UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ABORDAGENS PARA PREDIÇÃO DO NÚMERO DE CASOS SEMANAIS DE DENGUE EM REGIÕES TROPICAIS

Manaus - AM
Setembro de 2024

GIOVANNI ESCÓSSIO ZANARDO

ABORDAGENS PARA PREDIÇÃO DO NÚMERO DE CASOS SEMANAIS DE DENGUE EM REGIÕES TROPICAIS

Dissertação apresentada ao Programa de Pós-Graduação em Mestrado em Informática do Instituto de Computação da Universidade Federal do Amazonas, Campus Universitário Senador Arthur Virgílio Filho, como requisito parcial para a obtenção do grau de Mestre em Mestrado em Informática.

Orientador: Eduardo Freire Nakamura

Manaus - AM

Setembro de 2024

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Zanardo, Giovanni Escóssio
Z27a Abordagens para predição do número de casos semanais de dengue em Regiões Tropicais / Giovanni Escóssio Zanardo . 2024
101 f.: il. color; 31 cm.

Orientador: Eduardo Freire Nakamura
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Dengue. 2. Séries temporais. 3. Aprendizado de máquina. 4. Wavelets. I. Nakamura, Eduardo Freire. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"IABORDAGENS PARA PREDIÇÃO DO NÚMERO DE CASOS SEMANAIS DE DENGUE EM REGIÕES TROPICAIS"

GIOVANNI ESCÓSSIO ZANARDO

DISSERTAÇÃO DE MESTRADO DEFENDIDA E APROVADA PELA BANCA EXAMINADORA CONSTITUÍDA PELOS PROFESSORES:

Prof. Dr. Eduardo Freire Nakamura - PRESIDENTE

Prof. Dr. Efrén Lopes de Souza - MEMBRO EXTERNO

Prof. Dr. Juan Gabriel Colonna - MEMBRO INTERNO

MANAUS, 27 de setembro de 2024.

Documento assinado eletronicamente por **Eduardo Freire Nakamura**, **Professor do Magistério Superior**, em 18/10/2024, às 17:59, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Juan Gabriel Colonna**, **Professor**

do Magistério Superior, em 21/10/2024, às 11:04, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Efren Lopes de Souza**, **Usuário Externo**, em 22/10/2024, às 10:24, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2278097** e o código CRC **9E1B1A67**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Referência: Processo nº 23105.041601/2024-85 SEI nº 2278097

Agradecimentos

Agradeço inicialmente à minha família. Sem o suporte de todos eles, dificilmente teria chegado até aqui.

Agradeço ao professor Leandro Galvão por ter me incentivado a fazer o mestrado no PPGI desde o momento em que o contatei com esse objetivo, oferecendo conselhos, direcionamentos e referências bibliográficas para estudo.

Agradeço ao professor Eduardo Nakamura por aceitar ser meu orientador, pelos seus conselhos profissionais e pessoais, suas excelentes indicações bibliográficas e orientações que me permitiram desenvolver um pensamento mais crítico, apaixonado e amplo pela Ciência, tornando-me verdadeiramente um pesquisador. Agradeço à professora Fabíola pelas sugestões, orientações em relação ao trabalho e pelas oportunidades de atuar em tutoria acadêmica, o que me proporcionou ganhar cada vez mais experiência em ensino. Agradeço ao professor Éfren pela troca de experiências e sugestões para este trabalho e na produção de artigos. Agradeço ao professor Juan por sugerir a experimentação de modelos utilizados neste trabalho, por ter disponibilizado o uso do Laboratório de Aprendizagem de Máquina e pela dedicada prestação de orientações e informações durante sua época como coordenador do PPGI.

Agradeço aos meus excepcionais professores do PPGI: ao professor Altigran, pela dinâmica das aulas de Projeto e Análise de Algoritmos que estimulavam o verdadeiro estudo; à professora Eulanda, por compartilhar sua experiência e conhecimento em Aprendizado de Máquina; e ao professor Eduardo Souto, pelas valiosas dicas de escrita na matéria de Processo de Pesquisa em Ciência da Computação.

Agradeço às amizades que o mestrado me proporcionou: ao Daniel Gohl, ao Richard, ao Yuto, ao Andrés, à Yunevda e ao Oziel. Pela maior experiência que possuíam, eles foram uma fonte de inspiração para que eu me tornasse um profissional cada vez mais imerso na área de Ciência da Computação.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POSGRAD.

Resumo

As arboviroses transmitidas pelos mosquitos *Aedes aegypti* e *Aedes albopictus* representam um dos principais desafios de saúde pública, sendo a dengue a mais destacada. O manejo de epidemias de dengue requer preparação avançada; assim, predizer o número semanal de casos em uma região específica pode auxiliar nas estratégias de prevenção e controle do processo epidêmico. Neste estudo, avaliamos a eficácia de técnicas estatísticas clássicas e métodos de aprendizado de máquina na predição do número de casos semanais de dengue, utilizando dados geográficos de San Juan, Porto Rico, e das 27 unidades federativas brasileiras. Em San Juan, selecionamos as características baseando-nos na matriz de correlação cruzada com o número total de casos semanais de dengue e aplicamos transformações *wavelet* na característica selecionada, onde o modelo de Regressão Linear (LR), que utiliza níveis de precipitação e índices de vegetação filtrados pela *wavelet* Symmlet (sym20), apresentou os melhores resultados em métricas como MAE, R^2, MAPE, RMSE e BIAS, mostrando redução em 67,22% de RMSE se comparados ao LR univariado sem filtragem por *wavelets*. Para as unidades federativas brasileiras, um procedimento semelhante foi realizado, mas sem o uso da matriz de correlação cruzada, sendo o modelo LightGBM univariado, treinado em 26 cidades (*cross-learning*) e validado individualmente em cada unidade federativa através da técnica univariada utilizando *leave-one-out* e predições *one-step*, o que demonstrou superioridade tanto nas validações individuais quanto em comparação com a localidade geográfica de San Juan, onde o modelo não foi treinado, evidenciando uma melhor generalização dos resultados em comparação com outros modelos rasos, profundos e fundadores como TimeGPT-1 e MOIRAI.

Palavras-chave: Dengue, Séries temporais, Aprendizado de máquina, Wavelets.

Abstract

Arboviruses transmitted by *Aedes aegypti* and *Aedes albopictus* mosquitoes represent one of the major public health challenges, with dengue being the most prominent. Managing dengue epidemics requires advanced preparation; thus, predicting the weekly number of cases in a specific region can aid in prevention and control strategies for the epidemic process. In this study, we evaluated the effectiveness of classical statistical techniques and machine learning methods in predicting the number of weekly dengue cases, using geographic data from San Juan, Puerto Rico, and the 27 Brazilian federative units. In San Juan, we selected features based on the cross-correlation matrix with the total number of weekly dengue cases and applied *wavelet* transformations to the selected feature, where the Linear Regression (LR) model, using precipitation levels and vegetation indices filtered by the Symmlet *wavelet* (sym20), presented the best results in metrics such as MAE, R^2, MAPE, RMSE, and BIAS, showing a 67.22% reduction in RMSE compared to the univariate LR without wavelet filtering. For the Brazilian federative units, a similar procedure was carried out, but without using the cross-correlation matrix. The univariate LightGBM model, trained on 26 cities (*cross-learning*) and validated individually in each federative unit through the univariate *leave-one-out* technique and *one-step* predictions, demonstrated superiority both in individual validations and in comparison with the geographic locality of San Juan, where the model was not trained, evidencing better generalization of results compared to other shallow, deep, and founding models such as TimeGPT-1 and MOIRAI.

Keywords: Dengue, Time series, Machine learning, Wavelets.

Lista de Figuras

Lista de Tabelas

Sumário

1

Introdução

Arbovírus são vírus transmitidos pela picada de artrópodes hematófagos [da Silva & Angerami, 2008]. Temperatura, ciclos hidrológicos e umidade são fatores chave na introdução e disseminação de cada arbovírus em diferentes ambientes [Celentano et al., 2008]. Geralmente, países tropicais favorecem o desenvolvimento e proliferação de vetores, aumentando a incidência dessas doenças nessas localidades [Lopes et al., 2014]. Entre os arbovírus, a dengue é proeminente.

A dengue é causada pelo vírus da dengue, um arbovírus da família Flaviviridae, gênero Flavivirus, e possui quatro tipos imunológicos: DENV-1, DENV-2, DENV-3 e DENV-4 [Ross, 2010]. Em poucos casos, a dengue pode progredir para uma forma grave, aumentando significativamente o risco de morte. Esta etapa da doença é marcada por sangramento, redução nos níveis de plaquetas sanguíneas, vazamento de plasma e queda da pressão sanguínea para níveis perigosamente baixos [Ross, 2010]. Atualmente, a dengue é o arbovírus mais comum que afeta a humanidade, responsável por cerca de 100 milhões de casos/ano em uma população em risco de 2,5 a 3 bilhões de pessoas [World Health Organization, 2009]. Historicamente, a dengue tem recebido atenção das agências de saúde, como evidenciado pela resolução WHA58.3 da Assembleia Mundial da Saúde em 2005, referente à revisão dos Regulamentos Sanitários Internacionais (RSI), que inclui a dengue como um exemplo de doença que constitui uma emergência de saúde pública de interesse internacional [World Health Organization, 2009], com implicações para a segurança sanitária devido à sua capacidade de causar interrupções e rápida disseminação epidêmica além das fronteiras nacionais.

A disseminação epidêmica é verificada por meio de um aumento significativo na incidência global de dengue nas últimas duas décadas, representando um obstáculo à saúde pública. Entre 2000 e 2019, a Organização Mundial da Saúde (OMS) documentou um aumento de dez vezes nos casos relatados em todo o mundo, de

500.000 para 5,2 milhões [World Health Organization, 2023]. O ano de 2019 registrou um pico sem precedentes, com casos relatados em 129 países, seguido por uma leve queda nos casos entre 2020 e 2022, devido à pandemia de COVID-19 e à menor taxa de notificação. No entanto, o segundo semestre de 2023 testemunhou um aumento alarmante nos casos em todo o mundo. Esse aumento é caracterizado pelo crescimento significativo no número, escala e ocorrência simultânea de múltiplos surtos, expandindo-se para regiões anteriormente não afetadas pela dengue, de modo que o total acumulado de casos para 2023 excedeu todos os totais anuais anteriores e, em alguns países, a transmissão se estendeu além das áreas afetadas conhecidas (América do Sul, México, América Central e países do Caribe). Por exemplo, a dengue está se espalhando pela Europa devido às mudanças climáticas [Laverdeur et al., 2024], que aumentam as temperaturas e alteram os padrões de precipitação, possibilitando a reprodução dos vetores da dengue. Projeções baseadas em modelagem de nicho ecológico indicam um aumento nas áreas de adequação climática para A. albopictus nas partes central e ocidental da Europa até 2040 [Carvalho et al., 2016, Kamal et al., 2018, Laporta et al., 2023]. Portanto, nos próximos anos, a dengue se tornará uma preocupação significativa para os sistemas de saúde dos países europeus. Nas Américas, os casos de dengue aumentaram ao longo das últimas quatro décadas, de 1,5 milhão de casos de 1980 a 1989 para 17,5 milhões em 2010-2019. Antes de 2023, o maior número histórico de casos de dengue foi registrado em 2019, com mais de 3,18 milhões de casos, 28.208 casos graves e 1.823 mortes, resultando em uma taxa de fatalidade de 0,06 % [World Health Organization, 2023]. Especificamente, no Brasil, o número de casos notificados nas primeiras 23 semanas de 2024 foi de 7.866.769 casos, representando um aumento de 230% em relação ao mesmo período de 2023 [OPAS & OMS, 2024]. A taxa de incidência acumulada desde a primeira semana do ano é de 3.676 casos por 100.000 habitantes, com 5.210 casos (0,07%) caracterizados como casos graves e 3.643 como casos fatais.

Dessa forma, a dengue é uma doença de preocupação global. Estimar o número de casos de dengue em uma determinada localidade fornece uma base sólida para a tomada de decisão por parte das agências sanitárias, contribuindo para a melhor alocação de recursos financeiros destinados ao combate da doença e para a redução da mortalidade associada.

1.1 Problema

Diferenças climáticas são usadas para explicar a sazonalidade da dengue. No entanto, essa sazonalidade pode ser influenciada por múltiplos fatores demográficos,

como taxas de natalidade, imigração e mobilidade de curto prazo, bem como pela co-circulação de diferentes formas virais, complicando o uso de abordagens estatísticas tradicionais no cenário de previsão de surtos [San Martin et al., 2010]. Assim, entender e prever surtos de dengue é desafiador, devido à interação complexa entre determinantes ambientais—fatores que afetam a saúde e o bem-estar humano—e seus respectivos ambientes [Cabrera et al., 2022]. Essa dinâmica culmina em variações nos padrões de incidência da doença em diferentes regiões geográficas.

Na Região das Américas, o número de casos de dengue notificados durante os primeiros seis meses de 2024 superou o maior número de casos notificados em um único ano de todos os anos anteriores registrados. Conforme Organização Pan-Americana da Saúde / Organização Mundial da Saúde [2024], até a semana epidemiológica (SE) 23 de 2024, 43 países e territórios da Região das Américas reportaram 9.386.082 casos de dengue, o dobro do número de casos notificados durante todo o ano de 2023, que foi de 4.617.108 casos.

A dengue é predominantemente encontrada em regiões tipicamente tropicais, onde as condições de umidade e temperatura favorecem a reprodução dos mosquitos transmissores. Ademais, o Brasil responde por 82,4% (3643 mortes) dos casos fatais nas Américas, liderando o índice de mortalidade por dengue no continente americano [Organização Pan-Americana da Saúde / Organização Mundial da Saúde, 2024]. Assim, observa-se um aumento constante no número de casos de dengue no mundo, apesar do conhecimento das medidas de prevenção contra o vírus. A ausência de ferramentas eficazes para a predição prévia do número de casos de dengue revela-se como uma problemática significativa, uma vez que tais ferramentas possibilitam o planejamento prévio e a prevenção de surtos de dengue.

No entanto, o número de casos de dengue interage de forma complexa com variáveis ambientais. Por isso, devido a essa interação complexa entre o número de casos de dengue, representados como uma série temporal, e variáveis ambientais, os modelos de aprendizado de máquina emergem como ferramentas essenciais para a predição de séries temporais.

Recentemente, modelos de aprendizado profundo baseados em arquiteturas de codificador-decodificador com mecanismos de autoatenção, como Transformers [Vaswani et al., 2023], que se enquadram no paradigma de predição universal e são conhecidos como modelos fundadores [Bommasani et al., 2021], pré-treinados em grandes bancos de dados de vários domínios como saúde, economia e vendas para predizer séries temporais genéricas, surgiram como alternativas para predição do número semanal de casos de dengue. Dessa maneira, modelos pré-treinados como TimeGPT-1 [Garza et al., 2023] e MOIRAI [Woo et al., 2024] usam aprendizado

zero-shot [Rezaei & Shahidi, 2020] no processo de predição. No entanto, esses tipos de modelos carecem de ajuste de hiperparâmetros e desconsideram a influência de outras características do conjunto de dados a priori, concentrando todos os seus esforços em predizer apenas a série temporal de interesse. Estudos na área de predição de séries temporais [Elsayed et al., 2021, Makridakis et al., 2020, 2022] têm mostrado que modelos de aprendizado profundo muitas vezes não são a melhor opção, com modelos como LightGBM [Ke et al., 2017] e XGBoost [Chen & Guestrin, 2016] obtendo superioridade preditiva.

Atualmente, a literatura carece [Buczak et al., 2018, Necesito et al., 2021, Panja et al., 2023] de uma análise comparativa entre modelos estatísticos e modelos de aprendizado de máquina com dados pré-processados por diferentes estratégias de janelamento [Benidis et al., 2022], seguidos por técnicas de filtragem, como a filtragem por *wavelets*. Além disso, há a ausência de uma metodologia padrão ao abordar este problema de predição. Dessa maneira, esta carência motivou a (a) implementar métodos de predição de séries temporais e suas possíveis combinações, (b) comparar estes métodos e (c) avaliar sua eficiência de acordo com o ajuste do resultado do modelo em relação à curva do número semanal de casos de dengue e seu desempenho em métricas.

1.2 Objetivos

O principal objetivo deste estudo é determinar, dentre os modelos do estado-da-arte, qual possui maior eficácia na predição do número de casos semanais de dengue em San Juan e nas 27 unidades federativas do Brasil. Visando este propósito, definimos os seguintes objetivos específicos:

1. Demonstrar que o uso de aprendizado cruzado (*cross-learning*) têm a capacidade de extrair conhecimento de múltiplas séries temporais, aproveitando padrões comuns para superar limitações como disponibilidade de dados e incerteza;

2. Identificar características relevantes para predição do número de casos semanais de dengue;

3. Obter, dentre os modelos analisados, o melhor modelo por região para predição eficaz do número de casos semanais de dengue.

1.3 Contribuições

Neste trabalho, buscamos estratégias eficientes para a predição do número de casos semanais de dengue em San Juan e nas unidades federativas brasileiras, comparando o desempenho de modelos estatísticos clássicos e modelos de aprendizado de máquina.

O pré-processamento das bases de San Juan e InfoDengue seguiu etapas semelhantes. Na base de San Juan, foram selecionadas características relevantes por meio de uma matriz de correlação cruzada, seguidas da aplicação de transformadas *wavelet* para suavizar as séries temporais. As janelas temporais foram utilizadas para treinar e validar modelos em contextos univariado e multivariado, com estratégias *one-step* e *multi-step*.

Na base InfoDengue, foram testados métodos multivariados individualmente para cada unidade federativa e também abordagens de *cross-learning* univariado, incluindo o uso de validação *leave-one-out* (LOO) entre os diferentes estados brasileiros e San Juan. A fase final consistiu em uma análise comparativa dos modelos.

As principais contribuições deste estudo são: (i) Identificação de estratégias de pré-processamento que aprimoram a eficácia dos modelos; (ii) Identificação de características filtradas por *wavelets* que auxiliam no processo de predição da dengue; (iii) Determinação, entre os modelos analisados, daquele que apresenta melhor eficácia na generalização para a tarefa de predição do número de casos semanais de dengue nessas localidades.

1.4 Organização da Dissertação

O Capítulo 2 apresenta os conceitos fundamentais necessários para compreender os aspectos gerais das abordagens propostas. O Capítulo 3 descreve uma coletânea de trabalhos relacionados e relevantes para a pesquisa. O Capítulo 4 detalha a abordagem proposta. O Capítulo 5 exibe e discute os resultados obtidos nas bases de dados, evidenciando a eficácia das abordagens aplicadas. Por fim, o Capítulo 6 oferece as conclusões, discute as limitações do método e propõe direções para futuros trabalhos de pesquisa.

2

Fundamentação Teórica

Neste capítulo, apresentamos os conceitos fundamentais que embasam o desenvolvimento desta obra, organizados em onze seções. A primeira seção define série temporal e discute conceitos associados à sua definição. As segunda, terceira e quarta seções abordam métodos estatísticos clássicos. A quinta seção introduz técnicas de filtragem utilizando *wavelets*. Da sexta à décima seção, apresentamos as técnicas de aprendizado de máquina, tanto raso quanto profundo. Na décima primeira seção, descrevemos as métricas de avaliação utilizadas no trabalho.

2.1 Séries Temporais

Séries temporais são casos particulares de *processos estocásticos*.

Definição 1. *(processo estocástico) Considere* T *um conjunto arbitrário. Define-se processo estocástico como uma família* $X = \{X(t),\ t \in \mathrm{T}\}$*, onde para cada* $t \in \mathrm{T}$*, tem-se que* $X(t)$ *é uma variável aleatória.*

Dessa forma, um processo estocástico é uma família de variáveis aleatórias, que supomos estarem definidas num mesmo espaço de probabilidades $(\Omega,\ \mathcal{A},\ \mathcal{P})$, sendo Ω espaço amostral, $\mathcal{A}$ é uma σ-álgebra e $\mathcal{P}$ é uma medida de probabilidade sobre $\mathcal{A}$ [Berger & Casella, 2001]. Já que para cada $t \in \mathrm{T}$, $X(t)$ é variável aleatória definida sobre Ω, temos que $X(t) := X(t, w)$ é uma função de dois argumentos, em que $w \in \Omega$.

Definição 2. *(série temporal) Seja* X *um processo estocástico. Fixe* $w \in \Omega$*. Logo, para cada* $w \in \Omega$ *fixado, a função* $X(t) := X(t, w)$ *é dita trajetória do processo ou série temporal.*

Como para cada $w \in \Omega$ temos uma trajetória, o conjunto de trajetórias $X(t, w)$ é denotado por $X^{(1)}(t)$, $X^{(2)}(t)$, e assim por diante. Além disso, o conjunto de valores $\{X(t),\ t \in \mathrm{T}\}$ é chamado de *espaço de estados do processo estocástico*, e os valores $X(t)$ são chamados de *estados*. Se T for enumerável, como $\mathrm{T} \subseteq \mathbb{Z}$, o processo é denominado *processo com parâmetro discreto*. Por exemplo, o número de casos semanais de dengue ao longo de dez anos representa esta situação. Se T for não enumerável, como um intervalo de números reais, obtemos um *processo com parâmetro contínuo*. Para ilustrar, basta tomar medidas que variam de maneira contínua ao longo do tempo, como os valores da temperatura de uma região ao longo de uma semana. Diante da análise prática, tem-se a presença de uma das trajetórias do processo estocástico, isto é, temos informação a respeito de $X^{(j)}(t)$ para certo $j \in \mathbb{Z}$, que denotaremos por $X(t)$, e observações feitas em instantes discretos e equiespaçados no tempo, denominados $X_1, X_2, ..., X_N$. A definição de covariância é necessária para definição de estacionariedade.

Definição 3. *A função de autocovariância (momento de segunda ordem) de X é dada por*

$$\gamma(t_1, t_2) = E\{X(t_1)X(t_2)\} - E\{X(t_1)\}E\{X(t_2)\},\ t_1, t_2 \in \mathrm{T}. \tag{2.1}$$

Em (2.1), caso $t_1 = t_2 = t$, temos que $\gamma(t, t) = Var\{X(t)\}$. Ou seja, temos a variância do processo X. Comumente, denota-se $Cov\{Z(t_1), Z(t_2)\} = \gamma(t_1, t_2)$.

Quanto à estacionariedade, um processo pode ser *estritamente estacionário* ou *estacionário de segunda ordem.*

Definição 4. *(processo estacionário de segunda ordem) Um processo estocástico dado por $X = \{X(t), t \in \mathrm{T}\}$ é estacionário de segunda ordem se satisfaz as seguintes condições:*

(i) $E\{X(t)\} = \mu$, onde μ é constante, para todo $t \in \mathrm{T}$
(ii) $E\{X^2(t)\} < \infty$, para todo $t \in \mathrm{T}$
(iii) $\gamma(t_1, t_2) = Cov\{X(t_1), X(t_2)\}$ é uma função de $|t_1 - t_2|$

□

A condição *(iii)* da **Definição 4** é obtida tomando inicialmente $t = -t_1$,

$$\gamma(t_1, t_2) = \gamma(t_1 + t, t_2 + t) = \gamma(t_1 - t_2, 0) = \gamma(\tau) \text{ para } \tau = t_1 - t_2. \tag{2.2}$$

Depois, supondo $t = -t_2$, obtemos

$$\gamma(t_1, t_2) = \gamma(t_1 + t, t_2 + t) = \gamma(0, t_2 - t_1) = \gamma(-\tau) \text{ para } \tau = t_1 - t_2. \quad (2.3)$$

Portanto, $\gamma(t_1, t_2)$ é função de $|t_1 - t_2|$. A *estacionariedade estrita* é pouco explorada como hipótese na formação dos modelos de predição de séries temporais, de modo que a *estacionariedade de segunda ordem* recebe maior relevância. Sendo assim, *processos estacionários de segunda ordem* serão chamados simplesmente de *processos estacionários*. De maneira a simplificar a notação, se o conjunto $\mathcal{T} = \mathbb{Z}$, escrevemos $\{X_t, t \in \mathbb{Z}\}$.

Definição 5. *(sequência aleatória) Considere* $\{X_n, n = 1, 2, ...\}$ *uma sequência de variáveis aleatórias definidas no espaço amostral* Ω*. Neste caso, temos* $\mathrm{T} = \{1, 2, ...\}$ *e o processo é dito com parâmetro discreto, ou uma sequência aleatória.*

Definição 6. *(ruído branco discreto) O processo* $\{a_t, t \in \mathbb{Z}\}$ *é um ruído branco discreto se as variáveis aleatórias* a_t *são não correlacionadas, ou seja, temos que* $Cov\{a_t, a_s\} = 0$ *para* $t \neq s$*. Se* $E\{a_t\} = \mu_a$ *e* $Var\{a_t\} = \sigma_a^2$*, então o processo é estacionário. No intuito de facilitar a modelagem de processos estocásticos, assumimos que o ruído branco possui* $\mu_a = 0$*.*

Os modelos para séries temporais são divididos em duas classes, conforme o número de parâmetros envolvidos na análise: *modelos paramétricos* e *modelos não paramétricos*.

Os *modelos paramétricos* são analisados no domínio do tempo. Como exemplo de *modelos paramétricos*, temos o modelo autorregressivo de médias móveis (ARMA) e o modelo autorregressivo integrado e de médias móveis (ARIMA).

Os *modelos não paramétricos* são analisados no domínio da frequência. A *função de autocovariância* e a transformada de Fourier (espectro) do *processo estocástico* são exemplos de *modelos não paramétricos*.

2.2 Passeio Aleatório

Seja $\{\epsilon_t;\ t \in \mathbb{Z},\ t \geq 1\}$ uma dada sequência aleatória, onde esta sequência forma um conjunto de variáveis aleatórias independentes e identicamente distribuídas com $\epsilon_t \sim (\mu_\epsilon, \sigma_\epsilon^2)$. Chamamos de passeio aleatório (*random walk* - RW) a variável X_t que

representa a soma das primeiras t variáveis da sequência aleatória, ou seja,

$$X_t = \sum_{i=1}^{t} \epsilon_t. \tag{2.4}$$

Podemos reescrever como

$$X_t = X_{t-1} + \epsilon_t \implies X_t - X_{t-1} = \epsilon_t. \tag{2.5}$$

Portanto, os incrementos são não correlacionados. Além disso, RW é um processo não estacionário, pois X_t varia crescentemente ao longo do tempo e a autocovariância, para instantes t_1 e t_2, não pode ser expressa apenas como uma função de $|t_1 - t_2|$ [Box et al., 2015].

2.3 Modelos de suavização exponencial

As técnicas de suavização exponencial tratam os valores extremos como as representações principais da aleatoriedade em uma série temporal. Por consequência, estes modelos idealizam que através da suavização desses extremos, é possível identificar o padrão subjacente da série.

Definição 7. *Seja X uma série temporal. O valor $\hat{X}_t(h)$ é a predição de X_{t+h}, de origem t e horizonte h.*

Dentre as técnicas de suavização exponencial, temos modelos adequados a séries localmente constantes (suavização exponencial simples), modelos adequados para séries que representam tendências (suavização exponencial de Holt) e métodos para séries sazonais (suavização exponencial de Holt-Winters).

2.3.1 Suavização Exponencial Simples (SES)

Seja a série temporal X com observações $X_1, X_2, ..., X_N$, em que

$$X_t = \mu_t + a_t, \ t = 1, ..., N \tag{2.6}$$

e a média de a_t é dada por $E(a_t) = 0$ e $Var(a_t) = \sigma_a^2$. Considera-se que o parâmetro μ_t varia lentamente ao longo do tempo. O valor exponencialmente suavizado é

definido por

$$\bar{X}_t = \alpha X_t + (1-\alpha)\bar{X}_{t-1},\ \bar{X}_0 = X_1,\ t = 1, ..., N, \tag{2.7}$$

em que $0 \leq \alpha \leq 1$ é a constante de suavização. A predição da SES é

$$\hat{X}_t(h) = \bar{X}_t,\ \forall h > 0, \tag{2.8}$$

isto é,

$$\hat{X}_t(h) = \alpha X_t + (1-\alpha)\hat{X}_{t-1}(h+1). \tag{2.9}$$

Expandindo (2.9) conforme a recorrência (2.8), temos

$$\hat{X}_t(h) = \alpha \sum_{k=0}^{t-1}(1-\alpha)^k X_{t-k} + (1-\alpha)^t \bar{X}_0. \tag{2.10}$$

Portanto, ao analisarmos a equação (2.10), quanto menor o valor de α, maior será a estabilidade nas predições finais, visto que pesos maiores serão atribuídos às observações passadas. Desse modo, qualquer flutuação aleatória no valor presente da série terá um peso menor no cálculo da predição. Das vantagens do SES, temos a flexibilidade de manipular a constante α e o baixo armazenamento de variáveis a cada predição (X_t, $\bar{X}_t$ e α).

2.3.2 Suavização Exponencial de Holt (SEH)

O modelo de suavização exponencial de Holt serve para suprir a imprecisão que SES possui em capturar a tendência linear dos modelos, pois caso a série temporal apresente tendência linear positiva, então SES tende a subestimar os valores da predição. Caso a tendência linear seja negativa, então ocorre superestimação na predição. Logo, o modelo linear de Holt é utilizado para séries temporais com tendência linear e sem sazonalidade, isto é,

$$X_t = \mu_t + T_t + a_t,\ t = 1, ..., N. \tag{2.11}$$

Novamente, consideramos o resíduo a_t tal que $E(a_t) = 0$ e $Var(a_t) = \sigma_a^2$. Além disso, há duas constantes de suavização. A constante $0 < A < 1$ suaviza o nível da série. A constante $0 < C < 1$ suaviza a tendência da série. A escolha das constantes baseia-se na seleção do par (A, C) que minimize o erro quadrático na predição. A predição de SEH inicia-se supondo duas condições iniciais. Estas são

$$\begin{cases} \hat{T}_2 = X_2 - X_1, \\ \bar{X}_2 = X_2. \end{cases} \tag{2.12}$$

Em seguida, calcula-se

$$\begin{cases} \bar{X}_t = AX_t + (1-A)(\bar{X}_{t-1} + \hat{T}_{t-1}), \\ \hat{T}_t = C(\bar{X}_t - \bar{X}_{t-1}) + (1-C)\hat{T}_{t-1}, \end{cases} \tag{2.13}$$

sendo o valor predito $X_t(h)$ calculado como

$$X_t(h) = \bar{X}_t + h\hat{T}_t, \ \forall h > 0. \tag{2.14}$$

O mecanismo de atualização é dado por

$$\begin{cases} \bar{X}_{t+1} = AX_{t+1} + (1-A)(\bar{X}_t + \hat{T}_t), \\ \hat{T}_{t+1} = C(\bar{X}_{t+1} - \bar{X}_t) + (1-C)\hat{T}_t, \end{cases} \tag{2.15}$$

em que o valor predito $X_{t+1}(h-1)$ é

$$X_{t+1}(h-1) = \bar{X}_{t+1} + (h-1)\hat{T}_{t+1}. \tag{2.16}$$

2.3.3 Suavização Exponencial de Holt-Winters (HW)

A suavização exponencial de Holt-Winters modela séries temporais de acordo com nível, tendência e sazonalidade. A sazonalidade pode ser aditiva ou multiplicativa. Considere as séries com sazonalidade de período s. Para a série sazonal aditiva com tendência aditiva e fator sazonal F_t multiplicativo, temos

$$X_t = \mu_t F_t + T_t + a_t, \ t = 1, 2, ..., N. \tag{2.17}$$

O modelo HW apresenta três constantes de suavização. A constante $0 < A < 1$ suaviza o nível da série. A constante $0 < C < 1$ suaviza a tendência. A constante $0 < D < 1$ suaviza o fator sazonal. A escolha das constantes baseia-se na seleção

do par (A, C, D) que minimize o erro quadrático na predição. O modelo HW para a série (2.17) é dado por

$$\begin{cases} \hat{F}_t = D\left(\dfrac{X_t}{\bar{X}_t}\right) + (1-s)\hat{F}_{t-s}, \\ \bar{X}_t = A\left(\dfrac{X_t}{\hat{F}_{t-s}}\right) + (1-A)(\bar{X}_{t-1} + \hat{T}_{t-1}), \\ \hat{T}_t = C(\bar{X}_t - \bar{X}_{t-1}) + (1-c)\hat{T}_{t-1}, \end{cases} \tag{2.18}$$

em que $t = s+1, s+2, ..., N$. A predição da série com fator sazonal multiplicativo é dada por

$$\hat{X}_t(h) = (\bar{X}_t + h\hat{T}_t)\hat{F}_{t+h-s}, h = 1, 2, ..., s, \tag{2.19}$$

$$\hat{X}_t(h) = (\bar{X}_t + h\hat{T}_t)\hat{F}_{t+h-2s}, h = s+1, ..., 2s, \tag{2.20}$$

etc. Os valores $\bar{X}_t$, $\hat{F}_t$ e $\hat{T}_t$ são dados por (2.18). Analogamente, tem-se a formulação para o fator sazonal F_t com tendência aditiva [Morettin & Toloi, 2018].

2.4 Autorregressivo Integrado de Médias Móveis (ARIMA)

Considere a série temporal X_t. Seja $\{a_t, t \in \mathbb{Z}\}$ um ruído branco, *i.e.*, $Cov(a_s, a_t) = 0$ para $t \neq s$ e μ é o nível da série. A filosofia do modelo ARIMA(p, d, q) deriva de três modelos: AR(p), dado por

$$X_t = \mu + a_t + \sum_{i=1}^{p} \phi_i X_{t-i}, \tag{2.21}$$

um polinômio autorregressivo de ordem p. MA(q),

$$X_t = \mu + a_t - \sum_{j=1}^{q} \theta_j a_{t-j}, \tag{2.22}$$

é o polinômio de médias móveis de ordem q; e a quantidade d de vezes que a série foi diferenciada para se tornar estacionária, onde "diferenciada" se refere ao número de vezes que o operador de diferença Δ foi aplicado à série temporal,

$$\Delta X_t = X_t - X_{t-1}, \tag{2.23}$$

e não se relaciona ao conceito usual de derivada. É possível reescrever as expressões (2.21) e (2.22) utilizando o operador de translação passada B, definido como

$$BX_t = X_{t-1}, \tag{2.24}$$

e, indutivamente,

$$B^m X_t = X_{t-m}. \tag{2.25}$$

Portanto, (2.21) torna-se

$$\phi(B)X_t = \mu + a_t, \tag{2.26}$$

onde

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p \tag{2.27}$$

é o operador autorregressivo. Similarmente,

$$X_t = \mu + \theta(B)a_t, \tag{2.28}$$

com

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - ... - \theta_q B^q \tag{2.29}$$

sendo o operador de médias móveis. O modelo ARMA(p, q)

$$X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} -\theta_j a_{t-j} + a_t \tag{2.30}$$

ou

$$\phi(B)X_t = \theta(B)a_t \tag{2.31}$$

é a combinação de AR(p) e MA(q). Dessa forma, ARIMA(p, d, q) é dado por

$$\phi(B)\Delta^d X_t = \theta(B)a_t. \tag{2.32}$$

2.5 Transformada Wavelet

Historicamente, as transformadas *wavelet* apareceram pela primeira vez em sua forma contínua nos trabalhos de Morlet [1981], Grossmann & Morlet [1984], e Grossmann [1988]. Seja $f \in L^2(\mathbb{R})$. A transformada *wavelet* contínua (*continuous wavelet transform*- CWT) de f com respeito a uma *wavelet* ψ é dada por

$$W_f(a,b) = \int_{-\infty}^{\infty} f(t)\frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right)dt, \tag{2.33}$$

onde $a > 0$ representa o fator de escala, b é o parâmetro de translação, e ψ é a *wavelet* de análise, tipicamente satisfazendo certas condições como admissibilidade e normalização. O objetivo é gerar $L^2(\mathbb{R})$ a partir de uma única função ψ denominada *ondaleta mãe*. A geração do espaço é feita por meio de dilatações (ou compressões) e translações de ψ dadas por

$$\psi_{a,b}(t) = |a|^{-\frac{1}{2}}\psi\left(\frac{t-b}{a}\right),\ b \in \mathbb{R},\ a > 0. \tag{2.34}$$

Os valores usuais para a e b são $a = 2^{-j}$ e $b = k2^{-j}$ para $j, k \in \mathbb{Z}$, resultando em

$$\psi_{j,k}(t) = |2^{-j}|^{-\frac{1}{2}}\psi\left(\frac{t-k2^{-j}}{2^{-j}}\right) = 2^{\frac{j}{2}}\psi\left(2^j t - k\right). \tag{2.35}$$

O valor 2^j é dito *dilatação binária*. O termo $k2^{-j}$ é a *translação diática*. A transformada em (2.33) é o produto interno entre f e $\psi_{a,b}$. No entanto, devido à natureza dos dados neste trabalho, o foco será na *transformada wavelet discreta*. Para detalhes sobre transformadas contínuas, veja o trabalho de Daubechies [1992].

