Universidade Federal de Uberlândia
Instituto de Matemática e Estatística

Bacharelado em Estatística

Modelagem e Previsão de Variáveis Climáticas Usando Modelos SARIMA, VAR e LSTM

Izabella Cristina Nunes

Uberlândia-MG
2025

Izabella Cristina Nunes

Modelagem e Previsão de Variáveis Climáticas Usando Modelos SARIMA, VAR e LSTM

Trabalho de conclusão de curso apresentado à Coordenação do Curso de Bacharelado em Estatística como requisito parcial para obtenção do grau de Bacharel em Estatística.

Orientador: Prof. Dr. José Waldemar da Silva

Uberlândia-MG

2025

Universidade Federal de Uberlândia
Instituto de Matemática e Estatística

Coordenação do Curso de Bacharelado em Estatística

A banca examinadora, conforme abaixo assinado, certifica a adequação deste trabalho de conclusão de curso para obtenção do grau de Bacharel em Estatística.

Uberlândia, _____ de _____ de 20_____

BANCA EXAMINADORA

_____

Prof. Dr. José Waldemar da Silva

_____

Prof Dr. Marcelo Tavares

_____

Profa. Dra. Nádia Giaretta Biase

Uberlândia-MG

2025

Resumo

Este trabalho tem como objetivo a análise e previsão de dados climáticos, mais especificamente as variáveis de precipitação, velocidade do vento, temperatura e umidade, para a cidade de Uberlândia - MG, utilizando modelos de séries temporais. Serão empregados três modelos de previsão: o modelo SARIMA (*Seasonal Autoregressive Integrated Moving Average*), o modelo VAR (*Vector Autoregressive*) e a rede neural LSTM (*Long Short-Term Memory*). A escolha dessas variáveis climáticas se dá pela sua importância para diversos setores, como agricultura, meteorologia e gestão de recursos naturais, onde a previsão precisa desses dados é crucial. O estudo visa comparar a eficácia dos modelos, tradicionalmente utilizados na literatura (SARIMA e VAR) e uma metodologia nova (LSTM), na previsão dessas variáveis, além de avaliar a aplicabilidade de cada técnica em diferentes cenários. Para isso, serão utilizados dados históricos climáticos obtidos a partir da estação climatológica do Instituto Nacional de Meteorologia (INMET), localizada no campus Santa Mônica da Universidade Federal de Uberlândia (UFU), no período de Janeiro de 2014 á Dezembro de 2024, que serão tratados e analisados conforme as características de cada modelo. As métricas de validação utilizadas para comparar a performance dos modelos serão a RMSE (Raiz do Erro Quadrático Médio), a MSE (Erro Quadrático Médio) e o MAE (Erro Médio Absoluto), que permitem indicar a qualidade das previsões. A análise revelou que o SARIMA apresentou melhor desempenho para precipitação, temperatura e umidade, as LSTMs se destacaram na previsão do vento devido à sua flexibilidade frente a séries mais irregulares, enquanto o VAR apresentou erros mais elevados. Esses resultados reforçam a importância de avaliar comparativamente diferentes técnicas, considerando a natureza das séries temporais e os fenômenos climáticos analisados.

Palavras-chave: Métricas de Validação; Previsão; Redes Neurais; Seleção automática de Modelos; Variáveis Climáticas.

Abstract

This study aims to analyze and forecast climatic data, specifically the variables of precipitation, wind speed, temperature, and humidity, using time series models. Three forecasting models are employed: the SARIMA model (Seasonal Autoregressive Integrated Moving Average), the VAR model (Vector Autoregressive), and the LSTM neural network (Long Short-Term Memory). These climatic variables were chosen due to their importance in various sectors, such as agriculture, meteorology, and natural resource management, where accurate predictions are crucial. The study seeks to compare the effectiveness of the models in forecasting these variables and to evaluate the applicability of each technique in different scenarios. Historical climate data are used, processed, and analyzed according to the characteristics of each model. The validation metrics used to compare model performance are RMSE (Root Mean Squared Error), MSE (Mean Squared Error), and MAE (Mean Absolute Error), which allow assessing the precision and explanatory capacity of each model, as well as the quality of the forecasts. The analysis revealed that SARIMA performed best for precipitation, temperature, and humidity, LSTMs excelled in forecasting wind speed due to their flexibility in handling more irregular series, while VAR presented consistently higher errors. These results highlight the importance of comparatively evaluating different techniques, considering both the statistical properties of the series and the nature of the climatic phenomena analyzed.

Keywords: Automatic Model Selection; Climatic Variables; Forecasting; Neural Networks; Validation Metrics.

Sumário

Lista de Figuras **I**

Lista de Tabelas **II**

1 Introdução **1**
1.1 Introdução . . . 1
1.2 Objetivos . . . 2

2 Fundamentação Teórica **3**
2.1 Séries Temporais . . . 3
2.2 Modelo SARIMA . . . 3
2.3 Modelo VAR . . . 5
2.4 Modelo LSTM (Long Short-Term Memory) . . . 7
2.5 Métricas de Validação . . . 10

3 Metodologia **12**
3.1 Dados . . . 12
3.2 Modelo SARIMA . . . 12
3.3 Modelo VAR . . . 13
3.4 Modelo LSTM (Long Short-Term Memory) . . . 14
3.5 Métricas de Validação . . . 14

4 Resultados **16**
4.1 Análise Descritiva dos Dados . . . 16
4.2 Modelo SARIMA . . . 18
4.3 Modelo VAR . . . 21
4.4 Modelo LSTM (Long Short-Term Memory) . . . 24

5 Conclusões **28**

Referências Bibliográficas **29**

Apêndice A Código SARIMA **31**

Apêndice B Código VAR **41**

Apêndice C Código LSTM **47**

Lista de Figuras

2.1 Ilustração de uma rede neural com duas variáveis de entrada, duas camadas ocultas, com 4 e 3 neurônios respectivamente e uma variável de saída. 7

4.1 Série temporal da variável precipitação. 16
4.2 Série temporal da variável temperatura. 17
4.3 Série temporal da variável umidade. 17
4.4 Série temporal da variável velocidade do vento. 17
4.5 Previsão da série temporal de precipitação utilizando o modelo SARIMA. 20
4.6 Previsão da série temporal de temperatura utilizando o modelo SARIMA. 20
4.7 Previsão da série temporal de umidade utilizando o modelo SARIMA. 20
4.8 Previsão da série temporal de vento utilizando o modelo SARIMA. 21
4.9 Previsão da série temporal de precipitação utilizando o modelo VAR. 23
4.10 Previsão da série temporal de temperatura utilizando o modelo VAR. 23
4.11 Previsão da série temporal de umidade utilizando o modelo VAR. 23
4.12 Previsão da série temporal de vento utilizando o modelo VAR. 24
4.13 Previsão da série temporal de precipitação utilizando o modelo LSTM. 25
4.14 Previsão da série temporal de temperatura utilizando o modelo LSTM. 26
4.15 Previsão da série temporal de umidade utilizando o modelo LSTM. 26
4.16 Previsão da série temporal de vento utilizando o modelo LSTM. 26

Lista de Tabelas

4.1 P-valores do teste ADF antes e após a diferenciação 16
4.2 Valores de AIC e BIC para cada variável analisada 18
4.3 Resultados dos testes de diagnóstico dos resíduos dos modelos. 18
4.4 Valores observados, previstos e intervalos de confiança (95%) para precipitação. 19
4.5 Valores observados, previstos e intervalos de confiança (95%) para temperatura. 19
4.6 Valores observados, previstos e intervalos de confiança (95%) para vento. 19
4.7 Valores observados, previstos e intervalos de confiança (95%) para umidade. 19
4.8 Resultados (valores-p) do teste de causalidade de Granger e testes de diagnóstico aplicados ao modelo VAR 21
4.9 Valores observados, previstos e intervalos de confiança (95%) para precipitação. 22
4.10 Valores observados, previstos e intervalos de confiança (95%) para temperatura. 22
4.11 Valores observados, previstos e intervalos de confiança (95%) para umidade. 22
4.12 Valores observados, previstos e intervalos de confiança (95%) para vento. 22
4.13 Valores observados e previstos para precipitação utilizando o modelo LSTM. 24
4.14 Valores observados e previstos para temperatura utilizando o modelo LSTM. 24
4.15 Valores observados e previstos para umidade utilizando o modelo LSTM. 25
4.16 Valores observados e previstos para velocidade do vento utilizando o modelo LSTM. 25
4.17 Arquitetura do modelo LSTM definida automaticamente para cada variável. 27

1. Introdução

1.1 Introdução

A análise e previsão de variáveis climáticas desempenham papel central em diversos setores da sociedade, como agricultura, gestão de recursos hídricos, planejamento urbano e formulação de políticas ambientais. A capacidade de antecipar o comportamento de elementos meteorológicos, tais como precipitação, temperatura, umidade relativa do ar e velocidade do vento, é fundamental para reduzir impactos socioeconômicos associados a eventos extremos e apoiar a tomada de decisões em cenários de incerteza. Na cidade de Uberlândia, localizada na região do Triângulo Mineiro, tal análise assume relevância adicional devido à sua posição geográfica, marcada por variações sazonais expressivas e forte relação entre condições climáticas e atividades econômicas locais, além disso a previsão dessas variáveis é de fundamental importância para o setor agrícola.

Tradicionalmente, a previsão de séries temporais climáticas tem sido realizada por meio de modelos estatísticos, como o ARIMA (*Autoregressive Integrated Moving Average*) e sua extensão sazonal, o SARIMA (*Seasonal Autoregressive Integrated Moving Average*), amplamente utilizados em situações em que há padrões de dependência temporal regulares. Outra abordagem comumente adotada é o modelo VAR (*Vector AutoRegressive*), que possibilita analisar simultaneamente múltiplas variáveis interdependentes, capturando suas interações dinâmicas ao longo do tempo. Embora eficazes, esses modelos assumem linearidade nas relações, o que pode limitar sua capacidade preditiva em sistemas complexos e não lineares, o que pode acontecer em processos meteorológicos.

Diante desse cenário, este trabalho propõe a utilização complementar de técnicas de aprendizado de máquina, em especial a rede neural LSTM (*Long Short-Term Memory*), cuja estrutura foi projetada para capturar dependências de longo prazo em dados sequenciais e lidar com não linearidades intrínsecas. Ao comparar o desempenho do LSTM com modelos estatísticos tradicionais, busca-se avaliar em que medida abordagens modernas de inteligência artificial podem superar ou complementar métodos consolidados na literatura.

A justificativa para a adoção dessa metodologia reside na possibilidade de combinar as vantagens de cada abordagem. Enquanto os modelos estatísticos fornecem interpretabilidade e fundamentação teórica consolidada, as redes neurais recorrentes, como o LSTM, oferecem flexibilidade para capturar padrões complexos que escapam às hipóteses lineares. Assim, espera-se que a comparação entre SARIMA, VAR e LSTM forneça uma visão abrangente sobre as potencialidades e limitações de cada técnica na previsão de variáveis climáticas em Uberlândia, contribuindo para o aprimoramento de métodos preditivos no campo da climatologia aplicada.

Estas modelagens já foram abordadas por outros autores. Mata [9] aplicou o modelo SARIMA para prever o índice de calor na Baixada Cuiabana, utilizando dados de temperatura do ar e umidade relativa do ar. Além disso, estudos como o de Rahman [14] compararam o desempenho do SARIMA e do modelo VAR na previsão de parâmetros climáticos, concluindo que o VAR apresentou maior acurácia ao capturar as interdependências entre temperatura, umidade e cobertura de nuvens. De forma semelhante, Amegashie [1] aplicaram o modelo VAR para analisar a influência de variáveis climáticas na incidência de malária em Gana, demonstrando a capacidade do modelo em identificar relações dinâmicas entre clima e saúde. Redes neurais do tipo LSTM também têm sido aplicadas em estudos recentes para previsão de variáveis ambientais. Este fato pode ser comprovado pelo trabalho de Soares [17], que utilizou LSTM para prever chuva e eventos climáticos severos, e por Bem-Haja [3], que aplicou LSTM para prever níveis de bacias hidrográficas, mostrando que essas redes podem superar abordagens tradicionais em cenários não lineares.

Neste trabalho, os objetivos serão apresentados na Seção 1.2, aspectos teóricos de cada modelo serão abordados no Capítulo 2, métodos e informações sobre os dados serão apresentados no Capítulo 3, resultados no Capítulo 4 e, por fim, considerações finais no Capítulo 5.

