exposto nas Gráficos 15.

Tabela 6 – Coeficientes de Determinação do Modelo de Regressão Linear

	Coef. de Det. Regressão Linear
Amostra de Treino	0,1421
Amostra de Teste	-0,4182

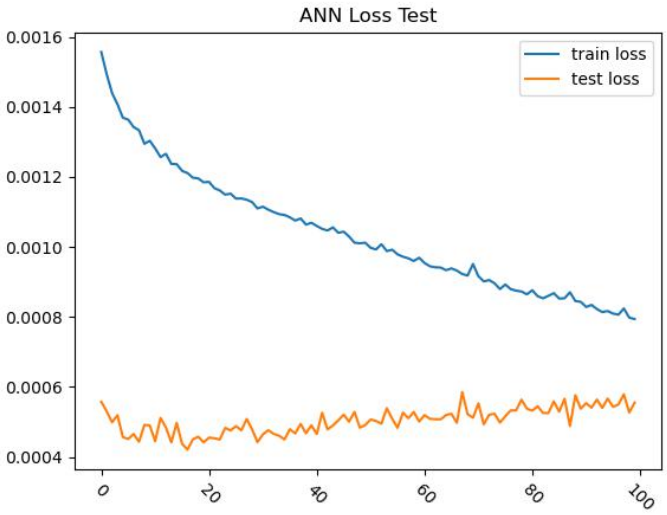
Fonte: Elaboração Própria (resultados obtidos no Python)

Gráfico 14: Função de Perda RNN (MSE)



Fonte: Elaboração Própria (software Python)

Gráfico 15: Função de Perda ANN (MSE)



Fonte: Elaboração Própria (software Python)

4.4 Análise dos Resultados Preditivos

Nesta seção, são discutidos os resultados do modelo ARIMA em comparação com as arquiteturas de *Machine e Deep Learning*. Como mencionado anteriormente, o modelo de regressão linear foi utilizado como referência, empregando uma janela deslizante de 12 períodos, juntamente com as métricas de avaliação RMSE, MAE e MAPE.

A Tabela 7 apresenta os resultados dessas métricas, fornecendo *insights* valiosos sobre os modelos desenvolvidos. O modelo de regressão linear obteve resultados preditivos notáveis, especialmente em previsões *multi-step*, com as melhores métricas (RMSE: 0,0351; MAE: 0,0301; MAPE: 0,6300), além de superar os modelos RF e ANN em todos os aspectos. Por outro lado, o modelo RNN destacou-se com os melhores resultados em previsões *one-step* (RMSE: 0,0203; MAE: 0,0139; MAPE: 0,2900).

Tabela 7: Métricas de Avaliação

Forecast	ARIMA (0,1,1)		Random Forest		ANN		RNN		Linear Regression	
	One-Step	Multi-Step	One-Step	Multi-Step	One-Step	Multi-Step	One-Step	Multi-Step	One-Step	Multi-Step
RMSE	0,0212	0,0533	0,0225	0,0861	0,0236	0,0637	0,0203	0,0525	0,0222	0,0351
MAE	0,0153	0,0455	0,0168	0,0743	0,0171	0,0532	0,0139	0,0447	0,0163	0,0301
MAPE	0,3194	1,5543	0,3516	1,5559	0,3573	1,1155	0,2900	0,9365	0,3420	0,6300

Legenda:

Melhor performance entre os modelos testados.

Performance inferior ao modelo de Regressão Linear.

Fonte: Elaboração Própria (resultados obtidos no Python)

No entanto, é importante notar que o modelo ARIMA não ficou atrás em relação às arquiteturas mais complexas de aprendizado supervisionado. Na prática, ele obteve resultados bastante próximos à arquitetura RNN no quesito RMSE (0,0212) um passo à frente. Para mais, ele foi capaz de superar os demais modelos nas métricas MAE (0,0153) e MAPE (0,3194). Fica

nítido, portanto, que o modelo ARIMA bem ajustado é capaz de produzir resultados similares e até superiores às arquiteturas *naive* de *Machine* e *Deep Learning*.