Seja $x = (x_0, x_1, x_2, \ldots, x_{n-1})$ uma série temporal com n observações. A transformada wavelet discreta (*discrete wavelet transform* - DWT) é expressa pela relação

$$d = Wx, \tag{2.36}$$

onde $W \in \mathbb{R}^{n\times n}$ é a matriz de transformação *wavelet*, e d são os coeficientes de detalhes. A matriz W é construída de modo que cada linha representa um filtro *wavelet* aplicado aos dados em uma escala diferente, decompondo efetivamente o sinal original em múltiplos componentes de frequência. Este processo permite a análise do sinal em várias resoluções, capturando tanto informações de alta frequência quanto de baixa frequência, essenciais para tarefas como compressão de dados, redução de ruído e extração de características.

Essa transformação é particularmente eficaz em aplicações onde os sinais contêm potência não estacionária em várias frequências diferentes, ou onde o sinal deve ser analisado com resoluções variáveis. Os coeficientes d capturam as informações do sinal em diferentes escalas e são usados para reconstruir o sinal ou para realizar análises adicionais.

Os coeficientes de detalhe *wavelet* geralmente não são obtidos através do produto matricial mostrado em (2.36). Em vez disso, o *algoritmo piramidal* é utilizado, que possui uma complexidade computacional de $O(n)$. Este algoritmo consiste em uma sequência de filtros passa-baixa e passa-alta. Em geral, o *algoritmo piramidal* usa filtros passa-baixa $\{l_k\}$ e filtros passa-alta $\{h_k\}$ calculados por

$$l_k = \sqrt{2}\int_{-\infty}^{\infty} \phi(t)\phi(2t-k)dt, \tag{2.37}$$

$$h_k = \sqrt{2}\int_{-\infty}^{\infty} \psi(t)\phi(2t-k)dt. \tag{2.38}$$

No filtro l_k, a função ϕ, também chamada de *ondaleta pai*, é dada por

$$\phi(t) = \sqrt{2}\sum_k l_k\phi(2t-k). \tag{2.39}$$

A ϕ gera uma família ortonormal em $L^2(\mathbb{R})$, expressa por

$$\phi_{j,k}(t) = 2^{\frac{j}{2}}\phi(2^j t - k),\ j,k \in \mathbb{Z}. \tag{2.40}$$

Dessa maneira, a ondaleta mãe ψ pode ser obtida de ϕ por meio da expressão

$$\psi(t) = \sqrt{2}\sum_k h_k \phi(2t-k), \tag{2.41}$$

em que Kumar & Mehra [2005] mostraram que

$$h_k = (-1)^k l_{1-k}, \tag{2.42}$$

e Strang & Nguyen [1996] mostraram que

$$\sum_k l_k = \sqrt{2}, \tag{2.43}$$

$$\sum_k l_k^2 = 1, \tag{2.44}$$

$$\sum_k l_k h_{k-2m} = \begin{cases} 1, \text{ se } m = 0 \\ 0, \text{ caso contrário,} \end{cases} \tag{2.45}$$

permitindo maior facilidade na computação dos filtros h_k e l_k.

No j-ésimo nível do algoritmo piramidal, calculam-se $c_{j,k}$ e $d_{j,k}$ pelas seguintes equações. Supondo as condições descritas acima, os coeficientes $c_{j,k}$ e $d_{j,k}$ usados pelo algoritmo piramidal são expressos por

$$c_{j,k} = \sum_n l_{n-2k} c_{j+1,n}, \tag{2.46}$$

$$d_{j,k} = \sum_n h_{n-2k} c_{j+1,n}. \tag{2.47}$$

Note que $c_{j,k}$ e $d_{j,k}$ são calculados em pares. Esta operação é chamada de *decimação*. Ou seja, no nível j teremos metade do número de coeficientes em comparação ao nível $j+1$, o que motiva o nome *piramidal* [Mallat, 1989].

Há interesse em ondaletas ortogonais e com *suporte compacto*. Ondaletas que possuem *suporte compacto* são funções que têm a propriedade de serem diferentes de zero apenas em um intervalo limitado. Esse *suporte compacto* permite que as ondaletas sejam especialmente úteis para analisar sinais ou imagens em diferentes escalas e posições, mantendo a localização exata das características de interesse [Strang & Nguyen, 1996]. A principal vantagem das ondaletas com *suporte compacto* sobre ondaletas que não possuem esta propriedade, é a sua capacidade de se adaptar

a características locais de um sinal. Por exemplo, em processamento de imagens ou sinais, elas permitem a análise de partes específicas do sinal com alta resolução, sem perder a visão geral ou o contexto em que essas partes ocorrem [Strang & Nguyen, 1996]. Esta propriedade é útil para aplicações como compressão de dados, remoção de ruídos, e análise multifractal, entre outras. O exemplo mais antigo e simples de *wavelet* ortogonal e com suporte compacto é a *wavelet de Haar* [Haar, 1910].

2.6 Aprendizado de Máquina

Uma *máquina aprende* a realizar uma tarefa T por meio da experiência E, se, após esta experiência, houver uma melhoria no desempenho D da máquina na tarefa T, conforme avaliação feita por um critério de desempenho adotado [Mitchell, 1997]. Em geral, há quatro tipos de aprendizado, sendo estes, o *supervisionado*, *não supervisionado*, *semi-supervisionado* e *por reforço* [Russell & Norvig, 2020].

Seja $A = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ tal que $f(x_i) = y_i$ para $1 \leq i \leq N$. O conjunto A é o *conjunto de treinamento*. A tarefa do *aprendizado supervisionado* objetiva encontrar a função h que se aproxima de f. O processo de aprendizado envolve procurar o conjunto de funções h que melhor se ajustam à realidade apresentada por f. Caso isto aconteça, h aprendeu com o treinamento. Entretanto, deseja-se a *generalização do aprendizado*. Por sua vez, dizemos que h *generaliza* o processo se predizer aproximadamente (ou corretamente) o valor de $y = f(x)$ para $(x, y) \notin A$. Na prática, esses pares ordenados $(x, y) \notin A$ pertencem ao dito *conjunto de teste*.

Em *aprendizado supervisionado*, existem *problemas de regressão* e *problemas de classificação*. Os *problemas de regressão* envolvem a predição de um valor numérico. Por exemplo, a predição do valor da temperatura em um certo instante t. Os *problemas de classificação* envolvem predizer um grupo ou uma classe que determinada instância se encontra. Como exemplo, saber qual dos quatro tipos imunológicos da dengue [Ross, 2010] estão presentes em um paciente.

Os dados pertinentes ao problema de aprendizado não são suficientes por si só para encontrarem uma função h que se aproxima de f. Portanto, é necessário assumir premissas (ou hipóteses) para inferirmos possíveis resultados. Estas premissas são chamadas de *viés indutivo* do processo de aprendizado [Alpaydin, 2020], e pertencem à classe $\mathcal{H}$ de hipóteses sobre o modelo. Por exemplo, em problemas de regressão, assumir que f possui comportamento linear é induzir viés na modelagem de h. Da mesma forma, dentre todas as retas geradas para modelar f, tomar a reta h' que minimiza o erro quadrático médio em relação à f, também é introduzir viés,

pois veja que a seleção da melhor reta foi obtida por meio de critério introduzido pelo analista.

Ademais, é possível estender a classe $\mathcal{H}$ ao adicionarmos premissas cada vez mais complexas, de maneira a restringir o espaço de funções que compõem a solução do problema. Em problemas de regressão, podemos aumentar ou reduzir o grau do polinômio h que se aproxima de f, visando melhorar a qualidade da aproximação. Logo, perceba que aumentar ou reduzir o nível de complexidade é introduzir viés no processo de modelagem, o que nos mostra que não é possível construir o modelo sem enviesá-lo. A problemática consiste em *o quanto de viés* será introduzido ao modelo. A escolha do modelo conforme as hipóteses pré-estabelecidas é denominada *seleção do modelo*. Esta seleção é feita visando obter resultados bons segundo critérios de avaliação feitos no *conjunto de teste*, ou seja, deseja-se que o viés seja escolhido de modo que o modelo tenha boa capacidade de *generalização*.

Dessa forma, se $\mathcal{H}$ for de menor complexidade, de maneira a gerar funções h com aproximações pobres em relação à função f (correspondente à realidade), então teremos o *subajuste* (*underfitting*). Por outro lado, se as hipóteses de $\mathcal{H}$ forem demasiado complexas, de maneira que h aprenda bem os dados de treino, mas falha na fase de *generalização* (testes), então teremos *sobreajuste* (*overfitting*).

Logo, conforme Alpaydin [2020], ao tentarmos encontrar uma função h que se aproxime de f, realizamos o procedimento a seguir. Primeiro, o conjunto de dados é dividido em dados de treino, validação e teste. Os dados de treino servem para encontramos as hipóteses da classe $\mathcal{H}$. O conjunto de validação serve para, dentre o conjunto de modelos possíveis segundo as hipóteses estabelecidas, selecionar o melhor modelo que generaliza o fenômeno. Para selecionar o melhor modelo, temos de selecionar as melhores hipóteses da classe $\mathcal{H}$ que o fundamentam. Este processo chama-se *validação cruzada*. O teste serve para exibir o erro médio do melhor modelo selecionado na fase de validação.

No campo do aprendizado de máquina, podemos classificar os modelos em dois tipos principais: *rasos* e *profundos*. Os *modelos rasos* têm uma ou poucas camadas de processamento entre a entrada e a saída, permitindo-lhes aprender padrões diretos e superficiais dos dados. Eles requerem menos dados para um treinamento eficaz e têm um custo computacional menor durante o treinamento. Por outro lado, os *modelos profundos* geralmente contêm duas ou mais camadas ocultas e são capazes de aprender padrões em múltiplos níveis de complexidade [Goodfellow et al., 2016]. Essa característica os torna úteis para tarefas como reconhecimento de imagem e processamento de linguagem natural, onde é desejável uma análise mais sutil e variada. O aprendizado profundo geralmente requer grandes quantidades de da-

dos para treinamento, pois precisa aprender um número expressivo de parâmetros. Além disso, a complexidade computacional e o tempo de treinamento são significativamente maiores em redes profundas. Quanto à interpretabilidade dos modelos, a natureza "*caixa preta*" da maioria dos *modelos de aprendizado profundo* é desafio significativo em diversos setores, principalmente no setor da saúde, que valoriza a transparência [Neto et al., 2022].

Na fase de experimentação deste trabalho, os modelos rasos utilizados incluíram *regressão linear*, *algoritmos baseados em árvore* (Árvore de Regressão, Bagging, Random Forest, Gradient Boosting, XGBoost e LightGBM) e *regressão por máquina de vetores suporte*. Já os modelos profundos adotados foram *perceptrons multicamada*, *redes neurais convolutivas*, *redes neurais recorrentes* e *redes com camadas de atenção*. Além disso, foram utilizados modelos fundadores como TimeGPT-1 e MOIRAI. A descrição de cada um dos métodos será apresentada a seguir.

2.7 Regressão linear

Na regressão linear, assume-se que a função f que modela os dados possui caráter linear [Alpaydin, 2020]. Ou seja, a regressão linear computa uma relação linear entre a variável dependente (valor alvo) e variáveis independentes ao ajustar uma equação linear aos dados observados.

Formalmente, considere um conjunto de m observações $\{(X_1, y_1), ..., (X_m, y_m)\}$. O valor $X_i \in \mathbb{R}^n$ é um vetor em que cada posição representa uma característica (*feature*) do problema em questão. Já y_i é o valor real da característica alvo do problema. A predição do modelo de regressão linear aplicada em uma observação i, para $1 \leq i \leq m$, é dado pela expressão

$$\hat{y}_i = \theta_0 + \sum_{j=1}^{n} \theta_j x_{ij}, \tag{2.48}$$

onde $\theta_j \in \mathbb{R}$, $\hat{y}_i \in \mathbb{R}$ é a variável dependente e $x_{ij} \in \mathbb{R}$, para $1 \leq j \leq n$, são as variáveis independentes. Dessa maneira, o formato matricial do modelo de regressão linear é

$$\hat{Y} = X\theta. \tag{2.49}$$

Se $n = 1$, temos a *regressão linear univariada*. Para $n > 1$, tem-se a *regressão linear multivariada*. Entretanto, é necessário encontrar os valores dos coeficientes $\theta_i \in \mathbb{R}$. Com efeito, se $y = (y_1, y_2, ..., y_m)$ é o valor real da f, isto é, $f(X_i) = y_i$

e, $\hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_m)$ são os valores preditos pela regressão linear, então define-se a *função de custo*

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2. \tag{2.50}$$

Desta expressão, obtemos que a forma matricial de J é

$$J(\theta) = \frac{1}{2}tr((X\theta)^T X\theta - (X\theta)^T Y - Y^T X\theta + Y^T Y). \tag{2.51}$$

onde tr é o traço da matriz. Derivando em relação à θ,

$$\nabla_\theta J(\theta) = X^T X\theta - X^T Y. \tag{2.52}$$

Portanto, se $\nabla_\theta J(\theta) = 0$, temos

$$X^T X\theta = X^T Y. \tag{2.53}$$

Caso $X^T X$ seja invertível, então o mínimo global de J é

$$\theta = (X^T X)^{-1} X^T Y. \tag{2.54}$$

A equação (2.54) é válida, caso $X^T X$ seja definida positiva, isto é,

$$x^T (X^T X)x > 0, \ \forall x \in \mathbb{R}^{n\times 1}. \tag{2.55}$$

Dessa forma, todos os autovalores de $X^T X$ são positivos, portanto $X^T X$ será invertível [Hoffman & Kunze, 1971]. Porém, há estratégias que lidam com a não invertibilidade de $(X^T X)$, como a *regularização de Ridge ou Tikhonov* ao adicionar o termo $\dfrac{\lambda\theta^T\theta}{2}$ à função $J(\theta)$, com $\lambda > 0$, de forma que

$$J(\theta) = \frac{1}{2}(\hat{Y} - Y)^T(\hat{Y} - Y) + \frac{\lambda\theta^T\theta}{2}. \tag{2.56}$$

Outra estratégia de regularização para regressão linear é a *Lasso* (*Least absolute shrinkage and selection operator*), onde a função de perda $J(\theta)$ é

$$J(\theta) = \frac{1}{2}(\hat{Y} - Y)^T(\hat{Y} - Y) + \sum_{j=0}^{n} |\theta_j|. \tag{2.57}$$

2.8 Algoritmos baseados em árvores

Modelos baseados em árvores segmentam o espaço gerado pelas variáveis preditoras em regiões mutuamente exclusivas [Hastie et al., 2009]. O algoritmo de construção dessas árvores é *top-down* e *guloso* [Cormen & Leiserson, 2022]. Inicialmente, todos os elementos estão localizados na mesma região do espaço de variáveis preditoras. A cada passo, busca-se a decisão que otimiza um determinado critério.

Dessa forma, seja $X_1, X_2, ..., X_p$ os p preditores que geram o espaço E. Este espaço pode ser expresso por uma partição formada por conjuntos $R_1, R_2, ..., R_M$ denominados *regiões*. Logo,

$$E = \bigcup_{i=1}^{M} R_j \text{ e } R_i \cap R_j = \varnothing,\ i \neq j. \tag{2.58}$$

Em geral, estas regiões são retângulos p-dimensionais construídos de modo a minimizar algum erro de predição definido. Considere o conjunto de treinamento $\{(x_i, y_i); i = 1, 2, ..., m\}$, em que $x_i \in \mathbb{R}^p$. A resposta y do modelo é dada por

$$y = f(x) = \sum_{j=1}^{M} f_j(x) I(x \in R_j), \tag{2.59}$$

sendo $I(x \in R_j) = 1$ se a condição de pertinência for satisfeita, e zero, caso contrário. Se o critério de otimalidade adotado for a minimização de quadrados , então o melhor preditor é a média dos y_i na região R_j, isto é,

$$\hat{f_j} = \text{média}(y_i : x_i \in R_j). \tag{2.60}$$

Com estas definições, constrói-se a árvore de regressão por meio de um método *top-down* e *guloso* expresso pelo **Algoritmo 1**. Seja n o número de elementos da região com menor quantidade de elementos. A notação $|R|$ indica o número de

elementos da região R. O parâmetro n_0 define o número mínimo de elementos em cada região.

Algoritmo 1: Árvore de Regressão

$R \leftarrow E$
`Dividir_Região(`R`)`
para $j \leftarrow 1$ **até** p **faça**
  $\hat{f}_j = \text{média}(y_i : x_i \in R_j)$
Calcule $f(x) = \sum_{j=1}^{M} \hat{f}_j(x) I(x \in R_j)$
Função `Dividir_Região(`R`):`
  se $|R| \leq n_0$ **então**
    retorne
  senão
    Escolha $X_j \in \{X_1, X_2, ..., X_p\}$ e o limiar $t \in \mathbb{R}$ que minimizam o erro de predição ao dividir R em $\{X : X_j < t\}$ e $\{X : X_j \geq t\}$
    $j_0 \leftarrow 0$
    para $j \leftarrow 1$ **até** p **faça**
      Defina $R_1(j,t) = \{X : X_j < t\}$ e $R_2(j,t) = \{X : X_j \geq t\}$
      se (j,t) *minimiza o erro de predição* **então**
        $j_0 \leftarrow j$
    `Dividir_Região(`$R_1(j_0, t)$`)`
    `Dividir_Região(`$R_2(j_0, t)$`)`

2.8.1 Bagging

O método *bagging*, ou *bootstrap aggregation*, gera múltiplas versões de um preditor a partir de distintas réplicas do conjunto de treinamento obtido por *bootstrap* e, com base nessas versões, constrói um preditor agregado [Hastie et al., 2009]. O objetivo é obter um preditor agregado que possua menor variância em comparação aos preditores individuais.

Tome o conjunto de treinamento $\tau = \{(x_i, y_i); i = 1, 2, ..., m\}$, com $x_i \in \mathbb{R}^p$, onde $f(x, \tau)$ é o preditor de Y baseado em τ. Seja τ_k um conjunto de B sequências de τ_k geradas por *bootstrap*, onde cada sequência possui n elementos independentes e τ_k possuem distribuição de probabilidade com valores aproximados aos de τ. Com as B réplicas de $\{\tau^{(b)},\ b = 1, 2, ..., B\}$, formamos preditores $f(x, \tau^{(b)})$. Por sua vez, o agregador *bootstrap* é dado por

$$f_{bag}(x) = \frac{1}{B}\sum_{i=1}^{B} f(x, \tau^{(b)}). \tag{2.61}$$

Um procedimento é *instável* se pequenas mudanças em τ implicam em grandes mudanças no preditor f. Em geral, *bagging* possui melhor eficiência preditiva em métodos *instáveis* [Breiman, 1996].

2.8.2 Random Forest

Florestas aleatórias (RF) regressivas são uma melhoria do método *bagging* aplicado a árvores de regressão [Breiman, 2001]. Em *bagging*, há a construção de B árvores a partir de um conjunto de treinamento obtido por *bootstrap*. No caso de RF, ao invés de considerarmos o conjunto completo de p preditores no momento de divisão do nó para formar novos nós, utilizamos apenas uma amostra aleatória de tamanho $k < p$ do conjunto de preditores. O valor padrão é $k = \lfloor p/3 \rfloor$ [Hastie et al., 2009]. Este ajuste tende a diminuir a correlação entre as árvores formadas anteriormente por *bagging*. No **Algoritmo** 2, considere o conjunto de bagging τ relatado anteriormente e $T_b(x)$ é a predição da árvore b para entrada x.

Algoritmo 2: Floresta Aleatória Regressiva

para $b \leftarrow 1$ **até** B **faça**
- Tome uma amostra *bootstrap* $\tau^{(b)}$ de tamanho m do conjunto τ
- Inicialize uma árvore T_b sem nós
- critério de parada: cada nó tenha uma quantia n_0 de amostras por nó
- **enquanto** *critério de parada não for satisfeito* **faça**
  - Selecione k variáveis aleatórias do conjunto de p variáveis preditoras
  - Tome a melhor variável preditora entre as k selecionadas
  - Divida o nó em dois nós filhos
  - **se** *nós filhos são não-terminais* **então**
    - Repita os passos de seleção e divisão para os nós filhos

Calcule $\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$

2.8.3 Boosting

Em problemas de regressão, métodos de *boosting* consistem na construção sequencial de modelos base (geralmente árvores de decisão), onde cada modelo subsequente é construído com foco nos resíduos gerados pelo modelo anterior [Hastie et al., 2009]. O objetivo é aprimorar continuamente a precisão do modelo agregado. Os principais hiperparâmetros do algoritmo de *boosting* incluem a quantidade B de árvores, o parâmetro de encolhimento λ (taxa de aprendizado), e o número d de divisões (*splits*) por árvore. Um valor alto de B pode levar a *overfitting*, enquanto valores típicos para λ são 0.01 e 0.001. A eficácia do *boosting* é frequentemente avaliada pela sua capacidade de reduzir o erro de forma iterativa e controlada.

Algoritmo 3: Algoritmo de Boosting

$\hat{f}(x) = 0$ e $r_i = y_i$ para todo conjunto de treino
para $b = 1, 2, ..., B$ **faça**
  Ajuste a árvore $\hat{f}^b$ com d splits para os dados de treino (X, r)
  $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$
  $r_i \leftarrow r_i - \lambda \hat{f}^b(x)$
Calcule $\sum_{b=1}^{B} \lambda \hat{f}^b(x)$

No **Algoritmo 3**, há o início do valor da predição com $\hat{f}(x) = 0$ e consideramos o i-ésimo resíduo r_i igual ao valor real de predição da i-ésima instância de treino y_i. Em seguida, B árvores são ajustadas visando compor o valor final para uma instância de treino x. Cada árvore de decisão $\hat{f}^b$ é construída com base nos dados de treino (x_i, y_i) para $i = 1, 2, ..., m$, onde são realizadas d *splits*, de forma que teremos $d + 1$ nós terminais (folhas). Logo, com a árvore construída, computamos o valor predito pela árvore dado a instância x, multiplicamos pelo parâmetro λ e atualizamos o valor da predição final $\hat{f}(x)$. Porém, provalmente esta predição possui erros. Logo, atualizamos o resíduo ao retirarmos $\lambda \hat{f}^b(x)$ de r_i. Portanto, perceba que a cada árvore construída, há a tendência a corrigir o erro cometido anteriormente. Ao término, a predição final é calculada considerando o valor individual de cada árvore $\lambda f^b(x)$.

2.8.4 Gradient Boosting

Ao invés de utilizar diretamente os resíduos do modelo, o *gradient boosting* (GBR) foca em minimizar uma função de perda L diferenciável, podendo ser, por exemplo, o erro quadrático. O algoritmo proposto por Friedman [2001] possui metodologia

inspirada no método do gradiente [Hastie et al., 2009], comum na área de otimização numérica [Nocedal & Wright, 2000], e serve como uma generalização do **Algoritmo 3**. No **Algoritmo 4**, a inicialização deseja encontrar um parâmetro γ que minimize a soma das funções de perda $L(y_i, \gamma)$, onde $\{(x_i, y_i); i = 1, 2, ..., m\}$ é o conjunto de treinamento. O laço externo com B repetições representa a quantidade de árvores formadas utilizando resíduos. Para cada árvore b, calculam-se m gradientes que irão compor a árvore b de regressão construída pelo **Algoritmo 1**. Em seguida, em cada uma das J_b regiões formadas, realiza-se o processo de *busca linear*, ou seja, calcula-se o *tamanho do passo* que auxilie na minimização dos gradientes (resíduos) da próxima iteração $b + 1$. Por fim, atualiza-se o regressor f_b de maneira que este venha ser usado na próxima iteração, caso $b < B$. Segundo Hastie et al. [2009], funções de perda L comumente usadas em problemas de regressão são a *erro médio quadrático*, *erro absoluto* e a *função de Huber*.

Algoritmo 4: Gradient Boosting

Seja $f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{m} L(y_i, \gamma)$

para $b \leftarrow 1$ **até** B **faça**

  para $i \leftarrow 1$ **até** m **faça**

    $r_{ib} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{b-1}}$

  Ajuste uma árvore de regressão conforme o **Algoritmo 1** aos resíduos r_{ib}, onde as regiões formadas sejam $R_{jb},\ j = 1, 2, ..., J_b$

  para $j \leftarrow 1$ **até** J_b **faça**

    $\gamma_{jb} = \arg\min_{\gamma} \sum_{x_i \in R_{jb}} L(y_i, f_{b-1}(x_i) + \gamma)$

  Atualize $f_b(x) = f_{b-1}(x) + \sum_{j=1}^{J_b} \gamma_{jb} I(x \in R_{jb})$

Tome $\hat{f}(x) = f_B(x)$

2.8.5 XGBoost

O algoritmo *Extreme Gradient Boosting* (XGBoost) [Chen & Guestrin, 2016] avança a abordagem tradicional de *boosting*, mantendo o uso dos gradientes da função de perda como resíduos, mas introduzindo melhorias em relação ao GBR. XGBoost aprimora a eficiência através de otimizações no nível do sistema, como o paralelismo, permitindo o uso simultâneo de múltiplas CPUs e GPUs, e uma gestão mais

eficaz da *memória cache*. Além disso, incorpora regularizações L1 e L2 no processo de aprendizado, ajudando a controlar o *overfitting* e a generalizar melhor o modelo. Diferentemente do GBR, que exige pré-processamento para tratar dados faltantes, o XGBoost gerencia automaticamente essas ausências. Para lidar com grandes *datasets*, utiliza a técnica de *quantile sketch*, superando as limitações de escalabilidade do GBR. Assim, o XGBoost não apenas melhora a eficiência e a flexibilidade, mas também simplifica o fluxo de trabalho em comparação ao seu predecessor.

2.8.6 LightGBM

O algoritmo *Light Gradient Boosting Machine* (LightGBM) [Ke et al., 2017] representa uma evolução no campo do *boosting*, otimizando ainda mais o desempenho e a eficiência em comparação com métodos anteriores, como o XGBoost. Uma das principais inovações do LightGBM é o uso da técnica *Gradient-based One-Side Sampling* (GOSS), que reduz o volume de dados durante o treinamento sem comprometer o desempenho, preservando instâncias com gradientes maiores, que são mais informativas. Além disso, o algoritmo emprega a técnica de *Exclusive Feature Bundling* (EFB), que agrupa características exclusivas, reduzindo a dimensionalidade dos dados sem perda significativa de informações. Essas técnicas tornam o LightGBM particularmente adequado para conjuntos de dados grandes e esparsos, oferecendo uma velocidade de treinamento superior e um uso mais eficiente da memória em comparação com outros algoritmos de *boosting*. Assim, o LightGBM não só acelera o processo de treinamento, como também mantém alta precisão, facilitando seu uso em aplicações práticas onde tempo e recursos são críticos.

2.9 Regressão por Máquina de Vetores Suporte

Originalmente, o algoritmo de *máquina de vetores suporte* (*support vector machine* - SVM) realiza classificação binária, em que a metodologia consiste em encontrar o *hiperplano ótimo* para separação de classes [Vapnik & Chervonenkis, 1964]. Posteriormente, Drucker et al. [1997] adaptaram o SVM para problemas de regressão, onde a filosofia do SVM permanece, porém o *support vector regressive* (SVR) prediz um valor numérico real ao invés de um número inteiro que representa uma classe. Devido à natureza deste trabalho, o enfoque será sobre o SVR.

Seja $\{(x_1, y_1), ..., (x_m, y_m)\} \subset \mathbb{R}^p \times \mathbb{R}$ um conjunto de treinamento. O objetivo é encontrar a função $f(x_i)$ tal que $|y_i - f(x_i)| < \epsilon$ para $i = 1, 2, ..., m$, sendo $\epsilon > 0$ o maior erro que deseja-se cometer. Para funções lineares (representam a equação de um hiperplano),

$$f(x) = \alpha + \beta^T x, \tag{2.62}$$

determinamos os coeficientes $\alpha \in \mathbb{R}$ e $\beta \in \mathbb{R}^{n\times 1}$ do hiperplano tal que

$$|f(x_i)| = |\alpha + \beta^T x_i| \leq \epsilon,\ i = 1, 2, ..., m. \tag{2.63}$$

Isto é, deseja-se f o mais *achatada* o possível, logo o valor de $\|\beta\|$ precisa ser minimizado. Para auxiliar na busca desses coeficientes, Vapnik [2013] define a função dos erros chamada $\epsilon-$*sensitiva*

$$E_\epsilon[f(x) - y] = \begin{cases} 0, \text{ se } |f(x) - y| < \epsilon \\ |f(x) - y| - \epsilon, \text{ caso contrário.} \end{cases} \tag{2.64}$$

Ou seja, há a geração de erro nulo, caso a diferença entre a predição $f(x)$ e o valor real y seja menor do que ϵ previamente definido. Portanto, o problema é minimizar

$$C \sum_{i=1}^{n} E_\epsilon[f(x_i) - y_i] + \frac{1}{2}\|\beta\|^2, \tag{2.65}$$

em que C é parâmetro de regularização, determinando um compromisso entre o achatamento de f e a disposição a tolerar erros maiores que ϵ. Na prática, há uma troca (*trade-off*) entre a complexidade do modelo (achatamento do hiperplano) e a tolerância quanto à quantidade de desvios maiores que ϵ. Valores maiores de C tendem a diminuir a tolerância a erros acima de ϵ, resultando em um modelo menos achatado e mais propenso ao *overfitting*, pois ajusta-se mais aos dados de treino. Por outro lado, valores menores de C aumentam a tolerância a tais erros, resultando em um modelo mais achatado que pode subestimar a complexidade dos dados, levando ao *underfitting*.

Assim como no SVM, o SVR usa o conceito de *margem flexível*, atribuindo variáveis de folga para cada ponto x_i, de maneira a permitir uma taxa de erro de predição para o modelo f ao predizer o valor de x_i. Neste caso, adicionamos duas variáveis de folga $\xi_i \geq 0$ e $\xi_i^* \geq 0$ para cada $i = 1, 2, ..., m$. Os valores $\xi_i > 0$ são para pontos x_i, onde $y_i > f(x_i) + \epsilon$. Já $\xi_i^* > 0$ são para os pontos x_i tais que $y_i < f(x_i) - \epsilon$. Para os pontos x_i situados dentro do espaço limitado pela fronteira,

temos $|y_i - f(x_i)| \leq \epsilon$. Esta representação pode ser vista na Figura 2.1.

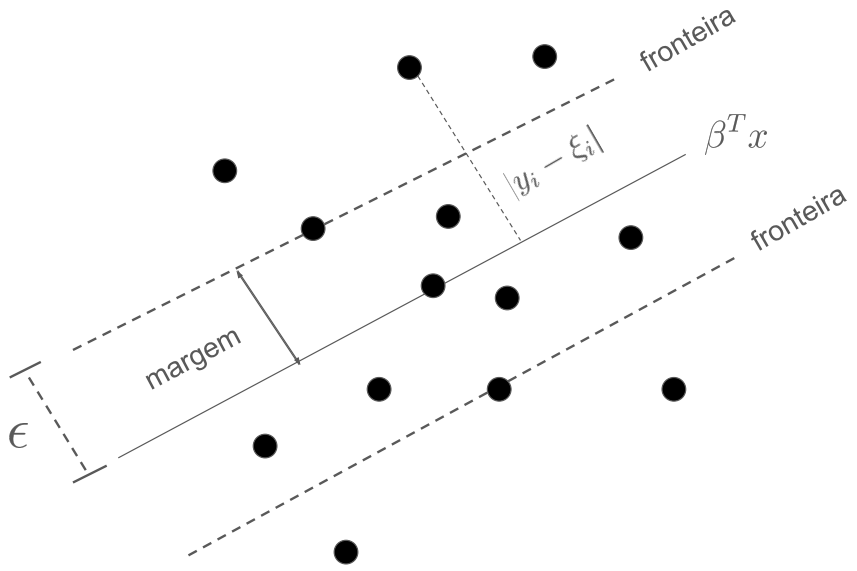


Figura 2.1. Representação do SVR.

2.10 Redes neurais

As *redes neurais* são modelos computacionais inspirados no funcionamento dos neurônios biológicos [Alpaydin, 2020]. Cada *neurônio* numa rede neural é uma unidade de cálculo que transforma os dados de entrada em uma saída, variável de acordo com o problema. Os neurônios são organizados em *camadas*: a *camada de entrada* capta os dados, as *camadas ocultas* processam esses dados por meio de transformações lineares e não lineares — permitindo a aprendizagem de representações complexas —, e a *camada de saída* entrega o resultado, como um valor numérico em problemas de regressão. Cada conexão entre neurônios tem um *peso*, e cada neurônio, um *viés*; ambos ajustados durante o treinamento via *algoritmo de retropropagação* [Hastie et al., 2009], uma forma eficiente do uso *método de gradiente descendente*. As *funções de ativação*, como a ReLU (unidade linear retificada) ou a sigmóide, aplicadas nas saídas das camadas, são cruciais para o aprendizado de padrões nos dados. Dependendo da configuração das camadas, diferentes *arquiteturas de rede* podem ser formadas, como as *redes densas*, *recorrentes*, *convolutivas* e de *atenção*.