1.2 OBJETIVOS

O presente trabalho teve como objetivo geral analisar séries temporais meteorológicas utilizando diferentes abordagens estatísticas e de aprendizado de máquina, a fim de comparar o desempenho dos modelos na previsão de variáveis climáticas. Para isso, serão explorados modelos tradicionais e avançados, incluindo SARIMA, VAR e redes neurais LSTM.

Além do objetivo geral, teve como objetivos específicos:

- Implementar analises das variáveis climáticas precipitação, umidade, temperatura e velocidade do vento, utilizando os modelos **SARIMA**, **VAR** e **LSTM** analisando seus desempenhos preditivos.
- Explorar funções de busca automática do melhor modelo SARIMA, VAR e LSTM.
- Comparar os modelos em termos de precisão preditiva, utilizando métricas como Raiz do Erro Quadrático Médio, Erro Quadrático Médio e Erro Médio Absoluto, em inglês, respectivamente, *Root Mean Squared Error* (RMSE), *Mean Squared Error* (MSE) e *Mean Absolut Error* (MAE).

2. Fundamentação Teórica

2.1 Séries Temporais

De acordo com Morettin e Toloi ([10]), uma **série temporal** pode ser definida como um conjunto de observações coletadas ao longo do tempo, geralmente em intervalos regulares. O principal objetivo da análise de séries temporais é modelar o comportamento dinâmico desses dados, levando em consideração a dependência entre as observações ao longo do tempo.

Uma característica fundamental das séries temporais é o fato de que as observações não são independentes, ou seja, o valor de uma variável em determinado momento pode depender dos valores passados. Por isso, técnicas estatísticas tradicionais, que assumem independência entre as observações, não são adequadas para este tipo de dado.

Segundo os autores, uma série temporal pode apresentar diferentes **componentes**, como:

- **Tendência (ou Trend):** Refere-se a uma mudança de longo prazo no nível médio da série, podendo ser crescente ou decrescente.
- **Sazonalidade:** Refere-se a padrões que se repetem em intervalos regulares de tempo, como ciclos anuais, mensais ou diários.
- **Ciclos:** Representam flutuações que ocorrem em períodos mais longos e menos regulares do que a sazonalidade.
- **Irregularidade (ou Ruído):** Refere-se a variações aleatórias que não podem ser explicadas por nenhum dos componentes anteriores.

Autores como Ferreira ([5]) ressaltam que a identificação e separação desses componentes é essencial para a construção de modelos de previsão mais robustos. Dentre os principais modelos utilizados na literatura para séries univariadas estão os modelos **AR (Auto-Regressivo)**, **MA (Média Móvel)** e **ARMA/ARIMA**, que servem de base para extensões mais complexas como o modelo SARIMA.

Além disso, destaca-se a importância da **análise exploratória**, da **verificação de estacionariedade** e da **avaliação de resíduos** como etapas fundamentais na modelagem de séries temporais ([10]; [5]).

2.2 Modelo SARIMA

Segundo Morettin e Toloi [10] o modelo SARIMA é uma extensão do ARIMA que incorpora componentes sazonais, sendo útil para modelar séries temporais com padrões repetitivos ao longo do tempo. Mesmo após remover a sazonalidade determinística, pode haver autocorrelação residual, tanto em lags de baixa ordem (indicando dependência temporal) quanto em lags sazonais (múltiplos do período s). Nesses casos, o SARIMA é uma abordagem adequada para capturar essa sazonalidade estocástica, permitindo previsões mais precisas.

O modelo SARIMA é representado pela seguinte expressão:

$$\phi(B)\Phi(B^{12})(1-B^{12})^{D}(1-B)^{d}Z_t = \theta(B)\Theta(B^{12})a_t \tag{2.1}$$

Onde:

- Z_t: Série temporal observada;
- B: Operador de defasagem (ou atraso), definido como $BZ_t = Z_{t-1}$;
- $(1-B)^d$: Diferenciação não sazonal de ordem d, usada para tornar a série estacionária. Para $d = 1$:

$$(1-B)Z_t = Z_t - Z_{t-1}$$

- $(1-B^{12})^D$: Diferenciação sazonal de ordem D, que elimina padrões sazonais de período 12. Para $D = 1$:

$$(1-B^{12})Z_t = Z_t - Z_{t-12}$$

- $\phi(B)$: Polinômio autorregressivo (AR) de ordem p:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$$

- $\Phi(B^{12})$: Polinômio autorregressivo sazonal de ordem P:

$$\Phi(B^{12}) = 1 - \Phi_1 B^{12} - \cdots - \Phi_P B^{12P}$$

- $\theta(B)$: Polinômio de médias móveis (MA) de ordem q:

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q$$

- $\Theta(B^{12})$: Polinômio de médias móveis sazonais de ordem Q:

$$\Theta(B^{12}) = 1 - \Theta_1 B^{12} - \cdots - \Theta_Q B^{12Q}$$

- a_t: Ruído branco, um processo com média zero, variância constante e sem autocorrelação.

Conforme apresentado em [8], o modelo SARIMA também pode ser representado pela Eq. 2.2:

$$\begin{aligned}\nabla^D \Delta^d Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + \cdots + \phi_p Z_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} \\ + \Phi_1 Z_{t-1S} + \Phi_2 Z_{t-2S} + \cdots + \Phi_P Z_{t-PS} - \Theta_1 \varepsilon_{t-1S} - \Theta_2 \varepsilon_{t-2S} - \cdots - \Theta_Q \varepsilon_{t-QS}\end{aligned} \tag{2.2}$$

em que:

- ∇^D: Ordem de integração sazonal com D diferenças;
- Δ^d: Ordem de integração não sazonal com d diferenças;
- ϕ_i: Coeficientes autorregressivos com $i = 1, 2, \ldots, p$;
- θ_i: Coeficientes de média móvel com $i = 1, 2, \ldots, q$;
- Φ_i: Coeficientes autorregressivos sazonais com $i = 1, 2, \ldots, P$;
- Θ_i: Coeficientes de média móvel sazonais com $i = 1, 2, \ldots, Q$;
- ε_t: Resíduo.

Essa equação define um modelo SARIMA$(p,d,q)(P,D,Q)[12]$, adequado para séries temporais com sazonalidade anual.

No contexto dos modelos SARIMA, a determinação dos parâmetros $(p,d,q)(P,D,Q)_s$ ocorre em etapas sucessivas. Primeiramente, busca-se identificar o número de diferenciações necessárias para tornar a série estacionária, sendo d a ordem de diferenciação regular e D a ordem de diferenciação sazonal, associada ao período s. Em seguida, a escolha das ordens dos termos autorregressivos (p e P) e de médias móveis (q e Q) é orientada pela análise dos correlogramas, isto é, pela inspeção dos padrões de corte e de decaimento nas funções de autocorrelação e autocorrelação parcial. Uma vez definidas essas ordens, procede-se à estimação dos parâmetros por métodos de máxima verossimilhança, buscando os valores que maximizam a probabilidade de observação da série sob o modelo proposto. Por fim, critérios de informação, como o AIC e o BIC, são empregados para selecionar a especificação mais parcimoniosa, enquanto a análise dos resíduos garante que o ajuste final se comporte como um processo de ruído branco [10].

2.3 Modelo VAR

O modelo VAR (Vector AutoRegressive) foi empregado como abordagem multivariada, permitindo a análise conjunta das variáveis climáticas e suas interações dinâmicas ao longo do tempo. Trata-se de uma generalização dos modelos autorregressivos univariados, em que cada variável é explicada não apenas por seus próprios valores passados, mas também pelos valores defasados das demais variáveis do sistema [2].

Matematicamente, um modelo VAR de ordem p, denotado por VAR(p), pode ser representado pela seguinte expressão:

$$Y_t = c + A_1 Y_{t-1} + A_2 Y_{t-2} + \cdots + A_p Y_{t-p} + \varepsilon_t, \tag{2.3}$$

em que:

- Y_t é um vetor $k \times 1$ contendo as k variáveis endógenas no instante t;
- c é um vetor de constantes;
- A_i são matrizes de coeficientes de dimensão $k \times k$, para $i = 1, \ldots, p$;
- ε_t é um vetor de erros aleatórios, com média zero e matriz de covariância constante.

A estimação dos parâmetros desse modelo é realizada pelo método dos mínimos quadrados ordinários (OLS) conforme apresentado em Barros, [2]. Cada equação do sistema pode ser tratada como uma regressão linear múltipla, na qual a variável dependente é explicada por seus próprios valores defasados e pelos valores defasados das demais variáveis. Para formalizar o procedimento, organiza-se o modelo na forma matricial, em que Y representa a matriz das observações das variáveis dependentes e X a matriz formada pelas defasagens de todas as séries incluídas. O estimador dos coeficientes é obtido pela expressão:

$$\hat{\beta} = (X'X)^{-1}X'Y, \tag{2.4}$$

sendo $\hat{\beta}$ o vetor dos parâmetros estimados. Essa expressão mostra que a estimação é feita a partir da inversão da matriz de correlação das variáveis defasadas $(X'X)$ e da multiplicação pelo produto cruzado entre os regressores e as variáveis dependentes $(X'Y)$. O uso do OLS nesse contexto garante consistência e eficiência dos estimadores quando as séries são estacionárias, além de permitir que cada equação seja estimada separadamente sem perda de eficiência estatística [2].

A expressão apresentada na Eq. 2.3 mostra a formulação geral do modelo VAR(p) em notação matricial compacta. Para o caso em que $k = 4$ variáveis endógenas são consideradas, e uma defasagem p qualquer é utilizada, a Eq. 2.3 pode ser expandida em um sistema de equações individuais, como mostrado nas Eqs. (2.4)–(2.7).

Nessas equações, cada variável $y_{i,t}$ é representada em função de suas próprias defasagens e das defasagens das demais variáveis, ponderadas pelos respectivos coeficientes $a_{i,k}^{(j)}$, além do termo de erro $u_{i,t}$. Essa representação detalhada torna explícita a forma como o modelo VAR capta as inter-relações dinâmicas entre as séries, enquanto a notação matricial fornece uma visão mais compacta e geral do mesmo modelo.

Para o caso em que $k = 4$ e uma defasagem p qualquer, as equações para cada variável são como na Eq. 2.8

$$\begin{aligned} y_{1,t} &= a_{11}^{(1)} y_{1,t-1} + a_{12}^{(1)} y_{2,t-1} + a_{13}^{(1)} y_{3,t-1} + a_{14}^{(1)} y_{4,t-1} \\ &+ a_{11}^{(2)} y_{1,t-2} + a_{12}^{(2)} y_{2,t-2} + a_{13}^{(2)} y_{3,t-2} + a_{14}^{(2)} y_{4,t-2} + \cdots \\ &+ a_{11}^{(p)} y_{1,t-p} + a_{12}^{(p)} y_{2,t-p} + a_{13}^{(p)} y_{3,t-p} + a_{14}^{(p)} y_{4,t-p} + u_{1,t} \end{aligned} \tag{2.5}$$

$$\begin{aligned} y_{2,t} &= a_{21}^{(1)} y_{1,t-1} + a_{22}^{(1)} y_{2,t-1} + a_{23}^{(1)} y_{3,t-1} + a_{24}^{(1)} y_{4,t-1} \\ &+ a_{21}^{(2)} y_{1,t-2} + a_{22}^{(2)} y_{2,t-2} + a_{23}^{(2)} y_{3,t-2} + a_{24}^{(2)} y_{4,t-2} + \cdots \\ &+ a_{21}^{(p)} y_{1,t-p} + a_{22}^{(p)} y_{2,t-p} + a_{23}^{(p)} y_{3,t-p} + a_{24}^{(p)} y_{4,t-p} + u_{2,t} \end{aligned} \tag{2.6}$$

$$\begin{aligned} y_{3,t} &= a_{31}^{(1)} y_{1,t-1} + a_{32}^{(1)} y_{2,t-1} + a_{33}^{(1)} y_{3,t-1} + a_{34}^{(1)} y_{4,t-1} \\ &+ a_{31}^{(2)} y_{1,t-2} + a_{32}^{(2)} y_{2,t-2} + a_{33}^{(2)} y_{3,t-2} + a_{34}^{(2)} y_{4,t-2} + \cdots \\ &+ a_{31}^{(p)} y_{1,t-p} + a_{32}^{(p)} y_{2,t-p} + a_{33}^{(p)} y_{3,t-p} + a_{34}^{(p)} y_{4,t-p} + u_{3,t} \end{aligned} \tag{2.7}$$

$$\begin{aligned} y_{4,t} &= a_{41}^{(1)} y_{1,t-1} + a_{42}^{(1)} y_{2,t-1} + a_{43}^{(1)} y_{3,t-1} + a_{44}^{(1)} y_{4,t-1} \\ &+ a_{41}^{(2)} y_{1,t-2} + a_{42}^{(2)} y_{2,t-2} + a_{43}^{(2)} y_{3,t-2} + a_{44}^{(2)} y_{4,t-2} + \cdots \\ &+ a_{41}^{(p)} y_{1,t-p} + a_{42}^{(p)} y_{2,t-p} + a_{43}^{(p)} y_{3,t-p} + a_{44}^{(p)} y_{4,t-p} + u_{4,t}. \end{aligned} \tag{2.8}$$

Uma etapa fundamental para a modelagem com VAR é a definição da ordem p adequada. Para tanto, utilizaram-se critérios de informação, como o AIC (Akaike Information Criterion) e o BIC (Bayesian Information Criterion), cujas expressões são dadas por:

$$AIC = -2\ln(\hat{L}) + 2k_1, \tag{2.9}$$

$$BIC = -2\ln(\hat{L}) + k_1 \ln(n), \tag{2.10}$$

em que $\hat{L}$ é a função de verossimilhança maximizada do modelo, k_1 é o número de parâmetros estimados e n o número de observações.