Para além, é basilar traçar um paralelo com outras evidências empíricas, afim de verificar a consistência dos resultados obtidos. Segundo Coelho et al. (2008), modelos de redes neurais se destacam pela capacidade de lidar com níveis elevados de não-linearidade, um aspecto acentuado em séries de maior frequência. Nesse sentido, a amostra mensal testada neste trabalho corroborou para um melhor desempenho do modelo ARIMA ajustado.

Ademais, além do estudo de Coelho et al. (2008), De Oliveira et al. (2020) também utilizou um modelo GARCH, para correção da autocorrelação residual, modelagem e previsão da volatilidade. Entretanto, optou-se por não utilizar tal estrutura neste trabalho, tendo em mente os resultados do teste de Ljung-Box do modelo ARIMA (Tabela 4).

5 CONSIDERAÇÕES FINAIS

Conforme exposto por De Oliveira et al. (2020), é evidente que, no contexto econômico atual, a taxa de câmbio real é uma variável de extrema importância. É o centro das atenções e preocupações dos agentes econômicos, sendo crucial para suas decisões. Além disso, a sua volatilidade pode gerar consequências adversas nas transações econômicas e financeiras, afetando diretamente a dinâmica e o desenvolvimento econômico, especialmente em países emergentes.

O presente trabalho buscou, além de comparar performances preditivas, introduzir um arcabouço contemporâneo de modelos via arquiteturas de aprendizado supervisionado. Ele expressa o paradigma hodierno entre métodos estatísticos tradicionais e técnicas modernas de *Machine Learning* para modelagem e previsão de séries temporais. Uma ideia para o futuro é retomar este debate aumentando o escopo de frequências analisadas, com séries semanais, diárias e até *intraday*.

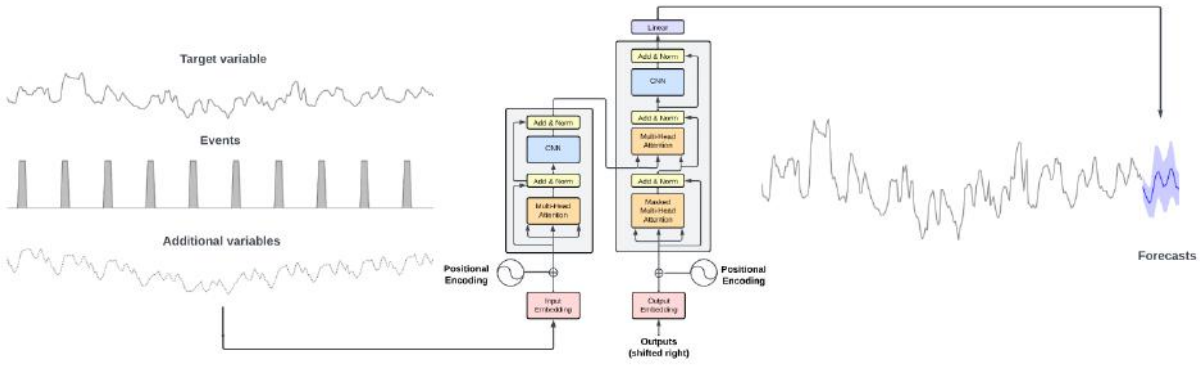
Isto posto, através dos resultados apresentados, pode-se concluir que o modelo ARIMA demonstrou ser de grande utilidade. Além de obter estatísticas pertinentes para análise, foi capaz de produzir previsões relevantes frente aos modelos *naive* de *Machine* e *Deep Learning*. Todavia, é fundamental comentar que para alcançar o "estado da arte" na modelagem de séries temporais é necessário grande conhecimento sobre os modelos, permitindo um ajuste eficaz dos hiperparâmetros, para que se possa obter máximo resultado dessas complexas estruturas.