2.10.1 Perceptron

Rosenblatt [1958] propôs o *perceptron* como um algoritmo supervisionado para classificação binária. Seja $l \in \mathbb{R}$ um limiar, $\{x_1, x_2, ..., x_N\} \subset \mathbb{R}^p$ o conjunto de treinamento, $w \in \mathbb{R}^N$ um *vetor de pesos*, $b \in \mathbb{R}$ um *viés*, $0 < \eta < 1$ a *taxa de aprendizado*, y_i é o valor real e $\hat{y}_i \in \{-1, 1\}$ *o valor predito*. O vetor w é inicializado aleatoriamente. A convergência do perceptron é garantida se as classes são linearmente separáveis [Alpaydin, 2020]. O número de passagens completas pelas N instâncias de treinamento denomina-se *época*.

2.10.2 Rede neural com camada oculta

Considere a *entrada* formada por elementos

$$x = (x_1, x_2, ..., x_p)^T, \tag{2.66}$$

a *camada oculta* formada por elementos

$$y = (y_1, y_2, ..., y_M)^T \tag{2.67}$$

com

$$y_j = h(a_{0j} + \alpha_j^T X),\ j = 1, 2, , ..., M, \tag{2.68}$$

e a *saída* constituída de elementos

$$z = (z_1, z_2, ..., z_K)^T \tag{2.69}$$

com

$$z_k = g(\beta_{0k} + \beta_k^T Y),\ k = 1, 2, ..., K. \tag{2.70}$$

Para problemas de regressão, usamos $K = 1$. Já para classificação binária, $K = 2$, e para problemas multiclasse, $K > 2$. Os valores a_{0j} e β_{0k} são *vieses*. As funções g e h são funções de ativação. Como exemplos de função de ativação, temos a *função logística* ou *sigmoide*,

$$f(x) = \frac{1}{(1 + e^{-x})}, \tag{2.71}$$

a *tangente hiperbólica*,

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{2.72}$$

a *unidade linear retificada* (ReLU)

$$f(x) = max(0, x), \tag{2.73}$$

e a *leaky* ReLU

$$f(x) = \begin{cases} x, x > 0 \\ 0,01x, \ x < 0. \end{cases} \tag{2.74}$$

O cálculo da saída z_k é feito pelo *algoritmo de proalimentação* (*forward propagation* ou *feed forward*). Os pesos da primeira camada são, para $j = 1, 2, ..., M$,

$$w_j^{(1)} = (w_{j0}^{(1)}, w_{j1}^{(1)}, ..., w_{jp}^{(1)})^T. \tag{2.75}$$

Os pesos da camada oculta são, para $k = 1, 2, ..., K$,

$$w_k^{(2)} = (w_{k0}^{(2)}, w_{k1}^{(2)}, ..., w_{kM}^{(2)})^T. \tag{2.76}$$

Algoritmo 5: Proalimentação

para $j \leftarrow 1$ **até** M **faça**
- $a_j \leftarrow \sum_{i=0}^{p} w_{ji}^{(1)} x_i$
- $y_j \leftarrow h(a_j)$
- **para** $k \leftarrow 1$ **até** K **faça**
  - $a_k \leftarrow \sum_{j=0}^{M} w_{kj}^{(2)} y_j$
  - $z_k = g(a_k)$

No **Algoritmo 5**, considera-se $x_0 = 1$, e $w_{j0}^{(1)}$ e $w_{k0}^{(2)}$ *vieses*. Em *problemas de regressão*, a função g é a *identidade*. Perceba que cada neurônio de uma camada possui conexões com a camada anterior. Esta característica define uma *camada densa* ou *totalmente conectada*. Uma rede formada por camadas densas é uma *rede*

densa. De forma sucinta, a expressão

$$f_k(x, w) = g\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{p} w_{ji}^{(1)} x_i\right)\right) \tag{2.77}$$

condensa as K saídas obtidas por meio de *proalimentação*. Ademais, a rede neural pode incluir camadas ocultas adicionais, de forma a torná-la uma *rede neural profunda*.

Enquanto o algoritmo de *proalimentação* é responsável por calcular os valores de predição, o *algoritmo de retropropagação* ajusta os pesos para minimizar uma função de perda especificada, como o *erro médio quadrático*. O processo de retropropagação atualiza os pesos movendo-os na direção oposta ao gradiente da função de perda. Adota-se esta abordagem, pois esta direção é a de máxima descida [Martínez & Augusta, 1998]. Dessa forma, supondo a função de perda acima, deseja-se encontrar w que minimize

$$Q_n(w) = \sum_{i=1}^{N} [z_i - f(x_i, w)]^2. \tag{2.78}$$

onde z_i é o valor real. De maneira geral, o objetivo é encontrar

$$\hat{w} = \arg\min_{w} [\lambda_1 Q_n(w) + \lambda_2 Q^*(w)], \tag{2.79}$$

em que $\lambda_1, \lambda_2 > 0$, e $Q^*(w)$ é um termo de regularização, podendo ser, por exemplo, regularização *Ridge*, *Lasso* ou *Elastic Net*. Obter o gradiente de Q_n e computar diretamente o método do gradiente é uma estratégia ineficiente [Hastie et al., 2009], isto é, inicializar $w^{(0)}$ e calcular a r-ésima iteração por

$$w^{(r)} = w^{(r-1)} - \lambda \nabla Q_n(w^{(r-1)}), \tag{2.80}$$

é um procedimento que não melhora a predição. Por esta razão, utiliza-se *retropropagação*. Seja

$$\begin{cases} w = (\alpha_0, ..., \alpha_M, \beta_0, ..., \beta_K), \\ \alpha_j = (\alpha_{j1}, ..., \alpha_{jp})^T, \ j = 1, 2, ..., M, \\ \beta_k = (\beta_{k1}, ..., \beta_{km})^T, \ k = 1, ..., K, \end{cases} \tag{2.81}$$

o vetor de pesos conforme exposto em (2.68) e (2.70). Explicitando as camadas em

(2.78), temos

$$Q_n(w) = \sum_{k=1}^{K} \sum_{i=1}^{n} [z_{ik} - f(x_i, w)]^2. \tag{2.82}$$

Nas camadas ocultas, considere

$$y_i = (y_{1i}, ..., y_{Mi})^T, \ 1 \leq m \leq M \tag{2.83}$$

em que

$$y_{mi} = h(\alpha_m^T x_i). \tag{2.84}$$

Derivando $Q_n(w)$ parcialmente em relação à β_{km} para um i fixo, temos

$$\frac{\partial}{\partial \beta_{km}} Q_i(w) = -2[z_{ik} - f(x_i, w)] f'(\beta_k^T y_i) h(\alpha_m^T x_i). \tag{2.85}$$

Para i fixo, alteramos a notação de Q_n para Q_i, e omitimos o somatório de índice i em Q_n. Analogamente, derivando $Q_n(w)$ parcialmente em relação à α_{ml} para i fixo, temos

$$\frac{\partial}{\partial \alpha_{ml}} Q_i(w) = -\sum_{k=1}^{K} 2[z_{ik} - f(x_i, w)] f'(\beta_k^T y_i) \beta_{km} h'(\alpha_m^T x_i) x_{il}. \tag{2.86}$$

Na iteração $r+1$, os pesos são atualizados segundo

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \lambda_r \sum_{i=1}^{n} \frac{\partial}{\partial \beta_{km}^{(r)}} Q_i(w) \tag{2.87}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \lambda_r \sum_{i=1}^{n} \frac{\partial}{\partial \alpha_{ml}^{(r)}} Q_i(w), \tag{2.88}$$

onde λ_r é a *taxa de aprendizado*. Além disso, considere

$$\delta_{ki} = -2[z_{ik} - f(x_i, w)] f'(\beta_k^T y_i) \tag{2.89}$$

o erro do modelo atual na saída, e

$$s_{mi} = h'(\alpha_m^T x_i) \sum_{k=1}^{K} \delta_{ki} \beta_{km} \tag{2.90}$$

o erro nas camadas escondidas. Portanto, (2.85) e (2.86) são reescritas como

$$\frac{\partial}{\partial \beta_{km}} Q_i(w), = \delta_{ki} y_{mi} \tag{2.91}$$

$$\frac{\partial}{\partial \alpha_{ml}} Q_i(w) = s_{mi} x_{il} \tag{2.92}$$

As equações (2.90) são as *equações de retropropagação*. Estas equações são usadas para implementar (2.91) e (2.92) em *dois passos*.

O *passo para frente* usa os pesos atuais fixos, calcula os valores preditos $f(x, w)$, e com estes valores, obtém (2.90).

O *passo para trás* calcula os erros cometidos (2.89), e propaga-os por (2.91) e (2.92), visando obter os erros s_{mi}.

Portanto, os dois conjuntos de erros (2.89) e (2.90) são usados para calcular os gradientes (2.85) e (2.86) por meio de (2.91) e (2.92).

2.10.3 Redes neurais recorrentes

As redes neurais recorrentes (RNN) foram inicialmente desenvolvidas nos trabalhos de Rumelhart et al. [1986] e Hopfield [1982]. Estas redes permitem a *retroalimentação* (*feedback*) entre as camadas ocultas. Ademais, elas usam estados internos (memórias) para processar sequências de tamanhos variados, possibilitando o uso destas redes em tarefas de reconhecimento de voz e processamento de séries temporais. Dessa maneira, seja h_t a sequência processada no estado t. A RNN processa iterativamente as sequências de entrada, pois

$$h_t = f(h_{t-1}, x_t), \tag{2.93}$$

onde as predições são feitas segundo

$$\hat{Y}_{t+h|t} = g(h_t), \tag{2.94}$$

sendo f e g funções previamente definidas. Além disso, a RNN pode ser parcialmente conectada, ou seja, nem todos os estados podem ser alcançados a partir de um estado

qualquer. Uma das dificuldades da RNN é o problema do *gradiente evanescente* [Goodfellow et al., 2016], motivando a criação de redes de memória curta e de longo prazo (LSTM).

A LSTM consiste em blocos de memória chamados células, que são conectadas por meio de camadas. As informações de uma célula estão contidas no estado C_t e no estado oculto h_t. O fluxo de informação é regulado por portas, que utilizam funções de ativação como a sigmoide (2.71) e a tangente hiperbólica (2.72). Pela definição da sigmoide σ, observe que $\sigma(x) = 0$ indica a negação da passagem de informação, enquanto $\sigma(x) = 1$ permite a passagem de informação. Dessa forma, a sigmoide pode adicionar ou descartar uma informação do estado da célula. No instante t, as portas recebem como entradas os estados ocultos h_{t-1} e a entrada atual X_t. Tanto o estado oculto quanto a entrada são multiplicados pela matriz de pesos W e somados a um viés. A implementação da LSTM é dada a seguir. Destaca-se que os pesos e vieses devem ser estimados ao longo do processo.

Para inicialização, tome $C_0 = 0$ e $h_0 = 0$. A *porta de esquecimento* determina qual informação será desprezada do estado da célula. Sua saída é

$$f_t = \sigma(W_f X_t + W_f h_{t-1} + b_f). \tag{2.95}$$

A função σ é a sigmoide. Caso $f_t = 1$, então a informação será lembrada. Se $f_t = 0$, então a informação será esquecida. A *porta de entrada* é dada por

$$\hat{C}_t = tanh(W_c X_t + W_c h_{t-1} + b_c). \tag{2.96}$$

Calcula-se o *filtro*

$$U_t = \sigma(W_u X_t + W_u h_{t-1} + b_u). \tag{2.97}$$

Atualize o estado C_{t-1}, utilizando os valores da *porta de esquecimento*, *a porta de entrada* e o *filtro*, resultando

$$C_t = f_t C_{t-1} + U_t \hat{C}_t. \tag{2.98}$$

A *porta de saída* fica

$$O_t = \sigma(W_o h_{t-1} + W_o X_t + b_o). \tag{2.99}$$

Em seguida, o estado C_t é passado por meio da função de ativação tangente hiperbólica para escalonar os valores no intervalo $[0, 1]$. Por fim, o estado escalonado é

multiplicado pela saída filtrada para se obter o estado oculto h_t, a ser passado para a próxima célula, em que

$$h_t = O_t tanh(C_t). \tag{2.100}$$

2.10.4 Redes convolutivas

Redes convolutivas são majoritariamente usadas para processamento e classificação de imagens [Mallat, 2016]. Entretanto, em processamento de séries temporais, utiliza-se a convolução 1D [Benidis et al., 2022]. Os *tensores* de entrada para convolução 1D têm a forma (N, L, C), onde N é o número de amostras, L é o comprimento da sequência (número de elementos na sequência), e C é o número de dimensões de cada elemento na sequência. O termo "1D" refere-se à dimensão ao longo da qual a convolução é aplicada. Embora o tensor de entrada possa ter várias dimensões (como número de amostras, comprimento da sequência e número de elementos na sequência), a operação de convolução em si ocorre ao longo de uma única dimensão — o comprimento da sequência. Das propriedades obtidas em camadas convolutivas, temos a *invariância* e *equivariância*. Uma função f é dita *invariante* em relação à uma transformação T se

$$f(T(x)) = f(x). \tag{2.101}$$

Já f é dita *equivariante* em relação à T se

$$f(T(x)) = T(f(x)). \tag{2.102}$$

Em redes convolutivas 1D, cada camada é *equivariante* em relação à translação. Os mecanismos de *pooling* dessas camadas induzem *invariância* de translação. As operações realizadas na camada de *convolução* em uma camada h_i são dadas por

$$h_i = a\left(\beta + \sum_{j=1}^{K} w_j x_{i+j-(K-1)}\right), \tag{2.103}$$

onde a é função de ativação, β é viés, e o somatório indica a operação de *convolução*. O valor K indica o *tamanho do kernel*, em que este é o vetor de pesos que será multiplicado com os valores de entrada da camada. Na prática, a operação do somatório é a *correlação cruzada*, não uma operação de convolução conforme definição tradicionalmente vista no âmbito matemático. Porém, convencionou-se em

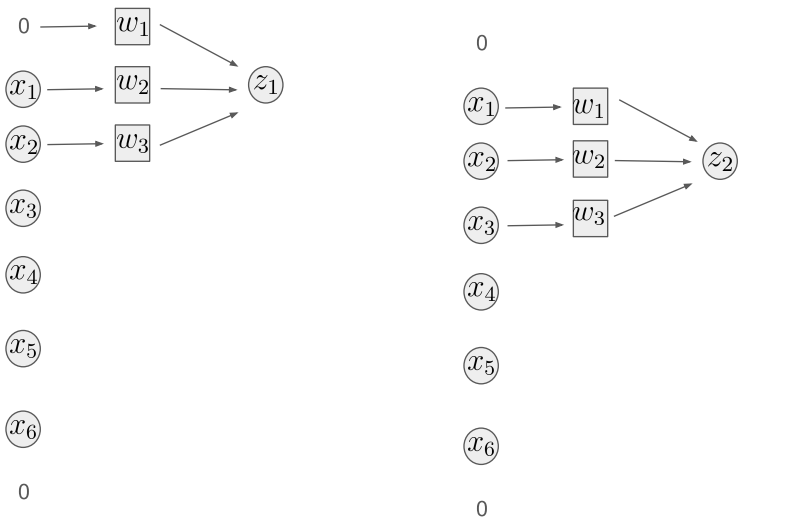


Figura 2.2. Uso de *padding* e *stride* em convolução 1D.

aprendizado de máquina a chamar esta operação de *convolução*. Ademais, perceba que a convolução transforma o vetor de entrada x em um vetor de saída z, de forma que cada saída z_i é uma *soma ponderada* das entradas próximas. Para expor os conceitos de *stride*, *padding* e *pooling*, suponha $K = 3$.

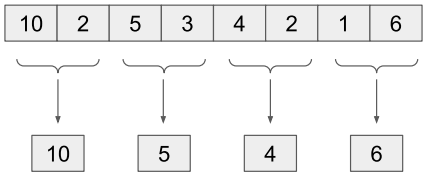
O *stride* (**Figura 2.2**) é o tamanho do passo de deslocamento do kernel ao longo da entrada. Se o *stride* for de tamanho um, o kernel desloca-se em uma unidade ao longo da entrada. O *padding* (**Figura 2.2**) assume entrada nula fora do intervalo válido (modo válido), e o kernel executa convolução considerando este termo nulo. Esta estratégia chama-se *zero padding*.

Em alguns momentos, aumenta-se o *kernel* acrescendo termos nulos entre os termos originais do vetor de pesos. Esta operação é chamada de *dilatação*. Por exemplo, se $w = (w_1, w_2, w_3)^T$, ao transformar w em $w = (w_1, 0, w_2, 0, w_3)^T$, estamos realizando uma *dilatação* com *taxa de dilatação* igual a dois.

Em geral, aplica-se uma quantidade fixa de convoluções em cada camada, onde cada convolução gera um conjunto de variáveis ocultas. Essa quantidade de convoluções aplicadas em uma camada representa o *número de filtros* (ou *canais*) ou o *mapa de características* da camada de convolução. A entrada e as camadas ocultas possuem diversos canais. Se uma camada possui C_i canais e um tamanho de kernel K, então as unidades ocultas de cada canal de saída são computadas como uma soma de todos os canais C_i e K posições de kernel usando a matriz $\Omega \in \mathbb{R}^{C_i \times K}$ e o

viés. Se houver C_0 canais na próxima camada, então precisamos de $\Omega \in \mathbb{R}^{C_i \times C_0 \times K}$ para a matriz de pesos e $\beta \in \mathbb{R}^{C_0}$ como o viés.

Após a aplicação das convoluções, é comum aplicar uma operação de *pooling* para reduzir a dimensionalidade e agregar informações importantes. No caso de convoluções 1D, uma técnica frequentemente utilizada é o *max pooling* 1D. O *max pooling* 1D funciona selecionando o valor máximo dentro de uma janela deslizante de tamanho P ao longo de cada canal. Essa operação reduz a dimensão da entrada, mantendo as características mais importantes, e resulta em uma representação mais compacta dos dados. Se a entrada para a camada de *pooling* possui N elementos em cada um dos C canais, a saída terá aproximadamente N/P elementos por canal, dependendo do tamanho da janela de *pooling* e do *stride* utilizado. Na **Figura 2.3**, $N = 8$ e $P = 2$. Como a janela deslizante é de tamanho $P = 2$, então temos a seleção do maior termo após compararmos dois números adjacentes no vetor de $N = 8$ termos. Ao término, após a aplicação de *max pooling*, teremos um total de $8/2 = 4$ termos.



(a) Max pooling com janela deslizante de tamanho dois

Figura 2.3. Exemplo do uso de *max pooling*.

A vantagem da *camada de convolução* em relação à *camada totalmente conectada* baseia-se no *compartilhamento de parâmetros* [Goodfellow et al., 2016]. Ao invés de ter um número independente de pesos para cada posição de entrada, o uso de um conjunto compartilhado de pesos reduz o número de parâmetros treinados. Com menos parâmetros, a eficiência computacional na fase de treino da camada é melhorada. Além disso, o *compartilhamento de parâmetros* possibilita que a camada de convolução aprenda características independentemente da posição de entrada, in-

duzindo a *invariância translacional* mencionada. Por fim, o aprendizado da rede, assim como em redes totalmente conectadas, também é realizado via *retropropagação*, comentada anteriormente.

2.10.5 Camadas de atenção

Diferentemente das camadas acima comentadas, um *bloco de autoatenção* toma N entradas $x_1, x_2, ..., x_N$ tal que $x_i \in \mathbb{R}^{D\times 1}$, e retorna N vetores do mesmo tamanho. Por exemplo, em *processamento de linguagem natural*, cada entrada x_i poderia ser uma palavra de uma sentença [Goodfellow et al., 2016]. Vejamos o cálculo das saídas de uma *camada de autoatenção*.

Primeiramente, um conjunto de valores v_m é computado para cada entrada,

$$v_m = \beta_v + \Omega_v x_m, \tag{2.104}$$

onde $\beta_v \in \mathbb{R}^{D\times 1}$ e $\Omega_v \in \mathbb{R}^{D\times D}$. A n-ésima saída da camada de atenção $sa_n[x_1, ..., x_N]$ é

$$sa_n[x_1, ..., x_N] = \sum_{m=1}^{N} a[x_m, x_n] v_m. \tag{2.105}$$

O escalar $a[x_m, x_n]$ é a *atenção* que a n-ésimo termo em relação à entrada x_m. Os N pesos $a[\bullet, x_n]$ satisfazem

$$a[\bullet, x_n] \geq 0 \text{ e } \sum_{m=1}^{N} a[x_m, x_n] = 1. \tag{2.106}$$

Para computar a *atenção*, aplicamos duas transformações lineares às entradas,

$$q_n = \beta_q + \Omega_q x_n, \tag{2.107}$$

$$k_n = \beta_k + \Omega_k x_m, \tag{2.108}$$

em que o conjunto $\{q_n\}$ são *consultas* e $\{k_m\}$ são *chaves*. O cálculo do peso $a[x_m, x_n]$ é o *produto interno euclidiano* entre as *consultas* e as *chaves*, cujo o resultado é

passado para a função $softmax$, ou seja,

$$a[x_m, x_n] = softmax[k_m^T q_n] \tag{2.109}$$

$$= \frac{exp(k_m^T q_n)}{\sum_{m'=1}^{N} exp(k_{m'}^T \cdot q_n)}. \tag{2.110}$$

Como o produto interno é uma medida de semelhança entre entrada, então os pesos $a[x_\bullet, x_n]$ dependem das semelhanças relativas entre a n-ésima *consulta* e todas as *chaves*. O uso da função $softmax$ indica que os vetores de *chaves* competem uns com os outros para contribuição do resultado final.

No contexto de séries temporais, considere uma sequência de observações temporais $(x_1, x_2, \ldots, x_N)$, onde cada x_i representa o estado do sistema em um instante i. As camadas de autoatenção permitem que cada ponto no tempo x_n preste atenção a todos os outros pontos x_m da sequência, permitindo que o modelo capture dependências de longo alcance. Para ilustrar, tome a predição da temperatura com base em dados históricos. Cada x_i poderia representar um vetor de características em um determinado tempo i (temperatura, umidade, pressão). Por meio do mecanismo de autoatenção, o modelo pondera a importância de observações passadas para predizer a temperatura futura. Isso é feito calculando as *consultas* e *chaves* a partir das entradas, possibilitando que o modelo se concentre dinamicamente nos períodos relevantes do passado para fazer predições mais precisas. Portanto, o uso de camadas de autoatenção em séries temporais permite modelar as relações temporais complexas e capturar dependências a longo prazo, oferecendo vantagem sobre métodos tradicionais que frequentemente dependem de janelas de tempo fixas ou métodos autorregressivos.

2.10.6 TimeGPT-1

O paradigma do modelo TimeGPT-1 [Garza et al., 2023] consiste em modelos de fundação [Bommasani et al., 2021], ou seja, modelos que são pré-treinados em uma ampla diversidade de dados para obter uma generalização eficaz em bases distintas das usadas no treinamento. Este modelo processa séries temporais com base em mecanismos de autoatenção [Vaswani et al., 2023]. A utilização do modelo pode ocorrer de duas formas: *zero-shot* [Rezaei & Shahidi, 2020], quando o modelo não é pré-treinado na base de dados de interesse, e *few-shot*, que ajusta os parâmetros pré-treinados para se adequarem à base de dados específica. Na arquitetura do TimeGPT-1, utiliza-se uma janela de valores históricos para realizar predições,

incluindo codificação posicional para enriquecer as entradas. Assim, a arquitetura é composta por um conjunto de camadas codificador-decodificador, cada uma com conexões residuais e normalização. Por fim, uma camada linear mapeia a saída do decodificador para um vetor com dimensões correspondentes ao tamanho da janela de predição. A intuição geral do modelo baseia-se na habilidade das camadas de atenção de capturar a diversidade de eventos passados e extrapolar corretamente para distribuições futuras. Ademais, embora o TimeGPT-1 seja um modelo fundacional, é importante ressaltar que não se trata de um Modelo Largo de Linguagem (LLM), já que seu treinamento é realizado com séries temporais e não com linguagem humana.

2.10.7 MOIRAI

Continuando no paradigma de modelos fundadores, o *Masked Encoder-based Universal Time Series Forecasting Transformer* (MOIRAI) [Woo et al., 2024] propõe-se a lidar com a vasta heterogeneidade de séries temporais, abordando frequências temporais distintas, séries univariadas e multivariadas, e séries com diferentes distribuições. Inicialmente, as séries temporais são divididas em pacotes para preservar o domínio temporal, segmentando-as em diversos pedaços. Se a série for multivariada, ela é achatada, convertendo-se em univariada. Esses pacotes são então transformados em vetores de *embedding* por meio de uma camada de projeção que suporta diferentes tamanhos de pacotes. Posteriormente, aplica-se o mascaramento (*masking*) em alguns desses vetores de *embedding*. Os pacotes mascarados são encaminhados para a camada de codificação, que utiliza camadas de atenção para focar em diferentes partes da entrada, aprendendo relações temporais e padrões na série. Finalmente, os tokens de saída são decodificados através da camada de projeção de saída, gerando parâmetros para a mistura de distribuições que resultará na predição final. Importante destacar que os autores empregam a normalização RMSNorm [Zhang & Sennrich, 2019] tanto na entrada quanto na saída, além da função de ativação SwiGLU [Shazeer, 2020] para o aprendizado de padrões não lineares.

2.11 Métricas

Para avaliar os modelos, seis métricas foram utilizadas: Erro Absoluto Médio (MAE), Percentual de Erro Médio Absoluto (MAPE), a Raiz Quadrada do Erro Quadrático Médio (RMSE), Viés Médio (BIAS) e R^2. Nas definições abaixo, considere y_i como os valores reais, $\hat{y}_i$ como os valores preditos e n como o número total

de observações. Dessa forma, a MAE,

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|, \tag{2.111}$$

é a média dos valores absolutos das diferenças entre as predições e os valores reais. Ou seja, é a média da magnitudes dos erros, sem considerar a direção dos vetores. A MAPE,

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|, \tag{2.112}$$

é a média percentual dos valores absolutos das diferenças entre as predições e os valores reais. Isto é, é o percentual da média das magnitudes dos erros, sem considerar a direção dos vetores. A RMSE,

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{2.113}$$

é a raiz quadrada da métrica MSE. Serve para trazer as unidades de medida de volta à escala original do valor alvo. A principal diferença entre a MSE e a RMSE é a interpretabilidade, pois como RMSE está na mesma unidade que a variável alvo, então há maior facilidade na interpretação da métrica.

O problema com MAE e RMSE consiste no fato de que essas métricas não são livres de escala, o que promove uma má interpretabilidade ao avaliar o desempenho do modelo usando mais de um conjunto de dados. Portanto, para evitar isso, usamos RMSE escalada e MAE escalada. Assim, RMSSE [Makridakis et al., 2022] é definido como

$$\text{RMSSE} = \sqrt{\frac{\frac{1}{h}\sum_{t=n+1}^{n+h}(y_t - \hat{y}_t)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(y_t - y_{t-1})^2}} \tag{2.114}$$

onde h é o horizonte de predição. Particularmente, RMSSE possui várias vantagens quando os processos de treinamento e validação de um modelo envolvem conjuntos de dados de diferentes escalas. Ela é livre de escala: como RMSSE é escalada pela variabilidade inerente à série temporal, permite comparações diretas entre modelos treinados em diferentes séries temporais. Ela normaliza o erro: dividindo pelo erro

da predição ingênua nos dados de treinamento, a RMSSE considera a variabilidade natural da série temporal, fornecendo uma medida de erro que é robusta a diferentes amplitudes e escalas das séries temporais. RMSSE é facilmente interpretável: valores iguais a um indicam que o modelo não adiciona informações significativas além da predição ingênua, valores menores que um indicam modelos com menos erros do que a predição ingênua, e valores maiores que um indicam predições piores do que a predição ingênua. Estabilidade: devido à raiz quadrada, RMSSE é menos sensível a *outliers*. Analogamente, MASE [Hyndman & Koehler, 2006],

$$\text{MASE} = \frac{\frac{1}{h}\sum_{t=n+1}^{n+h}|y_t - \hat{y}_t|}{\frac{1}{n-1}\sum_{t=2}^{n}|y_t - y_{t-1}|} \tag{2.115}$$

tem propriedades semelhantes à RMSSE, mas MASE é menos sensível a *outliers* do que RMSSE, uma vez que os termos de erro não são elevados ao quadrado. A interpretação das métricas RMSSE e MASE é a mesma. Uma observação importante sobre RMSSE e MASE é que ambas consideram a estrutura temporal dos dados ao dividir pela predição ingênua da fase de treinamento. Já o BIAS,

$$\text{BIAS} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\,, \tag{2.116}$$

mede o desvio dos valores preditos em relação aos valores reais. Esta métrica quantifica a tendência média do modelo em subestimar ou superestimar valores na predição. Um valor positivo indica superestimação. Um valor negativo indica subestimação. A métrica R^2, onde VAR indica a variância da amostra $\{y_1, y_2, ..., y_n\}$,

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\text{MSE}}{\text{VAR}}, \tag{2.117}$$

é uma medida estatística usada na análise de regressão para avaliar a adequação de um modelo. Ela fornece uma indicação de quão bem as variáveis independentes explicam a variância na variável dependente. Em outras palavras, R^2 mede o quão próximo os dados estão da linha de regressão ajustada. Um valor de R^2 igual a 0 indica que o modelo não explica nada da variabilidade dos dados em torno da média, enquanto um valor de R^2 igual a 1 indica que o modelo explica toda a variabilidade.

3

Trabalhos Relacionados

Neste capítulo, apresentamos uma revisão da literatura relacionada a este trabalho. Observamos que a pesquisa predominante tende a se concentrar em abordagens isoladas, utilizando métodos de estatística clássica, técnicas de aprendizado de máquina raso, ou aprendizado de máquina profundo. Além disso, há estudos que exploram a integração de métodos estatísticos clássicos com técnicas de aprendizado de máquina, buscando aproveitar as forças complementares dessas abordagens.

3.1 Trabalhos utilizando estatística clássica

Buczak et al. [2018] utilizaram um *ensemble* de três modelos distintos: o Método de Análogos Bidimensionais (MAB), que incorpora dados climáticos e o número de casos semanais de dengue; Holt-Winters, com componente aditivo e sazonal, aplicado tanto aos dados brutos quanto aos suavizados pela *wavelet* sym7; e Modelos Históricos Simples (MHS). Um conjunto de hiperparâmetros foi testado para cada modelo, e os melhores compuseram o preditor final. O conjunto de dados incluiu informações geográficas e o número de casos semanais de dengue das cidades de San Juan e Iquitos. As métricas utilizadas foram RMSE e Erro Médio Absoluto Relativo (MARE). As principais contribuições do estudo são a acurácia na predição do total de casos semanais e da altura do pico da série temporal para os dados de Iquitos.

Deb & Deb [2021] desenvolveram um *ensemble* de modelos composto por negative binomial regression model (NBM), ARIMA sazonal (SARIMA) e Modelo Linear Autorregressivo de Médias Móveis Generalizado (GLARMA) para analisar o número de casos semanais de dengue nas cidades de San Juan, em Porto Rico, e Iquitos, no Peru. Dados climáticos e informações sobre a vegetação (condições do terreno) foram prioritariamente utilizados. No pré-processamento, os autores começaram

preenchendo os dados faltantes nas séries temporais por meio de interpolação. Em seguida, aplicaram uma técnica de janelamento, considerando até quatro *lags* temporais, o que resultou na geração de um novo conjunto de variáveis. Para enfrentar o aumento potencial em multicolinearidade e sobreajuste decorrentes da introdução de muitas novas covariáveis, os autores implementaram um passo de seleção de variáveis em duas etapas. Primeiramente, utilizaram o critério de Fator de Inflação de Variância (VIF), com um limiar de 10, para eliminar variáveis que indicassem sinais de multicolinearidade. Posteriormente, adotaram o método de seleção de variáveis passo a passo em um modelo logístico contendo todos os preditores, visando escolher o conjunto final de características relacionadas ao clima e ao terreno. Para a escolha dos hiperparâmetros dos modelos SARIMA e GLARMA, foi utilizado o critério de informação de Akaike (AIC). Como resultado, os autores alcançaram um erro absoluto médio (MAE) inferior a 10 para predições que se estendiam por um período de oito semanas à frente.

3.2 Trabalhos com modelos rasos de aprendizado de máquina

Wu et al. [2008] desenvolveram um modelo para a predição do número de casos de dengue em Singapura. A base de dados consistia em 312 observações, cada uma contendo nove informações: data de registro, pluviometria semanal, nível de nublagem, temperaturas máxima, mínima e média, umidade e o número de casos semanais de dengue. A estratégia de janelamento adotada foi a *one-step*, com um lag temporal de tamanho 4. Inicialmente, os autores processaram cada série temporal por meio de uma transformada *wavelet*, com o objetivo de segmentar o sinal. Posteriormente, utilizaram um algoritmo genético, que segue os princípios da seleção natural e da genética e que incorpora validação cruzada para refinar a seleção de fatores, com o apoio de SVM, para selecionar os componentes mais significativos dos sinais. Após esta etapa de seleção de características, os autores compararam o desempenho de uma Regressão Linear e SVR. Para avaliar os resultados, utilizaram as métricas MSE e R^2. Entre as contribuições do estudo, os autores destacaram que a *wavelet* de Daubechies (db6) proporcionou melhor desempenho preditivo. Além disso, ressaltaram que a pluviometria semanal e a umidade possuem uma estreita relação com o número de casos semanais de dengue.

Guo et al. [2017] reuniram casos semanais de dengue, consultas de busca do Baidu e fatores climáticos (temperatura média, umidade relativa e chuva) de 2011 a 2014 da província de Guangdong, China. Com base nesses dados, algoritmos de

aprendizado de máquina foram testados para predizer a incidência de dengue, incluindo Support Vector Regression (SVR), Step-down Linear Regression, Gradient Boosting Regressor (GBM), Negative Binomial Regression Model (NBM), LASSO e Generalized Additive Model (GAM). Utilizou-se predição *one-step*. O desempenho desses modelos foi avaliado usando a Raiz Quadrada do Erro Quadrático Médio (RMSE) e medidas R-quadrado. Por fim, os autores ressaltaram que o algoritmo SVR realizou a predição com precisão das epidemias durante as últimas 12 semanas do período analisado pelo estudo, bem como o pico do grande surto de 2014, apresentando as menores taxas de erro em comparação aos demais algoritmos.

3.3 Trabalhos com modelos profundos de aprendizado de máquina

Mussumeci & Coelho [2020] compararam modelos de aprendizado de máquina, como LASSO (*Least Absolute Shrinkage and Selection Operator*), *Random Forest* e uma rede neural recorrente profunda (LSTM), para predizer a incidência semanal de dengue em 790 cidades no Brasil. Os modelos usaram séries temporais multivariadas como entrada para o conjunto de preditores e também incorporaram dados de cidades similares para capturar o componente espacial da transmissão da doença. Além disso, os autores afirmaram que LSTM mostrou o melhor desempenho na predição da incidência futura de dengue em cidades de diferentes tamanhos. O estudo conclui que modelos de aprendizado profundo, como o LSTM, podem ser usados com bom desempenho em problemas de predição em larga escala, levantando a expectativa da adoção mais ampla de modelos de aprendizado de máquina para a gestão de doenças sazonais.