Antes da estimação, a estacionariedade das series deve ser verificada e o teste de Dickey-Fuller Aumentado (ADF) é indicado para verificar a estabilidade da série conforme descrito em [2, 10]. Quando constatada não estacionariedade, aplicam-se transformações como diferenciação, de modo a tornar a série estacionária, assegurando que os parâmetros do modelo permaneçam estáveis ao longo do tempo e que as relações entre variáveis não resultem em correlações espúrias.

O teste de causalidade de Granger, conforme discutido por Bueno [16], é amplamente utilizado para analisar a relação temporal entre duas séries, buscando identificar se uma variável pode prever a outra. Nesse sentido, segundo Morettin e Toloi (2018) [10], é necessário verificar se os resíduos do modelo apresentam características desejáveis, como a ausência de autocorrelação, normalidade e heterocedasticidade. Para tanto, testes como o de Portmanteau podem ser aplicados para verificar a autocorrelação dos resíduos, enquanto o teste de Jarque-Bera é utilizado para avaliar a normalidade desses resíduos. Além disso, o teste ARCH é empregado para identificar a presença de heterocedasticidade condicional. Esses diagnósticos são fundamentais para garantir maior robustez às inferências feitas, proporcionando maior confiabilidade nas conclusões sobre a causalidade entre as séries temporais [2, 10]obtidas.

2.4 Modelo LSTM (Long Short-Term Memory)

Uma rede neural é um sistema matemático, inspirado no cérebro humano, que aprende padrões a partir de dados, ajustando automaticamente seus parâmetros ou pesos para melhorar suas previsões ou classificações.

Uma rede neural clássica é a MLP (Multi-Layer Perceptron) composta por múltiplas camadas de perceptrons totalmente conectados, capaz de aprender padrões complexos a partir de dados.

Na Figura 2.1 é ilustrada a arquitetura de uma rede neural MLP com duas variáveis de entrada, quatro e três neurônios, respectivamente, na primeira e segunda camadas ocultas, os vieses e os pesos em cada conexão.

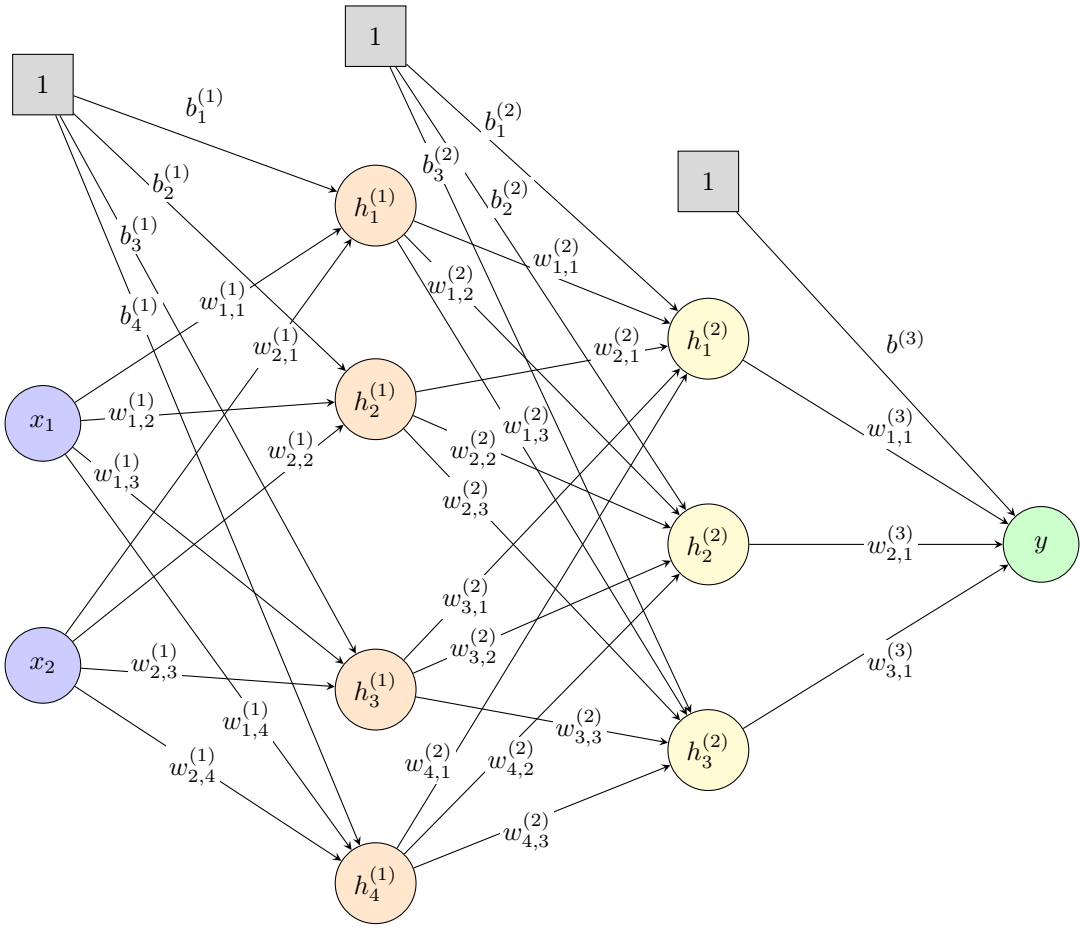

Figura 2.1: Ilustração de uma rede neural com duas variáveis de entrada, duas camadas ocultas, com 4 e 3 neurônios respectivamente e uma variável de saída.

A partir da arquitetura ilustrada na Figura 2.1, a propagação direta (forward propagation) pode ser representada pela operações matriciais indicada na Eq.2.11

$$
\begin{aligned}
\mathcal{Z}^{(1)} &= \mathcal{H}^{(0)} \cdot \mathcal{W}^{(1)} \\
\mathcal{H}^{(1)} &= f^{(1)}\left(\mathcal{Z}^{(1)}\right) \\
\\
\mathcal{Z}^{(2)} &= \mathcal{H}^{(1)} \cdot \mathcal{W}^{(2)} \\
\mathcal{H}^{(2)} &= f^{(2)}\left(\mathcal{Z}^{(2)}\right) \\
\\
\mathcal{Z}^{(3)} &= \mathcal{H}^{(2)} \cdot \mathcal{W}^{(3)} \\
\mathcal{Y} &= f^{(3)}\left(\mathcal{Z}^{(3)}\right)
\end{aligned}
\tag{2.11}
$$

em que os elementos de $\mathcal{Z}$ são resultantes da combinação linear dos pesos com as entradas adicionada do viés e, portanto, é a operação linear da camada. Este é o cálculo realizado imediatamente antes da aplicação da função de ativação e sendo assim, a $\mathcal{Z}$ também é denominada de matriz de pré-ativação.

Nas equações apresentadas em Eq. 2.11, $H^0 = \begin{bmatrix} \mathbf{1} & \boldsymbol{x}_1 & \boldsymbol{x}_2 \end{bmatrix}$ em que $\mathbf{1}$ é um vetor coluna de uns de dimensão $n \times 1$, $\boldsymbol{x_1}$ e $\boldsymbol{x_2}$ são vetores coluna de dimensão $n \times 1$ e as matrizes $\boldsymbol{W^{(1)}}$, $\boldsymbol{W^{(2)}}$ e $\boldsymbol{W^{(3)}}$ são como apresentadas nas Eqs. 2.12, 2.13 e 2.14, respectivamente.

$$\boldsymbol{W^{(1)}} = \begin{array}{r} \\ \text{Viés (1)} \\ \text{Entrada 1} \\ \text{Entrada 2} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Neurônio 1} & \text{Neurônio 2} & \text{Neurônio 3} & \text{Neurônio 4} \end{array} \\ \begin{bmatrix} b_1^{(1)} & b_2^{(1)} & b_3^{(1)} & b_4^{(1)} \\ w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} & w_{14}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} & w_{24}^{(1)}. \end{bmatrix} \end{array} \tag{2.12}$$

$$\boldsymbol{W^{(2)}} = \begin{array}{r} \\ \text{Viés (1)} \\ \text{Entrada 1} \\ \text{Entrada 2} \\ \text{Entrada 3} \\ \text{Entrada 4} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Neurônio 1} & \text{Neurônio 2} & \text{Neurônio 3} \end{array} \\ \begin{bmatrix} b_1^{(2)} & b_2^{(2)} & b_3^{(2)} \\ w_{11}^{(2)} & w_{12}^{(2)} & w_{13}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & w_{23}^{(2)} \\ w_{31}^{(2)} & w_{32}^{(2)} & w_{33}^{(2)} \\ w_{41}^{(2)} & w_{42}^{(2)} & w_{43}^{(2)}. \end{bmatrix} \end{array} \tag{2.13}$$

$$\boldsymbol{W^{(3)}} = \begin{array}{r} \\ \text{Viés (1)} \\ \text{Entrada 1} \\ \text{Entrada 2} \\ \text{Entrada 3} \end{array} \begin{array}{c} \begin{array}{c} \text{Neurônio 1} \end{array} \\ \begin{bmatrix} b^{(3)} \\ w_{11}^{(3)} \\ w_{21}^{(3)} \\ w_{31}^{(3)}. \end{bmatrix} \end{array} \tag{2.14}$$

A propagação na rede desde a entrada até a saída é feita, realizando as operaçãoes, de forma sequencial, definidas pelas equações apresentadas na Eq. 2.11 e, $\boldsymbol{f^{(1)}}$, $\boldsymbol{f^{(2)}}$ e $\boldsymbol{f^{(3)}}$ são funções de ativação.

Várias funções de ativação são definidas na literatura [17] e a decisão sobre qual utilizar, em geral, é dependente do tipo de problema abordado ou do tipo da variável a ser modelada.

As redes neurais recorrentes (RNNs) introduzem conexões recorrentes que permitem o armazenamento de informações de estados anteriores, tornando-as especialmente indicadas para lidar com dados sequenciais ou temporais, como séries temporais, linguagem natural e sinais de áudio. Assim, enquanto a MLP aprende padrões estáticos, a RNN é capaz de capturar dependências ao longo do tempo, expandindo o alcance de aplicação das redes neurais.

No entanto, as RNNs sofrem com o problema do gradiente desaparecendo ou explodindo, o que dificulta o aprendizado de dependências de longo prazo em séries temporais ou sequências muito longas. A rede neural LSTM (Long Short-Term Memory) pode sanar este problema.

O modelo LSTM é uma variante avançada das RNNs, projetada para superar o problema do desvanecimento e explosão de gradientes, o que permite capturar dependências de longo prazo em séries temporais. De acordo com Haykin [6], as unidades LSTM incorporam mecanismos chamados portas — de entrada, esquecimento e saída — que regulam seletivamente o fluxo de informações, permitindo à rede decidir quais dados devem ser retidos ou descartados.

O modelo LSTM (Long Short-Term Memory) foi utilizado como técnica de aprendizado de máquina para previsão das variáveis climáticas. Essa rede neural recorrente é capaz de capturar dependências de longo prazo em dados sequenciais, sendo adequada para lidar com padrões complexos e não lineares presentes em séries temporais. De acordo com Haykin ([5]), as redes neurais são especialmente eficazes em tarefas que envolvem reconhecimento de padrões e previsão, justamente pela capacidade de adaptação de seus parâmetros durante o

treinamento.