APÊNDICE I

Olhando para os futuros desenvolvimentos no campo, é explícito que os avanços recentes com o TimeGPT-1 (Garza e Mergenthaler-Canseco, 2023) e o MOIRAI (Woo et al, 2024), que combinam métodos estatísticos com técnicas de *Machine Learning* e *Deep Learning*, estão moldando o futuro da modelagem e previsão de séries temporais. Essas abordagens introduzem a inteligência artificial generativa através de modelos pré-treinados em larga escala, dando origem ao conceito de "Modelo de Previsão Universal". A premissa subjacente é que um modelo suficientemente grande e previamente treinado pode efetivamente prever dados não observados.

O TimeGPT-1 é um modelo de previsão de séries temporais que baseado na arquitetura *Transformer* e utiliza mecanismos de "autoatenção", inspirados no trabalho de Vaswani et al. (2017). De acordo com Garza e Mergenthaler-Canseco (2023), sua estrutura é composta por um conjunto de camadas codificadoras e decodificadoras, cada uma equipada com conexões residuais e normalização de camada. Após o processamento dos dados, uma camada linear é aplicada para adaptar a saída do decodificador ao tamanho da janela de previsão. A premissa é que os mecanismos de "atenção" sejam capazes de capturar a diversidade de eventos passados e fazer projeções precisas sobre possíveis distribuições futuras. De forma simplificada, um modelo *Transformer* com mecanismos de "autoatenção" é uma arquitetura de rede neural que possibilita que o modelo foque sua capacidade de processamento em partes específicas (mais relevantes) dos dados, conforme apresentado na Figura 4.

Figura 4: Estrutura do TimeGPT-1

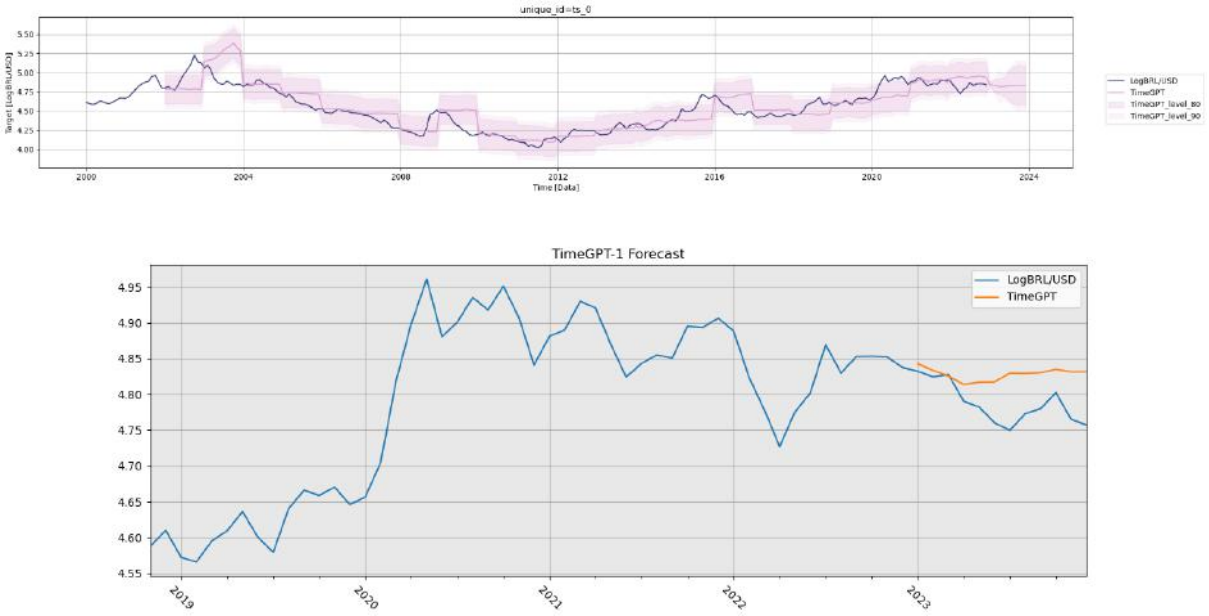


Fonte: Garza e Mergenthaler-Canseco, 2023: "TimeGPT-1"