Shaikh et al. [2023] propuseram a criação de um sistema de aviso prévio para predizer dengue em San Juan e Iquitos, além de prover um sistema inteligente que apresenta possíveis medidas preventivas para os pacientes da região. Um pré-processamento é feito na base de dados para retirar *outliers* e dados faltantes. Posteriormente, é feita uma seleção de características utilizando o algoritmo heurístico *Neighbour Count-based Dragonfly Electric Fish Optimization* (NC-DEFO). Tais características são levadas ao comitê de classificadores formado pelos modelos ANN, CNN e SVM. Uma vez que a predição para febre de dengue é realizada, o sistema informa possíveis medicamentos para fortalecer o sistema imunológico e prescrições médicas. Os autores utilizaram as métricas RMSE, norma L2, norma L1, MAE, norma L-infinito, SMAPE, MASE e MEP para avaliação dos modelos. Por fim, concluem que seu comitê de classificadores intitulado Optimized Ensemble

Classifier (OEC) obtém superioridade de predição em relação ao ANN, CNN, SVM, LSTM-RF e o comitê de classificadores (sem uso de NC-DEFO) destes três últimos modelos.

Panja et al. [2023] propuseram um conjunto (*ensemble*) de redes neurais autorregressivas chamado XEWNet (*External Ensemble Wavelet Neural Network*). Essas redes processam séries temporais discretas previamente tratadas pela transformada *wavelet* de Haar. Após o processamento, os valores gerados pelas redes são associados aos níveis de precipitação da região em estudo, resultando na predição final. A precipitação da região demonstrou uma alta causalidade de Granger em relação ao número de casos semanais de dengue, motivando o uso dessa característica como componente auxiliar ou externo do *ensemble*. No total, os autores usaram séries temporais contendo informações geográficas e registros semanais de casos de dengue das regiões de San Juan, Iquitos e Ahmedabad. A preparação dos dados envolveu a aplicação de previsões de curto e longo prazo, com janelas temporais de 26 e 52 semanas visando predição *one-step*, de tal sorte que a janela de 26 semanas apresentou os melhores resultados nas métricas de RMSE e MAE, com valores de 7,69 e 5,66, respectivamente, para a cidade de San Juan.

Bogado et al. [2023] discutiram a aplicação de redes neurais do tipo LSTM para predizer o número de casos semanais de dengue no Paraguai. No entanto, os autores destacaram que a precisão desses modelos pode ser reduzida em áreas com menor número de casos, devido à necessidade de mais dados. Para mitigar esse problema, realizaram-se abordagens de *clustering* (k-Means, DBScan e agrupamento hierárquico) em séries temporais, avaliando a qualidade do agrupamento em 217 cidades do Paraguai. De forma específica, os dados são do período de janeiro de 2009 até dezembro de 2013, e apresentam informações semanais sobre a temperatura mínima, média e máxima; valor mínimo, médio e máximo de pressão atmosférica; taxa pluviométrica; velocidade mínima, média e máxima do vento e nebulosidade. Os autores compararam a eficácia do uso de dados brutos e dados agrupados, concluindo que o agrupamento hierárquico com a correlação de Spearman, ao pré-processar os dados, obteve melhores resultados quando utilizados como entrada para a LSTM. As métricas utilizadas pelos autores foram RMSE, Desvio Padrão do Erro Percentual (STDERR) e Erro Percentual Máximo (MAXERR). Por fim, ao comparar modelos LSTM construídos a partir dos resultados do agrupamento, o erro quadrático médio confirmou que os modelos agrupados aumentam a precisão em 19,48 $\pm$ 18,80 %.

Necesito et al. [2021] utilizaram um modelo profundo univariado composto por camadas LSTM para a previsão do número de casos semanais de dengue. A base de dados foi composta pelo número de casos mensais de dengue em Manila,

capital das Filipinas, no período de janeiro de 1994 a dezembro de 2018, totalizando 300 observações. A estratégia de janelamento adotada envolveu *lags* temporais de tamanho zero e um. Na fase de pré-processamento, os autores eliminaram ruídos da série temporal usando uma transformada *wavelet* discreta; no entanto, o tipo de *wavelet* utilizado não foi especificado. Para avaliar a eficácia da rede proposta, foram utilizadas as métricas RMSE, NSE (Eficiência de Nash-Sutcliffe), CC (Coeficiente de Correlação), KGE (Eficiência de Kling-Gupta) e o Índice de Concordância. Uma das limitações do estudo é que a modelagem se restringiu a uma abordagem univariada, não considerando fatores geográficos que influenciam a manifestação da dengue.

Sebastianelli et al. [2024] propuseram um *ensemble* de modelos de aprendizado de máquina (Catboost, SVR, LSTM) para predizer a taxa do índice de dengue na população com faixa etária abaixo de dezenove anos. O conjunto de dados de treinamento consistiu no número de casos de dengue nas 27 unidades federativas brasileiras, enquanto o conjunto de dados de teste incluiu os casos de dengue das regiões de Loreto e Madre de Dios, localizadas no Peru. Para o pré-processamento, os autores utilizaram uma CNN para obter a correlação espacial entre as unidades federativas do Brasil, adicionando os valores obtidos como uma nova característica ao conjunto de dados. Além disso, empregaram a técnica de Mínimos Quadrados Parciais (PLS) para a redução da dimensionalidade dos dados. Em seguida, combinaram as predições de cada modelo no conjunto para obter a predição final. Entre as contribuições do estudo, destaca-se o bom desempenho do modelo na métrica nRMSE em uma região distinta (Peru) do conjunto original de treinamento (Brasil), evidenciando um modelo com maior potencial de generalização. Adicionalmente, a construção de um conjunto de dados com informações de múltiplas fontes também se apresenta como uma contribuição relevante do estudo.

Ferdousi et al. [2021] construíram um *framework* que integra metodologias quantitativas para a seleção de dados sobre a incidência de dengue em localidades próximas, visando criar características adicionais em modelos preditivos. Estas foram elaboradas com base na correlação cruzada com *lag* temporal e métricas de distância e prevalência, definidas no estudo, com o objetivo de identificar possíveis relações causais entre os surtos. Utilizando dados do projeto *InfoDengue*, que rastreia o número de casos de dengue em 700 municípios brasileiros, os autores avaliaram o desempenho preditivo de redes neurais recorrentes dos tipos GRU e LSTM, em comparação com uma ANN (*baseline*). Os modelos foram testados com uma estratégia de janelamento *multi-output*, onde o tamanho da janela de entrada usada foi de oito semanas e a janela de saída foi de quatro semanas, demonstrando eficácia em predizer o número de casos de dengue com quatro semanas de antecedência, a

partir dos dados das oito semanas anteriores. A métrica avaliada no trabalho foi a MAE. Como resultados, as redes neurais recorrentes obtiveram um percentual de pelo menos 41% de superioridade preditiva em relação ao *baseline*.

3.4 Trabalhos utilizando tanto estatística clássica quanto aprendizado de máquina

Carvajal et al. [2018] compararam a eficiência dos modelos GAM, ARIMA, SARIMA, RF e GBR na predição do número de casos de dengue em Manila, Filipinas. O conjunto de dados foi construído a partir de 260 observações do período de 1º de janeiro de 2009 até 31 de dezembro de 2013, incluindo dados meteorológicos como nível de inundação, precipitação, temperatura, umidade relativa, velocidade e direção do vento, Índice de Oscilação Sul (pressão atmosférica) e a incidência semanal de dengue em Manila. Durante o pré-processamento, os autores aplicaram a correlação de Pearson para identificar características com correlação positiva com o número de casos de dengue. O modelo RF, utilizando janelas temporais de tamanho 25 e focando principalmente no nível de umidade, apresentou desempenho superior nas métricas MAE e RMSE em comparação com os demais modelos. Além disso, os autores destacaram a possível interferência da vacinação contra dengue nos números de casos. Como limitação, apontaram que o potencial de generalização do estudo é limitado devido ao número restrito de observações e à aplicação do modelo em uma única localidade.

Chakraborty et al. [2019] realizaram comparação univariada com estratégia de janelamento *multi-step* entre os modelos ARIMA, SVM, ANN, LSTM e NNAR (modelo que utiliza observações passadas da série temporal como entrada em uma ANN), e os modelos híbridos ARIMA-SVM, ARIMA-ANN, ARIMA-LSTM e ARIMA-NNAR na predição de séries temporais que representam o número de casos semanais de dengue nas regiões de Iquitos – Peru, San Juan – Porto Rico e Filipinas. A estratégia dos modelos híbridos consiste no fato de que ARIMA capta a linearidade da série temporal, enquanto SVM, ANN, LSTM e NNAR captam a não linearidade, onde esta foi obtida através dos resíduos obtidos na predição do modelo ARIMA. As métricas de avaliação do trabalho foram RMSE, MAE e SMAPE. O modelo com melhor desempenho foi o ARIMA-NNAR.

Appice et al. [2020] propuseram o modelo de aprendizado de máquina AutoTiC-NN (*Autoencoding based time series clustering with nearest neighbour*) composto por quatro estágios: camadas de codificação e decodificação, janelamento usando a temperatura das regiões e o número de casos totais de dengue, agrupamento

(*clustering*) de séries temporais e predição do vizinho mais próximo. O conjunto de dados utilizado é proveniente da temperatura e do número de casos mensais de dengue de 32 estados do México. Além disso, houve comparação do AutoTiC-NN com regressores como Auto ARIMA, VAR (Autorregressão Vetorial), M5', SVR e kNN. Por fim, os autores ressaltam que o modelo proposto obteve superioridade na métrica RMSE.

Tabela 3.1. Resumo dos trabalhos relacionados.

Autor	Local	Métrica	Melhor Modelo
Buczak et al. [2018]	San Juan e Iquitos	RMSE, MARE	MAB-MHS-HW (sym7)
Deb & Deb [2021]	San Juan e Iquitos	MAE	NBM-SARIMA-GLARMA
Wu et al. [2008]	Singapura	MSE, R^2	SVR (db6)
Guo et al. [2017]	China	RMSE, R^2	SVR
Mussumeci & Coelho [2020]	Brasil	Erro Médio	LSTM
Shaikh et al. [2023]	San Juan e Iquitos	RMSE, MAE, MASE	CNN-ANN-SVM
Panja et al. [2023]	San Juan, Iquitos e Ahmedabad	RMSE, MAE	XEWNet (haar)
Bogado et al. [2023]	Paraguai	RMSE, Spearman	LSTM-*clustering*
Necesito et al. [2021]	Filipinas	RMSE, NSE, CC, KGE	LSTM
Sebastianelli et al. [2024]	Brasil e Peru	nRMSE	Catboost-SVR-LSTM
Ferdousi et al. [2021]	Brasil	MAE	GRU, LSTM
Carvajal et al. [2018]	Filipinas	MAE, RMSE	RF
Chakraborty et al. [2019]	San Juan, Iquitos e Filipinas	RMSE, MAE, SMAPE	ARIMA-NNAR
Appice et al. [2020]	México	RMSE	AutoTiC-NN

3.5 Síntese dos Trabalhos Relacionados

Em geral, os autores concentram-se em abordagens que envolvem exclusivamente o uso de modelos estatísticos, aprendizado de máquina raso, a combinação de modelos estatísticos clássicos com aprendizado de máquina ou modelos profundos de aprendizado de máquina. A maioria dos estudos não adota o pré-processamento por *wavelets* em suas metodologias, limitando assim a possibilidade de comparações preditivas amplas sobre as melhorias ou deteriorações decorrentes do uso de *wavelets*. Além disso, muitos autores não detalham as estratégias de janelamento utilizadas, o que compromete a reprodutibilidade dos estudos.

Buczak et al. [2018] e Deb & Deb [2021] não exploraram técnicas de aprendizado profundo nas regiões de San Juan e Iquitos, o que impede a realização de comparações entre modelos estatísticos clássicos e modelos de aprendizado de máquina.

Dos trabalhos de aprendizado de máquina raso, Wu et al. [2008] utilizaram modelos rasos e *wavelets*, porém seus resultados carecem de generalização devido à pequena base de 312 instâncias. Já Guo et al. [2017] analisou modelos de aprendizado de máquina raso na mesma base em que havia treinado, não permitindo concluir se seu modelo possuía generalização para outros *datasets*.

Dos trabalhos sobre modelos profundos, Mussumeci & Coelho [2020] analisaram 790 cidades do Brasil, mas sua avaliação em métricas baseou-se apenas no erro médio, sem fornecer uma análise abrangente em outras métricas. Além disso, a validação de seus resultados foi realizada apenas em cidades brasileiras, limitando o modelo ao cenário brasileiro.

Shaikh et al. [2023] se destaca pelo pré-processamento com algoritmos genéticos e uso de *ensembles*, mas a validação no conjunto reduzido de San Juan e Iquitos não foi detalhada, assim como a estratégia de reformatação do *dataset* não foi descrita, dificultando a reprodutibilidade dos resultados e a comparação com outros modelos.

Carvajal et al. [2018] treinou e validou seus modelos em uma pequena base (300 observações), o que compromete a generalização de seus resultados. De igual maneira, Necesito et al. [2021] possui uma base pequena para treino e validação, o que limita a expansão de seus resultados.

Panja et al. [2023], Bogado et al. [2023], Sebastianelli et al. [2024], Appice et al. [2020], Ferdousi et al. [2021] e Chakraborty et al. [2019] treinaram e validaram seus métodos em *datasets* de mesma localidade, não explorando o uso de *cross-learning*, prejudicando a generalização do modelo devido ao viés constante do *dataset* de treino.

Por fim, nenhum desses trabalhos explora o uso de modelos fundadores, o que impede uma comparação abrangente com os novos paradigmas na interseção entre aprendizado de máquina e a área de predição de séries temporais.

4

Abordagem Proposta

Neste capítulo, apresentamos as bases de dados de San Juan e das unidades federativas brasileiras (InfoDengue), além de detalharmos as etapas de pré-processamento. Para a base de San Juan, exploramos a seleção de características por meio da correlação cruzada e o uso de transformadas *wavelet* para suavização das séries temporais. Na base InfoDengue, aplicamos uma abordagem análoga, com ajustes específicos devido à menor disponibilidade de características. Além disso, abordamos diferentes estratégias de modelagem, incluindo predições *one-step*, *multi-step*, e o método de *cross-learning* com *leave-one-out*, destacando a engenharia de características realizada antes do uso dos modelos.

4.1 Base de Dados

Para o desenvolvimento deste trabalho, utilizaram-se duas bases de dados: a base de dados de San Juan [US National Oceanic and Atmospheric Administration, 2017] e a base InfoDengue [Codeco et al., 2018].

A base de dados de San Juan é formada da quantidade de casos de dengue reportados semanalmente na cidade de San Juan (Porto Rico) no período de 1990 a 2010. A base possui formato tabular, composta por 936 instâncias, constituída de 24 atributos, em que a maioria dos atributos representa uma série temporal. De forma explícita, os atributos são provenientes de quatro fontes distintas: GHCNd (*Global Historical Climatology Network daily*) – banco de dados integrado de resumos climáticos diários de estações de superfície terrestres ao redor do mundo; PERSIANN – algoritmo de recuperação de precipitação baseado em satélite que fornece informações de precipitação quase em tempo real usando redes neurais; NOAA NCPE (National Centers for Environmental Prediction); e NDVI (Índice de Vegetação de Diferença Normalizada) da NOAA. Abaixo listamos os atributos usados por fonte:

- GHCNd: cidade em questão, semana de registro, ano de registro, temperatura máxima, temperatura mínima, temperatura média, total de precipitação em milímetros e faixa de temperatura diurna;

- PERSIANN: precipitação total em milímetros;

- NOAA NCPE: precipitação total em milímetros, temperatura média do ponto de orvalho, temperatura média do ar, umidade relativa média, umidade específica média, precipitação em quilograma por metro quadrado, temperatura máxima do ar, temperatura mínima do ar, temperatura, média do ar e faixa de temperatura média do período diurno;

- NDVI: quantidade de pixels a sudeste do centróide da cidade, quantidade de pixels a sudoeste do centróide da cidade, quantidade de pixels a nordeste do centróide da cidade e quantidade de pixels a noroeste do centróide da cidade, representando variações na vegetação em San Juan.

Em particular, o NDVI é um indicador utilizado para medir a densidade da vegetação em uma área baseada na reflectância de diferentes comprimentos de onda de luz. A reflectância é a proporção de energia radiante refletida por uma superfície em relação à energia radiante incidente. Explicitamente,

$$\mathrm{NDVI} = \frac{\mathrm{NIR} - \mathrm{RED}}{\mathrm{NIR} + \mathrm{RED}}, \tag{4.1}$$

onde NIR (*near infrared*) é a reflectância no espectro do infravermelho próximo, fortemente refletida pela vegetação saudável, e RED (*red*) é a reflectância no espectro do vermelho visível, absorvida pela clorofila nas plantas.

A interpretação do NDVI consiste na variação de valores entre -1 e 1. Os valores próximos a 1 indicam vegetação densa e saudável (alta reflectância no NIR e baixa reflectância no RED). Valores próximos a 0 indicam pouca ou nenhuma vegetação, como rochas, solo exposto ou neve. Valores negativos indicam superfícies como água, neve, ou áreas não vegetadas.

Em suma, NDVI é sensível à quantidade de biomassa e permite o monitoramento de áreas de cultivo, florestas e ecossistemas naturais, além de ser usado para detectar mudanças ao longo do tempo.

InfoDengue é um sistema brasileiro de alerta de arbovírus, utilizando dados híbridos gerados pela análise integrada de dados extraídos das redes sociais, juntamente com dados climáticos e epidemiológicos. Todas as cidades do Brasil são

elegíveis para participar. Atualmente, o sistema inclui 5.570 cidades registradas, abrangendo 26 capitais e o Distrito Federal (Brasília). A coleta de dados começou na primeira semana de 2010. Os principais conteúdos de cada conjunto de dados usados em nossas análises incluem o número de casos semanais de dengue, dados meteorológicos e dados demográficos. Especificamente, nos concentramos nas seguintes características, uma vez que outros pontos de dados disponíveis referem-se ao processo de modelagem usado pelos autores, que não são relevantes para o nosso estudo. As características usadas em nossa análise são:

- data inicial: Primeiro dia da semana epidemiológica;
- casos: Número de casos notificados;
- temperatura mínima: Temperaturas mínimas médias da semana;
- temperatura máxima: Temperaturas máximas médias da semana;
- umidade mínima: Umidade relativa mínima diária média ao longo da semana;
- umidade média: Umidade relativa diária média ao longo da semana;
- umidade máxima: Umidade relativa máxima diária média ao longo da semana;
- temperatura média: Temperaturas diárias médias ao longo da semana.

4.2 Pré-processamento da base de San Juan

Primeiro, os dados ausentes são preenchidos por meio de interpolação polinomial. Em seguida, apenas a coluna de casos totais de dengue é selecionada para a aplicação do Z-score [Kreyszig, 2010]. Logo, cada instância dessa coluna é subtraída da média dos valores da coluna e dividida pelo desvio padrão. Após este processo, as instâncias com Z-score maior que -3 e menor que 3 são selecionadas. Esta abordagem identifica valores aberrantes (*outliers*) ao excluir pontos que estão a mais de três desvios padrão da média.

Uma série temporal é caracterizada por sua tendência, sazonalidade e resíduo [Morettin & Toloi, 2018]. Algoritmos estatísticos clássicos como ARIMA, exigem estacionariedade da série por hipótese. Portanto, aplica-se o Teste Aumentado de Dickey-Fuller [Fuller, 1976] para verificação da estacionariedade das séries.

Com este simples pré-processamento, alguns modelos são testados e verificamos, após a visualização gráfica da série temporal do total de casos e dos valores preditos, que certos pontos não são captados pelos modelos, por mais que um ajuste

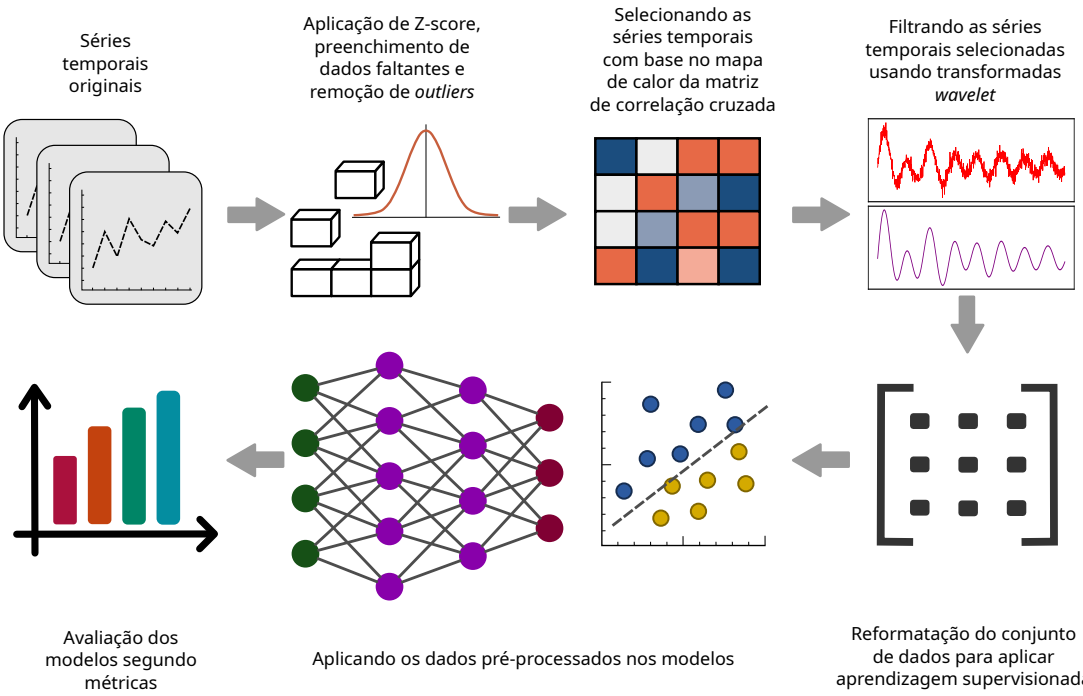


Figura 4.1. Visão geral do experimento. Primeiramente, aplicamos normalização Z-score e removemos instâncias com valores abaixo de -3 ou acima de +3 desvios padrão. Em seguida, selecionamos características com base na matriz de correlação entre as características e o total semanal de casos de dengue. Após isso, as séries temporais selecionadas foram filtradas usando *wavelets*. Então, reformatamos o conjunto de dados para previsões *one-step* e *multi-step*. Finalmente, aplicamos os dados aos modelos e avaliamos os resultados usando métricas de desempenho.

fino fosse realizado. Assim, ainda no processo de retirada de *outliers*, há a extração manual destes pontos aberrantes na base de dados de San Juan. Com este tratamento inicial feito, passamos para formatação dos dados nos modelos testados.

Para os modelos estatísticos clássicos abordados, como Holt-Winters ou Suavização Exponencial, modelo linear de Holt, Suavização Exponencial Simples, ARIMA e Passeio Aleatório, tomamos apenas a coluna de casos totais dos conjuntos de dados, porque esses modelos realizam predição apenas em contexto univariado [Morettin & Toloi, 2018].

Os modelos rasos e profundos de aprendizado de máquina necessitam de reformatação da base de dados para aplicação de aprendizado supervisionado. A literatura aborda três estratégias principais: a predição *one-step*, *multi-step* e a *multi-output* [Benidis et al., 2022]. Ressalta-se que a estratégia *multi-ouput* possui maior relevância na predição de múltiplas características simultaneamente, fato não explorado neste trabalho, pois desejamos apenas a predição da coluna de casos semanais de dengue. Dentro destas estratégias, duas abordagens são testadas, sendo estas, a abordagem inocente (univariada) e a não inocente (multivariada). De um

lado, a inocente consiste na construção de janelas temporais baseadas no uso exclusivo da coluna de casos totais semanais de dengue. Por outro lado, a não inocente usa mais de uma coluna para formar a janela temporal, na qual cada elemento da janela representa um vetor de dimensão correspondente ao número de características escolhidas.

De maneira intuitiva, para a formatação supervisionada desejada, necessitamos formar duas matrizes denotadas por X (matriz formada pelas janelas temporais) e Y (a matriz com os valores alvos, formada por valores da coluna de casos totais e/ou mais colunas). De maneira precisa, seja $T > 0$ o tamanho da janela, $c > 0$ o número de colunas (características selecionadas) e $n > 0$ o número de instâncias do conjunto de dados.

No caso *one-step* (**Figura 4.2**), $X \in \mathbb{R}^{(n-T)\times T\times c}$ é a matriz de janelas temporais, onde temos $n-T$ janelas, em que cada janela tem tamanho T e cada elemento da janela tem tamanho c (c -upla). A matriz $Y \in \mathbb{R}^{(n-T)\times 1}$ é a coluna alvo, formada apenas pelos casos totais.

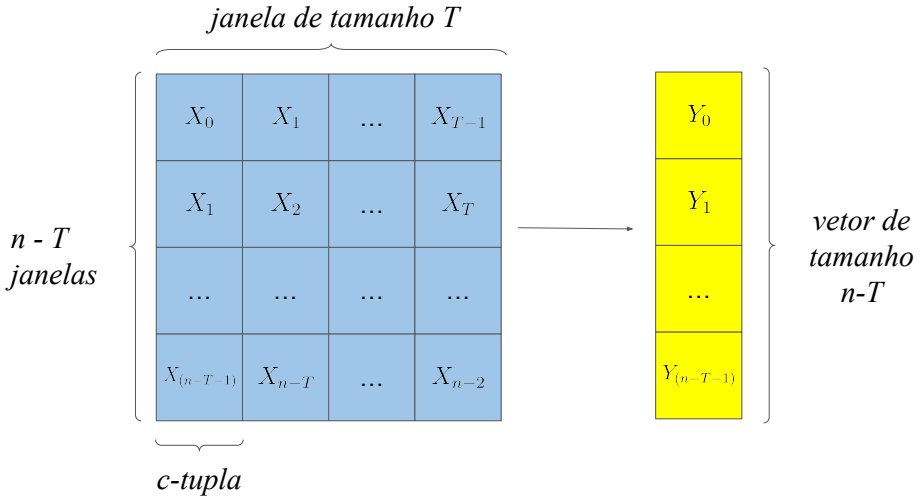


Figura 4.2. Abordagem *one-step*. Cada linha da matriz de rótulos (colorida de azul) contém T *timesteps*, o que reflete o tamanho da janela escolhida. Cada j-ésimo *timestep* X_j é formado por uma c-upla, onde c representa o número de atributos selecionados. Cada linha na coluna alvo (colorida de amarela) corresponde ao valor a ser predito utilizando, respectivamente, cada linha da matriz de rótulos, ou seja, uma janela de tamanho T. Note o deslizamento de uma unidade em cada janela, o que justifica o nome *one-step*.

Para o caso *multi-step* (**Figura 4.3**), aproveitamos o treinamento do modelo feito de maneira *one-step*. Inicialmente, define-se o vetor $J \in \mathbb{R}^{T\times c}$ como a primeira janela dos dados de teste. Realiza-se a predição do valor $P \in \mathbb{R}^c$ com base em J. Em seguida, cada elemento de J com índice i é deslocado para o índice $i-1$, em que o primeiro elemento de J ocupará agora a última posição de J. Ao término,

substitua o elemento da última posição do vetor J com o valor P e repita o processo até que chegue na quantidade de amostras de teste previamente definida.

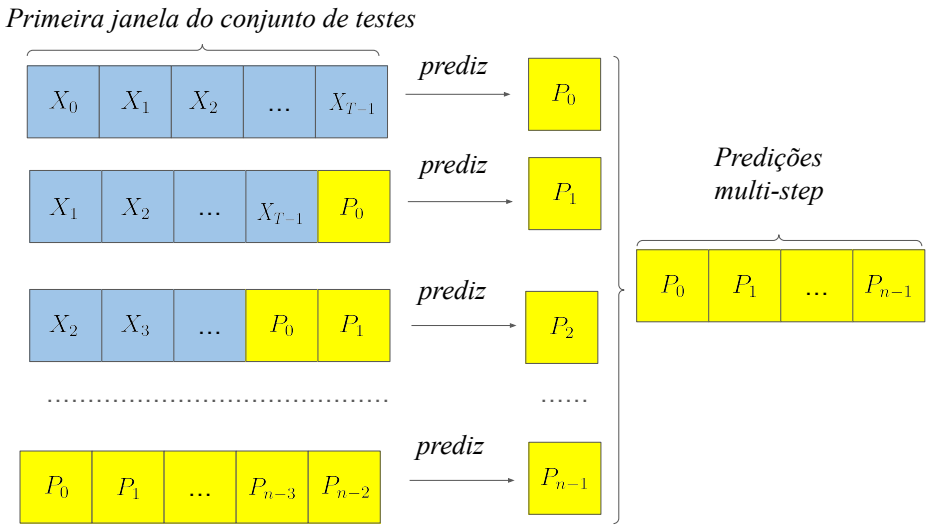


Figura 4.3. Abordagem *multi-step*. Inicialmente, o modelo é treinado de maneira *one-step* utilizando o conjunto de dados designado para treinamento. Após isso, o processo de predição começa com a primeira janela do conjunto de teste. Utilizando o primeiro valor predito P_0, a janela original é deslocada uma unidade para a esquerda, fazendo com que o primeiro elemento X_0 da janela anterior se torne o último elemento da janela atual. Substitui-se então X_0 por P_0 e, com esta nova configuração da janela, realiza-se a próxima predição do modelo, obtendo P_1. Esse processo é repetido sucessivamente até alcançar o horizonte de predição estipulado, que neste caso é definido pelo valor n.

No contexto não inocente, faz-se uma engenharia de características para selecionar aquelas que geram maior poder preditivo, ou seja, escolhem-se as características que captam a tendência geral dos dados e que possuem melhor desempenho em métricas. Com efeito, inicialmente observamos a matriz de correlação cruzada [Derrick & Thomas, 2004], observada na **Figura 4.4**, entre o número de casos semanais de dengue e as características representadas pelos índices das linhas da **Figura 4.4**: quantidades de pixels relativos à vegetação (0 - 3), total de precipitação em milímetros da mensuração feita pelo NOAA (4), reanálise da temperatura do ar (5), reanálise da temperatura média (6), reanálise do ponto do orvalho (7), reanálise da temperatura máxima do ar (8), reanálise da temperatura mínima do ar (9), reanálise do nível de precipitação em quilograma por metro quadrado (10), reanálise do percentual de umidade relativa (11), total de precipitação semanal (12) reanálise de umidade específica (13), reanálise da faixa de temperatura média do período diurno (14), temperatura média da semana (15), faixa de temperatura diurna semanal (16) temperatura máxima da semana (17), temperatura mínima da semana (18), precipitação semanal em milímetros (19), número de casos semanais (20).

Diante desta análise, realizamos diversas combinações, onde uma delas consiste na escolha de características (linhas do mapa de calor) com maior correlação negativa (linhas de índices 5, 6, 7, 8, 9, 13, 15, 18) e *lag* temporal de tamanho 17. Porém, a melhor combinação de características obtida neste trabalho consiste nas características com índices de 0, 1, 2, 3, 18, 19, 20 com *lag* temporal de tamanho 22. Portanto, constatamos que o nível de precipitação e a vegetação do ambiente são fatores mais relevantes para a predição do número de casos semanais, o que ressalta, de forma empírica, os resultados obtidos por Santos et al. [2019] e Panja et al. [2023].

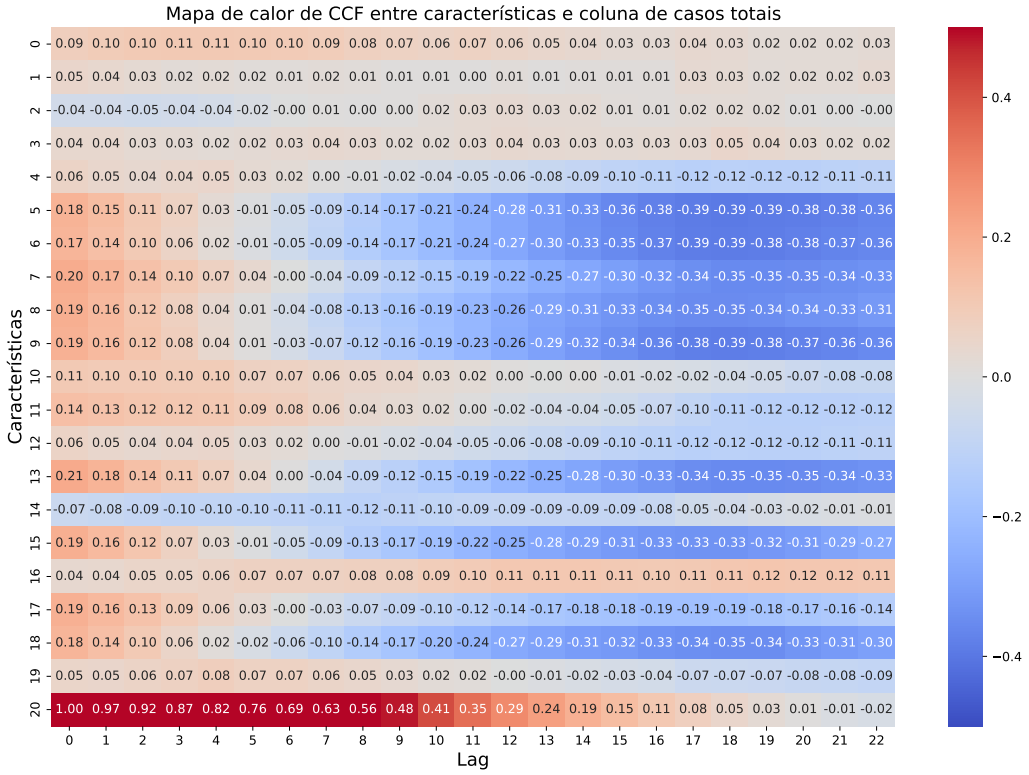


Figura 4.4. Mapa de Calor da Correlação Cruzada entre as características em San Juan e a coluna de casos totais em San Juan.

Após a seleção de características relevantes, a transformada *wavelet* é aplicada a cada uma das séries temporais selecionadas. O objetivo é suavizar cada característica visando reduzir ruído [Morettin, 2014].

As transformadas *wavelet* [Strang & Nguyen, 1996] dividem o sinal em componentes de alta frequência (detalhes) e baixa frequência (aproximação). A escolha do nível de decomposição é um hiperparâmetro. Tem-se que a cada nível de decom-

posição, o componente de aproximação do nível anterior é dividido novamente em uma nova aproximação (frequência ainda mais baixa) e detalhes (frequências mais altas dentro da banda de frequência mais baixa). Os coeficientes de detalhe em cada nível representam diferentes bandas de frequência. Por exemplo, no primeiro nível, os detalhes representam as frequências mais altas do sinal original. No segundo nível, representam as frequências mais altas dentro da metade inferior do espectro de frequência do sinal original, e assim por diante. O valor máximo para o nível depende do comprimento do sinal e do tipo de *wavelet* utilizada. Ressalta-se que um valor muito alto para o nível pode resultar em decomposições em que os coeficientes de aproximação ou detalhe são insignificantes ou não informativos. Por consequência, a escolha do número de níveis de decomposição depende do problema específico e das características do sinal. Em geral, níveis mais altos permitem analisar detalhes mais finos do sinal, enquanto níveis mais baixos focam nas tendências gerais.