Matematicamente, cada unidade LSTM segue a dinâmica apresentada nas Eqs. 2.15:

$$\begin{aligned} f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \\ C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t \cdot \tanh(C_t) \end{aligned} \tag{2.15}$$

em que:

- f_t é a porta de esquecimento, no instante t, que decide quais informações da célula anterior serão descartadas;
- i_t é a porta de entrada, que controla quais novas informações serão armazenadas no estado da célula;
- $\tilde{C}_t$ é o vetor candidato a estado, contendo novas informações que podem ser adicionadas à memória;
- C_t é o estado da célula, que armazena as informações relevantes ao longo do tempo;
- o_t é a porta de saída, que decide quais informações do estado da célula serão usadas para calcular a saída;
- h_t é o estado oculto, ou saída da unidade LSTM no instante t;
- x_t é a entrada no instante t;
- h_{t-1} é o estado oculto do instante anterior;
- W_f, W_i, W_C, W_o são as matrizes de pesos associadas a cada porta;
- a notação $W_f[h_{t-1}, x_t]$ representa a multiplicação da matriz de pesos W_f pelo vetor resultante da concatenação de h_{t-1} e x_t;
- b_f, b_i, b_C, b_o são os vetores de viés correspondentes;
- σ representa a função logística sigmoide, que mapeia valores para o intervalo $(0, 1)$;
- tanh é a função tangente hiperbólica, que mapeia valores para o intervalo $(-1, 1)$.

Em um estudo, Bem-Haja [3] propôs a utilização de redes neurais artificiais do tipo LSTM na previsão do nível de bacias hidrográficas, destacando que "os modelos LSTM mostraram-se eficazes em capturar padrões complexos de séries temporais hidrológicas, possibilitando previsões de curto prazo com desempenho satisfatório". De fato, a estrutura recorrente dessas redes permite lidar com a natureza sequencial e não linear dos dados climáticos. Essa combinação de boa retenção de informações e controle de fluxo torna o LSTM uma escolha robusta para previsão de séries climáticas complexas.

A arquitetura de uma rede LSTM é caracterizada pelo número de camadas e pela quantidade de neurônios em cada uma delas, elementos que definem a capacidade de representação do modelo. De acordo com Haykin ([6]), a escolha da quantidade de unidades em uma rede neural influencia diretamente sua habilidade de capturar padrões complexos, sendo necessário encontrar um equilíbrio entre complexidade e capacidade de generalização. Nesse sentido, Bem-Haja ([3]) destaca que a definição adequada da arquitetura em modelos LSTM é fundamental para garantir previsões satisfatórias em séries temporais, pois redes muito simples podem não capturar dependências relevantes, enquanto arquiteturas excessivamente complexas podem resultar em sobreajuste (*overfitting*) quando aplicadas a bases de dados limitadas.

O processo de aprendizado em uma rede neural artificial, como nas MLPs, tem como principal objetivo a minimização de uma função de erro, geralmente associada ao erro quadrático médio (MSE), que corresponde à soma dos quadrados das diferenças entre os valores previstos e os observados, conforme descrito por [6]. Esse aprendizado ocorre de forma iterativa e envolve três etapas fundamentais. Na primeira, chamada de propagação direta (*forward propagation*), os dados de entrada percorrem a rede camada por camada até a obtenção da saída. Em seguida, calcula-se o erro comparando a saída da rede com o valor real, por meio da função de perda. A terceira etapa corresponde ao ajuste dos pesos sinápticos, realizado através do algoritmo de retropropagação do erro (*backpropagation*), no qual os gradientes da função de perda em relação aos pesos são calculados e usados para atualizar os parâmetros da rede segundo uma regra de descida de gradiente. Esse ciclo de propagação direta, cálculo do erro e retropropagação se repete diversas vezes ao longo do treinamento, em múltiplas épocas, até que o erro seja minimizado ou atinja um nível satisfatório, como enfatiza [6].

As redes LSTM compartilham esse mesmo princípio de aprendizado baseado na minimização do erro, mas apresentam uma diferença essencial em relação às MLP tradicionais: a capacidade de lidar com dados sequenciais e dependências de longo prazo. Para isso, incorporam mecanismos internos de memória, organizados em portas de entrada, esquecimento e saída, que regulam de forma seletiva o fluxo de informações relevantes ao longo do tempo, segundo [6]. Essa característica as torna particularmente eficazes na modelagem de séries temporais, uma vez que permitem capturar padrões complexos e não lineares que se estendem por diversos períodos. [3] destaca que os modelos LSTM mostraram-se eficazes em capturar padrões complexos de séries hidrológicas, permitindo previsões de curto prazo com bom desempenho. De forma complementar, [4] reforça que as LSTM e suas variantes, têm se mostrado eficazes na modelagem de sistemas hidrológicos e climáticos não lineares, evidenciando a adequação dessa arquitetura para aplicações preditivas em séries temporais.

2.5 Métricas de Validação

A avaliação da qualidade preditiva dos modelos de séries temporais constitui uma etapa essencial no processo de modelagem, uma vez que possibilita verificar a proximidade entre os valores estimados e aqueles efetivamente observados. Para este trabalho, buscou-se respaldo na literatura especializada, de onde foram selecionadas as métricas de validação mais frequentemente empregadas. Assim, optou-se pela utilização do *Erro Médio Absoluto* (MAE) [12], da *Raiz do Erro Quadrático Médio* (RMSE) [15] e do *Erro Quadrático Médio* (MSE) [15].

As expressões para a MSE, bem como para a RMSE e MAE, são apresentadas, respectivamente, nas Eqs. 2.16, 2.17 e 2.18.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2 \tag{2.16}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{2.17}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{2.18}$$

em que:

- y_t: Valor observado no instante t;
- $\hat{y}_t$: Valor estimado ou previsto pelo modelo no instante t;
- n: Número total de observações da série temporal;
- $(y_t - \hat{y}_t)$: Erro de previsão no instante t;
- $(y_t - \hat{y}_t)^2$: Erro quadrático no instante t (utilizado no MSE e RMSE);

- $|y_t - \hat{y}_t|$: Erro absoluto no instante t (utilizado no MAE).

3. METODOLOGIA

3.1 DADOS

Os dados foram obtidos da base do Instituto Nacional de Meteorologia (INMET) [7], abrangendo variáveis medidas por hora na cidade de Uberlândia, no período de 2014 a 2024, período em que os dados apresentam melhor qualidade no sentido de não possuir dados faltantes.

Foram identificados valores inconsistentes, isto é, valores negativos nas séries de precipitação e umidade. No caso da precipitação, valores negativos foram substituídos por zero, uma vez que não é esperado que essa variável apresente registros abaixo deste limite. Para a série de umidade relativa do ar, os valores negativos foram substituídos pela média mensal correspondente, de modo a preservar a coerência física da variável.

Após tais correções, para as quatro variáveis, os dados diários foram agregados em médias mensais, com o objetivo de reduzir a presença de zeros excessivos e facilitar a modelagem. Os dados diários são oriundos de dados coletados por hora, em que a precipitação diária é o acumulo do dia e as variáveis temperatura, umidade e velocidade do vento é a média de 24 valores.

Não foram realizadas transformações adicionais nos dados. A divisão da base foi feita em conjuntos de treino e teste, respeitando uma proporção usualmente adotada na literatura para séries temporais, com o objetivo de avaliar a capacidade de generalização dos modelos. Os dados utilizados para treinamento compreende o período de janeiro de 2014 a setembro de 2024 e de outubro a dezembro de 2024 foram utilizados para validação dos modelos.

As análises e modelagens foram desenvolvidas utilizando a linguagem de programação Python, que oferece diversas bibliotecas para manipulação de dados, modelagem estatística e aprendizado de máquina. O ambiente escolhido para o desenvolvimento foi o Google Colab, uma plataforma colaborativa baseada na nuvem, que permite a execução de códigos Python [13] de forma interativa e com acesso facilitado a recursos computacionais.

A implementação dos modelos SARIMA, VAR e LSTM foi realizada por meio dos códigos apresentados nos Apêndices A.1, B.1 e C.1.

3.2 MODELO SARIMA

O modelo SARIMA (Seasonal Autoregressive Integrated Moving Average) foi escolhido como uma das abordagens estatísticas para previsão das variáveis climáticas consideradas neste estudo. Conforme discutido em [10], trata-se de uma extensão do modelo ARIMA que incorpora componentes sazonais, sendo particularmente útil para séries temporais que apresentam padrões de repetição ao longo do tempo.

Durante a etapa de implementação deste modelo, verificou-se que algumas bibliotecas necessitaram ser instaladas em versões específicas, dado que os pacotes utilizados eram compatíveis apenas com estas versões. Em especial, a biblioteca *NumPy* foi fixada na versão **1.25.2** e o *Pandas* na versão **2.3.1**, garantindo a consistência entre os módulos estatísticos empregados no estudo.

Inicialmente, aplicou-se o teste de Dickey-Fuller Aumentado (ADF) para verificar a estacionariedade das séries originais. Quando necessário, procedeu-se à diferenciação das séries, tanto em nível quanto sazonal, de modo a atender ao pressuposto fundamental de estacionariedade exigido para a modelagem [10]. Para a variável precipitação, por exemplo, não foi necessária diferenciação, enquanto para temperatura, umidade e velocidade

do vento realizou-se uma diferenciação em nível.

Em seguida, a seleção dos parâmetros do modelo SARIMA foi realizada por meio de uma função de busca automática [19], disponibilizada nas biblioteca do ambiente Python [13], que permite identificar de forma sistemática as ordens dos polinômios autorregressivos (AR), de médias móveis (MA) e seus equivalentes sazonais. Essa escolha é baseada em critérios de informação como o AIC (Akaike Information Criterion) e o BIC (Bayesian Information Criterion), possibilitando a determinação do modelo mais parcimonioso, isto é, aquele que equilibra qualidade de ajuste e complexidade, conforme recomendação da literatura [2].

Após a estimação, avaliou-se a adequação dos modelos por meio da análise dos resíduos, que foram submetidos aos testes de Ljung-Box (para detecção de autocorrelação), Jarque-Bera (para normalidade) e ARCH (para heterocedasticidade). Esses testes forneceram uma avaliação abrangente da validade dos pressupostos do modelo e da consistência das estimativas obtidas. Embora em alguns casos tenha havido violações parciais dos pressupostos de normalidade e homocedasticidade (Tabela 4.3), os modelos foram mantidos com o intuito de permitir a comparação com os demais métodos.

Por fim, as previsões foram realizadas para o período de outubro a dezembro de 2024, abrangendo as quatro variáveis de interesse: precipitação, temperatura, umidade e velocidade do vento. O desempenho dos modelos foi avaliado por meio das métricas RMSE e MAE, já apresentadas na Seção 2, e os resultados estão detalhados na Seção 4.

3.3 Modelo VAR

Para o modelo VAR, a escolha da ordem ótima foi realizada de forma automática, similar ao que foi feito para o modelo ARIMA, utilizando o **Critério de Informação Akaike (AIC)**, como apresentado na Seção 2.3. A estimativa dos parâmetros foi realizada de acordo com o método dos **mínimos quadrados ordinários (OLS)**, descrito nessa mesma seção, em que cada equação do modelo VAR é tratada como uma regressão linear múltipla, com as variáveis defasadas como preditores para cada variável dependente. Esse procedimento garante estimadores consistentes e eficientes quando as séries são estacionárias.

A busca automática pela melhor ordem do modelo foi realizada utilizando a função de busca automática [18] disponível no Python [13]. Este procedimento facilita a seleção da ordem mais adequada para o modelo, sem a necessidade de uma análise manual extensiva das defasagens. A utilização dessa função automatizada assegura a obtenção do modelo VAR com a melhor configuração, garantindo eficiência e precisão na modelagem de séries temporais multivariadas.

A ordem ótima do modelo foi definida a partir do critério de informação AIC, assegurando a escolha de uma estrutura parcimoniosa e ajustada aos dados. Antes da estimação, foi verificada a estacionariedade das séries por meio do teste ADF, com aplicação de diferenciação quando necessário. Após o ajuste, a adequação do modelo foi avaliada por meio de testes estatísticos aplicados aos resíduos, incluindo o teste de Portmanteau (autocorrelação), o teste ARCH (heterocedasticidade condicional) e o teste de Jarque-Bera (normalidade), garantindo maior robustez às inferências.