Visando vislumbrar tais avanços, testou-se um modelo *naive* de TimeGPT-1 para série da taxa de câmbio real (BRL/USD) em nível, conforme Gráficos 16 e 17. É possível observar que o teste de Ljung-Box (Tabela 8) apontou autocorrelação em todos os *lags*, indicando que

há dependência serial nos dados. Além disso, conforme apontado pela Tabela 9, ele também não foi capaz de superar os resultados preditivos *multi-step* do modelo de regressão linear. No entanto, de acordo com a coluna "*2nd Bests*", que exibe os segundos melhores resultados *multi-step*, o TimeGPT-1 superou a RNN na métrica de RMSE (0,0487) e obteve as terceiras melhores métricas de MAE (0,0415) e MAPE (0,8699), ficando atrás apenas do modelo CNN entre todos os modelos testados, inclusive os que apresentaram problema de especificação.

Gráficos 16 e 17: Previsão TimeGPT-1



Fonte: Elaboração Própria (software Python)

Tabela 8: Teste de Ljung-Box TimeGPT-1

Ljung-Box Test	**Time-GPT**	
Lags	Estatística	P-valor
1	202,0486	7,46E-46
2	348,0875	2,59E-76
3	439,8831	5,07E-95
4	492,4515	2,87E-105
5	517,4283	1,38E-109
6	524,6466	4,12E-110
7	524,9590	3,44E-109
8	526,4230	1,50E-108
9	533,1669	4,52E-109
10	547,5630	2,98E-111

Fonte: Elaboração Própria (resultados obtidos no Python)

Tabela 9: Métricas de Avaliação TimeGPT-1 (*Multi-Step*)

Forecast	Time-GPT	LR Model	2nd Bests
RMSE	0,0487	0,0351	0,0525
MAE	0,0415	0,0301	0,0380
MAPE	0,8699	0,6300	0,7958

Melhor performance entre os modelos testados.**

**com exceção do modelo de Regressão Linear.

Fonte: Elaboração Própria (resultados obtidos no Python)

Com base nos dados apresentados, assim como evidenciado por Garza e Mergenthaler-Canseco (2023), fica evidente que o TimeGPT-1 representa uma oportunidade excepcional para democratizar o acesso a previsões precisas e reduzir a incerteza, tirando proveito dos avanços contemporâneos em *Deep Learning*. No entanto, apesar do progresso alcançado, desafios significativos ainda persistem, como a necessidade de lidar com a implementação arbitrária de variáveis de ajuste e a correlação serial dos resíduos. Contudo, é justamente na superação desses desafios que emergem as oportunidades, e é isso que mantém a relevância contínua dos pesquisadores neste campo em constante evolução.

APÊNDICE II

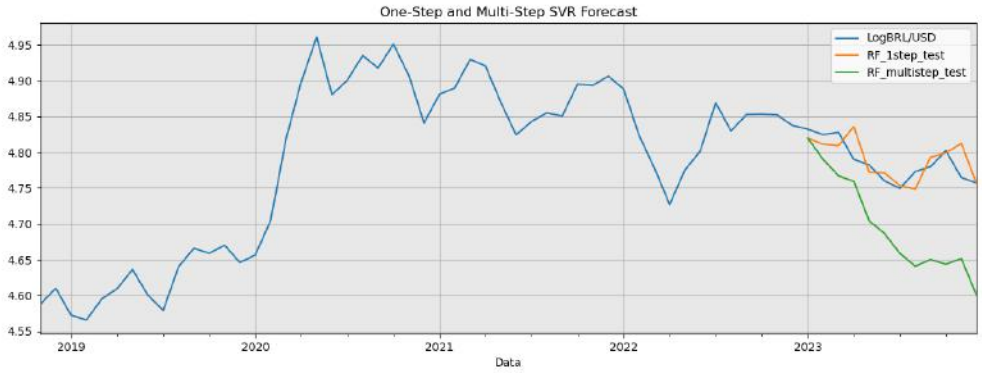
Esta sessão é destinada aos modelos com problemas de especificação. Ou seja, foram rejeitados no teste de Ljung-Box (Tabela 10), apresentando correlação serial nos resíduos. Os Gráficos 18 e 19 exibem as previsões do modelo SVR e do CNN. Entretanto, é importante notar que eles obtiveram resultados preditivos distintos apesar da autocorrelação nos resíduos. Com base na Tabela 11, por um lado o SVR obteve os piores resultados preditivos, enquanto o CNN exibe métricas similares aos outros modelos de redes neurais.