Ao longo das experimentações, testamos todas as *wavelets* discretas disponíveis na biblioteca PyWavelets [Lee et al., 2019] e deixamos o nível de decomposição igual a um. A transformação *wavelet* expressa a série temporal em termos de coeficientes de aproximação e de detalhe. Os coeficientes de detalhe (alta frequência) capturam mudanças de curto prazo na série temporal e tendem a destacar transições rápidas nos dados. Os coeficientes de aproximação (baixa frequência) capturam tendências de longo prazo na série temporal e tendem a revelar padrões gerais nos dados. Em nossos experimentos, utilizamos um filtro passa-baixa, utilizando apenas os coeficientes de aproximação para reconstruir o sinal. Na biblioteca PyWavelets, esse processo é realizado utilizando os coeficientes de aproximação obtidos a partir da decomposição pela *wavelet*, enquanto os coeficientes de detalhe são substituídos por valores nulos. Esta operação garante que, ao reconstruir o sinal, tenhamos a mesma quantidade de *timesteps* da série temporal original, porém sem o uso de coeficientes de detalhe.

Por fim, com as características desejadas e transformadas pela *wavelet* escolhida em dois contextos distintos, construímos as matrizes X e Y definidas acima. Posteriormente, respeitado a escala temporal, dividimos estes dados em treino/validação na proporção 80/20 e passamos para a fase de experimentação dos modelos e comparação de resultados obtidos.

4.3 Pré-processamento da base InfoDengue

Analogamente, para cada unidade federativa brasileira disponível na base InfoDengue, aplica-se estratégia similar ao pré-processamento (**Figura 4.1**) da base de San

Juan. Entretanto, devido à menor quantidade de características disponíveis nesta base, não realizamos a seleção de características por meio da matriz de correlação cruzada entre as características e o número de casos semanais de dengue. Ademais, realizamos a filtragem das características por *wavelets* apenas na análise individual de cada *dataset*, já que para as abordagens envolvendo *cross-learning* (uso dos *datasets* de todas as capitais), usamos apenas o número de casos semanais de dengue para treinamento e validação.

Na análise individual de cada *dataset*, utilizamos o método multivariado *one-step*, empregando todas as características relevantes disponíveis no conjunto de dados, conforme detalhado anteriormente. Em seguida, particionamos os dados em conjuntos de treinamento e validação, reservando as últimas 100 instâncias exclusivamente para validação. Finalmente, realizamos uma análise comparativa dos modelos e dos resultados obtidos.

Para a abordagem de *cross-learning* aleatorizada (**Figura 4.5**), utilizamos apenas o número de casos semanais de dengue (dados univariados) para treinar e validar o modelo. Inicialmente, reformatamos todos os 27 conjuntos de dados para aprendizado supervisionado e concatenamos os rótulos e alvos nas matrizes X e Y, respectivamente. Em seguida, o número n_f de elementos em cada *fold* foi definido tomando o piso do quociente entre o número n_X de linhas da matriz X pelo número K de *folds* pré-definido, isto é,

$$n_f = \left\lfloor \frac{n_X}{K} \right\rfloor. \tag{4.2}$$

Após a divisão em K *folds*, $K-1$ *folds* foram selecionados aleatoriamente para formar o conjunto de treinamento, e o *fold* restante foi utilizado para validação. Esse processo é denominado uma rodada. O número de rodadas, R, é um hiperparâmetro. Os resultados finais para cada métrica são a média das métricas individuais calculadas em cada rodada.

Analisou-se também o desempenho da abordagem de *cross-learning* por meio do uso de *leave-one-out* (LOO), conforme exposto na **Figura 4.6**. Ou seja, para cada unidade federativa, dos 27 *datasets* reformatados para o aprendizado supervisionado, usamos 26 para treinamento do modelo e o *dataset* da unidade federativa em questão para validação. Para esta estratégia, experimentamos tanto esquema *one-step*, quanto *multi-step*.

Por fim, aplicamos esta mesma estratégia de *cross-learning* utilizando LOO, treinando o modelo com a base de dados do InfoDengue e validando-o com a base de dados de San Juan, visando avaliar o potencial de generalização do modelo.

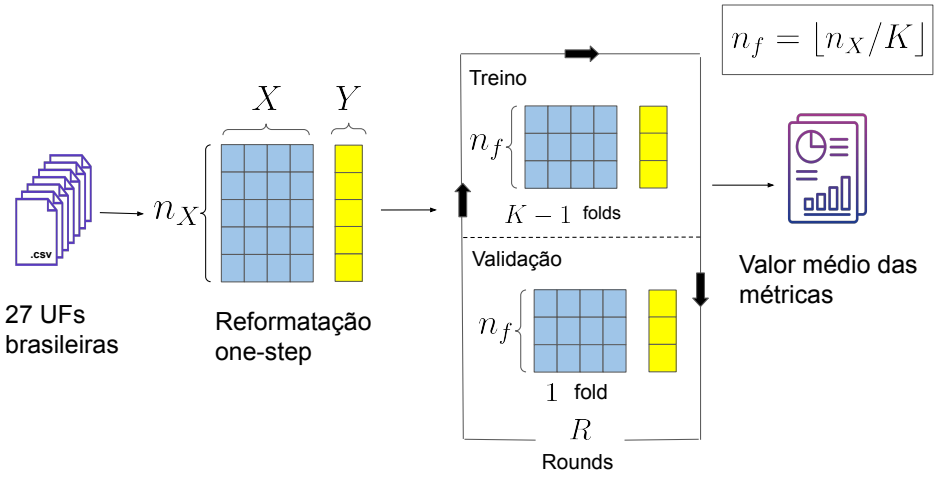


Figura 4.5. Abordagem *cross-learning* aleatorizada. Inicialmente, realizamos a reformatação *one-step* nos 27 *datasets*. Em seguida, concatenamos as matrizes de rótulos X (coloridas de azul) e as colunas alvo Y (coloridas de amarelo) de modo a preservar a relação original entre a janela e o valor alvo em cada *dataset*. O número de linhas do *dataset* concatenado é n_X. Após a concatenação, o *dataset* resultante é dividido em K *folds*. O número de instâncias por *fold* é n_f. A cada *round* r, selecionamos $K-1$ *folds* de forma aleatória para o treinamento e o *fold* restante é utilizado para validação. Ao final de cada *round* r, calculamos e armazenamos as métricas de desempenho obtidas. Finalmente, cada métrica final é calculada como a média das métricas de todos os R *rounds*.

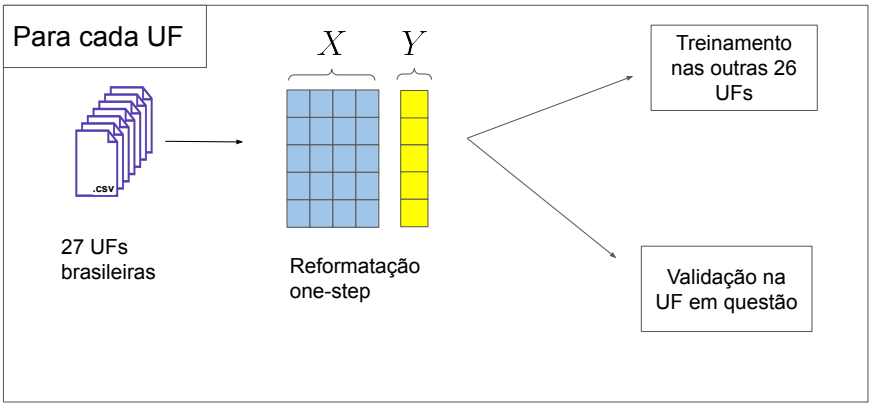


Figura 4.6. Abordagem *leave-one-out* (LOO) para cidades brasileiras. Inicialmente, reformate cada *dataset* das unidades federativas (UFs) brasileiras no esquema *one-step*. Em seguida, para cada UF, utilize os 26 *datasets* das demais UFs para treinar o modelo. O *dataset* da UF em questão é reservado para validação. Esta estratégia permite que cada modelo seja testado de forma isolada com dados não utilizados no treinamento, proporcionando uma avaliação robusta do desempenho do modelo em diferentes contextos regionais.

5

Experimentação

Neste capítulo, detalhamos as características e os hiperparâmetros dos modelos aplicados às bases de dados de San Juan e InfoDengue, que foram pré-processadas conforme descrito no capítulo anterior. Por fim, apresentamos os resultados obtidos com essas configurações e discutimos os achados observados.

Os modelos estatísticos clássicos usados são provenientes da biblioteca *Statsmodels* [Seabold & Perktold, 2010]. Para os modelos rasos de aprendizado de máquina usamos a biblioteca *Scikit-Learn* [Pedregosa et al., 2011]. Para os modelos profundos, utilizamos a versão 2.15.0 da biblioteca *TensorFlow* [Abadi et al., 2015] e *PyTorch* [Paszke et al., 2019] versão 2.0.1. Ao longo dos experimentos, tanto para os modelos rasos quanto profundos de aprendizado, testamos todas as *wavelets* discretas da biblioteca PyWavelets [Lee et al., 2019]. Em ambas as bases de dados, realizou-se execução iterativa dos métodos em um intervalo de janelas de tamanho $1 \leq T \leq 30$. Destaca-se o uso *Standard Scaler* antes do processamento dos dados pelos modelos profundos [Goodfellow et al., 2016], de maneira que cada característica formada na estratégia de janelamento contribua de maneira uniforme.

5.1 Descrição dos modelos

Na implementação de modelos estatísticos clássicos utilizou-se a configuração padrão do *Statsmodels*. Nos modelos rasos do *scikit-learn*, a única alteração nos hiperparâmetros para os dados de San Juan foi o uso do *kernel* linear no SVR. Na base InfoDengue, realizamos a otimização dos hiperparâmetros do modelo LightGBM, utilizado na abordagem de *cross-learning*, por meio de *Grid Search*. Quanto aos modelos profundos, houveram particularidades para cada base.

5.1.1 Modelos profundos na base InfoDengue

Para os modelos profundos, exploramos algumas arquiteturas, incluindo CNN (com e sem camada de atenção no final), LSTM (com e sem camada de atenção no final) e a combinação de CNN-LSTM (com e sem um mecanismo de atenção no final). Também explorou-se modelos que utilizam exclusivamente camadas de atenção, bem como uma versão modificada da CNN-LSTM que incorporou um componente autorregressivo sem camada de atenção ao término (CNN-LSTM-LR). Na base InfoDengue, cada rede neural foi treinada utilizando 200 épocas, o erro quadrático médio (MSE) como função de perda e o otimizador *Adam* com uma taxa de aprendizado de 0.001.

Para a CNN sem camadas de atenção, a entrada da rede assume uma forma tridimensional, onde N é o número de amostras, D representa o número de características, e T denota o comprimento da sequência. O modelo é composto por três camadas convolucionais consecutivas. A primeira camada usa 7 canais de entrada (correspondentes às 7 características) e aumenta a dimensionalidade para 16 canais de saída com um tamanho de *kernel* de 3. Essa camada não emprega *padding*, tem um *stride* de 1 e uma taxa de dilatação de 1. A segunda camada convolucional transforma os 16 canais de entrada em 32 canais de saída, mantendo o mesmo tamanho de *kernel*, *stride* e dilatação da primeira. A terceira camada convolucional expande o tamanho do canal de 32 para 64, novamente usando os mesmos parâmetros para tamanho de *kernel*, *stride* e dilatação. Após as camadas convolucionais, uma camada de *average pooling* reduz a dimensionalidade da saída de cada amostra para um único valor por mapa de características (64 no total), preparando os dados para a última camada densa. A saída da camada de *pooling* é remodelada de um tensor tridimensional para um tensor bidimensional para se adequar aos requisitos de entrada da camada totalmente conectada subsequente. A camada final na arquitetura é uma camada linear com 64 características de entrada e uma única característica de saída, ideal para tarefas de regressão. Essa camada produz diretamente a saída final do modelo. A função de ativação não linear usada ao longo do modelo é a unidade linear retificada (ReLU), aplicada após cada camada convolucional para introduzir não linearidade no modelo, aumentando sua capacidade de aprender padrões complexos nos dados.

Por outro lado, a CNN com camadas de atenção visa melhorar a capacidade da rede focar em características relevantes dentro da sequência de dados, aumentando a precisão da predição para tarefas como regressão. O componente de atenção utilizado é a camada *Multihead Attention*, configurada com 32 dimensões de *embedding* para corresponder aos canais de saída da última camada convolucional, e 2 a*ttention*

heads. Essa configuração permite que o modelo aprenda simultaneamente diferentes subespaços dos dados, focando em vários aspectos das sequências de entrada. Após o processamento através da camada de atenção, uma operação de *max pooling* é aplicada ao longo do comprimento da sequência para reduzir a dimensionalidade dos dados a um único vetor por amostra. Esse vetor, que representa as características mais relevantes segundo o mecanismo de atenção, é então passado para uma camada linear para produzir a saída final.

A camada LSTM sem atenção é configurada com um tamanho de entrada de D (número de características), um tamanho da camada oculta de 20 e três camadas. Após a camada LSTM, a forma do tensor de saída é (N, T, M), onde M é o tamanho da camada oculta (20 neste caso), encapsulando a saída do estado oculto para cada passo de tempo. Para processamento adicional, apenas a saída do último passo de tempo é utilizada, efetivamente reduzindo a dimensionalidade do tensor para (N, M). Uma camada linear segue, mapeando a dimensão de saída da LSTM (20) para uma única característica de saída. Os estados iniciais para a LSTM (tanto o estado oculto h_0 quanto o estado de célula c_0) são inicializados como tensores zero com dimensões correspondentes ao número de camadas, tamanho do lote e tamanho da camada oculta (3, N, 20). Essa configuração fornece um ponto de partida neutro para a acumulação de estados durante o processamento da sequência.

De forma semelhante à CNN com atenção, a arquitetura da LSTM com atenção integra um mecanismo de atenção dentro de uma rede LSTM. Após a camada LSTM, o modelo incorpora uma camada de *Multihead Attention*. Esta camada é configurada com 32 dimensões de *embedding*, correspondendo à saída do LSTM, e 2 *attention heads*, permitindo que o modelo se concentre em diferentes aspectos dos dados de sequência simultaneamente. Após a camada de atenção, é aplicada uma operação de *max pooling* ao longo do comprimento da sequência para evidenciar as características mais significativas em um único vetor por amostra. O componente final é uma camada linear que mapeia a saída de 32 dimensões da operação de *pooling* para um valor de saída.

O modelo híbrido CNN-LSTM é projetado para tarefas de processamento de sequência, combinando as capacidades de extração de características espaciais CNNs com o uso processamento temporal das redes LSTM. O modelo inicia com uma sequência de camadas convolucionais destinadas a processar a entrada no formato (N, T, D), onde N é o número de amostras, T o comprimento da sequência, e D a dimensão das características. Essa entrada é primeiramente permutada para (N, D, T) para corresponder ao formato de entrada esperado pelas camadas convolucionais. Após as camadas convolucionais, uma camada de *max pooling* reduz a dimensão

de cada mapa de características para 1, resumindo efetivamente as características extraídas, enquanto mantém as dimensões de lote e canal. A saída após o *pooling* é, portanto, remodelada de $(N, 64, 1)$ para $(N, 1, 64)$, alinhando-a para a entrada na camada LSTM. A porção LSTM do modelo tem um tamanho de entrada de 64 (correspondente aos canais de saída da CNN) e contém 2 camadas com um tamanho oculto de 20. Essa configuração permite que o LSTM processe os dados da sequência, mantendo as dependências temporais ao longo da sequência. A saída da LSTM, com dimensão $(N, T, 20)$, é passada por uma camada linear após selecionar apenas a saída do último passo de tempo, o que é típico em muitas tarefas de processamento de sequência onde apenas a saída final é usada para predição. A camada linear mapeia a saída de 20 dimensões da LSTM para um valor de saída.

O modelo baseado puramente em atenção aproveita as capacidades do mecanismo de *Multihead Attention* para processar dados em sequência. Este modelo é projetado para focar especificamente nas relações e interações entre os elementos na sequência, sem o uso de camadas convolucionais ou recorrentes. Cada camada de atenção é configurada com uma dimensão de *embedding* de D (definida como 7, o número de características) e D cabeças de atenção. Após a sequência passar por todas as três camadas de atenção, uma operação de *max pooling* é aplicada ao longo do comprimento da sequência para agregar as características, reduzindo cada sequência aos seus elementos mais significativos. O componente final do modelo é uma camada linear que mapeia o vetor de características agregado (dimensão D) para um valor de saída.

O modelo CNN-LSTM-LR combina CNNs, LSTMs e Regressão Linear (LR) em um modelo híbrido projetado para tarefas envolvendo processamento de sequências, integrando tanto a extração de características espaciais quanto temporais com a análise de regressão. A parte CNN começa com três camadas convolucionais, cada uma com um tamanho de canal de entrada de 7 e um tamanho de canal de saída de 16, tamanho de *kernel* de 2, *stride* de 1, *padding* de 1 e dilatação de 1. Essas camadas são projetadas para extrair características espaciais dos dados de entrada. Cada camada convolucional é seguida por uma camada de *max pooling* com tamanho de *pool* de 2 e camadas de *dropout* com taxa de 0.2 para prevenir *overfitting* e promover generalização. A saída da última camada convolucional passa por *max pooling* para reduzir a dimensionalidade a um único vetor de características por amostra. A componente LSTM é configurada com um tamanho de entrada igual à dimensão das características D, um tamanho oculto de 32 e quatro camadas, com um dropout de 0.2. Ela processa os aspectos temporais dos dados de entrada, capturando dependências de longo prazo. A saída da LSTM também passa por *max*

pooling, reduzindo de forma semelhante a sequência temporal a um único vetor por sequência. Uma camada personalizada de regressão linear é definida para processar os dados de entrada de forma linear, com parâmetros otimizados para tarefas de regressão. O modelo de regressão recebe uma dimensionalidade de entrada de $T \times D$ e produz uma única característica, refletindo um resultado de regressão com base nas características de entrada. Finalmente, as saídas dos componentes CNN, LSTM e LR são concatenadas e, em seguida, alimentadas em uma camada linear final, que integra essas características diversas em uma única predição de saída. Esta última camada densa tem uma dimensão de entrada de 18 (combinando características das saídas da CNN, LSTM e regressão linear) e produz um valor de saída.

MOIRAI e TimeGPT-1 são modelos pré-treinados e não requerem reformatação de *dataset* para predição como os modelos tradicionais de aprendizado supervisionado, logo utilizamos predições univariadas na coluna de casos semanais totais de dengue tomando um horizonte de predição de cem semanas. No MOIRAI, podemos escolher a complexidade do modelo (pequena, média e grande), em que complexidade baseia-se na quantidade de hiperparâmetros pré-treinados, o comprimento do contexto, ou seja, o tamanho da janela, e o número de horizontes de predição ou comprimento da predição. Para o TimeGPT-1, podemos escolher apenas o horizonte de predição.

5.1.2 Modelos profundos na base de San Juan

Quanto aos modelos profundos, três arquiteturas foram utilizadas: LSTNet [Lai et al., 2017]; uma combinação entre rede neural convolutiva (CNN) e rede neural recorrente (LSTM); uma combinação entre CNN e LSTM com um componente autorregressivo (CNN-LSTM-LR) ao término exposto na **Figura 5.1**. A configuração deste modelos é análoga à utilizada na base InfoDengue.

Quanto às especificações do modelo, usamos função de ativação como a unidade linear retificada (ReLU), função de perda sendo o erro médio absoluto (MAE), otimizador Adam com taxa de aprendizado de 0.01, parada antecipada com paciência de 10 épocas (monitorada pela MAE) e pré-definimos 150 épocas (geralmente não atingidas, devido à parada antecipada).

Nas camadas convolutivas, o número de filtros utilizados foi dezesseis e com tamanho de kernel igual a dois. Para as camadas recorrentes utilizou-se um total de 32 unidades escondidas. O uso e o não uso de camadas de *pooling* foi testado na arquitetura da CNN-LSTM, em que esta comparação foi motivada pelo fato de que a LSTNet não usa tais camadas em sua arquitetura.

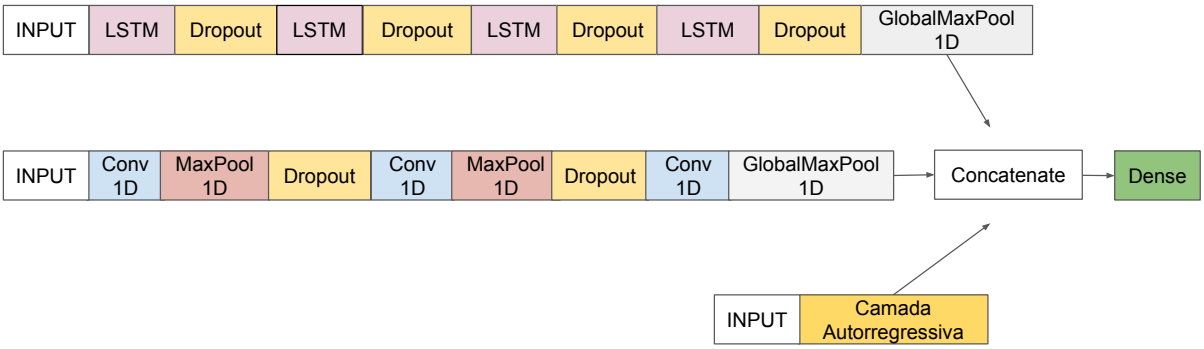


Figura 5.1. Modelo CNN-LSTM-LR com uso de *pooling*.

5.2 Resultados com a base de San Juan

A **Tabela A.1** (veja Apêndice) apresenta os resultados dos melhores modelos na base de San Juan, tanto da perspectiva univariada quanto multivariada. A **Tabela 5.1** exibe as métricas dos melhores modelos da **Tabela A.1**. A **Figura 5.2** mostra o ajuste do modelo LR (sym20) aos dados de validação.

	TREINO					VALIDAÇÃO				
MODELO	MAE	R^2	MAPE	RMSE	BIAS	MAE	R^2	MAPE	RMSE	BIAS
LR (sym20)	3.15	0.97	0.18	4.87	-0.003	2.72	0.98	0.20	3.80	0.05
CNN-LSTM-LR (sym20)	3.15	0.97	0.18	4.87	-1.39	2.79	0.98	0.20	3.80	-9.64

Tabela 5.1. Métricas de treino e teste dos melhores modelos (comparação com a coluna original de casos totais).

5.3 Discussão sobre os resultados da base de San Juan

Na base de San Juan, investigamos três tipos de modelagem: modelagem estatística tradicional, aprendizado de máquina raso e aprendizado de máquina profundo. Realizamos análises univariadas e multivariadas para estes modelos.

O Passeio Aleatório (RW) destacou-se entre os modelos estatísticos (SES, SEH, HW, ARIMA, RW), enquanto o ARIMA, apesar de convergir, teve predições impre-

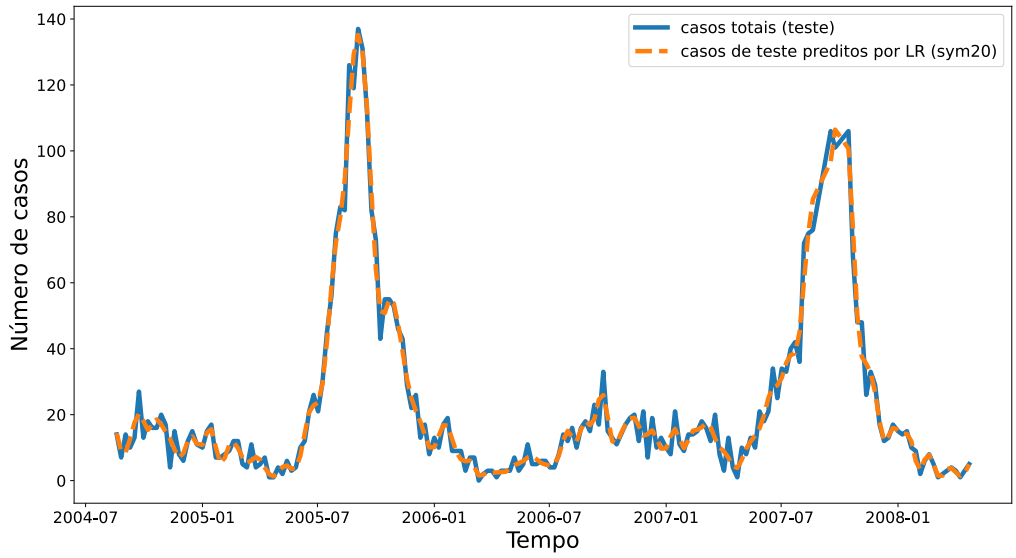


Figura 5.2. Comparação entre predições de LR e casos totais de dengue não filtrados.

cisas. Nos modelos rasos (RF, SVR, XGBoost, GBR, LR) e profundos (CNN-LSTM, CNN-LSTM-LR, LSTNet), optamos por estratégias univariadas e multivariadas, priorizando predições *one-step* após baixo desempenho na abordagem *multi-step*.

A análise da **Tabela A.1**, mostra que o modelo de Regressão Linear (LR) com pré-processamento via *wavelet* sym20 foi o mais eficaz entre os rasos, e o CNN-LSTM-LR se destacou entre os profundos. Ao comparar os resultados com dados não filtrados, observamos na **Tabela 5.1** que o modelo LR (sym20) se destacou em termos de MAE, apresentando um BIAS de 0.05. Por outro lado, CNN-LSTM-LR mostrou subnotificação de casos, com um BIAS de -9.64. A investigação indicou que o CNN-LSTM-LR, em grande parte, apenas replicava o desempenho de seu componente autorregressivo, alcançando resultados similares ao do modelo LR (sym20).

Recorde que o melhor tamanho de janela para convergência em San Juan fixou-se em $T = 22$. Por esta razão, nas tabelas de San Juan, não destacou-se o tamanho da janela utilizada na base. Na **Tabela A.1**, no intuito de se obter consistência nos resultados, comparamos os resultados preditos com a coluna de casos totais filtrada pela *wavelet* em questão. Entretanto, como o objetivo principal consiste em comparar os resultados preditos com a realidade apresentada, temos nas **Tabela 5.1** os resultados preditos comparados com a coluna de casos totais não filtrada dos melhores modelos obtidos na **Tabela A.1**. Além disso, a **Figura 5.2** mostra o ajuste dos resultados preditos para com a coluna de casos totais não filtrada.

Em comparação ao *baseline* [Shaikh et al., 2023], que usa algoritmos genéticos na fase de pré-processamento e um *ensemble* de modelos (ANN, SVR e CNN)

para predição, os modelos da **Tabela 5.1** obtiveram uma superioridade de 1.90 em RMSE. Além disso, o modelo LR (sym20) apresenta resultados melhores do que os de Panja et al. [2023] em termos de MAE e RMSE, com reduções de 2.94 e 3.89, respectivamente.

5.4 Resultados com a base InfoDengue

Os melhores tamanhos de janelas foram para $T \in \{5, 10, 15, 17, 22, 30\}$. Para o melhor modelo (LightGBM), os hiperparâmetros testados foram:

- taxa de aprendizado: {0.005, 0.01, 0.02, 0.05, 0.1};
- número de folhas: {31, 63, 127, 255};
- máxima profundidade: {5, 7, 10, 12, 15};
- regularização L1: {0, 1, 3, 5, 7, 9, 11};
- regularização L2: {0, 1, 3, 5, 7, 9}.

Os melhores hiperparâmetros obtidos por *GridSearch* para os modelos LightGBM *K-fold* e LightGBM LOO (*one-step* e *multi-step*) foram:

- taxa de aprendizado: 0.01;
- número de folhas: 127;
- máxima profundidade: 10;
- regularização L1: 9;
- regularização L2: 7.

Primeiramente, apresentamos o melhor resultado do *cross-learning* com validação *K-fold*. Em seguida, os resultados da fase de validação das abordagens para cada cidade foram organizados em tabelas individuais. Por fim, apresentamos o resultado do LightGBM treinado em todas as unidades federativas e validado no *dataset* de San Juan.

5.4.1 Resultados de cross-learning com validação K-fold

Das janelas testadas, o melhor resultado para a abordagem aleatorizada foi $T = 30$. Em geral, utilizou-se $R = 50$ para o número de rodadas, e $K = 15$ *folds*. O número n_f de elementos por *folds* varia segundo a quantia de linhas n_X da matriz X de rótulos, onde esta quantia, por sua vez, depende do tamanho da janela T adotado no experimento. Para o melhor caso apresentado na **Tabela 5.2**, temos $T = 30$, logo, segundo (4.2), tem-se

$$n_f = \left\lfloor \frac{n_X}{K} \right\rfloor = \left\lfloor \frac{17710}{15} \right\rfloor = 1180. \tag{5.1}$$

	TREINO					VALIDAÇÃO				
Tamanho da janela (T)	**MASE**	**RMSSE**	**MAE**	**RMSE**	**BIAS**	**MASE**	**RMSSE**	**MAE**	**RMSE**	BIAS
30	0.64	**0.67**	27.95	141.66	-0.21	0.75	**0.52**	31.87	104.97	0.90

Tabela 5.2. Melhor resultado da abordagem *cross-learning* LightGBM com validação K-fold.

5.4.2 Resultados para cada unidade federativa brasileira

Em cada uma das **Tabelas A.2** — **A.28** expostas no Apêndice, os dados processados por modelos estatísticos clássicos e LightGBM LOO não foram filtrados por *wavelets*. Os modelos estatísticos clássicos não exigem reformatação do conjunto de dados. Os modelos rasos e profundo de aprendizado de máquina usaram a estratégia *one-step* multivariada. Dos dois modelos pré-treinados, MOIRAI é o único que permite uma estratégia de janelamento, de maneira que explorou-se esta característica. Ao final de cada tabela relativa a uma unidade federativa, foi exposto o desempenho do LightGBM utilizando *leave-one-out* (LOO) tanto na predição *one-step* quanto *multi-step*. Destaca-se que a validação dos modelos estatísticos clássicos, dos modelos de aprendizado de máquina raso e profundo foram feitas com as cem últimas instâncias do *dataset*. Já a validação do LightGBM LOO (*one-step* e *multi-step*) foi feita com todas as instâncias do *dataset* da cidade em questão (acima de 700 instâncias). O *dataset* da cidade de Teresina (PI) possuía apenas a coluna de casos totais semanais com informações preenchidas, logo a abordagem neste caso foi apenas univariada.

5.4.3 Resultados da validação em San Juan

As **Tabelas 5.3** e **5.4** apresentam os resultados do modelo LightGBM treinado com dados das 27 unidades federativas brasileiras e validado no *dataset* de San Juan. A

Tabela 5.3 mostra a validação do modelo em todas as 936 instâncias do *dataset* de San Juan. Para fins de comparação com os resultados obtidos pelo modelo LR (sym20) treinado em San Juan, a **Tabela 5.4** apresenta a validação nas últimas 187 instâncias.

	TREINO					**VALIDAÇÃO**				
Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS	MASE	RMSSE	MAE	RMSE	BIAS
$T = 30$	0.67	**0.68**	28.73	140.78	-0.24	0.16	**0.06**	6.80	**11.49**	1.89

Tabela 5.3. Melhor resultado do modelo LightGBM LOO (one-step) validado em todo *dataset* de San Juan.

Tamanho da janela (T)	**MASE**	**RMSSE**	**MAE**	**RMSE**	**BIAS**
30	0.12	0.04	4.98	**7.55**	2.24

Tabela 5.4. Melhor resultado do modelo LightGBM LOO (one-step) validado nas últimas 187 instâncias do *dataset* de San Juan.

5.5 Discussão sobre os resultados da Base InfoDengue

Pela **Tabela 5.2**, as métricas MASE e RMSSE indicam que o modelo *cross-learning* LightGBM com validação *K-fold* apresenta um desempenho superior ao do passeio aleatório. A MAE no treino foi menor que no teste, o que é esperado, dado que há uma quantidade maior de dados no treino (16530 instâncias) em comparação com a validação (1180 instâncias). A RMSE sugere que o modelo é robusto em relação a possíveis *outliers*. A métrica BIAS mostra que o modelo não tende a subestimar ou superestimar de forma significativa, considerando que o número de casos de dengue é um valor inteiro. Devido à aleatoriedade na construção dos *folds*, a avaliação do modelo perde parte de seu significado, pois o *dataset* original da série temporal não é completamente preservado. No entanto, o resultado é útil, pois demonstra que o modelo tem um desempenho razoável, independentemente da ordem das janelas apresentada a ele, já que os *folds* de validação nas 50 rodadas são aleatoriamente selecionados. Logo, se o modelo é robusto o suficiente para manter um bom desempenho mesmo quando as janelas são apresentadas de forma aleatória, este comportamento indica que ele pode lidar bem com a variação e com a ordem não previsível dos dados futuros. Por esta razão, este resultado nos motivou a testar o LightGBM LOO em cada uma das unidades federativas brasileiras.