Além disso, para investigar a relação de causalidade entre as variáveis do modelo VAR, foi aplicado teste de causalidade de Granger. Esse teste busca determinar se uma variável pode ajudar a prever outra, considerando que uma variável "causa"a outra no sentido de que suas defasagens fornecem informação relevante para prever seu comportamento futuro. A causalidade de Granger é um método amplamente utilizado para identificar relações dinâmicas entre séries temporais, sendo essencial para a análise de dependência temporal entre as variáveis em estudo.

Por fim, as previsões foram realizadas para o período de outubro a dezembro de 2024, abrangendo as quatro variáveis de interesse. O desempenho do modelo foi avaliado por meio das métricas RMSE e MAE, já apresentadas na Seção 2, e os resultados estão detalhados na Seção 4.

3.4 Modelo LSTM (Long Short-Term Memory)

A arquitetura do modelo LSTM foi definida por meio de uma função de busca automática, responsável por selecionar a configuração mais adequada quanto ao número de camadas e de neurônios da rede. Essa abordagem, disponível em bibliotecas do ambiente Python, permite identificar automaticamente a melhor estrutura de rede para cada variável analisada, assegurando maior eficiência no processo de modelagem.

O número de camadas e neurônios variou de acordo com as características específicas de cada série, garantindo equilíbrio entre capacidade preditiva e eficiência computacional. O treinamento da rede foi realizado com a função de perda baseada no erro quadrático médio (MSE) e otimização por meio do algoritmo Adam, em múltiplas épocas até a convergência.

As previsões foram realizadas para o período de outubro a dezembro de 2024. Para avaliar o desempenho do modelo, utilizaram-se as métricas RMSE e MAE, já apresentadas na Seção 2, permitindo a comparação direta entre o LSTM e os modelos estatísticos tradicionais.

Estudos recentes reforçam a eficácia dessa abordagem. Fernandes ([4]) destaca que as unidades LSTM e suas variantes, como as GRU, têm se mostrado altamente eficazes na modelagem de sistemas hidrológicos não lineares e variantes no tempo. De forma semelhante, Bem-Haja ([3]) evidencia que modelos LSTM são capazes de capturar padrões complexos em séries temporais hidrológicas, possibilitando previsões de curto prazo com desempenho satisfatório. Tais constatações corroboram a adequação da aplicação de redes LSTM em contextos climáticos e ambientais, em razão de sua estrutura de memória e mecanismos de portas que potencializam a captura de padrões de longo alcance.

Durante o processo de treinamento, os pesos sinápticos da rede são ajustados de forma iterativa com o objetivo de minimizar o erro entre os valores previstos e observados. Esse procedimento ocorre em três etapas principais: propagação direta, retropropagação do erro e atualização dos pesos. Na propagação direta, os dados percorrem a rede camada por camada, sendo transformados por meio de combinações lineares dos pesos e vieses, seguidas de funções de ativação. Em seguida, o erro é calculado pela função de perda, neste caso o MSE. A retropropagação utiliza derivadas parciais para estimar o gradiente da função de perda em relação a cada peso, indicando o quanto cada parâmetro contribuiu para o erro final. Como explica Haykin ([6]), esse mecanismo é essencial para orientar os ajustes dos parâmetros internos. Por fim, a atualização dos pesos é realizada por algoritmos de otimização, como o gradiente descendente e suas variantes, em pequenas quantidades controladas pela taxa de aprendizado.

Esse processo é repetido ao longo de várias iterações ou épocas, permitindo que a rede aprimore progressivamente sua capacidade preditiva. Como ressalta Bem-Haja ([3]), a definição adequada da arquitetura, aliada ao treinamento iterativo, possibilita às LSTM representar de forma eficiente dependências complexas em séries temporais. No presente trabalho, a arquitetura da rede foi definida por uma função de busca automatica [11] implementada no python [13], garantindo que cada variável fosse treinada em uma estrutura ajustada às suas características específicas.

3.5 Métricas de Validação

As métricas utilizadas para avaliar o desempenho dos modelos foram o *Erro Quadrático Médio* (MSE), a *Raiz do Erro Quadrático Médio* (RMSE) e o *Erro Médio Absoluto* (MAE), já apresentadas na Seção 2. Essas medidas permitem quantificar a acurácia das previsões, sendo o RMSE mais sensível a erros de maior magnitude, enquanto o MAE expressa a magnitude média dos erros absolutos, e o MSE fornece uma medida direta da variabilidade dos erros quadráticos. Dessa forma, as três métricas complementam-se na avaliação do ajuste e da qualidade preditiva dos modelos.

Assim, todos os modelos ajustados (SARIMA, VAR e LSTM) foram avaliados utilizando as mesmas métricas, o que possibilitou a comparação direta de seus desempenhos preditivos. Essa abordagem assegura que as diferenças nos resultados estejam associadas às características próprias de cada técnica, e não a critérios de

avaliação distintos.

4. RESULTADOS

4.1 ANÁLISE DESCRITIVA DOS DADOS

O teste de Dickey-Fuller Aumentado (ADF) foi aplicado para avaliar a estacionariedade das séries. Como apenas a série de precipitação apresentou estacionariedade, aplicou-se diferenciação nas demais séries, enquanto a série de precipitação foi mantida em sua forma original. Os resultados antes e após a diferenciação são apresentados na Tabela 4.1.

Tabela 4.1: P-valores do teste ADF antes e após a diferenciação

Variável	**Antes da diferenciação**	**Após a diferenciação**
Precipitação	$2,99 \times 10^{-3}$	$2,99 \times 10^{-3}$
Temperatura	0,88	$4,59 \times 10^{-21}$
Umidade	0,49	$2,07 \times 10^{-20}$
Vento	0,98	$5,44 \times 10^{-12}$

Os gráficos das séries temporais, após as transformações descritas na seção de metodologia, são apresentados nas Figuras 4.1 a 4.4. As adequações aplicadas foram: (i) na série de precipitação, substituição de valores negativos por zero, por não haver suporte físico para medições abaixo desse limite; (ii) na série de umidade relativa do ar, substituição de valores menores ou iguais a zero pela média mensal correspondente, preservando a coerência física da variável; e (iii) agregação dos registros diários em médias mensais, com o objetivo de reduzir a presença de zeros excessivos e favorecer a modelagem.

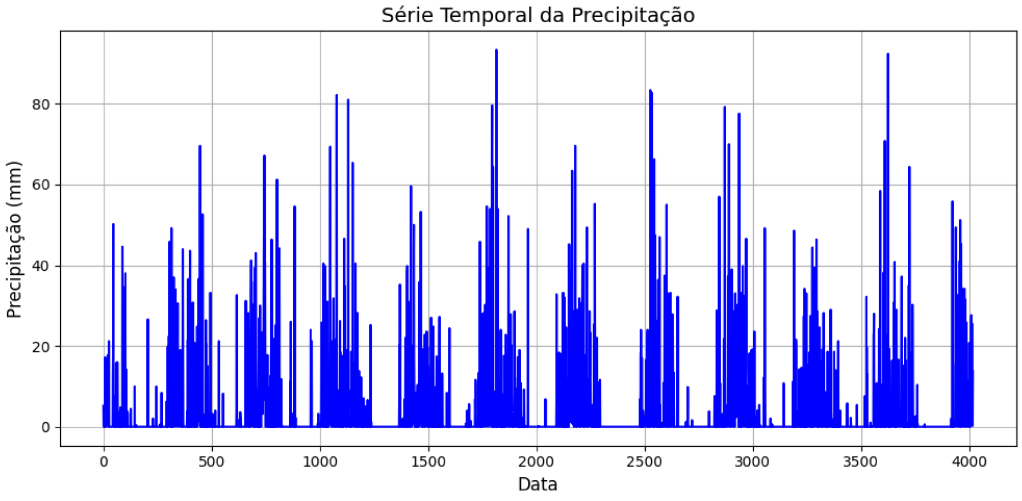

Figura 4.1: Série temporal da variável precipitação.

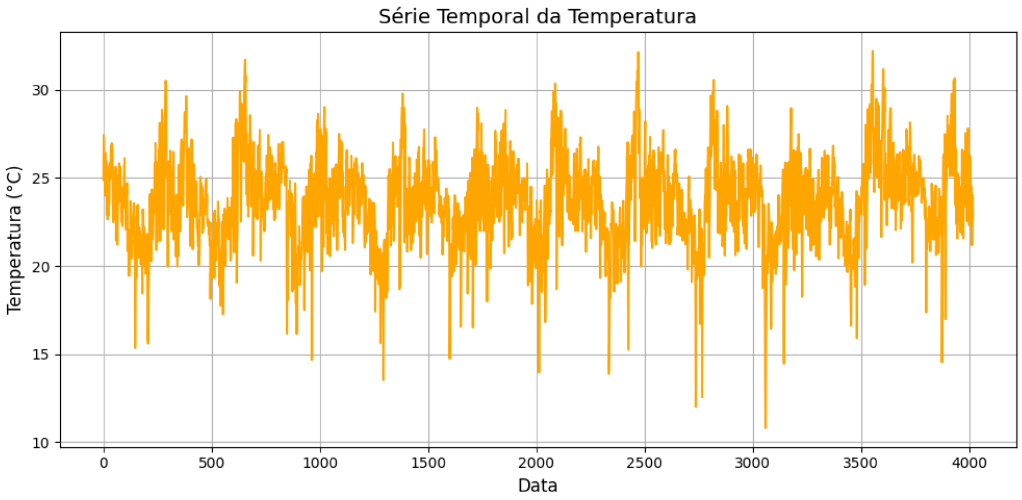

Figura 4.2: Série temporal da variável temperatura.

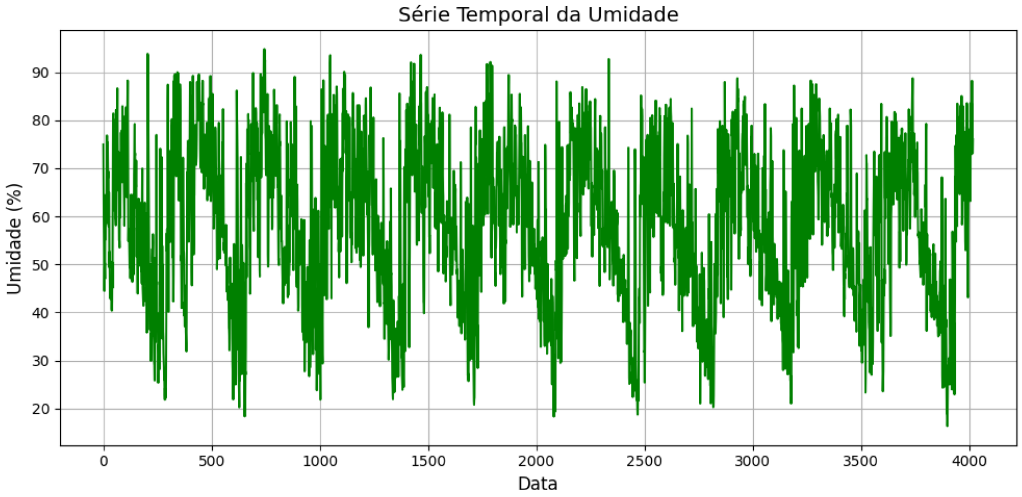

Figura 4.3: Série temporal da variável umidade.

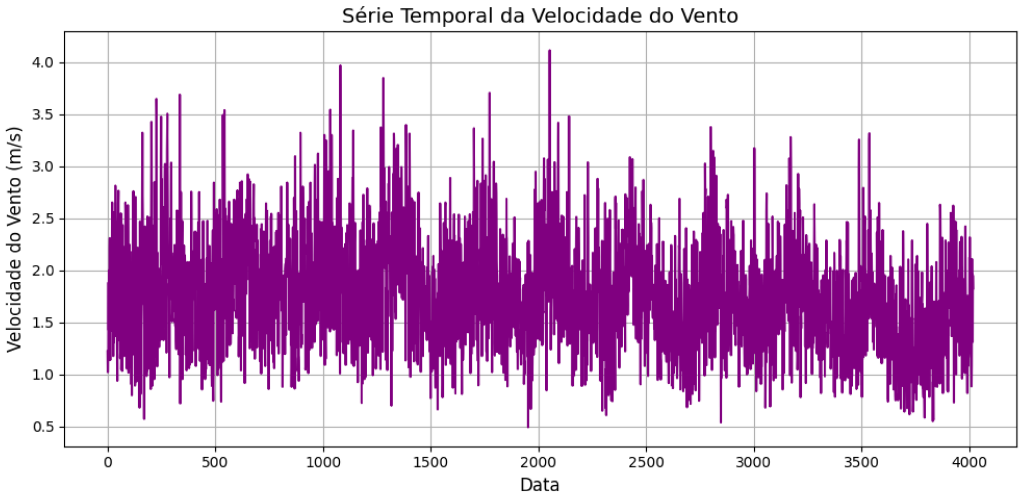

Figura 4.4: Série temporal da variável velocidade do vento.