Tabela 10: Teste de Ljung-Box SVR e CNN

Ljung-Box Test	**SVR**		**CNN**	
Lags	Estatística	P-valor	Estatística	P-valor
1	17,9390	2,3E-05	1,2704	0,2597
2	19,3342	6,3E-05	15,5747	0,0004
3	20,8477	0,00011	15,8686	0,0012
4	22,5712	0,00015	16,4660	0,0025
5	26,2832	7,9E-05	16,5232	0,0055
6	29,0013	6,1E-05	17,4535	0,0078
7	30,7319	0,00007	17,5900	0,0140
8	31,9594	9,5E-05	17,9635	0,0215
9	32,1367	0,00019	18,4257	0,0305
10	32,2762	0,00036	19,4589	0,0348

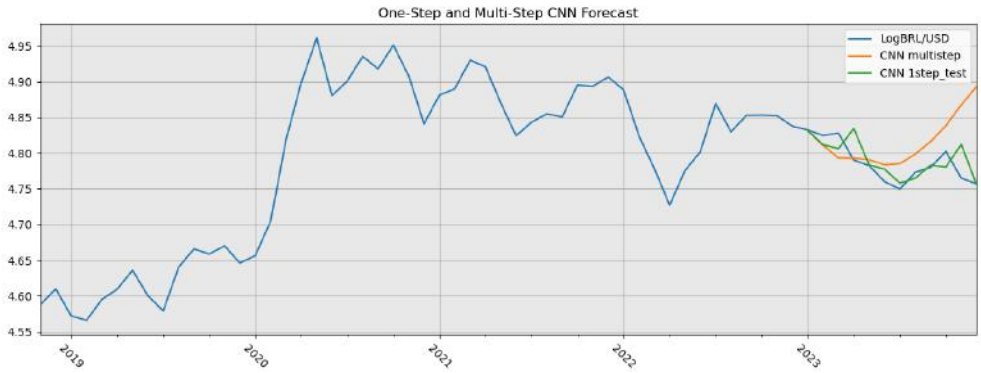
Fonte: Elaboração Própria (software Python)

Gráfico 18: Previsões Support Vector Regressor



Fonte: Elaboração Própria (software Python)

Gráfico 19: Previsões CNN



Fonte: Elaboração Própria (software Python)

Tabela 11: Métricas de Avaliação

Forecast	SVR		CNN	
	One-Step	Multi-Step	One-Step	Multi-Step
RMSE	0,0382	0,1651	0,0218	0,0545
MAE	0,0334	0,1497	0,0156	0,0380
MAPE	0,6974	3,1351	0,3252	0,7958

Legenda:

Performance inferior ao modelo de Regressão Linear.