Na análise individual, o modelo LightGBM LOO (*one-step*) obteve o melhor desempenho na RMSE em todos os *datasets* de cidades brasileiras. Recorde que este

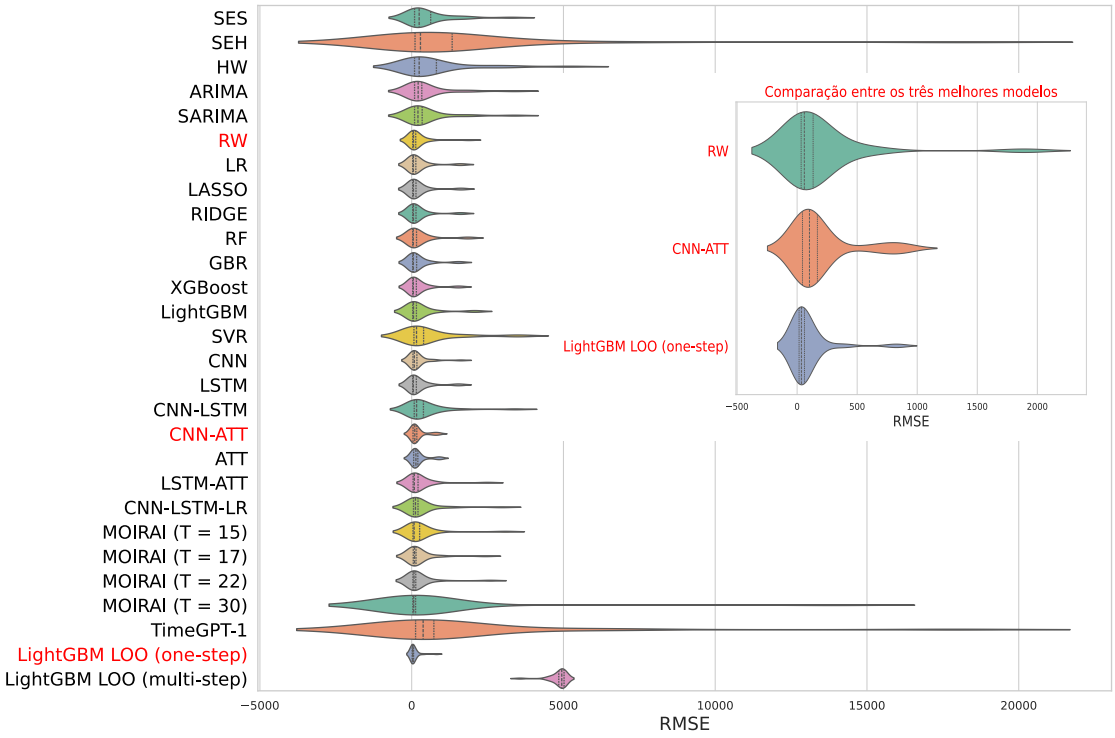


Figura 5.3. Distribuição numérica da métrica RMSE

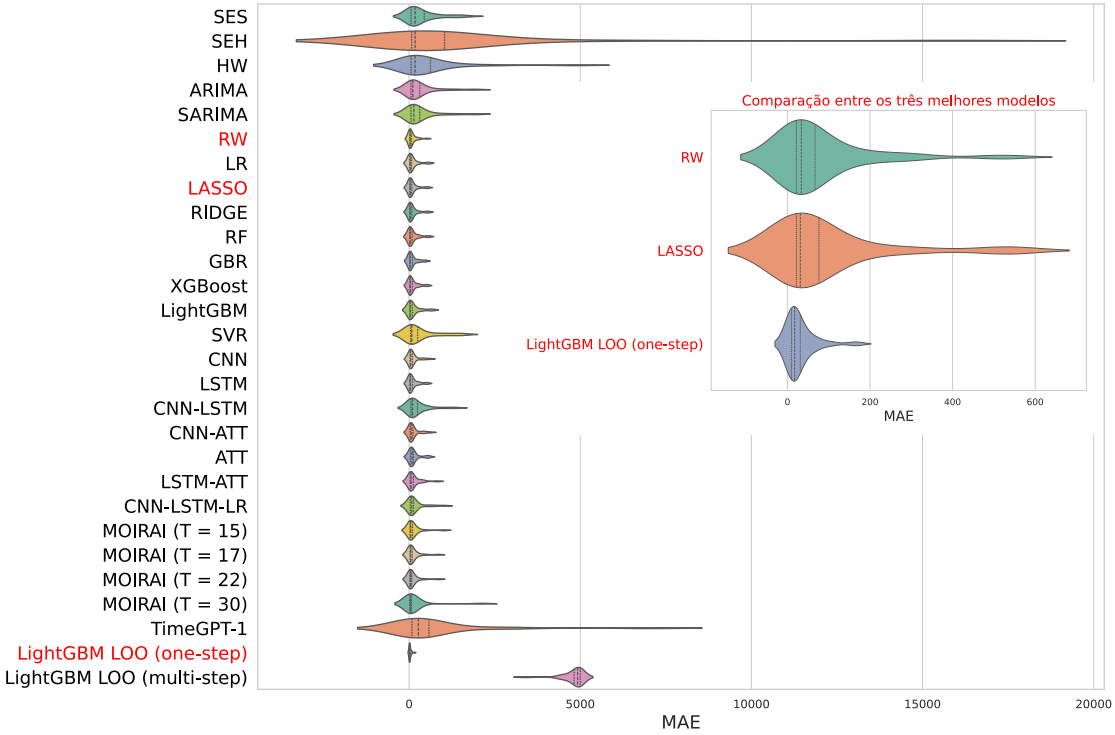


Figura 5.4. Distribuição numérica da métrica MAE

modelo foi validado usando todas as instâncias do *dataset* da cidade em questão, enquanto os outros modelos foram validados com as 100 últimas instâncias. Por-

tanto, embora haja aumento no número de instâncias na validação, o LightGBM LOO (*one-step*) errou menos e mostra maior robustez no aprendizado. O gráfico de violino exposto na **Figura 5.3** evidencia a superioridade do LightGBM LOO (*one-step*) em cada uma das cidades, mostrando que a faixa de valores da RMSE se concentra em valores inferiores a 100.

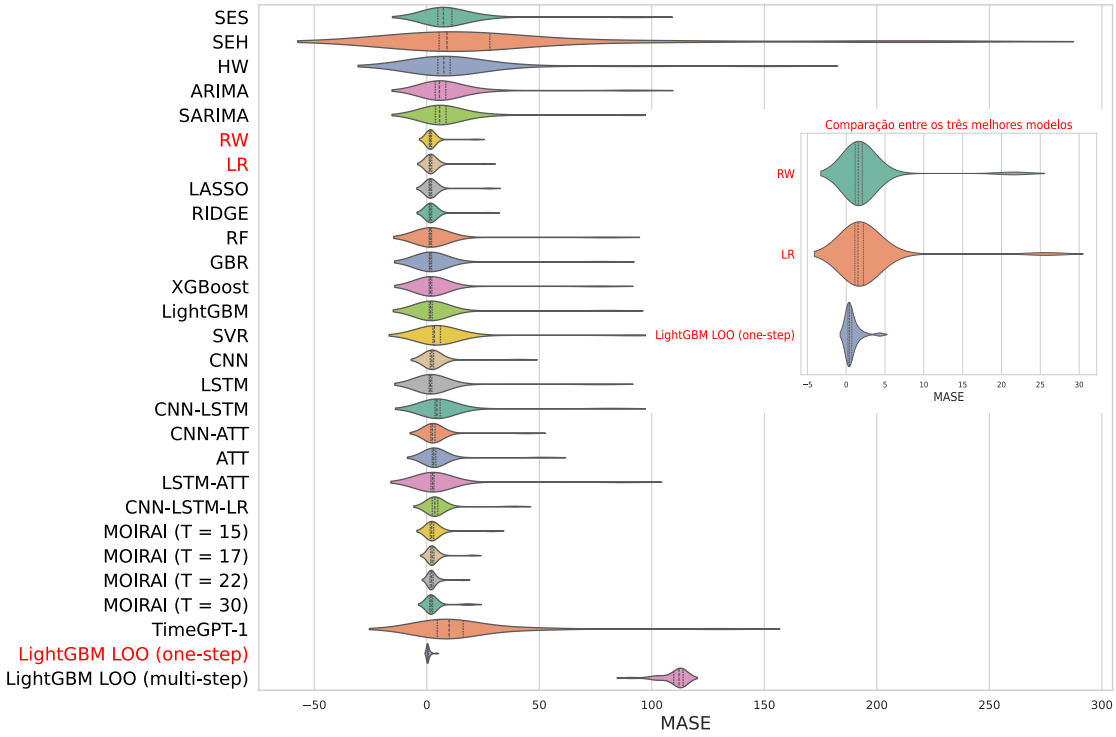


Figura 5.5. Distribuição numérica da métrica MASE

Quanto à MAE, o modelo LightGBM LOO (*one-step*) foi superior em todas as cidades, com exceção de Belém (8.60), onde o RW obteve um desempenho melhor (8.42), e Cuiabá (10.22), onde o LASSO (9.26) foi superior. No entanto, como RW e LASSO foram validados com as últimas 100 instâncias, é possível que o LightGBM LOO *one-step* também sugerisse um desempenho superior caso fosse validado nas mesmas 100 últimas instâncias. No entanto, analisando o gráfico de violino da **Figura 5.4**, os valores de MAE do modelo LightGBM LOO (*one-step*) formam distribuições com maior densidade ao redor do valor nulo, mostrando superioridade.

LightGBM LOO (*multi-step*), ao contrário do excelente desempenho do LightGBM (*one-step*), apresentou-se como um dos piores modelos, o que pode ser explicado pela acumulação de erros anteriores gerados a cada passo da predição. Portanto, devido ao acúmulo consecutivo de erros, o modelo tende a apresentar piores desempenhos nas métricas, fato anteriormente observado nos resultados *multi-step* na **Tabela A.1** para os modelos de San Juan.

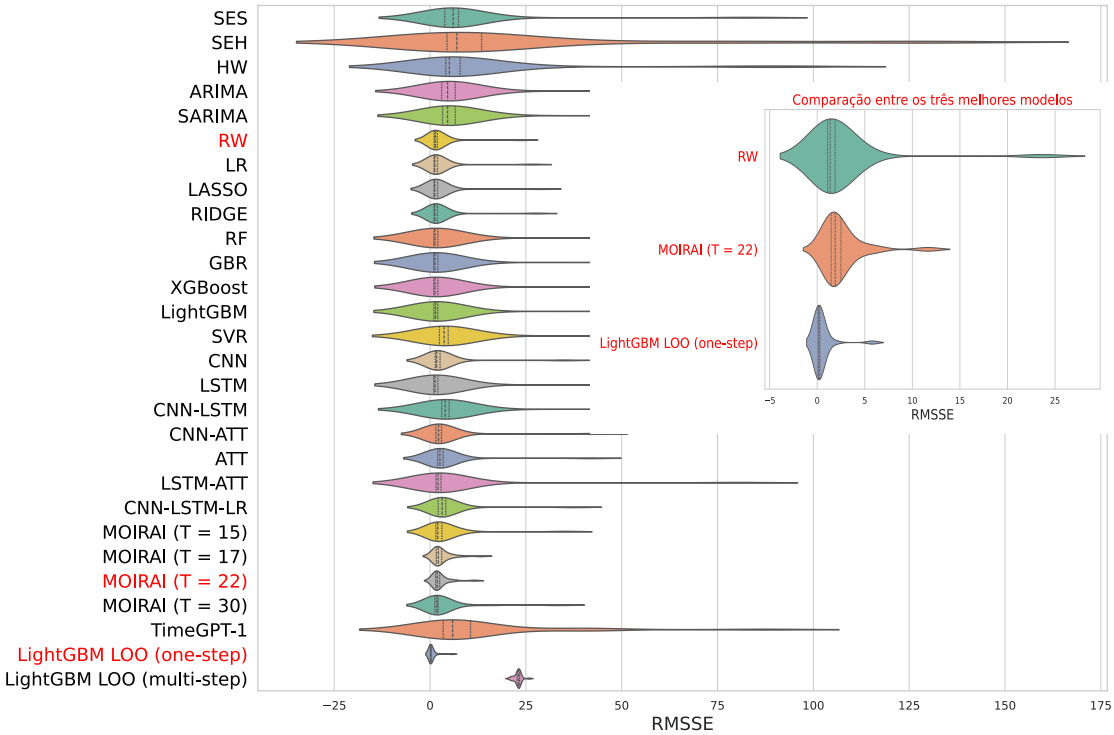


Figura 5.6. Distribuição numérica da métrica RMSSE

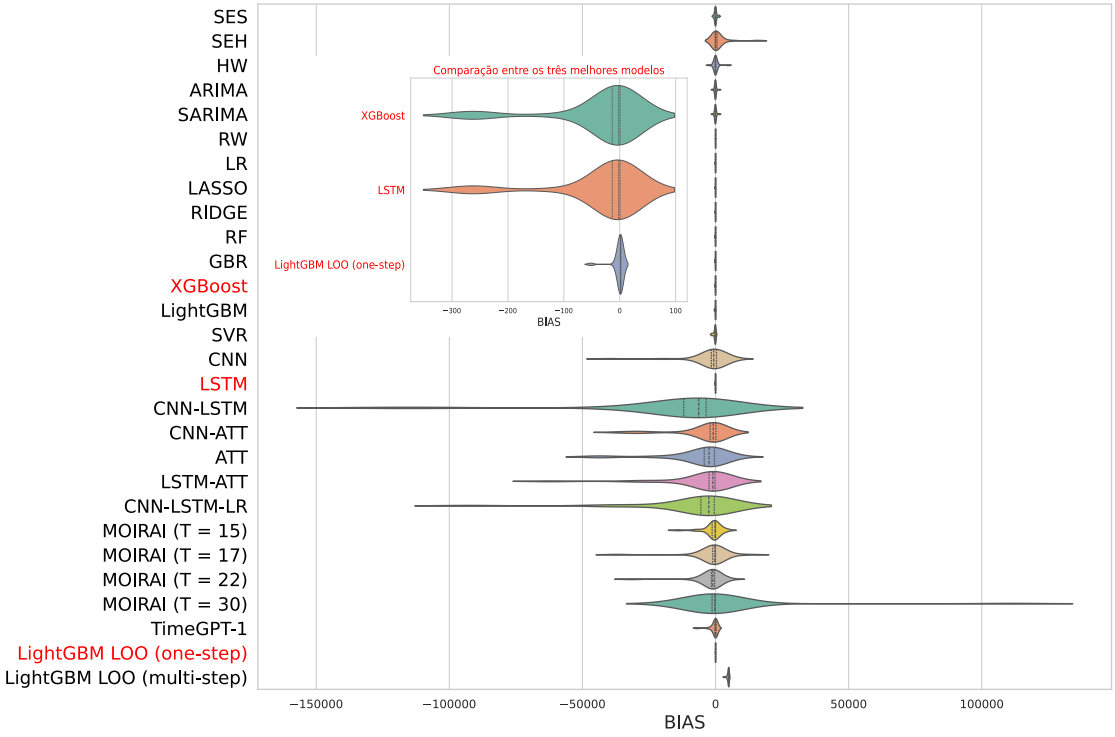


Figura 5.7. Distribuição numérica da métrica BIAS

Modelos estatísticos clássicos erraram em demasia, com destaque para SEH. Esse fato pode ser explicado pela tentativa de ajuste de um modelo linear (SEH) a

dados com natureza não linear. O SES obteve melhor desempenho que SEH e HW, apresentando desempenho similar ao ARIMA e SARIMA. Note que, apesar de sua simplicidade, o RW possui valores de métricas com distribuição concentrada ao redor dos menores valores quando comparados a modelos de aprendizado de máquina, não estando entre os três menores quanto ao BIAS; contudo, observe que, nesta métrica, sua distribuição encontra-se ao redor do valor nulo.

Modelos rasos apresentam resultados similares entre si, sendo o SVR o de pior desempenho. A regressão LASSO apresentou a segunda melhor MAE (**Figura 5.4**). Modelos profundos apresentaram resultados similares entre si. Entretanto, houve um aumento de BIAS no processo, indicando que esses modelos não permitiram uma boa generalização. O modelo CNN-ATT apresentou a segunda menor distribuição em RMSE (**Figura 5.3**) e os menores resultados em métricas quando comparado aos outros modelos profundos (**Figuras 5.3 − 5.7**), caracterizando-o como o melhor modelo profundo. Além disso, percebemos que modelos rasos possuem desempenho similar ou superior aos modelos profundos sem pré-treinamento. Entretanto, os modelos profundos demandam mais tempo para treinamento, dependendo do nível de complexidade de sua arquitetura. Portanto, os modelos rasos saem em vantagem devido ao seu desempenho e à menor complexidade computacional ao gerar seus resultados, o que nos motiva a questionar a necessidade de aprendizado profundo em tarefas de predição de séries temporais [Elsayed et al., 2021].

Em relação aos modelos fundadores, o TimeGPT-1 não demonstrou robustez, como evidenciado pela distribuição de sua RMSE na **Figura 5.3**. Por outro lado, o MOIRAI tende a apresentar um desempenho superior, pois permite predições *one-step* e a escolha do tamanho da janela. No entanto, observamos uma deterioração nas métricas à medida que o tamanho da janela aumenta (**Figuras 5.3 − 5.7**). A falha dos modelos fundadores pode ser atribuída ao treinamento em bases de dados de contextos distintos, à falta de afinamento de hiperparâmetros, e à ausência de um pré-processamento adequado em modelos *zero-shot* como TimeGPT-1 e MOIRAI, quando aplicados a tarefas de predição como o número de casos semanais de dengue. Por exemplo, estudos como o de Zeng et al. [2023] mostram que preditores lineares simples superaram Transformers pré-treinados em uma variedade de *datasets* em tarefas específicas. Isso ocorre porque, embora as arquiteturas de Transformers sejam excepcionais em tarefas de Processamento de Linguagem Natural, a natureza textual difere significativamente da natureza de uma série temporal. Séries temporais tendem a apresentar distribuições com maior aleatoriedade em comparação à estrutura sintática linguística. Este fato é corroborado pela superioridade do modelo RW em relação à maioria dos resultados observados.

As métricas MASE e RMSSE foram analisadas devido ao uso do LightGBM LOO (*one-step* e *multi-step*), pois as diferentes escalas dos *datasets* poderiam afetar a avaliação do desempenho pelas métricas RMSE e MAE. É interessante notar que o LightGBM LOO (*one-step*) obteve os menores valores em comparação aos outros modelos, mesmo tendo sido treinado em um maior número de *datasets*. Isso evidencia que, mesmo ao avaliar modelos rasos, profundos e fundadores em um único *dataset*, esses modelos, em geral, não superaram o desempenho do passeio aleatório.

Por fim, para avaliar o potencial de generalização do melhor modelo individual nas unidades federativas brasileiras, validamos o LightGBM LOO (*one-step*) em toda a base de San Juan e nas 187 últimas instâncias, com o objetivo de compará-lo aos modelos especificamente treinados na base de San Juan. O modelo superou todos os modelos univariados da **Tabela A.1**. Entretanto, o LightGBM LOO (*one-step*) não conseguiu superar os modelos LR (sym20) e CNN-LSTM-LR (sym20), o que possivelmente está relacionado às características particulares do *dataset* de San Juan nas quais esses modelos foram treinados, visto que são multivariados, e à ausência dessas características na base InfoDengue. No entanto, a **Tabela 5.3** mostra que o modelo não sofre de *overfitting*, apresentando boa capacidade de generalização. Portanto, embora o LightGBM LOO *one-step* não seja o melhor modelo no contexto específico de San Juan, ele demonstra potencial de aplicação em bases além desta, evidenciando que a estratégia de *cross-learning* com um modelo raso como o LightGBM mostrou-se eficiente em sua forma univariada, mesmo sem o uso de filtragem por *wavelets* específicas.

6

Conclusões

Neste trabalho, discutimos metodologias eficientes para a predição do número de casos semanais de dengue na cidade de San Juan (Porto Rico) e nas 27 unidades federativas brasileiras. O principal desafio consistiu em selecionar uma abordagem que melhor generalizasse as predições, considerando as variadas possibilidades na área de processamento de séries temporais, abrangendo desde métodos clássicos até técnicas baseadas em aprendizado de máquina.

Em nossa abordagem, testamos diversos modelos clássicos e de aprendizado de máquina (modelos rasos, profundos e fundadores), exploramos a utilidade de diferentes características ambientais disponíveis no auxílio à predição de dengue, e avaliamos distintas estratégias de reformatação de *datasets* visando a aprendizagem supervisionada, como as estratégias *one-step* e *multi-step*, bem como as perspectivas univariada e multivariada. Além disso, experimentamos uma ampla gama de *wavelets* na fase de pré-processamento, com o objetivo de avaliar sua eficiência na predição final.

6.1 Considerações finais

Na base de San Juan, verificamos que a melhor abordagem foi o modelo multivariado LR (sym20). Esse modelo, com o auxílio de características como temperatura e índice de vegetação, mostrou-se o mais adequado para a predição do número de casos semanais de dengue nessa região, superando arquiteturas profundas como a CNN-LSTM-LR. Embora a CNN-LSTM-LR tenha proporcionado um desempenho semelhante ao do LR (sym20), ela apresentou um aumento de viés e um custo computacional significativamente maior na fase de treinamento.

Na base InfoDengue, devido à maior quantidade de dados disponíveis, diferentes estratégias foram testadas. Contudo, a estratégia de maior êxito foi o modelo

univariado LightGBM, utilizando *leave-one-out* e predição *one-step*. Nem os modelos profundos, como as arquiteturas já experimentadas em San Juan, nem os modelos fundadores *zero-shot*, como TimeGPT-1 e MOIRAI, obtiveram resultados satisfatórios nas métricas avaliadas. Além disso, embora a superioridade do LightGBM LOO *one-step* em relação a todos os modelos analisados já indicasse uma boa generalização dos resultados no contexto brasileiro, mostramos que ele apresenta resultados similares, embora inferiores, aos melhores modelos de uma base de localidade distinta e sem o uso de características específicas da base de San Juan. Isso ressalta a possibilidade de estendê-lo para outros *datasets* e questiona a necessidade de características individuais de uma localidade geográfica, bem como o uso de modelos fundadores como modelos epidemiológicos de dengue.

6.2 Publicações

Os resultados obtidos na base de San Juan foram publicados no *LI Seminário Integrado de Software e Hardware* (SEMISH) [Zanardo et al., 2024]. Um resumo estendido com os resultados de San Juan foi publicado no *2º Simpósio do CI-IA Saúde* da UFMG. A extensão dos resultados para as unidades federativas brasileiras foi submetida ao *Journal of the Brazilian Computer Society* (JBCS).

6.3 Limitações

Analisando apenas a base de San Juan, devido ao pequeno tamanho do conjunto de dados, é possível que modelos rasos tenham se beneficiado em comparação com modelos profundos. Na base InfoDengue, embora tenha sido testado um amplo intervalo de tamanhos de janelas e de *wavelets* discretas, uma das limitações foi a ausência de uma verificação exaustiva dos hiperparâmetros de cada modelo. É possível que outros modelos pudessem ter apresentado um desempenho superior caso tivessem sido afinados de forma mais detalhada. Portanto, apesar de uma quantidade expressiva de modelos ter sido treinada, a avaliação do potencial individual de cada modelo foi comprometida. Outras técnicas de filtragem, como a Transformada Rápida de Fourier (FFT) e o Filtro de Kalman, poderiam ter sido aplicadas, considerando que o trabalho focou principalmente na filtragem por *wavelets*.

Além disso, é importante notar que, dadas n instâncias, ao utilizar estratégias como *one-step* e *multi-step*, há um aumento no número de características conforme o produto entre o tamanho da janela T e a quantidade D de características usadas, e uma diminuição no número de instâncias para $n - T$. Portanto, problemas como a

maldição da dimensionalidade [Hastie et al., 2009] podem ter afetado o desempenho de certos modelos.

Outra limitação é o fato de que, embora tenhamos duas bases distintas, isto não é suficiente para afirmar que o modelo LightGBM LOO (*one-step*) é generalizável para qualquer base de dados de dengue; os resultados mostram apenas um indicativo de possível generalização.

6.4 Trabalhos Futuros

Como trabalhos futuros, podemos testar outras estratégias de pré-processamento, como o uso de PCA, levando em consideração o domínio temporal [Jolliffe, 2002], com o objetivo de diminuir a dimensionalidade do *dataset*, buscar características mais significativas e aumentar a interpretabilidade do modelo. A dengue, assim como a maioria das doenças epidemiológicas, apresenta características sazonais. Portanto, construir modelos que considerem essa sazonalidade pode melhorar o processo de aprendizado. Outra possibilidade é que predições *multi-step* são interessantes para planejamentos a longo prazo, enquanto predições *one-step* são úteis para predições de curto prazo. Assim, visando obter resultados expressivos ao longo de meses (maior horizonte de predição), estudar maneiras de melhorar as predições *multi-step* torna-se uma alternativa promissora. Ademais, expandir a base de dados para incluir treinamento e validação em dados de localidades distintas possibilitaria uma melhor generalização dos resultados obtidos.

A

Tabelas de resultados em San Juan e cada unidade federativa brasileira

Neste apêndice, apresentamos uma tabela com os resultados gerais dos modelos aplicados em San Juan, além das tabelas com os resultados para todas as unidades federativas brasileiras da base InfoDengue. Os modelos univariados utilizam apenas a quantidade de casos semanais de dengue para treinamento e validação, enquanto os multivariados incorporam outras variáveis para o treinamento, sendo validados com base no número de casos semanais. A **Tabela A.1** exibe os resultados de San Juan para ambos os tipos de modelos. As **Tabelas A.2 – A.28** apresentam os resultados para as unidades federativas brasileiras, onde modelos multivariados são identificados pelo campo **Wavelet** preenchido, e os demais são univariados. Em cada unidade federativa, os modelos LightGBM LOO (*one-step* e *multi-step*) foram treinados com dados das outras 26 unidades federativas e validados na cidade indicada na legenda da tabela. Estes modelos foram validados em todo o *dataset* da unidade federativa em questão, enquanto os demais modelos foram validados nas 100 últimas instâncias.

		TREINO					VALIDAÇÃO				
ANÁLISE	**MODELO**	**MAE**	R^2	**MAPE**	**RMSE**	**BIAS**	**MAE**	R^2	**MAPE**	**RMSE**	**BIAS**
UNIVARIADA	SES	8.24	0.93	2.41e+13	14.13	-0.01	3.13e+14	-0.14	17.86	33.38	-11.75
	SEH	8.31	0.93	4.13e+13	14.17	0.96	2.03e+15	-8.16	80.43	94.56	75.52
	HW	9.46	0.92	2.88e+13	15.17	-0.14	5.32e+15	-95.08	271.04	306.27	0.00
	RW	8.25	0.93	2.41e+13	14.14	-0.01	1.69e+14	0.87	6.94	11.37	0.04
	ARIMA	8.21	0.94	6.02e+13	13.58	-0.04	8.70e+14	-0.13	27.53	33.17	0.00
	LR one-step	8.28	0.94	5.58e+13	13.37	0.00	2.41e+14	0.86	7.37	11.59	0.75
	SVR one-step	8.03	0.94	4.80e+13	13.86	-1.13	1.94e+14	0.87	6.97	11.09	-0.33
	GBR one-step	4.48	0.99	7.49e+13	5.81	0.00	1.38e+14	0.87	6.67	11.39	1.084
	RFR one-step	3.19	0.99	2.08e+13	6.03	0.04	1.63e+14	0.86	7.04	11.64	1.786
	XGB one-step	0.11	0.99	1,70e+12	0.16	0.00	2.27e+14	0.83	7.75	12.92	0.64
	LR multi-step	8.28	0.94	5.58e+13	13.37	0.00	7.55e+14	-0.03	24.49	31.68	5.49
	SVR multi-step	8.03	0.94	4.80e+13	13.86	-1.13	1.87e+14	-0.30	19.17	35.59	-17.08
	GBR multi-step	4.48	0.99	7.49e+13	5.81	0.00	4.16e+14	-0.02	17.99	31.56	-9.37
	RFR multi-step	3.19	0.98	2.08e+13	6.03	0.04	1.87e+14	-0.30	19.19	35.59	-17.14
	XGB multi-step	0.11	0.99	1.70e+12	0.16	0.00	5.96e+14	-0.28	26.44	35.41	6.56
MULTIVARIADA	RF (rbio2.4)	3.24	0.95	0.13	6.00	-0.05	3.76	0.95	0.20	6.18	-0.65
	XGBoost (rbio2.4)	2.29	0.98	0.13	3.05	0.00	3.71	0.94	0.20	6.64	-0.32
	GBR (rbio2.4)	1.95	1.00	0.11	2.53	0.00	3.84	0.94	0.22	6.50	-0.41
	LR (rbio2.4)	2.16	0.99	0.17	3.37	0.00	3.09	0.97	0.53	4.22	0.30
	SVR (rbio2.4)	1.85	0.98	0.12	4.24	0.19	2.29	0.98	0.25	3.86	-0.09
	CNN-LSTM (rbio2.4)	3.53	0.96	0.21	5.71	-684.93	4.18	0.95	0.30	6.27	-391.85
	LSTNet (rbio2.4)	19.20	-10.80	1.61	29.20	12420.50	14.80	-9.35	1.40	25.27	1850.00
	CNN-LSTM-LR (rbio2.4)	1.89	0.98	0.12	3.22	1.15	3.14	0.97	0.51	4.23	-89.67
	RF (coif8)	3.12	0.96	0.12	5.63	-0.03	3.74	0.95	0.21	5.6	-0.70
	XGBoost (coif8)	1.72	0.99	0.10	2.25	0.00	3.93	0.93	0.20	6.84	-0.59
	GBR (coif8)	1.79	0.99	0.10	2.33	0.00	3.84	0.94	0.21	6.47	-0.69
	LR (coif8)	0.28	0.99	0.02	0.39	0.00	0.40	0.99	0.08	0.51	0.00
	SVR (coif8)	0.81	0.99	0.05	1.62	0.10	0.99	0.99	0.10	1.48	0.00
	CNN-LSTM (coif8)	4.91	0.94	0.39	6.72	2158.42	6.76	0.88	0.64	8.48	20.16
	LSTNet (coif8)	18.20	-80.02	1.02	29.50	9641.68	14.42	-68.40	0.80	25.85	1083.45
	CNN-LSTM-LR (coif8)	1.45	0.99	0.09	2.20	-49.01	5.20	0.94	2.33	6.68	-7.86
	RF (db11)	3.02	0.96	0.12	5.36	0.01	3.65	0.95	0.20	5.70	-0.78
	XGBoost (db11)	2.47	0.99	0.13	3.23	0.00	3.66	0.95	0.20	6.10	-0.69
	GBR (db11)	1.70	0.99	0.09	2.20	0.00	3.80	0.95	0.21	5.88	-0.53
	LR (db11)	0.51	0.99	0.04	0.74	0.00	0.76	0.99	0.17	0.96	-0.07
	SVR (db11)	1.04	0.99	0.05	1.96	0.09	1.25	0.99	0.11	1.92	0.00
	CNN-LSTM-LR (db11)	0.26	0.99	0.02	0.38	12.70	0.40	0.99	0.08	0.51	0.19
	RF (sym20)	3.24	0.96	0.13	5.46	0.00	3.89	0.95	0.21	6.20	-0.84
	XGBoost (sym20)	1.80	0.99	0.11	2.32	0.00	4.01	0.93	0.21	6.83	-0.82
	GBR (sym20)	1.69	0.99	0.10	2.15	0.00	4.20	0.92	0.22	7.32	-0.46
	LR (sym20)	0.22	0.99	0.02	0.30	0.00	0.29	0.99	0.03	0.38	-0.01
	SVR (sym20)	0.80	0.99	0.05	1.46	0.09	1.04	0.99	0.08	1.52	-0.06
	LSTNet (sym20)	17.90	-94.91	0.92	28.85	8410.00	14.94	-77.93	0.75	26.00	731.16
	CNN-LSTM-LR (sym20)	0.21	0.99	0.01	0.30	31.43	0.30	0.99	0.03	0.38	7.00

Tabela A.1. Melhores resultados no treino e validação em San Juan.

Modelo	**Wavelet**	**Tamanho da janela (T)**	**MASE**	**RMSSE**	**MAE**	**RMSE**	**BIAS**
SES	—	—	5.18	3.88	550.14	762.39	19.26
SEH	—	—	5.32	5.20	564.90	1022.50	-453.64
HW	—	—	5.33	4.59	566.08	901.91	-433.22
ARIMA	—	—	4.58	3.98	486.76	781.73	-203.38
SARIMA	—	—	4.65	3.97	493.99	780.78	-168.55
RW	—	—	1.28	1.16	136.30	228.55	5.35
LR	bior3.1	17	1.44	1.11	152.16	217.05	35.38
LASSO	bior3.1	17	1.33	1.05	140.90	206.25	31.43
RIDGE	bior3.1	15	1.42	1.09	149.49	213.79	34.40
RF	db15	30	1.12	1.01	120.63	200.28	1.37
GBR	bior2.4	22	1.25	1.03	132.90	202.32	21.29
XGBoost	bior3.7	22	1.20	0.96	127.37	189.12	0.11
LightGBM	rbio1.3	22	1.19	1.00	126.33	196.73	16.80
SVR	coif16	15	4.73	4.40	499.07	860.82	-449.37
CNN	bior1.3	15	1.08	0.95	113.70	186.04	378.71
LSTM	bior1.1	30	4.64	3.95	497.63	781.62	-17339.32
CNN-LSTM	bior1.1	30	4.64	3.95	497.57	781.77	-17409.96
CNN-ATT	bior1.5	15	1.08	0.93	114.11	182.38	357.70
ATT	bior3.3	15	1.16	1.03	122.79	202.28	-1933.03
LSTM-ATT	bior4.4	15	1.35	1.23	142.65	240.96	-2808.74
CNN-LSTM-LR	db11	15	1.42	1.29	149.59	253.01	-623.44
MOIRAI	—	15	2.13	2.41	221.01	448.40	-701.23
MOIRAI	—	17	2.13	2.16	220.65	401.33	-761.41
MOIRAI	—	22	1.93	2.02	200.52	375.77	-3448.22
MOIRAI	—	30	1.66	1.78	172.00	331.38	-4228.88
TimeGPT-1	—	—	4.56	3.28	563.42	780.41	496.94
LightGBM LOO (one-step)	—	30	1.58	0.56	**68.67**	**119.07**	-5.19
LightGBM LOO (multi-step)	—	17	99.46	20.73	4369.25	4451.74	4369.25

Tabela A.2. Melhores resultados na validação para cidade de Goiânia (GO).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	10.87	7.84	1295.08	3270.31	-854.87
SEH	—	—	12.51	7.35	1490.27	3066.79	-198.84
HW	—	—	10.87	7.84	1295.08	3270.31	-854.87
ARIMA	—	—	11.19	8.12	1333.90	3387.59	-1230.62
SARIMA	—	—	11.19	8.12	1333.90	3387.59	-1230.62
RW	—	—	4.36	4.52	520.30	1888.16	-29.37
LR	db4	30	4.72	3.82	566.74	1602.99	-217.20
LASSO	db5	30	4.43	3.86	532.05	1622.03	-200.19
RIDGE	db5	30	4.56	3.83	547.85	1608.10	-204.91
RF	bior1.3	5	4.61	4.48	560.90	1853.33	-267.19
GBR	sym2	5	3.91	3.74	475.20	1544.50	-270.80
XGBoost	sym6	5	4.24	3.73	515.18	1540.65	-261.37
LightGBM	bior3.1	5	5.46	5.01	664.15	2070.48	-389.32
SVR	db19	5	12.57	8.46	1528.11	3496.93	-1521.21
CNN	db10	15	5.10	3.90	613.88	1620.98	-39598.58
LSTM	bior1.3	30	11.34	8.10	1362.89	3400.64	-126499.09
CNN-LSTM	bior1.5	30	11.34	8.10	1362.89	3400.64	-126499.14
CNN-ATT	db37	15	5.23	2.17	629.96	901.34	-35573.21
ATT	rbio2.2	22	4.87	2.07	580.85	861.27	-43434.11
LSTM-ATT	sym19	15	6.78	6.03	816.44	2505.33	-62052.50
CNN-LSTM-LR	rbio1.5	15	8.46	7.14	1017.98	2964.62	-92559.80
MOIRAI	—	15	8.00	7.49	1010.69	3092.29	-14134.44
MOIRAI	—	17	6.75	5.86	852.32	2421.29	-36980.67
MOIRAI	—	22	6.86	6.27	865.95	2589.39	-31520.54
MOIRAI	—	30	17.00	33.50	2147.58	13837.75	111576.56
TimeGPT-1	—	—	55.77	43.29	7044.43	17883.45	-6869.28
LightGBM LOO (one-step)	—	10	4.34	5.80	**165.37**	**825.04**	-51.68
LightGBM LOO (multi-step)	—	22	90.57	25.87	3404.62	3574.02	3171.36