4.2 Modelo SARIMA

Para a modelagem, utilizou-se a função de busca automática, que selecionou os modelos mais adequados para cada série de acordo com os critérios de informação. Os resultados foram:

- **Precipitação**: ARIMA(1, 0, 0)(1, 0, 1, 12)[12]

$$Y_t = 2{,}648 \times 10^{-5} + 0{,}3275\, Y_{t-1} + 0{,}9886\, e_{t-1} + 0{,}9885\, Y_{t-12} + 0{,}9099\, e_{t-12} - 0{,}1312\, e_{t-24}$$

- **Temperatura**: ARIMA(1, 0, 1)(1, 0, 2, 12)[12]

$$Y_t = 0{,}0405 - 0{,}1598\, Y_{t-1} + 0{,}9910\, Y_{t-12} + 0{,}7709\, e_{t-12}$$

- **Vento**: ARIMA(2, 0, 1)(1, 0, 1, 12)[12]

$$Y_t = 0{,}1802\, Y_{t-1} + 0{,}1714\, Y_{t-2} + 0{,}9770\, Y_{t-12} + 0{,}9247\, e_{t-1} + 0{,}8080\, e_{t-12}$$

- **Umidade**: ARIMA(5, 0, 1)(1, 0, 1, 12)[12]

$$Y_t = 0{,}2905\, Y_{t-1} + 0{,}0274\, Y_{t-2} - 0{,}0939\, Y_{t-3} - 0{,}0441\, Y_{t-4} - 0{,}1418\, Y_{t-5} + 0{,}9886\, e_{t-1} + 0{,}9641\, Y_{t-12} + 0{,}6466\, e_{t-12}$$

Os critérios de avaliação AIC e BIC para cada modelo ajustado foram:

Tabela 4.2: Valores de AIC e BIC para cada variável analisada

Variável	**AIC**	**BIC**
Precipitação	600,50	614,76
Temperatura	374,61	394,57
Vento	-103,05	-85,93
Umidade	844,46	870,12

A verificação dos pressupostos do modelo VAR foi realizada por meio de três testes distintos: Ljung-Box (autocorrelação dos resíduos), Jarque-Bera (normalidade dos resíduos) e ARCH (heterocedasticidade). Cada um desses testes avalia um pressuposto específico do modelo, possibilitando uma análise mais abrangente de sua adequação. Os valores de *p-valor* obtidos encontram-se na Tabela 4.3.

Tabela 4.3: Resultados dos testes de diagnóstico dos resíduos dos modelos.

Variável	**Ljung-Box (lag=12)**	**Jarque-Bera**	**ARCH**
Umidade	1,00	0,01	0,02
Precipitação	0,05	0,00	0,57
Temperatura	0,91	0,65	0,21
Vento	0,85	0,33	0,01

No teste de Ljung-Box, os valores de p obtidos foram superiores a 0,05 em todas as variáveis, indicando ausência de autocorrelação serial significativa nos resíduos e, portanto, atendendo a esse pressuposto. Já no teste de Jarque-Bera, os resultados mostraram que, em alguns casos, o pressuposto de normalidade não foi satisfeito, uma vez que determinados p-valores ficaram abaixo de 0,05, sugerindo rejeição da hipótese nula de normalidade dos resíduos. Por fim, o teste ARCH revelou indícios de heterocedasticidade para certas variáveis, dado que alguns p-valores também se mostraram inferiores a 0,05, caracterizando violação do pressuposto de homocedasticidade.

De forma geral, conclui-se que os resíduos do modelo VAR não apresentam autocorrelação serial, porém há violações nos pressupostos de normalidade e homocedasticidade para algumas variáveis. Esses resultados evidenciam que, embora o modelo capture adequadamente a dinâmica temporal das séries, certas limitações permanecem na distribuição e na variabilidade dos resíduos.

As Tabelas 4.4 a 4.7 apresentam a comparação entre os valores observados e os valores previstos pelo modelo, bem como os respectivos intervalos de confiança de 95% para cada variável analisada (*precipitação, temperatura, vento e umidade*). Além disso, no rodapé de cada tabela, são reportados os valores das métricas **RMSE** (Root Mean Squared Error), **MSE** (Mean Squared Error) e **MAE** (Mean Absolute Error), que permitem quantificar a acurácia das previsões do modelo. Ressalta-se que a precisão das estimativas para precipitação e umidade é baixa. No caso da precipitação (Tabela 4.4) inclui, inclusive valores negativos e no caso da umidade (Tabela 4.7), valores acima de 100%. Essas estimativas não são coerentes com os valores que estas variáveis podem, de fato, assumirem.

Tabela 4.4: Valores observados, previstos e intervalos de confiança (95%) para precipitação.

Mês	Observado	Previsto	IC 95%
10-2024	9,05	4,07	[-0,26 ; 8,40]
11-2024	12,34	7,24	[2,86 ; 11,63]
12-2024	6,85	9,53	[5,14 ; 13,92]
	RMSE: 2,36	**MSE:** 5,58	**MAE:** 2,31

Tabela 4.5: Valores observados, previstos e intervalos de confiança (95%) para temperatura.

Mês	Observado	Previsto	IC 95%
10-2024	25,51	26,06	[24,28 ; 27,83]
11-2024	24,03	24,61	[20,71 ; 28,51]
12-2024	24,29	24,32	[18,26 ; 30,38]
	RMSE: 0,67	**MSE:** 0,44	**MAE:** 0,59

Tabela 4.6: Valores observados, previstos e intervalos de confiança (95%) para vento.

Mês	Observado	Previsto	IC 95%
10-2024	1,65	1,65	[1,36 ; 1,93]
11-2024	1,71	1,53	[0,88 ; 2,17]
12-2024	1,43	1,47	[0,46 ; 2,47]
	RMSE: 0,19	**MSE:** 0,03	**MAE:** 0,18

Tabela 4.7: Valores observados, previstos e intervalos de confiança (95%) para umidade.

Mês	Observado	Previsto	IC 95%
10-2024	59,07	52,99	[41,78 ; 64,21]
11-2024	70,50	59,35	[34,46 ; 84,23]
12-2024	71,34	67,63	[28,93 ; 106,33]
	RMSE: 2,04	**MSE:** 4,15	**MAE:** 1,57

Nas Figuras de 4.5 a 4.8 são apresentados os gráficos das previsões obtidas para cada variável, acompanhados de seus respectivos intervalos de confiança, de modo a ilustrar o desempenho preditivo do modelo.

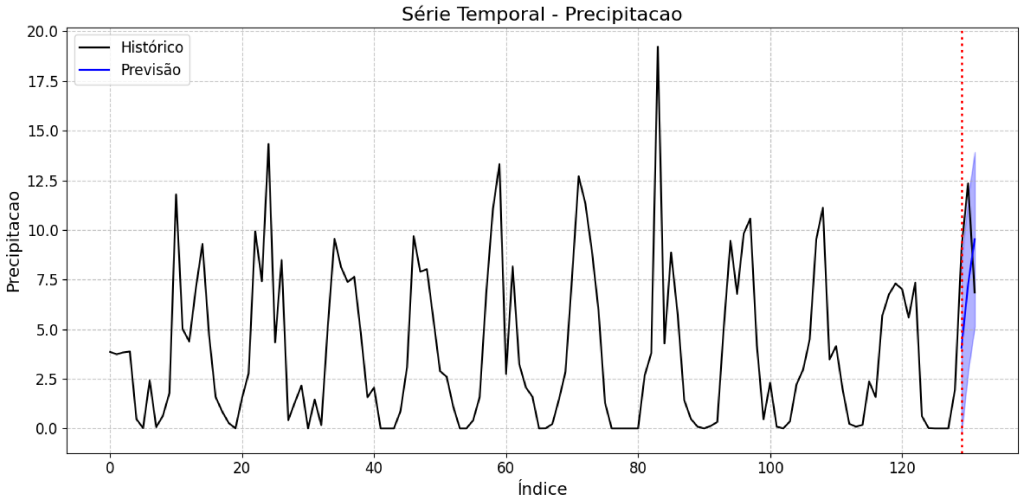

Figura 4.5: Previsão da série temporal de precipitação utilizando o modelo SARIMA.

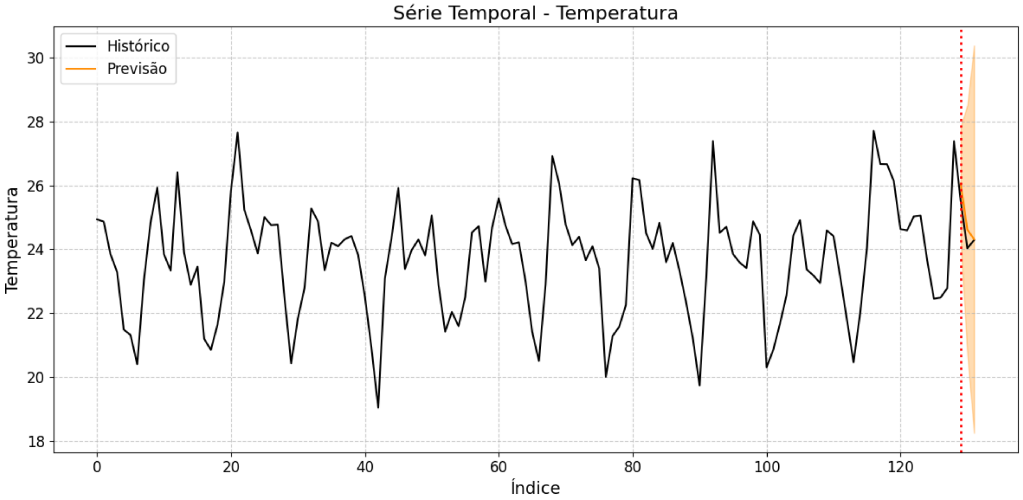

Figura 4.6: Previsão da série temporal de temperatura utilizando o modelo SARIMA.

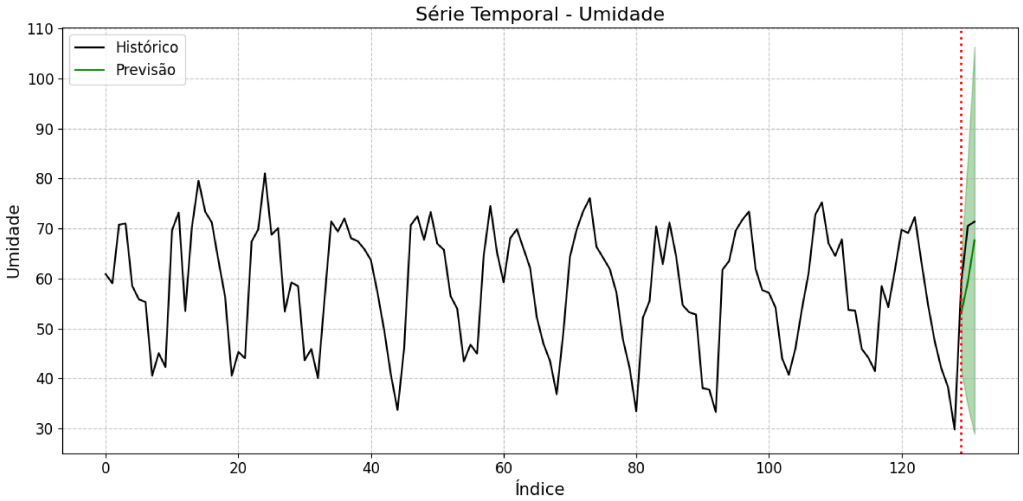

Figura 4.7: Previsão da série temporal de umidade utilizando o modelo SARIMA.