Fonte: Elaboração Própria (resultados obtidos no Python)

REFERÊNCIAS

1. **BANCO CENTRAL DO BRASIL (BACEN)**. Índice da taxa de câmbio real (IPCA) - Jan/2000=100 - Dólar americano. Sistema de Séries Temporais. Disponível em: https://www.bcb.gov.br/
2. BORGES, T. A.; NEVES, R. F. **Ensemble of machine learning algorithms for cryptocurrency investment with different data resampling methods**. Applied Soft Computing, v. 90, p. 106187, 2020.
3. BREIMAN, L. **Machine learning**, Random forests v. 45, p. 5-32, 2001.
4. BRESSER-PEREIRA, L. C. **A taxa de câmbio no centro da teoria do desenvolvimento**. 4. ed. São Paulo: Editora 34, 2012.
5. COELHO, L. S.; SANTOS, A. A. P.; COSTA JR, N. C. A. **Podemos prever a taxa de câmbio brasileira? Evidência empírica utilizando inteligência computacional e modelos econométricos.** Gestão & Produção, v. 15, p. 635-647, 2008.
6. CRISTIANINI, N.; SHAWE-TAYLOR, J. **Intelligent Data Analysis**: An Introduction (Support vector and kernel methods**).** Berlin, Heidelberg: Springer Berlin Heidelberg, p. 169-197. 2007.
7. DE OLIVEIRA, B. M.; RODRIGUES, L. L. M.; FRANCHINI, A. A. **Modelagem da Taxa de Câmbio Real no Brasil**: Uma Aplicação ARIMA-GARCH para o Período 2010-2018.
8. DERBENTSEV, V. et al. **Machine learning approaches for financial time series forecasting**. CEUR Workshop Proceedings, 2020.
9. DERBENTSEV, V.; DATSENKO, N.; STEPANENKO, O.; BEZKOROWAINYI, V. **Forecasting Cryptocurrency Prices Time Series Using Machine Learning**. CEUR Workshop Proceedings 2422, 320–334 (2019).
10. GAMBOA, J. C. B. **Deep learning for time-series analysis**. arXiv preprint arXiv:1701.01887 (2017).
11. GARDNER, M. W.; DORLING, S. **Artificial neural networks(the multilayer perceptron)**—a review of applications in theatmospheric sciences.Atmospheric Environment,32(14-15),2627–2636. (1998).
12. GARZA, A.; MERGENTHALER-CANSECO, M. **TimeGPT-1**. arXiv preprint arXiv:2310.03589, 2023.
13. GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. MIT Press, 2016.

14. HAMID, S.A.; HABIB, A. **Financial Forecasting with Neural Networks**. Academy of Accounting and Financial Studies Journal 18(4), 37–55 (2014).

15. HYNDMAN, R. J. **ARIMA processes**. Datajobs: Data science knowledge, 2001.

16. LECUN, Y.; BENGIO, Y.; HINTON, G. **Deep learning.** nature, v. 521, n. 7553, p. 436-444, 2015.

17. MORETTIN, P. A.; TOLOI, C. M. C. **Análise de Séries Temporais**, 2.ed., São Paulo: Edgard Blucher, 2004.

18. **Python**: software (bibliotecas: *sklearn, tensorflow e timegpt)*.

19. ROCHA, M.; CURADO, M.; DAMIANI, D. **Taxa de câmbio real e crescimento econômico**: uma comparação entre economias emergentes e desenvolvidas. Brazilian Journal of Political Economy, v. 31, p. 528-550, 2011.

20. SANTOS, V. **Séries Temporais**: ARIMA. Universidade Federal do Pará - Instituto de Ciências Exatas e Naturais, Belém, Pará, 2014.

21. SICSÚ, J. **Taxa de câmbio dentro de uma estratégia de desenvolvimento**. Econômica, v. 11, n. 1, 2009.

22. STAUDEMEYER, R. C.; MORRIS, E. R**. Understanding LSTM**: a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:1909.09586, 2019.

23. **Time Series Lab**: software.

24. VASWANI, A. et al. **Attention is all you need**: Advances in neural information processing systems, v. 30, 2017.

25. WOO, G. et al. **Unified training of universal time series forecasting transformers**. arXiv preprint arXiv:2402.02592, 2024.