Tabela A.3. Melhores resultados na validação para cidade de São Paulo (SP).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	6.40	4.04	652.98	1089.85	-521.56
SEH	—	—	30.77	13.18	3137.18	3556.98	3135.87
HW	—	—	7.80	4.24	794.95	1145.53	486.92
ARIMA	—	—	6.14	3.82	626.46	1031.74	-387.20
SARIMA	—	—	6.14	3.80	625.62	1026.54	-374.25
RW	—	—	1.65	1.11	168.09	301.05	0.78
LR	db35	15	1.51	1.02	152.56	273.18	-26.32
LASSO	coif16	22	1.54	1.03	156.51	278.02	-42.38
RIDGE	db38	30	1.53	1.01	157.40	273.31	-47.32
RF	rbio3.3	15	1.63	1.02	164.57	274.80	20.52
GBR	rbio3.1	30	1.35	0.88	139.54	239.78	4.38
XGBoost	db17	30	1.61	1.07	165.68	290.01	-13.11
LightGBM	coif4	30	1.65	1.16	170.19	314.39	-8.18
SVR	coif14	15	7.11	4.37	717.96	1172.54	-716.53
CNN	db33	30	2.39	1.40	246.20	380.00	3921.84
LSTM	bior1.1	30	6.07	3.75	625.59	1019.90	-35277.36
CNN-LSTM	bior1.1	30	6.07	3.75	625.57	1020.12	-35340.66
CNN-ATT	db20	30	3.74	2.22	386.05	603.08	-26894.00
LSTM-ATT	db24	15	3.04	1.98	307.42	531.98	-14192.73
ATT	sym13	15	3.08	1.88	311.64	505.23	-13701.57
CNN-LSTM-LR	db22	15	3.66	2.47	369.41	662.61	-24531.37
MOIRAI	—	15	2.28	1.69	220.13	429.13	-7328.12
MOIRAI	—	17	2.41	1.70	233.17	431.44	-7871.62
MOIRAI	—	22	2.10	1.46	202.93	371.56	-5607.99
MOIRAI	—	30	2.12	1.50	204.86	381.65	-3584.11
TimeGPT-1	—	—	12.61	9.91	1567.49	3467.08	-1554.27
LightGBM LOO (one-step)	—	30	1.50	0.81	**65.19**	**172.36**	0.87
LightGBM LOO (multi-step)	—	17	103.04	21.61	4526.13	4641.36	4489.71

Tabela A.4. Melhores resultados na validação para cidade de Rio de Janeiro (RJ).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	5.87	3.76	877.03	1788.94	-757.91
SEH	—	—	9.00	3.46	1345.06	1644.67	620.03
HW	—	—	9.93	4.73	1484.32	2252.32	524.53
ARIMA	—	—	6.16	3.58	921.66	1702.14	-530.90
SARIMA	—	—	6.16	3.58	921.66	1702.14	-530.90
RW	—	—	2.05	1.36	305.38	647.53	3.36
LR	bior1.5	15	2.12	1.13	332.66	546.31	53.52
LASSO	bior1.5	15	2.03	1.13	317.74	544.79	60.44
RIDGE	bior1.5	15	2.10	1.14	329.34	547.75	54.47
RF	db9	30	1.99	1.28	298.52	614.99	-34.01
GBR	db6	30	2.00	1.32	301.32	631.28	23.02
XGBoost	sym12	30	1.85	1.18	277.86	565.23	-13.66
LightGBM	sym11	15	2.72	1.82	427.52	876.70	162.02
SVR	db17	15	6.01	3.85	942.90	1857.33	-931.44
CNN	bior2.2	15	2.84	1.52	446.15	734.53	5325.06
LSTM	sym20	30	5.12	3.26	769.10	1560.23	-71076.23
CNN-ATT	db37	15	2.45	1.58	384.78	760.20	1925.26
LSTM-ATT	rbio3.1	15	2.60	1.56	407.80	753.72	858.34
ATT	sym9	22	3.42	1.95	511.43	928.46	-24129.12
CNN-LSTM-LR	db37	15	4.69	2.92	736.49	1410.57	-39484.02
MOIRAI	—	15	3.08	2.08	485.13	956.83	-6316.67
MOIRAI	—	17	3.22	2.24	507.65	1029.24	-2114.90
MOIRAI	—	22	2.82	1.99	444.43	916.28	-7106.27
MOIRAI	—	30	2.68	1.91	422.17	878.24	-10885.15
TimeGPT-1	—	—	13.31	10.24	2897.83	6584.44	-2858.44
LightGBM LOO (one-step)	—	30	2.38	1.82	**103.66**	**385.60**	-3.14
LightGBM LOO (multi-step)	—	17	102.83	21.42	4517.21	4599.45	4326.24

Tabela A.5. Melhores resultados na validação para cidade de Belo Horizonte (MG).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	7.59	6.04	178.95	293.99	-172.30
SEH	—	—	7.27	5.87	171.37	285.79	-161.95
HW	—	—	7.54	6.01	177.73	292.69	-170.28
ARIMA	—	—	7.06	5.39	166.28	262.45	-110.57
SARIMA	—	—	7.54	6.01	177.67	292.52	-169.84
RW	—	—	2.32	2.06	54.48	100.43	0.40
LR	db3	30	2.41	1.93	56.28	93.96	-6.07
LASSO	bior1.1	10	2.26	2.02	54.54	99.12	-6.09
RIDGE	db3	30	2.25	1.91	52.71	93.23	-7.48
RF	coif1	10	2.00	1.94	48.12	95.06	0.16
GBR	coif2	10	2.04	1.97	49.15	96.41	-4.93
XGB	sym16	15	2.12	1.93	50.68	94.32	-9.24
LightGBM	db38	5	2.24	1.94	55.60	97.48	-4.53
SVR	coif17	15	6.14	5.07	146.63	247.98	-123.80
CNN	coif6	15	2.47	1.97	59.03	96.50	-1043.21
LSTM	coif8	15	6.99	5.37	166.86	262.65	-11063.38
CNN-LSTM	coif15	15	6.99	5.36	166.85	261.87	-10878.83
CNN-ATT	bior1.3	15	3.57	2.88	85.10	140.94	88.80
LSTM-ATT	coif14	15	2.86	2.49	68.16	121.73	-1529.27
ATT	rbio5.5	22	3.71	2.85	87.36	138.94	-4708.05
CNN-LSTM-LR	db38	15	5.07	4.04	121.04	197.37	-4631.75
MOIRAI	—	15	2.79	2.65	69.88	138.34	-1809.09
MOIRAI	—	17	2.37	1.91	59.26	99.83	-1311.83
MOIRAI	—	22	2.23	1.79	55.89	93.67	-1177.37
MOIRAI	—	30	2.46	2.12	61.60	110.90	-1631.31
TimeGPT-1	—	—	5.99	4.31	193.80	320.52	-188.14
LightGBM LOO (one-step)	—	22	0.48	0.20	**20.93**	**41.95**	2.06
LightGBM LOO (multi-step)	—	17	111.71	22.94	4907.07	4926.36	4906.98

Tabela A.6. Melhores resultados na validação para cidade de Rio Branco (AC).

Model	Wavelet	Tamanho da Janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	11.51	10.93	51.87	91.09	-49.63
SEH	—	—	10.97	10.68	49.46	89.02	-45.57
HW	—	—	11.09	10.72	49.98	89.39	-46.22
ARIMA	—	—	11.59	10.95	52.23	91.30	-50.19
SARIMA	—	—	11.59	10.95	52.23	91.30	-50.19
RW	—	—	1.65	1.11	168.09	301.05	0.78
LR	db4	30	4.72	3.82	566.74	1602.99	-217.20
LASSO	db5	30	4.43	3.86	532.05	1622.03	-200.19
RIDGE	db5	30	4.56	3.83	547.85	1608.10	-204.91
RF	bior1.3	5	4.61	4.48	560.90	1853.33	-267.19
GBR	sym2	5	3.91	3.74	475.20	1544.50	-270.80
XGBoost	sym6	5	4.24	3.73	515.18	1540.65	-261.37
LightGBM	bior3.1	5	5.46	5.01	664.15	2070.48	-389.32
SVR	db19	5	12.57	8.46	1528.11	3496.93	-1521.21
CNN	sym4	15	5.97	5.22	26.73	43.31	-721.22
LSTM	db20	15	7.75	6.84	34.70	56.68	-2203.00
CNN-LSTM	db14	15	7.04	6.62	31.52	54.87	-2133.96
CNN-ATT	coif14	15	6.14	5.38	27.49	44.63	-568.04
LSTM-ATT	bior1.1	15	7.23	6.44	32.36	53.41	-2155.62
ATT	coif14	22	8.16	7.19	36.77	59.94	-1994.03
CNN-LSTM-LR	coif1	15	7.59	7.55	33.96	62.59	-2112.19
MOIRAI	—	15	5.72	4.79	30.55	59.52	-35.46
MOIRAI	—	17	5.20	3.98	27.73	49.45	-273.96
MOIRAI	—	22	4.78	3.84	25.53	47.74	-699.23
MOIRAI	—	30	4.65	3.74	24.80	46.47	-717.56
TimeGPT-1	—	—	10.21	5.57	81.09	164.01	12.62
LightGBM LOO (one-step)	—	30	0.16	0.07	**7.00**	**15.74**	2.23
LightGBM LOO (multi-step)	—	17	114.27	23.45	5019.77	5035.06	5019.77

Tabela A.7. Melhores resultados na validação para cidade de Porto Alegre (RS).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	24.22	12.59	1688.59	1803.25	1246.62
SEH	—	—	227.72	125.86	15874.14	18026.52	15828.57
HW	—	—	9.80	7.86	683.19	1126.37	12.31
ARIMA	—	—	27.50	14.24	1916.73	2039.56	1566.16
SARIMA	—	—	27.50	14.24	1916.73	2039.56	1566.16
RW	—	—	3.79	3.80	264.01	545.07	23.40
LR	bior1.5	5	3.32	3.71	238.62	533.78	-28.99
LASSO	sym6	5	3.31	3.71	237.98	533.34	-32.28
RIDGE	coif13	30	3.74	3.69	263.07	531.51	-63.42
RF	coif4	5	3.98	4.06	285.73	583.43	-73.55
GBR	sym8	5	3.91	3.92	280.51	564.01	-74.11
XGB	sym18	5	4.05	4.04	291.02	580.41	-115.12
LightGBM	sym4	17	4.77	4.69	336.10	673.04	-113.25
SVR	coif17	5	16.06	12.15	1153.53	1746.87	-1153.53
CNN	sym16	22	5.54	4.70	386.47	673.38	-19628.90
LSTM	sym17	15	14.64	11.59	1033.14	1662.27	-103222.42
CNN-LSTM	bior1.3	30	14.71	11.54	1033.90	1662.77	-103301.94
CNN-ATT	db20	15	6.80	5.79	479.91	830.62	-29302.70
LSTM-ATT	coif15	15	6.11	4.89	431.02	700.85	-23939.25
ATT	db6	30	7.67	6.46	539.29	931.14	-43640.65
CNN-LSTM-LR	db31	15	8.19	6.93	577.96	994.24	-38179.13
MOIRAI	—	15	3.27	3.53	260.34	562.92	4188.45
MOIRAI	—	17	3.94	5.02	313.98	800.19	12076.46
MOIRAI	—	22	2.94	3.44	233.95	548.87	4516.57
MOIRAI	—	30	3.23	3.68	257.02	587.06	7742.50
TimeGPT-1	—	—	36.39	39.48	2896.98	6293.22	-1715.71
LightGBM LOO (one-step)	—	30	1.33	0.78	**58.07**	**166.55**	3.28
LightGBM LOO (multi-step)	—	17	101.88	21.25	4475.42	4563.67	4468.37

Tabela A.8. Melhores resultados na validação para cidade de Brasília (DF).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	7.39	6.98	175.99	288.75	-151.74
SEH	—	—	7.40	6.99	176.20	289.03	-152.19
HW	—	—	7.39	6.98	176.06	288.85	-151.90
ARIMA	—	—	6.98	6.46	166.37	267.18	-109.17
SARIMA	—	—	7.03	6.39	167.37	264.48	-151.74
RW	—	—	2.32	3.03	55.28	125.29	0.90
LR	coif14	22	2.31	2.74	54.93	113.48	-4.98
LASSO	coif12	30	2.33	2.73	55.38	113.23	-0.50
RIDGE	db36	30	2.75	2.82	65.16	116.84	-6.38
RF	dmey	22	2.40	2.86	57.26	118.15	-15.74
GBR	coif14	30	2.58	2.94	61.27	121.65	-17.95
XGBoost	db38	17	2.77	2.86	66.01	118.22	-27.97
LightGBM	db37	30	2.61	2.79	61.90	115.57	-22.87
SVR	coif14	15	6.54	6.41	155.82	264.63	-133.24
CNN	coif15	15	3.20	3.09	76.09	127.57	-680.30
LSTM	coif13	15	7.05	6.46	167.86	266.54	-10339.28
CNN-LSTM	sym16	22	3.69	3.17	87.89	131.22	-4333.35
CNN-ATT	bior1.5	15	3.28	3.16	78.20	130.19	-1496.66
LSTM-ATT	bior1.5	15	3.57	3.13	84.98	128.94	-3210.86
ATT	db38	30	3.76	3.55	89.27	146.95	-2795.73
CNN-LSTM-LR	coif15	15	5.23	4.94	124.51	203.62	-5642.33
MOIRAI	—	15	4.87	8.28	113.05	333.98	2629.01
MOIRAI	—	17	3.11	3.32	72.12	134.00	-1009.18
MOIRAI	—	22	3.16	3.65	73.20	147.40	-1518.38
MOIRAI	—	30	2.94	3.31	68.21	133.59	-1584.80
TimeGPT-1	—	—	9.94	8.65	260.28	414.68	-237.84
LightGBM LOO (one-step)	—	17	0.52	0.20	**22.98**	**43.15**	2.48
LightGBM LOO (multi-step)	—	17	111.93	22.99	4916.82	4936.03	4916.78

Tabela A.9. Melhores resultados na validação para cidade de Vitória (ES).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	5.66	5.03	60.48	83.35	-56.31
SEH	—	—	4.75	4.47	50.73	73.94	-35.21
HW	—	—	5.67	5.04	60.52	83.39	-56.36
ARIMA	—	—	5.62	5.01	60.04	82.98	-55.77
SARIMA	—	—	5.62	5.01	60.04	82.98	-55.77
RW	—	—	1.89	1.72	20.25	28.45	0.33
LR	rbio2.6	17	1.95	1.71	20.74	28.21	-3.82
LASSO	db32	30	1.86	1.66	19.92	27.64	-1.65
RIDGE	bior4.4	22	2.03	1.70	21.70	28.08	-1.79
RF	coif7	15	1.89	1.74	20.01	28.67	-2.21
GBR	coif7	15	2.14	1.88	22.74	30.98	-2.57
XGBoost	coif17	15	1.96	1.78	20.79	29.34	-1.70
LightGBM	bior3.9	15	2.05	1.84	21.70	30.35	-4.48
SVR	coif17	15	4.37	4.00	46.40	65.99	-40.06
CNN	coif16	15	2.49	2.21	26.44	36.42	-620.97
LSTM	bior1.1	30	5.71	5.04	61.23	83.97	-5722.17
CNN-LSTM	bior1.1	30	5.71	5.04	61.23	83.97	-5722.17
CNN-ATT	coif12	15	3.09	2.71	32.80	44.70	-1232.57
LSTM-ATT	bior2.4	15	2.59	2.24	27.52	36.83	-1002.59
ATT	dmey	15	3.63	3.12	38.45	51.42	-2449.19
CNN-LSTM-LR	rbio3.1	15	3.59	3.46	38.06	56.99	-2057.19
MOIRAI	—	15	2.44	2.09	25.48	34.24	-82.50
MOIRAI	—	17	2.37	2.05	24.79	33.48	-32.37
MOIRAI	—	22	2.29	2.14	23.99	35.02	-192.21
MOIRAI	—	30	2.12	1.86	22.17	30.43	-144.47
TimeGPT-1	—	—	7.22	4.76	84.48	92.14	58.79
LightGBM LOO (one-step)	—	30	0.25	0.08	**10.80**	**16.85**	1.53
LightGBM LOO (multi-step)	—	17	113.59	23.32	4989.76	5007.03	4989.76

Tabela A.10. Melhores resultados na validação para cidade de Aracaju (SE).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	3.21	2.41	24.70	26.48	16.41
SEH	—	—	11.83	8.95	90.96	98.24	90.96
HW	—	—	3.45	2.57	26.52	28.25	19.14
ARIMA	—	—	3.16	2.38	24.29	26.12	15.82
SARIMA	—	—	3.19	2.40	24.55	26.35	16.20
RW	—	—	0.99	1.07	7.54	11.55	0.12
LR	db34	15	1.05	1.03	8.24	11.48	-2.91
LASSO	db23	15	0.95	1.01	7.40	11.30	-0.58
RIDGE	coif12	10	1.03	1.05	8.13	11.83	1.07
RF	rbio3.7	5	0.92	0.99	7.38	11.24	1.50
GBR	coif13	17	0.99	0.99	7.75	10.98	1.21
XGBoost	db6	22	1.12	1.10	8.62	12.03	1.57
LightGBM	db19	17	0.99	1.10	7.76	12.27	1.14
SVR	db38	5	1.03	1.14	8.26	12.96	-1.13
CNN	bior1.1	15	1.12	0.96	118.31	186.84	360.82
LSTM	bior1.1	30	4.64	3.95	497.89	780.89	-17007.35
CNN-LSTM	bior1.1	30	4.64	3.95	497.88	780.91	-17019.66
CNN-ATT	bior2.4	30	1.21	1.09	129.42	215.87	718.30
LSTM-ATT	sym20	15	1.41	1.28	149.15	250.62	679.37
ATT	bior3.5	15	1.15	1.04	121.43	203.28	-222.72
CNN-LSTM-LR	db32	30	1.52	1.54	11.53	16.62	-183.52
MOIRAI	—	15	1.15	1.17	8.96	13.06	-90.53
MOIRAI	—	17	1.13	1.22	8.81	13.62	-60.89
MOIRAI	—	22	1.18	1.28	9.21	14.36	-198.16
MOIRAI	—	30	1.14	1.24	8.89	13.83	-211.61
TimeGPT-1	—	—	1.59	1.18	15.07	20.83	1.57
LightGBM LOO (one-step)	—	30	0.16	0.05	**7.16**	**10.15**	1.45
LightGBM LOO (multi-step)	—	17	114.11	23.42	5012.68	5028.47	5012.66

Tabela A.11. Melhores resultados na validação para cidade de Boa Vista (RR).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	3.11	2.66	28.59	37.10	12.92
SEH	—	—	2.39	2.87	21.95	40.02	-3.93
HW	—	—	3.12	2.66	28.73	37.13	13.17
ARIMA	—	—	3.14	2.67	28.90	37.25	13.34
SARIMA	—	—	3.12	2.65	28.71	36.98	13.34
RW	—	—	0.91	0.96	8.42	13.36	0.22
LR	coif17	5	0.90	0.95	8.63	13.81	-0.81
LASSO	bior1.1	5	0.91	0.94	8.63	13.63	1.03
RIDGE	coif17	5	0.90	0.95	8.56	13.75	-0.54
RF	coif6	5	0.95	0.92	9.03	13.39	1.15
GBR	coif15	5	0.91	0.90	8.72	13.09	0.75
XGBoost	coif3	5	1.01	0.95	9.65	13.70	3.04
LightGBM	coif16	5	1.02	0.96	9.72	13.97	0.61
SVR	coif8	5	1.24	1.55	11.86	22.46	-2.12
CNN	coif17	15	1.54	1.50	14.24	20.95	318.26
LSTM	bior1.5	15	1.87	2.63	17.20	36.67	-1161.30
CNN-LSTM	bior2.2	15	1.87	2.63	17.21	36.64	-1151.55
CNN-ATT	db31	15	1.92	2.37	17.71	32.95	-317.62
ATT	coif15	15	2.40	2.46	22.14	34.26	595.98
LSTM-ATT	bior3.7	15	1.57	1.99	14.48	27.77	-448.65
CNN-LSTM-LR	sym9	30	1.91	2.28	17.50	31.71	-215.04
MOIRAI	—	15	1.04	1.10	9.80	15.68	-149.54
MOIRAI	—	17	1.16	1.25	10.90	17.84	-126.86
MOIRAI	—	22	1.12	1.22	10.51	17.33	-323.36
MOIRAI	—	30	1.10	1.17	10.35	16.63	-274.07
TimeGPT-1	—	—	3.36	5.24	33.20	78.87	-27.98
LightGBM LOO (one-step)	—	30	0.20	0.06	**8.60**	**12.68**	1.69
LightGBM LOO (multi-step)	—	17	113.91	23.38	5003.97	5020.15	5003.97

Tabela A.12. Melhores resultados na validação para cidade de Belém (PA).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	91.59	82.79	315.13	494.37	-315.13
SEH	—	—	91.05	82.53	313.30	492.83	-313.30
HW	—	—	91.58	82.79	315.11	494.35	-315.11
ARIMA	—	—	91.59	82.79	315.14	494.37	-315.14
SARIMA	—	—	91.59	82.79	315.14	494.37	-315.14
RW	—	—	21.53	23.63	74.16	141.23	-5.62
LR	coif17	30	25.66	26.60	89.26	159.85	-45.70
LASSO	rbio2.2	5	27.48	28.69	93.81	169.90	-42.25
RIDGE	bior1.1	17	27.20	27.79	94.61	167.02	-33.48
RF	coif15	5	79.15	77.81	270.25	460.72	-268.17
GBR	coif17	5	77.17	76.74	263.50	454.37	-259.79
XGBoost	db23	30	76.75	76.33	266.98	458.78	-264.18
LightGBM	bior2.6	10	80.43	77.98	275.14	462.89	-274.80
SVR	db15	5	91.95	83.92	313.95	496.90	-313.95
CNN	bior1.5	30	41.27	34.99	143.54	210.33	-5783.74
LSTM	haar	22	85.54	80.01	294.33	477.79	-29340.65
CNN-LSTM	coif10	15	82.23	79.17	282.91	471.72	-28224.49
CNN-ATT	bior1.3	15	44.36	43.38	152.64	258.49	-11605.07
LSTM-ATT	bior1.1	15	87.47	80.34	300.95	478.70	-30036.66
ATT	coif13	22	51.91	41.94	178.63	250.43	-13645.53
CNN-LSTM-LR	bior4.4	30	38.92	37.70	135.38	226.58	-8367.49
MOIRAI	—	15	28.79	35.45	148.25	497.70	3608.05
MOIRAI	—	17	20.33	13.35	104.71	187.46	-138.54
MOIRAI	—	22	16.11	11.64	82.94	163.46	-1404.54
MOIRAI	—	30	19.70	14.97	101.44	210.09	-2774.06
TimeGPT-1	—	—	129.99	87.19	669.37	1223.98	-648.99
LightGBM LOO (one-step)	—	22	0.24	0.17	**10.37**	**35.83**	2.37
LightGBM LOO (multi-step)	—	17	113.48	23.30	4984.85	5003.17	4984.85

Tabela A.13. Melhores resultados na validação para cidade de Florianópolis (SC).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	2.36	2.34	26.94	43.33	-23.92
SEH	—	—	3.83	3.28	43.69	60.73	-43.67
HW	—	—	2.36	2.34	26.95	43.35	-23.95
ARIMA	—	—	2.36	2.34	26.99	43.42	-24.07
SARIMA	—	—	2.36	2.34	26.99	43.42	-24.07
RW	—	—	0.86	0.84	9.82	15.48	0.14
LR	bior5.5	17	0.94	0.81	10.71	14.92	0.59
LASSO	db5	17	0.81	0.76	9.26	14.09	0.70
RIDGE	sym16	10	0.86	0.81	9.76	14.83	-1.51
RF	db22	30	0.93	0.87	10.65	16.15	1.41
GBR	db16	22	0.89	0.80	10.17	14.85	-1.01
XGBoost	rbio3.1	22	0.98	0.86	11.20	15.93	-1.25
LightGBM	db24	20	0.92	0.85	10.44	15.65	-0.02
SVR	coif15	5	1.12	1.06	12.90	19.75	-2.27
CNN	bior2.4	15	1.25	1.01	14.16	18.62	431.32
LSTM	bior1.5	15	2.31	2.19	26.27	40.32	-1790.08
CNN-LSTM	bior2.2	15	2.31	2.17	26.29	40.05	-1728.55
CNN-ATT	coif13	15	1.65	1.51	18.79	27.82	-370.62
LSTM-ATT	db38	15	1.38	1.50	15.67	27.69	-183.36
ATT	db9	22	1.71	1.56	19.51	28.79	-286.08
CNN-LSTM-LR	bior2.8	22	1.55	1.33	17.71	24.65	107.04
MOIRAI	—	15	1.09	1.10	12.23	19.37	-179.11
MOIRAI	—	17	1.05	1.10	11.80	19.41	-198.61
MOIRAI	—	22	1.04	1.08	11.63	19.09	-180.45
MOIRAI	—	30	1.01	1.05	11.36	18.47	-217.35
TimeGPT-1	—	—	1.40	1.16	18.97	29.67	-16.42
LightGBM LOO (one-step)	—	30	0.23	0.08	**10.22**	**15.97**	1.95
LightGBM LOO (multi-step)	—	17	113.90	23.38	5003.22	5020.13	5003.19

Tabela A.14. Melhores resultados na validação para cidade de Cuiabá (MT).

Model	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	5.15	2.37	277.47	309.41	146.13
SEH	—	—	5.63	2.59	303.73	336.90	196.63
HW	—	—	5.15	2.37	277.47	309.41	146.13
ARIMA	—	—	3.67	2.08	198.06	271.45	-56.88
SARIMA	—	—	3.84	2.13	206.86	277.09	-51.03
RW	—	—	1.01	0.61	54.57	79.99	4.02
LR	db10	10	0.96	0.52	51.50	67.10	8.20
LASSO	sym10	10	0.91	0.52	49.01	67.48	6.31
RIDGE	sym10	10	0.93	0.52	49.82	67.83	11.62
RF	db24	10	0.92	0.55	49.61	71.37	9.80
GBR	bior6.8	10	0.85	0.49	45.68	63.76	5.99
XGBoost	coif10	17	0.96	0.56	51.18	72.34	-0.29
LightGBM	db17	10	1.04	0.61	55.95	78.98	0.26
SVR	db35	5	3.36	2.20	185.45	288.27	-163.51
CNN	coif16	15	1.55	0.84	82.90	108.46	-1671.22
LSTM	db6	15	3.84	2.33	205.17	302.32	-13044.04
CNN-LSTM	db17	15	3.84	2.33	205.17	302.45	-13074.04
CNN-ATT	rbio5.5	15	2.01	1.20	107.48	155.35	-4016.83
LSTM-ATT	db17	15	2.01	1.28	107.34	165.74	-743.90
ATT	sym17	15	2.61	1.47	139.78	190.46	-2625.00
CNN-LSTM-LR	coif15	15	2.58	1.64	137.78	212.38	-4517.39
MOIRAI	—	15	1.45	1.01	79.98	119.33	-921.33
MOIRAI	—	17	1.73	1.22	95.86	144.66	81.63
MOIRAI	—	22	1.28	0.84	70.52	100.05	-1134.84
MOIRAI	—	30	1.16	0.79	63.99	93.66	1202.15
TimeGPT-1	—	—	3.46	1.64	271.47	332.41	203.22
LightGBM LOO (one-step)	—	30	0.79	0.35	**34.28**	**73.92**	-2.64
LightGBM LOO (multi-step)	—	17	109.25	22.51	4799.17	4833.76	4799.16

Tabela A.15. Melhores resultados na validação para cidade de Campo Grande (MS).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	4.01	3.95	104.27	246.54	-95.78
SEH	—	—	5.26	4.39	136.85	274.42	-135.52
HW	—	—	4.01	3.95	104.24	246.51	-95.74
ARIMA	—	—	3.93	3.89	102.18	243.07	-86.45
SARIMA	—	—	3.92	3.88	102.04	242.72	-85.43
RW	—	—	1.33	1.63	34.55	101.89	0.14
LR	coif16	15	1.43	1.32	36.92	81.86	-4.33
LASSO	coif17	17	1.32	1.35	34.15	84.31	-0.85
RIDGE	coif16	15	1.46	1.33	37.87	82.83	-3.19
RF	bior3.1	30	1.25	1.32	32.81	82.69	0.20
GBR	bior3.1	10	1.24	1.26	31.96	78.29	0.97
XGBoost	db34	10	1.27	1.31	32.57	81.27	-7.69
LightGBM	bior1.1	17	1.55	1.35	40.30	84.13	-14.27
SVR	coif14	5	3.29	3.48	84.11	214.37	-57.58
CNN	coif17	22	2.54	2.54	65.99	158.64	-2641.39
LSTM	bior4.4	15	3.90	3.84	100.93	238.93	-7402.58
CNN-LSTM	coif2	15	3.90	3.84	100.93	238.90	-7393.16
CNN-ATT	db24	15	2.23	2.40	57.63	149.28	-2547.61
LSTM-ATT	db10	22	2.79	2.84	72.47	177.74	-1983.59
ATT	sym7	30	3.74	3.27	97.84	205.48	-457.06
CNN-LSTM-LR	coif13	30	3.75	3.16	98.19	198.67	-2658.94
MOIRAI	—	15	1.91	2.32	47.84	139.16	-1857.52
MOIRAI	—	17	2.05	2.26	51.18	135.13	-1115.66
MOIRAI	—	22	1.68	1.96	41.96	117.46	-2084.09
MOIRAI	—	30	1.67	2.05	41.67	122.84	-2247.46
TimeGPT-1	—	—	2.60	1.83	103.10	247.75	-98.05
LightGBM LOO (one-step)	—	30	0.48	0.23	**20.87**	**48.97**	1.04
LightGBM LOO (multi-step)	—	17	112.09	23.03	4923.86	4944.33	4923.85

Tabela A.16. Melhores resultados na validação para cidade de Manaus (AM).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	17.86	9.51	1381.40	1401.49	1381.40
SEH	—	—	25.33	13.69	1959.55	2017.19	1959.55
HW	—	—	31.88	19.92	2466.21	2935.21	-2373.50
ARIMA	—	—	2.66	1.59	205.82	234.70	138.44
SARIMA	—	—	3.85	2.18	298.21	321.91	271.39
RW	—	—	0.77	0.64	59.43	94.65	15.25
LR	db29	10	0.77	0.53	58.64	77.79	7.26
LASSO	coif5	15	0.77	0.53	58.76	78.17	1.99
RIDGE	db29	10	0.80	0.56	61.04	81.07	11.14
RF	db31	17	0.72	0.55	55.28	81.33	-0.48
GBR	haar	17	0.71	0.54	54.52	79.62	14.49
XGBoost	db28	15	0.71	0.50	54.62	73.16	-6.05
LightGBM	db18	30	0.75	0.55	58.33	81.86	16.64
SVR	db32	5	1.77	1.48	134.20	214.82	-86.54
CNN	coif17	15	0.93	0.64	71.04	93.65	956.70
LSTM	db30	15	2.20	1.68	168.61	245.71	-6676.72
CNN-LSTM	rbio3.1	15	2.20	1.68	168.61	245.96	-6770.24
CNN-ATT	db37	15	0.96	0.78	73.50	113.89	-918.09
LSTM-ATT	coif13	15	0.98	0.71	74.95	103.56	-1387.92
ATT	db33	15	1.48	1.09	113.58	159.99	5350.07
CNN-LSTM-LR	coif17	15	1.50	1.22	115.49	178.42	-6296.32
MOIRAI	—	15	1.16	0.91	84.63	127.76	-305.13
MOIRAI	—	17	1.19	0.96	86.63	134.60	-200.95
MOIRAI	—	22	1.09	0.87	79.77	121.30	-1011.58
MOIRAI	—	30	0.92	0.72	67.32	100.76	79.39
TimeGPT-1	—	—	4.81	3.59	473.88	786.32	466.61
LightGBM LOO (one-step)	—	30	1.04	0.37	**45.40**	**79.40**	1.61
LightGBM LOO (multi-step)	—	17	105.27	21.75	4624.44	4670.83	4624.44

Tabela A.17. Melhores resultados na validação para cidade de Fortaleza (CE).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	10.87	6.20	267.88	292.42	257.83
SEH	—	—	21.73	12.04	535.42	567.68	535.42
HW	—	—	8.07	5.04	198.81	237.49	81.89
ARIMA	—	—	4.44	3.06	109.40	144.32	-78.86
SARIMA	—	—	5.70	3.55	140.59	167.43	19.33
RW	—	—	1.53	1.12	37.72	53.04	2.32
LR	coif14	5	1.52	1.14	37.35	53.34	0.01
LASSO	coif14	5	1.51	1.14	37.07	53.18	-0.75
RIDGE	coif14	5	1.52	1.14	37.24	53.30	-0.48
RF	bior3.7	15	1.41	1.07	34.81	50.56	-3.44
GBR	db18	22	1.54	1.11	38.00	52.44	-1.10
XGBoost	rbio5.5	5	1.59	1.13	39.06	52.61	-5.25
LightGBM	rbio3.7	30	1.56	1.12	38.75	53.17	-7.04
SVR	sym19	5	3.21	2.58	78.80	120.51	-64.52
CNN	db29	15	1.94	1.36	47.79	63.84	-1022.83
LSTM	bior1.1	30	4.48	3.21	111.35	152.11	-6409.62
CNN-LSTM	bior1.1	30	4.48	3.21	111.35	152.11	-6409.58
CNN-ATT	haar	15	2.75	2.09	67.78	98.57	-856.58
LSTM-ATT	db37	22	2.21	1.71	54.42	80.49	-2558.15
ATT	bior3.3	15	3.46	2.58	85.50	121.57	1122.31
CNN-LSTM-LR	bior1.3	15	2.69	2.02	66.34	95.12	98.74
MOIRAI	—	15	2.22	1.85	52.67	73.94	-1262.55
MOIRAI	—	17	2.46	2.17	58.48	86.44	-804.69
MOIRAI	—	22	2.10	1.74	49.86	69.52	-1661.68
MOIRAI	—	30	1.90	1.65	45.15	65.75	-1003.86
TimeGPT-1	—	—	12.87	5.88	360.58	385.57	360.58
LightGBM LOO (one-step)	—	15	0.60	0.23	**26.16**	**49.32**	4.04
LightGBM LOO (multi-step)	—	17	111.50	22.90	4897.96	4917.35	4897.96

Tabela A.18. Melhores resultados na validação para cidade de Palmas (TO).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	4.68	4.52	82.84	117.96	-23.47
SEH	—	—	9.32	7.01	165.03	182.87	106.53
HW	—	—	4.67	4.52	82.72	117.98	-23.80
ARIMA	—	—	4.24	4.65	75.12	121.38	-52.21
SARIMA	—	—	4.22	4.63	74.69	120.78	-51.91
RW	—	—	1.56	1.64	27.59	42.65	1.00
LR	db14	5	1.49	1.55	25.91	39.98	-1.28
LASSO	db5	5	1.51	1.57	26.22	40.45	-1.54
RIDGE	db14	5	1.48	1.55	25.86	40.02	-0.98
RF	db11	10	1.59	1.59	27.85	41.20	-4.84
GBR	coif11	10	1.48	1.54	26.01	39.84	-5.41
XGBoost	db34	10	1.58	1.55	27.76	40.14	-4.48
LightGBM	db9	10	1.59	1.53	27.87	39.64	-4.15
SVR	coif16	5	3.37	3.96	58.77	102.13	-50.02
CNN	coif14	15	1.50	1.52	26.48	39.56	-169.42
LSTM	bior1.1	22	4.50	4.91	79.77	127.96	-5485.79
CNN-LSTM	bior1.1	22	4.50	4.91	79.77	127.96	-5485.79
CNN-ATT	coif17	15	2.32	2.32	40.92	60.33	193.91
LSTM-ATT	coif15	15	2.16	2.09	37.98	54.27	1098.81
ATT	db24	22	2.74	2.83	48.50	73.94	-2664.24
CNN-LSTM-LR	rbio3.9	15	3.20	3.49	56.33	90.58	-1771.65
MOIRAI	—	15	1.84	1.98	32.19	51.85	41.37
MOIRAI	—	17	1.91	2.09	33.37	54.60	-227.63
MOIRAI	—	22	2.06	2.56	36.03	66.81	-506.70
MOIRAI	—	30	1.71	1.81	29.93	47.20	-191.85
TimeGPT-1	—	—	16.09	11.75	302.41	337.00	252.87
LightGBM LOO (one-step)	—	30	0.37	0.11	**16.10**	**24.21**	1.16
LightGBM LOO (multi-step)	—	17	112.47	23.09	4940.75	4958.98	4940.69

Tabela A.19. Melhores resultados na validação para cidade de João Pessoa (PB).