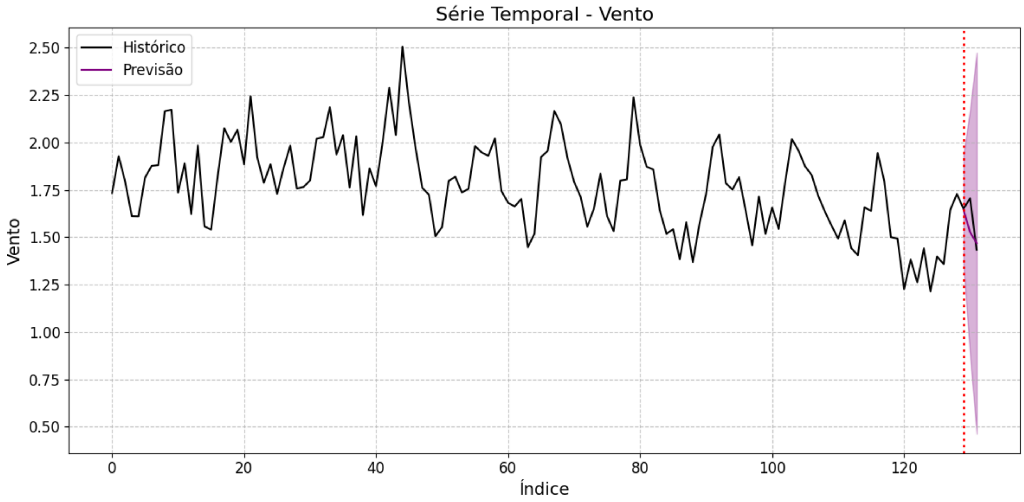

Figura 4.8: Previsão da série temporal de vento utilizando o modelo SARIMA.

4.3 Modelo VAR

Para o modelo VAR, a escolha da ordem ótima foi realizada de forma automática, utilizando o Critério de Informação Akaike (AIC), conforme apresentado na Seção 3.3. Esse critério foi adotado para selecionar a ordem do modelo, com o objetivo de garantir a melhor configuração para as variáveis do modelo. A busca automática pela ordem ideal foi realizada com a função de busca automática no Python, que ajusta modelos VAR e seleciona a ordem com base no critério AIC. A função de busca automática selecionou o número máximo de defasagens arbitrado, ou seja, 20. Como resultado dessa busca, foi escolhida a ordem 20, pois ela apresentou o menor valor de AIC entre as opções testadas.

O modelo VAR estimado resultou em um conjunto de equações que descrevem a relação dinâmica entre as variáveis analisadas. Cada equação representa uma variável dependente em função das defasagens de todas as séries incluídas. As tabelas completas com os coeficientes estimados encontram-se no Apêndice, organizadas em blocos para cada variável dependente.

Para avaliar a adequação do modelo VAR, foram aplicados diferentes testes estatísticos. Primeiramente, utilizou-se o teste de causalidade de Granger para verificar relações de precedência temporal entre as variáveis. Além disso, foram aplicados testes de diagnóstico sobre os resíduos do modelo, incluindo: (i) o teste de Portmanteau, para identificar autocorrelação; (ii) o teste ARCH, para verificar heterocedasticidade condicional; e (iii) o teste de normalidade de Jarque-Bera. A Tabela 4.8 resume os resultados obtidos.

Tabela 4.8: Resultados (valores-p) do teste de causalidade de Granger e testes de diagnóstico aplicados ao modelo VAR

Variável	**Granger**	**Portmanteau**	**ARCH**	**Jarque-Bera**
Precipitação	0,00	0,93	0,05	0,50
Temperatura	0,32	0,57	0,40	0,84
Umidade	0,01	0,03	0,76	0,20
Vento	0,12	0,91	0,79	0,70

A análise dos resultados mostra que o teste de Granger indica causalidade estatisticamente significativa apenas para as variáveis Precipitação (p-valor = 0.000) e Umidade (p-valor = 0.009), enquanto as demais não apresentaram evidências de causalidade. No contexto de um modelo VAR com objetivo preditivo, o teste de causalidade de Granger é uma ferramenta complementar que auxilia na compreensão das interações dinâmicas entre as variáveis, mas não é um critério obrigatório para a continuidade da modelagem. Ainda que nem todas as relações apresentem causalidade significativa, o modelo VAR permanece válido, pois sua principal finalidade

é fornecer previsões. Assim, mesmo na ausência de causalidade estatisticamente significativa entre todas as variáveis, a análise com o VAR pode prosseguir normalmente. Em geral a manutenção das variáveis, ainda que com casualidade não significativa, tende a contribuir com a capacidade de explicação do modelo. No teste de Portmanteau, observa-se que apenas a variável Umidade (p-valor = 0,028) sugere a presença de autocorrelação dos resíduos, o que pode comprometer a qualidade do ajuste para essa série. Os testes ARCH não indicaram problemas de heterocedasticidade condicional, já que todos os p-valores são superiores ao nível de 5%. Por fim, o teste de Jarque-Bera não rejeitou a hipótese de normalidade dos resíduos em nenhuma das variáveis, com p-valores todos acima de 0,05.

As previsões foram realizadas para o período de outubro a dezembro de 2024, utilizando o comando como apresentado do Apêndice B.1.

As Tabelas 4.9 a 4.12 apresentam a comparação entre os valores observados e os valores previstos pelo modelo, bem como os respectivos intervalos de confiança de 95% para cada variável analisada (*precipitação, temperatura, umidade e vento*). Além disso, no rodapé de cada tabela, são reportados os valores das métricas **RMSE** (Root Mean Squared Error), **MSE** (Mean Squared Error) e **MAE** (Mean Absolute Error), que permitem quantificar a acurácia das previsões do modelo.

Tabela 4.9: Valores observados, previstos e intervalos de confiança (95%) para precipitação.

Mês	Observado	Previsto	IC 95%
10-2024	9,05	11,30	[8,00 ; 14,60]
11-2024	12,34	10,26	[6,31 ; 14,20]
12-2024	6,85	11,62	[7,59 ; 15,64]
	RMSE: 3,27	**MSE:** 10,72	**MAE:** 3,03

Tabela 4.10: Valores observados, previstos e intervalos de confiança (95%) para temperatura.

Mês	Observado	Previsto	IC 95%
10-2024	25,51	29,50	[27,47 ; 31,53]
11-2024	24,03	27,70	[23,33 ; 32,08]
12-2024	24,29	25,95	[19,02 ; 32,86]
	RMSE: 3,27	**MSE:** 10,72	**MAE:** 3,11

Tabela 4.11: Valores observados, previstos e intervalos de confiança (95%) para umidade.

Mês	Observado	Previsto	IC 95%
10-2024	59,07	56,09	[46,36 ; 65,81]
11-2024	70,50	63,69	[40,63 ; 86,75]
12-2024	71,34	80,41	[43,42 ; 117,40]
	RMSE: 6,77	**MSE:** 45,85	**MAE:** 6,29

Tabela 4.12: Valores observados, previstos e intervalos de confiança (95%) para vento.

Mês	Observado	Previsto	IC 95%
10-2024	1,65	1,32	[1,04 ; 1,59]
11-2024	1,71	1,37	[0,71 ; 2,04]
12-2024	1,43	1,29	[0,23 ; 2,35]
	RMSE: 0,28	**MSE:** 0,08	**MAE:** 0,27

Os gráficos de previsão são apresentados nas Figuras 4.9 a 4.12. Destaca-se que, em alguns casos, a umidade

ultrapassou 100% nos limites superiores, o que reforça que o modelo não conta com restrições matemáticas para impor restrições físicas às variáveis.

Nas Figuras 4.9 a 4.12 são apresentados os gráficos das previsões obtidas para cada variável, acompanhados de seus respectivos intervalos de confiança, de modo a ilustrar o desempenho preditivo do modelo.

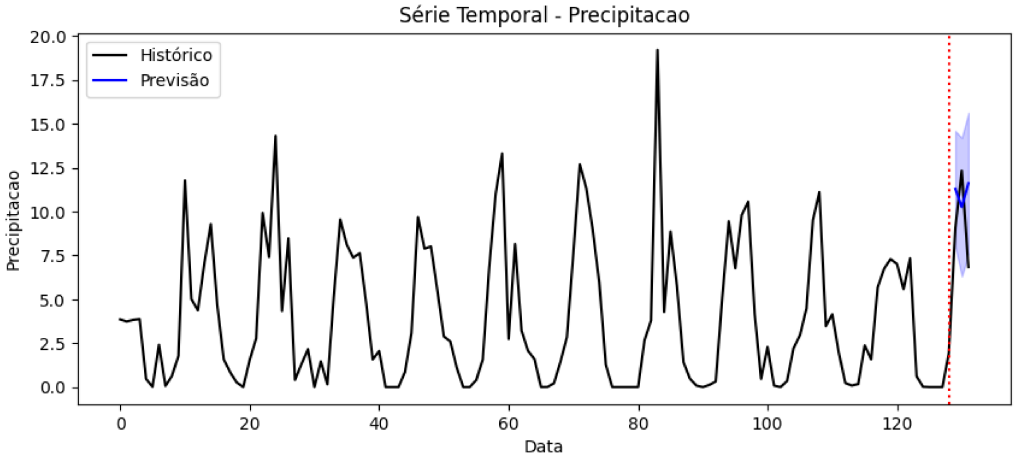

Figura 4.9: Previsão da série temporal de precipitação utilizando o modelo VAR.

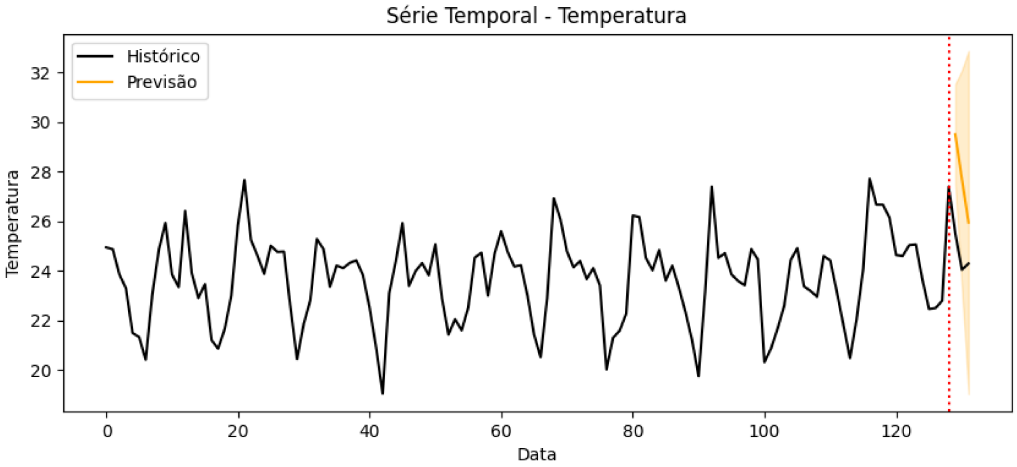

Figura 4.10: Previsão da série temporal de temperatura utilizando o modelo VAR.

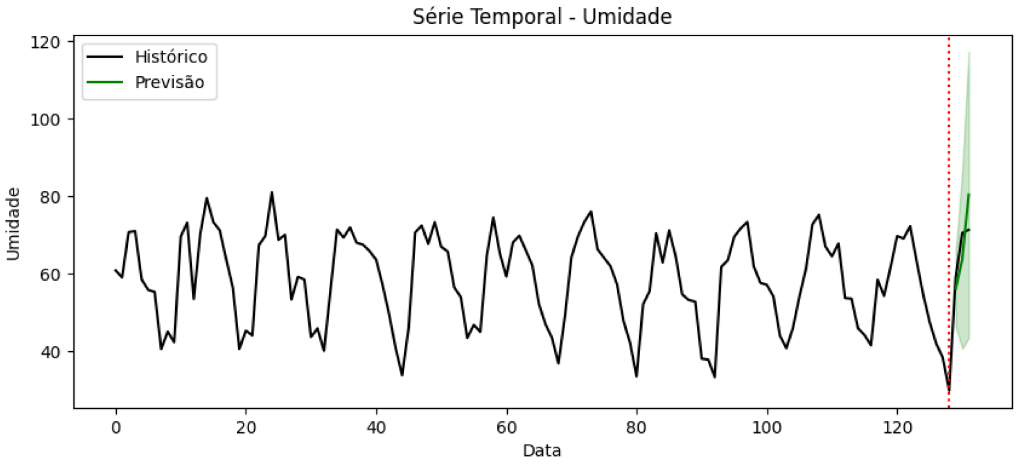

Figura 4.11: Previsão da série temporal de umidade utilizando o modelo VAR.