Modelo	Wavelet	Tamanho da Janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	2.60	2.67	16.98	25.95	-15.02
SEH	—	—	2.43	2.51	15.83	24.36	-12.82
HW	—	—	2.60	2.67	16.99	25.96	-15.04
ARIMA	—	—	2.62	2.69	17.12	26.11	-15.30
SARIMA	—	—	2.60	2.67	16.96	25.93	-14.99
RW	—	—	1.26	1.11	8.22	10.81	0.12
LR	db20	5	1.19	1.07	8.02	10.75	-0.07
LASSO	coif17	5	1.17	1.08	7.90	10.90	-1.05
RIDGE	db27	5	1.18	1.06	7.95	10.73	-0.06
RF	bior1.1	5	1.16	1.03	7.86	10.39	0.62
GBR	rbio3.1	22	1.26	1.13	8.25	10.94	0.23
XGBoost	db6	5	1.05	0.99	7.11	10.03	-0.20
LightGBM	bior4.4	5	1.18	1.04	7.96	10.54	0.34
SVR	db27	5	1.53	1.64	10.33	16.55	-5.28
CNN	rbio5.5	15	1.88	1.92	12.41	18.88	136.35
LSTM	bior2.4	15	2.30	2.25	15.14	22.07	-626.72
CNN-LSTM	bior4.4	15	1.73	2.14	11.40	20.98	-435.28
CNN-ATT	rbio2.2	15	1.97	1.78	12.99	17.46	-426.73
LSTM-ATT	db21	22	1.93	2.07	12.61	20.14	-447.13
ATT	coif8	15	2.90	2.82	19.09	27.64	-365.55
CNN-LSTM-LR	coif6	22	2.31	2.19	15.07	21.31	188.72
MOIRAI	—	15	1.64	1.62	11.05	16.16	-69.43
MOIRAI	—	17	1.57	1.58	10.58	15.77	-47.91
MOIRAI	—	22	1.62	1.53	10.93	15.30	-115.63
MOIRAI	—	30	1.33	1.16	8.91	11.60	-0.11
TimeGPT-1	—	—	5.11	6.43	35.06	66.06	-33.30
LightGBM LOO (one-step)	—	30	0.15	0.05	**6.63**	**9.64**	1.94
LightGBM LOO (multi-step)	—	17	114.28	23.45	5020.00	5036.07	5020.00

Tabela A.20. Melhores resultados na validação para cidade de Macapá (AP).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	22.87	13.91	733.32	757.48	733.32
SEH	—	—	195.73	129.22	6276.82	7038.07	6276.82
HW	—	—	149.62	95.49	4798.26	5200.93	4798.26
ARIMA	—	—	5.28	3.91	169.33	212.98	72.70
SARIMA	—	—	4.90	3.79	157.02	206.48	80.26
RW	—	—	1.05	0.96	33.68	52.13	7.15
LR	rbio2.6	5	0.96	0.90	30.23	48.50	8.36
LASSO	dmey	10	0.98	0.91	31.16	49.40	5.08
RIDGE	rbio2.6	5	0.96	0.91	30.37	48.82	9.24
RF	db10	5	1.05	0.97	33.12	52.39	9.05
GBR	db2	15	1.01	0.91	32.23	49.36	9.68
XGBoost	db13	15	0.92	0.88	29.39	47.49	0.95
LightGBM	db24	5	0.99	0.92	31.36	49.42	1.17
SVR	bior1.1	5	2.71	2.97	85.76	159.79	-59.12
CNN	db35	15	1.34	1.23	42.87	66.94	1612.90
LSTM	coif6	15	3.74	3.85	119.60	209.01	-8755.21
CNN-LSTM	coif13	15	3.74	3.84	119.56	208.30	-8583.48
CNN-ATT	bior1.5	15	1.65	1.49	52.69	81.05	912.96
LSTM-ATT	db23	22	1.83	1.58	58.83	85.89	3024.73
ATT	db5	15	2.30	2.18	73.40	118.09	-3680.00
CNN-LSTM-LR	coif17	15	2.80	2.81	89.32	152.26	-5402.16
MOIRAI	—	15	1.55	1.52	47.23	78.35	-253.39
MOIRAI	—	17	2.00	2.73	61.05	140.93	1028.71
MOIRAI	—	22	1.55	1.47	47.37	75.89	-811.83
MOIRAI	—	30	1.30	1.24	39.70	64.28	-34.99
TimeGPT-1	—	—	16.29	9.15	594.79	618.26	594.79
LightGBM LOO (one-step)	—	30	0.58	0.19	**25.45**	**40.45**	1.46
LightGBM LOO (multi-step)	—	17	110.83	22.77	4868.42	4889.54	4868.41

Tabela A.21. Melhores resultados na validação para cidade de Natal (RN).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	6.87	4.68	48.70	57.15	11.70
SEH	—	—	8.17	5.40	57.86	65.94	34.67
HW	—	—	6.88	4.68	48.76	57.17	11.86
ARIMA	—	—	5.83	5.33	41.32	65.18	-29.94
SARIMA	—	—	5.83	5.33	41.32	65.18	-29.94
RW	—	—	2.10	2.04	14.87	25.00	0.77
LR	db35	15	2.06	1.93	14.52	23.52	-1.11
LASSO	rbio3.1	22	1.90	1.91	13.43	23.33	-2.40
RIDGE	bior1.1	22	1.95	1.93	13.82	23.53	-3.00
RF	coif16	17	2.21	2.10	15.61	25.55	-0.93
GBR	db17	17	2.14	2.08	15.07	25.29	-1.64
XGBoost	db12	15	2.36	2.13	16.61	25.94	1.22
LightGBM	db5	15	2.19	2.07	15.38	25.22	-0.37
SVR	coif14	5	4.31	4.27	30.08	51.58	-25.91
CNN	db35	15	3.20	2.69	22.53	32.69	-699.99
LSTM	coif15	15	3.65	3.58	25.66	43.51	-1079.31
CNN-LSTM	bior1.1	30	6.14	5.48	43.72	67.28	-3738.96
CNN-ATT	db13	15	3.83	3.38	26.95	41.05	-1306.46
LSTM-ATT	db37	22	3.33	2.90	23.61	35.50	-884.72
ATT	sym3	22	4.29	3.65	30.40	44.61	-620.64
CNN-LSTM-LR	db16	30	4.62	4.12	32.87	50.58	-1392.70
MOIRAI	—	15	3.02	2.64	22.04	35.51	-340.11
MOIRAI	—	17	2.93	2.53	21.43	34.04	-588.40
MOIRAI	—	22	2.84	2.43	20.77	32.68	-624.34
MOIRAI	—	30	2.38	1.99	17.41	26.77	-402.71
TimeGPT-1	—	—	6.30	2.88	57.44	68.14	35.39
LightGBM LOO (one-step)	—	30	0.17	0.06	**7.51**	**12.91**	1.79
LightGBM LOO (multi-step)	—	17	114.19	23.44	5016.12	5032.89	5016.12

Tabela A.22. Melhores resultados na validação para cidade de Porto Velho (RO).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	7.59	6.95	54.05	122.18	-49.83
SEH	—	—	7.59	6.95	54.05	122.18	-49.83
HW	—	—	7.59	6.95	54.04	122.17	-49.81
ARIMA	—	—	7.26	6.69	51.66	117.55	-36.13
SARIMA	—	—	7.26	6.69	51.66	117.55	-36.13
RW	—	—	3.25	4.25	23.18	74.83	-1.34
LR	db10	30	3.38	3.80	24.29	67.23	-2.90
LASSO	sym19	30	3.28	4.09	23.55	72.31	-3.34
RIDGE	sym10	30	3.32	3.88	23.85	68.57	-3.39
RF	dmey	5	3.34	3.32	23.46	57.65	-9.70
GBR	sym12	5	3.19	3.07	22.44	53.34	-8.18
XGBoost	coif15	5	3.46	3.10	24.30	53.79	-13.67
LightGBM	coif10	10	3.82	3.77	26.91	65.71	-9.13
SVR	db19	5	5.96	6.41	41.91	111.29	-38.59
CNN	rbio1.3	30	4.63	4.86	33.25	85.96	-1601.09
LSTM	rbio1.3	15	4.25	3.91	30.01	68.27	-1784.13
CNN-LSTM	db22	15	4.24	4.81	29.98	84.03	-1157.49
CNN-ATT	db37	15	4.91	4.16	34.67	72.75	-1238.62
LSTM-ATT	db1	15	4.70	3.79	33.22	66.26	-1012.91
ATT	coif4	22	5.15	3.94	36.63	69.22	-3139.52
CNN-LSTM-LR	haar	15	5.28	5.32	37.33	92.93	-2518.11
MOIRAI	—	15	5.10	5.32	36.46	92.13	-1369.45
MOIRAI	—	17	5.04	5.46	36.07	94.58	-1058.87
MOIRAI	—	22	4.77	5.42	34.11	93.94	-1184.04
MOIRAI	—	30	4.28	5.18	30.62	89.68	-782.76
TimeGPT-1	—	—	40.27	39.37	287.93	682.15	-283.50
LightGBM LOO (one-step)	—	30	0.19	0.11	**8.41**	**22.77**	0.77
LightGBM LOO (multi-step)	—	17	114.04	23.41	5009.33	5026.67	5009.33

Tabela A.23. Melhores resultados na validação para cidade de Curitiba (PR).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	19.65	11.82	338.32	355.04	338.32
SEH	—	—	42.18	25.35	726.23	761.43	726.23
HW	—	—	19.65	11.82	338.32	355.04	338.32
ARIMA	—	—	19.20	11.58	330.57	347.67	330.57
SARIMA	—	—	20.44	12.27	351.99	368.53	351.99
RW	—	—	1.45	1.29	24.90	38.76	3.91
LR	—	5	1.41	1.22	24.11	36.32	0.42
LASSO	—	5	1.41	1.22	24.08	36.28	0.43
RIDGE	—	5	1.41	1.22	24.11	36.32	0.42
RF	—	5	1.43	1.20	24.44	35.62	2.61
GBR	—	5	1.28	1.10	21.86	32.76	3.15
XGBoost	—	5	1.45	1.29	24.75	38.44	0.48
LightGBM	—	22	1.67	1.49	28.67	44.64	6.20
SVR	—	5	3.11	2.79	53.12	82.85	-37.33
CNN	—	22	2.52	2.23	43.32	66.93	-260.00
LSTM	—	30	5.70	4.79	97.54	143.78	-9526.76
CNN-LSTM	—	30	5.83	4.86	99.89	145.91	-9845.73
CNN-ATT	—	22	1.43	1.22	24.67	36.65	-341.50
LSTM-ATT	—	30	1.69	1.40	28.95	41.97	395.70
ATT	—	30	5.62	4.70	96.25	141.16	-8941.17
CNN-LSTM-LR	—	15	4.58	3.38	78.95	101.11	571.43
MOIRAI	—	15	2.19	2.01	36.33	58.21	114.63
MOIRAI	—	17	2.40	2.18	39.87	62.89	323.84
MOIRAI	—	22	2.09	1.87	34.73	54.07	86.26
MOIRAI	—	30	2.17	1.92	35.94	55.61	-112.00
TimeGPT-1	—	—	16.35	7.88	358.08	394.64	358.08
LightGBM LOO (one-step)	—	30	0.33	0.11	**14.44**	**23.32**	1.48
LightGBM LOO (multi-step)	—	17	112.86	23.17	4957.71	4975.34	4957.71

Tabela A.24. Melhores resultados na validação para cidade de Teresina (PI).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	9.62	6.47	88.45	98.83	84.61
SEH	—	—	50.85	34.21	467.40	522.80	467.40
HW	—	—	9.62	6.47	88.45	98.83	84.61
ARIMA	—	—	9.62	6.47	88.45	98.83	84.61
SARIMA	—	—	9.62	6.47	88.45	98.83	84.61
RW	—	—	1.61	1.39	14.73	21.16	1.31
LR	sym11	10	1.57	1.31	14.29	19.89	-1.94
LASSO	db3	5	1.60	1.34	14.52	20.27	-1.57
RIDGE	bior3.1	5	1.65	1.36	14.99	20.52	-3.46
RF	db1	10	1.65	1.32	15.08	19.98	3.13
GBR	db27	17	1.61	1.38	14.75	21.10	-2.60
XGBoost	db37	30	1.63	1.38	14.55	20.57	0.68
LightGBM	dmey	30	1.74	1.43	15.53	21.30	-1.02
SVR	db30	5	2.82	2.58	25.61	38.99	-18.72
CNN	coif17	15	1.79	1.53	16.40	23.32	-566.12
LSTM	bior1.1	22	4.95	3.98	45.54	60.87	-3312.98
CNN-LSTM	bior1.1	22	4.95	3.98	45.54	60.87	-3312.98
CNN-ATT	coif17	15	2.14	1.68	19.57	25.49	117.05
ATT	db35	15	2.51	2.24	22.94	34.04	-25.36
LSTM-ATT	db36	22	2.67	2.28	24.58	34.90	-998.26
CNN-LSTM-LR	db34	15	3.28	2.88	29.99	43.78	-1076.76
MOIRAI	—	15	2.10	1.86	18.45	27.51	-117.24
MOIRAI	—	17	3.63	8.62	31.93	127.49	1199.10
MOIRAI	—	22	2.15	1.90	18.91	28.03	-454.87
MOIRAI	—	30	1.92	1.62	16.90	23.98	-242.32
TimeGPT-1	—	—	6.08	3.64	64.50	72.06	51.06
LightGBM LOO (one-step)	—	30	0.21	0.07	**9.19**	**14.54**	2.01
LightGBM LOO (multi-step)	—	17	113.91	23.38	5003.80	5020.98	5003.80

Tabela A.25. Melhores resultados na validação para cidade de São Luís (MA).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	11.56	7.00	217.31	228.70	173.35
SEH	—	—	110.58	72.32	2079.19	2363.54	2063.91
HW	—	—	36.84	22.25	692.60	727.17	670.31
ARIMA	—	—	16.30	9.76	306.55	319.05	277.16
SARIMA	—	—	16.43	9.83	308.83	321.40	279.53
RW	—	—	1.43	1.37	26.67	44.65	3.68
LR	bior2.4	10	1.34	1.23	26.69	42.39	2.56
LASSO	rbio1.5	10	1.23	1.20	24.39	41.51	0.98
RIDGE	sym4	10	1.31	1.22	26.01	42.03	2.26
RF	haar	10	1.27	1.19	25.35	40.99	6.59
GBR	db1	10	1.32	1.20	26.27	41.54	8.57
XGBoost	coif1	15	1.25	1.11	24.44	37.84	2.05
LightGBM	db12	10	1.31	1.23	25.99	42.53	2.06
SVR	coif16	5	2.96	3.60	59.57	125.02	-42.64
CNN	db25	15	2.02	1.72	39.52	58.61	-676.84
LSTM	bior3.9	15	4.48	4.71	87.49	160.60	-5947.78
CNN-LSTM	bior6.8	15	4.48	4.73	87.47	161.45	-6173.89
CNN-ATT	bior3.5	15	2.79	2.64	54.58	89.87	2079.48
LSTM-ATT	db24	15	2.40	2.13	46.88	72.75	1916.23
ATT	db37	22	2.84	2.74	53.40	89.60	-2093.26
CNN-LSTM-LR	db27	15	3.43	3.65	67.05	124.48	-3242.20
MOIRAI	—	15	1.49	1.52	29.63	52.68	-8.01
MOIRAI	—	17	1.78	1.90	35.40	65.69	757.79
MOIRAI	—	22	1.39	1.50	27.65	52.06	-396.33
MOIRAI	—	30	1.30	1.29	25.94	44.67	-235.75
TimeGPT-1	—	—	26.50	15.33	605.61	618.33	605.61
LightGBM LOO (one-step)	—	30	0.34	0.11	**15.01**	**24.33**	1.16
LightGBM LOO (multi-step)	—	17	112.47	23.08	4940.55	4955.96	4940.55

Tabela A.26. Melhores resultados na validação para cidade de Maceió (AL).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	2.55	2.97	96.46	202.80	-62.75
SEH	—	—	2.68	3.13	101.52	214.03	-80.37
HW	—	—	2.89	3.26	109.17	222.47	-96.36
ARIMA	—	—	2.55	2.97	96.46	202.97	-63.30
SARIMA	—	—	2.55	2.97	96.46	202.97	-63.30
RW	—	—	0.85	0.88	31.94	60.06	0.69
LR	coif14	10	0.81	0.81	31.54	56.52	-2.34
LASSO	db30	5	0.78	0.82	30.15	57.00	-0.60
RIDGE	coif17	5	0.80	0.82	30.93	57.24	1.07
RF	coif13	10	0.84	0.75	32.47	52.00	7.36
GBR	db7	5	0.88	0.78	34.03	54.31	7.22
XGBoost	coif4	10	0.96	0.76	37.20	52.94	10.87
LightGBM	db25	30	0.93	0.89	34.36	59.72	4.56
SVR	coif16	5	1.91	2.29	74.14	159.29	-38.68
CNN	coif17	15	1.57	1.69	61.15	117.98	-1636.16
LSTM	coif6	15	2.48	2.90	96.46	202.79	-6272.19
CNN-LSTM	coif12	15	2.48	2.90	96.46	202.96	-6326.50
CNN-ATT	db26	15	1.55	1.59	60.38	111.10	-1165.32
LSTM-ATT	bior2.2	15	2.01	2.04	78.21	142.98	-184.07
ATT	db31	15	2.03	2.06	78.93	144.19	-3776.26
CNN-LSTM-LR	coif8	22	2.25	2.07	85.25	141.52	-340.96
MOIRAI	—	15	1.21	1.21	45.10	79.04	-1284.76
MOIRAI	—	17	1.22	1.31	45.78	85.36	-1212.00
MOIRAI	—	22	1.24	1.25	46.19	81.06	-1121.58
MOIRAI	—	30	1.06	1.18	39.79	77.08	-1017.68
TimeGPT-1	—	—	2.16	2.47	97.37	212.95	-83.78
LightGBM LOO (one-step)	—	30	0.64	0.23	**27.78**	**49.13**	1.73
LightGBM LOO (multi-step)	—	17	110.18	22.62	4840.09	4857.36	4840.09

Tabela A.27. Melhores resultados na validação para cidade de Recife (PE).

Modelo	Wavelet	Tamanho da janela (T)	MASE	RMSSE	MAE	RMSE	BIAS
SES	—	—	7.56	6.50	154.76	213.50	-154.74
SEH	—	—	7.49	6.46	153.24	212.07	-153.22
HW	—	—	8.37	6.97	171.33	228.84	-171.32
ARIMA	—	—	4.42	4.57	90.39	150.17	-68.96
SARIMA	—	—	4.42	4.57	90.39	150.17	-68.96
RW	—	—	1.55	1.54	31.54	50.23	-0.53
LR	rbio3.1	5	1.54	1.47	32.00	48.57	1.76
LASSO	rbio3.1	5	1.49	1.46	31.05	48.34	1.65
RIDGE	rbio3.1	5	1.52	1.46	31.68	48.43	1.76
RF	sym20	10	1.52	1.43	31.60	47.50	-3.83
GBR	bior1.3	5	1.55	1.47	32.20	48.77	3.03
XGBoost	coif9	15	1.59	1.50	33.24	49.91	-1.49
LightGBM	sym4	5	1.67	1.59	34.73	52.47	-5.46
SVR	sym12	5	3.30	3.67	68.54	121.51	-54.38
CNN	coif6	15	2.25	2.21	46.86	73.28	-1558.27
LSTM	coif14	15	5.04	4.65	105.07	154.34	-4671.46
CNN-LSTM	db6	15	5.04	4.64	105.07	154.01	-4561.89
CNN-ATT	db14	15	3.26	3.09	68.04	102.68	-4429.40
LSTM-ATT	coif11	22	2.67	2.55	54.63	83.70	-2275.81
ATT	db23	22	2.94	2.48	60.09	81.60	-2217.07
CNN-LSTM-LR	sym16	15	3.58	3.33	74.67	110.48	-3042.29
MOIRAI	—	15	2.98	5.85	60.90	189.74	898.03
MOIRAI	—	17	2.25	2.14	45.92	69.42	-1048.95
MOIRAI	—	22	2.12	1.89	43.35	61.44	-1570.96
MOIRAI	—	30	1.91	1.78	38.93	57.87	-64.16
TimeGPT-1	—	—	11.88	10.85	250.40	377.18	-249.74
LightGBM LOO (one-step)	—	30	0.40	0.13	**17.53**	**28.00**	1.46
LightGBM LOO (multi-step)	—	17	111.86	22.96	4913.93	4929.91	4913.86

Tabela A.28. Melhores resultados na validação para cidade de Salvador (BA).

Referências Bibliográficas

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y. & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Alpaydin, E. (2020). *Introduction to Machine Learning*. MIT Press, Cambridge, MA, 4 edição.

Appice, A.; Gel, Y. R.; Iliev, I.; Lyubchich, V. & Malerba, D. (2020). A multi-stage machine learning approach to predict dengue incidence: A case study in mexico. *IEEE Access*, 8:52713--52725.

Benidis, K.; Rangapuram, S. S.; Flunkert, V.; Wang, Y.; Maddix, D.; Turkmen, C.; Gasthaus, J.; Bohlke-Schneider, M.; Salinas, D.; Stella, L.; Aubet, F.-X.; Callot, L. & Januschowski, T. (2022). Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6):1–36.

Berger, R. & Casella, G. (2001). *Statistical Inference*. Duxbury Press, Florence, AL, 2 edição.

Bogado, J. V.; Schaerer, C. E.; Stalder, D. H. & Martínez, G. (2023). Cluster-based lstm models to improve dengue cases forecast. *CLEI Eletronic Journal (CLEIej)*, 26(1).

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S. & et. al. (2021). On the opportunities and risks of foundation models. *ArXiv*. Report available at https://crfm.stanford.edu/assets/report.pdf.

Box, G. E. P.; Jenkins, G. M.; Reinsel, G. C. & Ljung, G. M. (2015). *Time series analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, Nashville, TN, 5 edição.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Buczak, A. L.; Baugher, B.; Moniz, L. J.; Bagley, T.; Babin, S. M. & Guven, E. (2018). Ensemble method for dengue prediction. *PLOS ONE*, 13(1):e0189988.

Cabrera, M.; Leake, J.; Naranjo-Torres, J.; Valero, N.; Cabrera, J. C. & Rodríguez-Morales, A. J. (2022). Dengue prediction in latin america using machine learning and the one health perspective: A literature review. *Tropical Medicine and Infectious Disease*, 7(10).

Carvajal, T.; Viacrusis, K.; Hernandez, L.; Ho, H.; Amalin, D. & Watanabe, K. (2018). Machine learning methods reveal the temporal pattern of dengue incidence using meteorological factors in metropolitan manila, philippines. *BMC Infect. Dis.*, 18(1):183.

Carvalho, B.; Rangel, E. & Vale, M. (2016). Evaluation of the impacts of climate change on disease vectors through ecological niche modelling. *Bulletin of Entomological Research*, 107(4):419–430.

Celentano, D. D.; Sifakis, F.; Go, V. & Davis, W. (2008). Changing sexual mores and disease transmission. Em *The Social Ecology of Infectious Diseases*, pp. 50--76. Elsevier.

Chakraborty, T.; Chattopadhyay, S. & Ghosh, I. (2019). Forecasting dengue epidemics using a hybrid methodology. *Physica A*, 527.

Chen, T. & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Em *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM.

Codeco, C.; Coelho, F.; Cruz, O.; Oliveira, S.; Castro, T. & Bastos, L. (2018). Infodengue: A nowcasting system for the surveillance of arboviruses in brazil. *Revue d'Épidémiologie et de Santé Publique*, 66:S386. European Congress of Epidemiology "Crises, epidemiological transitions and the role of epidemiologists".

Cormen, T. H. & Leiserson, C. E. (2022). *Introduction to Algorithms, fourth edition*. MIT Press, London, England.

da Silva, L. J. & Angerami, R. N. (2008). *Viroses emergentes no Brasil*.

Daubechies, I. (1992). *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics.

Deb, S. & Deb, S. (2021). An ensemble method for early prediction of dengue outbreak. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 185(1):84–101.

Derrick, T. & Thomas, J. (2004). *Time-Series Analysis: The Cross-Correlation Function*, pp. 189–205. Human Kinetics Publishers, Champaign, Illinois. Posted with permission.

Drucker, H.; Burges, C. J.; Kaufman, L.; Smola, A. & Vapnik, V. (1997). Support vector regression machines. pp. 155–161. Bell Labs and Monmouth University, Department of Electronic Engineering, West Long Branch, NJ 07764, AT&T Labs.

Elsayed, S.; Thyssens, D.; Rashed, A.; Jomaa, H. S. & Schmidt-Thieme, L. (2021). Do we really need deep learning models for time series forecasting?

Ferdousi, T.; Cohnstaedt, L. W. & Scoglio, C. M. (2021). A windowed correlation-based feature selection method to improve time series prediction of dengue fever cases. *IEEE Access*, 9:141210–141222.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5).

Fuller, W. A. (1976). *Introduction to statistical time series*. Probability & Mathematical Statistics S. John Wiley & Sons, Nashville, TN.

Garza, A.; Challu, C. & Mergenthaler-Canseco, M. (2023). Timegpt-1.

Goodfellow, I.; Courville, A. & Bengio, Y. (2016). Deep learning. Adaptive Computation and Machine Learning series. MIT Press, London, England.

Grossmann, A. (1988). *Wavelet Transforms and Edge Detection*. Reidel, Boston.

Grossmann, A. & Morlet, J. (1984). Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM Journal of Mathematics*, 15:723–736.

Guo, P.; Liu, T. & Zhang, Q. e. a. (2017). Developing a dengue forecast model using machine learning: a case study in china. *PLoS Negl. Trop. Dis.*, 11(10).

Haar, A. (1910). Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69:331--371.

Hastie, T.; Tibshirani, R. & Friedman, J. (2009). Introduction. Em *The Elements of Statistical Learning*, Springer series in statistics, pp. 1--8. Springer New York, New York, NY.

Hoffman, K. M. & Kunze, R. (1971). *Linear Algebra*. Pearson, Upper Saddle River, NJ, 2 edição.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554--2558.

Hyndman, R. J. & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679--688.

Jolliffe, I. (2002). *Principal Component Analysis for Time Series and Other Non-Independent Data*, pp. 299--337. Springer New York, New York, NY.

Kamal, M.; Kenawy, M. A.; Rady, M. H.; Khaled, A. S. & Samy, A. M. (2018). Mapping the global potential distributions of two arboviral vectors aedes aegypti and ae. albopictus under changing climate. *PLOS ONE*, 13(12):e0210122.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q. & Liu, T.-Y. (2017). Lightgbm: a highly efficient gradient boosting decision tree. Em *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 3149–3157, Red Hook, NY, USA. Curran Associates Inc.

Kreyszig, E. (2010). *Advanced Engineering Mathematics 10E*. John Wiley & Sons, Chichester, England.

Kumar, V. R. & Mehra, M. (2005). Wavelet based preconditioners for sparse linear systems. *Applied Mathematics and Computation*, 171:203--224.

Lai, G.; Chang, W.; Yang, Y. & Liu, H. (2017). Modeling long- and short-term temporal patterns with deep neural networks. *CoRR*, abs/1703.07015.

Laporta, G. Z.; Potter, A. M.; Oliveira, J. F. A.; Bourke, B. P.; Pecor, D. B. & Linton, Y.-M. (2023). Global distribution of aedes aegypti and aedes albopictus in a climate change scenario of regional rivalry. *Insects*, 14(1):49.

Laverdeur, J.; Desmecht, D.; Hayette, M.-P. & Darcis, G. (2024). Dengue and chikungunya: future threats for northern europe? *Frontiers in Epidemiology*, 4.

Lee, G. R.; Gommers, R.; Waselewski, F.; Wohlfahrt, K. & Leary, A. (2019). Pywavelets: A python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237.

Lopes, N.; Nozawa, C. & Linhares, R. E. C. (2014). Características gerais e epidemiologia dos arbovírus emergentes no brasil. *Revista Pan-Amazônica de Saúde*, 5(3).

Makridakis, S.; Spiliotis, E. & Assimakopoulos, V. (2020). The m4 competition: 100, 000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74.

Makridakis, S.; Spiliotis, E. & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364.

Mallat, S. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674--693. 1989b.

Mallat, S. (2016). Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203.

Martínez, J. M. & Augusta, S. (1998). *Métodos Computacionais de Otimização*. Campinas.

Mitchell, T. (1997). Machine Learning. Em *McGraw-Hill Higher Education*.

Morettin, P. A. (2014). *Ondas e Ondaletas: Da Análise de Fourier à Análise de Ondaletas de Séries Temporais*. EDUSP, 2 edição. ISBN-10: 8531414784.

Morettin, P. A. & Toloi, C. M. C. (2018). *Análise de Séries Temporais: Modelos Lineares Univariados*, volume 1. Blucher, 3 edição.

Morettin, P. A. & Toloi, C. M. C. (2020). *Análise de Séries Temporais: Modelos Multivariados e Não Lineares*, volume 2. Blucher, 1 edição.

Morlet, J. (1981). Sampling theory and wave propagation. Em *51st Annual Meeting Society of Exploration Geophysicists*, Los Angeles.

Mussumeci, E. & Coelho, F. C. (2020). Large-scale multivariate forecasting models for dengue - lstm versus random forest regression. *Spatial and Spatio-temporal Epidemiology*, 35.

Necesito, I.; Velasco, J. M.; Kwak, J.; Lee, J.; Lee, M.; Kim, J. & Kim, H. (2021). Combination of univariate long-short term memory network and wavelet transform for predicting dengue case density in the national capital region, the philippines. *The Southeast Asian journal of tropical medicine and public health*, 52:479–494.

Neto, S. R. S.; Oliveira, T. T.; Teixeira, I. V. et al. (2022). Machine learning and deep learning techniques to support clinical diagnosis of arboviral diseases: A systematic review. pp. 1–37. PLoS Negl Trop Dis.

Nocedal, J. & Wright, S. J. (2000). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, 1 edição.

OPAS & OMS (2024). Atualização epidemiológica: Aumento de casos de dengue na região das américas. Acesso em: 13 jul. 2024.

Organização Pan-Americana da Saúde / Organização Mundial da Saúde (2024). Atualização epidemiológica: Aumento de casos de dengue na região das américas. 18 de junho de 2024.

Panja, M.; Chakraborty, T.; Nadim, S. S.; Ghosh, I.; Kumar, U. & Liu, N. (2023). An ensemble neural network approach to forecast dengue outbreak based on climatic condition. *Chaos, Solitons & Fractals*, 167:113124.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J. & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825--2830.

Rezaei, M. & Shahidi, M. (2020). Zero-shot learning and its applications from autonomous vehicles to covid-19 diagnosis: A review. *Intelligence-Based Medicine*, 3–4:100005.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386--408.

Ross, T. M. (2010). Dengue virus. *Clinics in Laboratory Medicine*, 30(1):149--160.

Rumelhart, D. E.; Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533--536.

Russell, S. & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4 edição.

San Martin, J.; Solorzano, J. & Guzman, M. e. a. (2010). The epidemiology of dengue in the americas over the last three decades: a worrisome reality. *Am. J. Trop. Med. Hyg.*, 82(1):128--135.

Santos, C. A. G.; Guerra-Gomes, I. C.; Gois, B. M.; Peixoto, R. F.; Keesen, T. S. L. & da Silva, R. M. (2019). Correlation of dengue incidence and rainfall occurrence using wavelet transform for joão pessoa city. *Science of The Total Environment*, 647:794–805.

Seabold, S. & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. Em *9th Python in Science Conference*.

Sebastianelli, A.; Spiller, D.; Carmo, R.; Wheeler, J.; Nowakowski, A.; Jacobson, L. V.; Kim, D.; Barlevi, H.; Cordero, Z. E. R.; Colón-González, F. J.; Lowe, R.; Ullo, S. L. & Schneider, R. (2024). A reproducible ensemble machine learning approach to forecast dengue outbreaks. *Scientific Reports*, 14(1).

Shaikh, M. S. G.; SureshKumar, D. B. & Narang, D. (2023). Development of optimized ensemble classifier for dengue fever prediction and recommendation system. *Biomedical Signal Processing and Control*, 85:104809.

Shazeer, N. (2020). GLU variants improve transformer. *CoRR*, abs/2002.05202.

Strang, G. & Nguyen, T. (1996). *Wavelets and filter banks*. Wellesley-Cambridge Press, Wellesley, MA, 2 edição.

US National Oceanic and Atmospheric Administration (2017). Dengue forecasting project website. Acessado em 21 de fevereiro de 2024.

Vapnik, V. (2013). *The nature of statistical learning theory*. Information Science and Statistics. Springer, New York, NY, 2 edição.

Vapnik, V. N. & Chervonenkis, A. Y. (1964). On a class of pattern-recognition learning algorithms. *Automation and Remote Control*, 25(6):838--845.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. & Polosukhin, I. (2023). Attention is all you need.

Woo, G.; Liu, C.; Kumar, A.; Xiong, C.; Savarese, S. & Sahoo, D. (2024). Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592*.

World Health Organization (2009). *Dengue: Guidelines for diagnosis, treatment, prevention and control*. World Health Organization, Genève, Switzerland.

World Health Organization (2023). Dengue - global situation. [Online; accessed 12-29-2023].

Wu, Y.; Lee, G.; Fu, X. & Hung, T. G. G. (2008). Detect climatic factors contributing to dengue outbreak based on wavelet, support vector machines and genetic algorithm.

Zanardo, G.; Éfren Souza; Nakamura, F. & Nakamura, E. (2024). Uma comparação entre métodos baseados em aprendizado de máquina para inferir número de casos semanais de dengue. Em *Anais do LI Seminário Integrado de Software e Hardware*, pp. 37--48, Porto Alegre, RS, Brasil. SBC.

Zeng, A.; Chen, M.; Zhang, L. & Xu, Q. (2023). Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128.

Zhang, B. & Sennrich, R. (2019). Root mean square layer normalization.