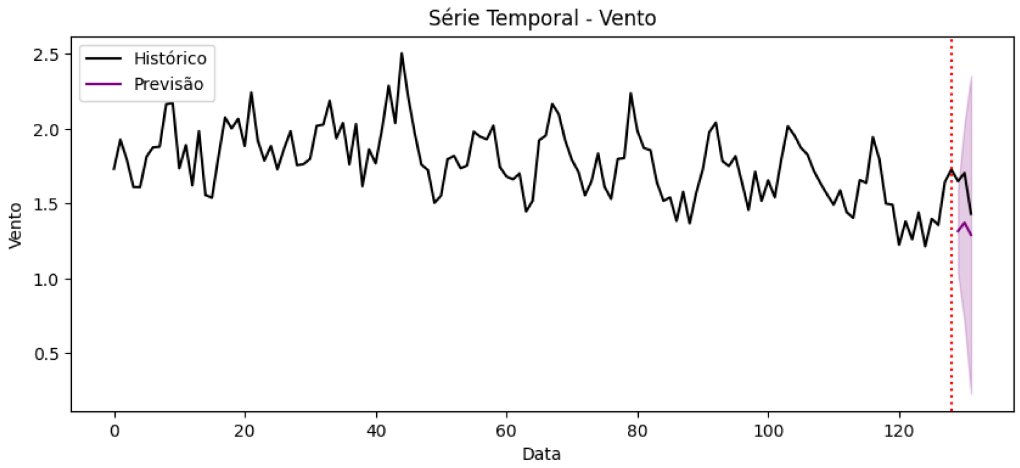

Figura 4.12: Previsão da série temporal de vento utilizando o modelo VAR.

4.4 Modelo LSTM (Long Short-Term Memory)

Nesta seção são apresentados os resultados obtidos a partir da rede neural LSTM. Este modelo é capaz de capturar dependências não lineares e de longo prazo nas séries temporais, mostrando-se adequado para aplicações preditivas em variáveis com padrões complexos. No entanto, ao contrário dos modelos SARIMA e VAR, o LSTM não fornece intervalos de confiança, motivo pelo qual os resultados apresentados a seguir incluem apenas os valores observados, previstos e as métricas de erro.

Assim como para as demais modelagens, as previsões foram realizadas para o período de outubro a dezembro de 2024. Nas Tabelas 4.13 a 4.16, respectivamente para as variáveis precipitação, temperatura, umidade e vento, são apresentados os valores observados e previstos, que permitem verificar a eficiência da modelagem. As métricas de avaliação RMSE, MSE e MAE também são apresentadas e permitem a comparação das diferentes modelagens.

Tabela 4.13: Valores observados e previstos para precipitação utilizando o modelo LSTM.

Mês	Observado	Previsto
10-2024	9,05	5,05
11-2024	1,34	7,45
12-2024	6,85	8,16
RMSE: 3,72	MSE: 8,05	MAE: 3,40

Tabela 4.14: Valores observados e previstos para temperatura utilizando o modelo LSTM.

Mês	Observado	Previsto
10-2024	25,51	28,64
11-2024	24,03	27,53
12-2024	24,29	26,58
RMSE: 3,02	MSE: 9,10	MAE: 2,97

Tabela 4.15: Valores observados e previstos para umidade utilizando o modelo LSTM.

Mês	Observado	Previsto
10-2024	59,07	55,28
11-2024	70,50	62,29
12-2024	71,34	62,48
RMSE: 7,31	MSE: 53,41	MAE: 6,95

Tabela 4.16: Valores observados e previstos para velocidade do vento utilizando o modelo LSTM.

Mês	Observado	Previsto
10-2024	1,65	1,71
11-2024	1,71	1,48
12-2024	1,43	1,49
RMSE: 0,14	MSE: 0,02	MAE: 0,12

A análise preditiva com a rede neural LSTM incluindo sua arquitetura, ou seja, suas camadas e unidades ajustadas, mostrou-se capaz de reter informações relevantes ao longo da série temporal, capturando tendências e padrões. Entretanto as previsões com este modelo apresentaram erros grandes, conforme métricas, quando comparado com o modelo SARIMA. Redes neurais, em geral, são treinadas com grande quantidade de dados e esta pode ser uma justificativa para a falta de precisão das estimativas obtidas com a LSTM. Além disso, a arquitetura da rede pode interferir na qualidade das predições.

Nas Figuras de 4.13 a 4.16, são apresentados os gráficos contendo os valores previstos em comparação com os valores observados, para cada uma das variáveis, o que permite verificar visualmente a capacidade do modelo em reproduzir comportamentos da série.

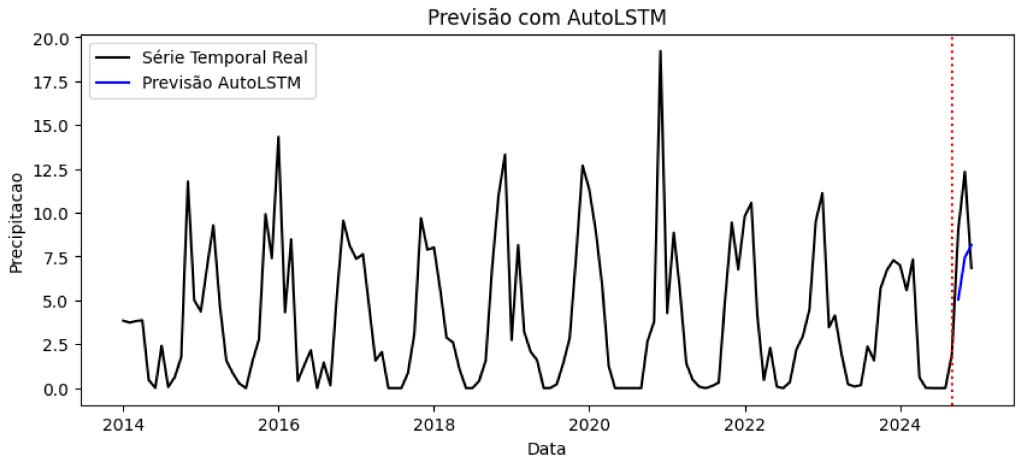

Figura 4.13: Previsão da série temporal de precipitação utilizando o modelo LSTM.

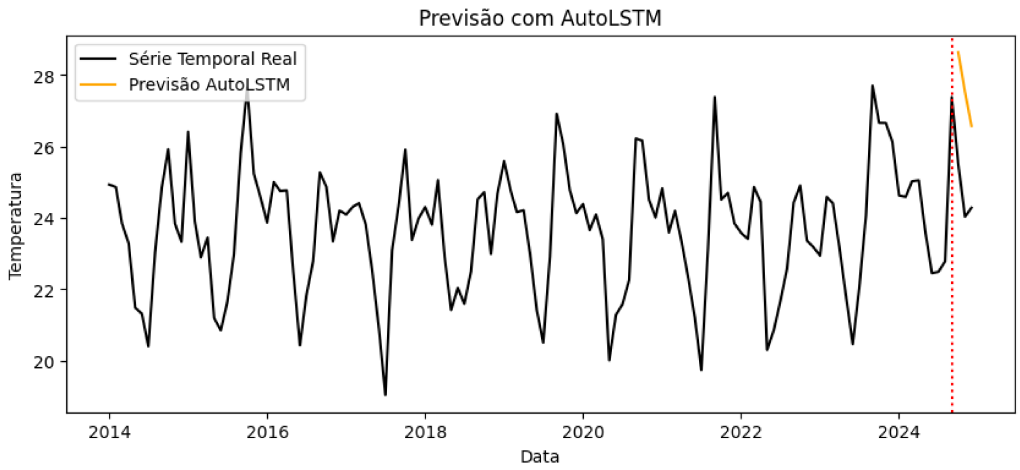

Figura 4.14: Previsão da série temporal de temperatura utilizando o modelo LSTM.

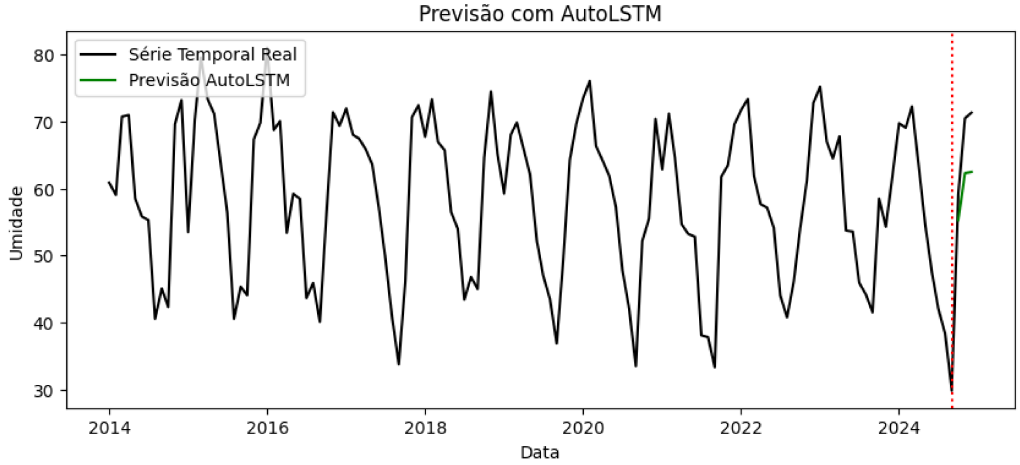

Figura 4.15: Previsão da série temporal de umidade utilizando o modelo LSTM.

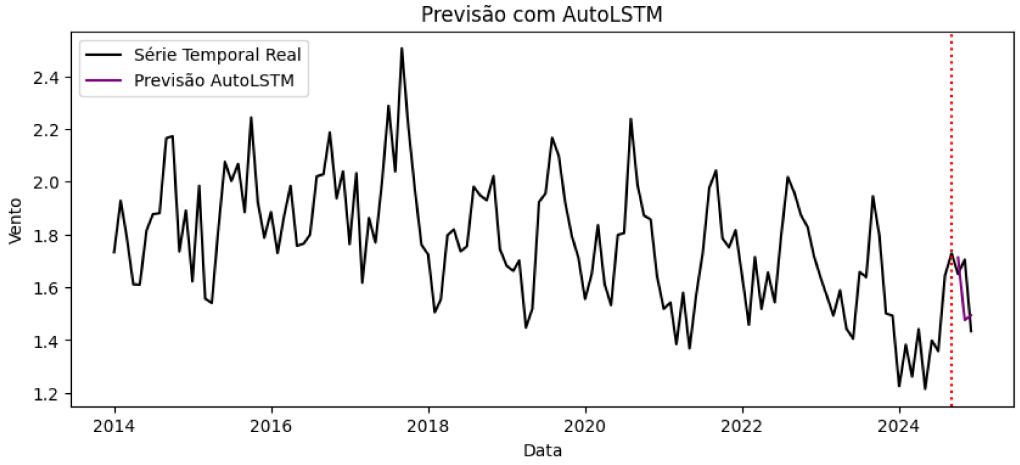

Figura 4.16: Previsão da série temporal de vento utilizando o modelo LSTM.

A arquitetura das redes LSTM utilizadas neste estudo foi definida por uma função de busca automatica, que selecionou a configuração mais adequada aos dados de cada variável.

Para a precipitação e temperatura, a rede apresentou duas camadas com 32 neurônios cada, enquanto para

a umidade a arquitetura foi mais simples, com duas camadas de 16 neurônios. Já a variável vento demandou uma rede mais profunda, composta por três camadas e 128 neurônios por camada.

Tabela 4.17: Arquitetura do modelo LSTM definida automaticamente para cada variável.

Variável	Número de camadas	Neurônios por camada
Precipitação	2	32
Temperatura	2	32
Umidade	2	16
Vento	3	128

Essas diferenças refletem a complexidade dos padrões presentes em cada série temporal. Variáveis como precipitação, que possuem forte variabilidade mas também padrões sazonais claros, podem ser representadas de forma eficiente com arquiteturas mais compactas. Por outro lado, a velocidade do vento tende a apresentar maior instabilidade e ruído, exigindo uma rede mais profunda e com mais unidades de memória para capturar suas flutuações. Nesse contexto, espera-se que a previsão da precipitação seja mais estável e precisa em comparação à do vento, enquanto variáveis intermediárias, como temperatura e umidade, apresentam comportamento equilibrado em termos de complexidade e exigência de arquitetura.

5. CONCLUSÕES

Este trabalho analisou séries temporais meteorológicas referentes às variáveis de precipitação, temperatura, umidade e velocidade do vento, empregando três diferentes abordagens: os modelos SARIMA, VAR e LSTM. A aplicação das técnicas permitiu compreender o comportamento das séries e avaliar o desempenho preditivo de cada modelo em contextos distintos.

As análises demonstraram que os procedimentos automáticos de seleção de parâmetros, como o uso das funções tanto para os modelos SARIMA, quanto VAR e LSTM, foram eficazes para identificar configurações parcimoniosas e adequadas às características das séries. No caso do VAR, a escolha da ordem com base em critérios de informação também garantiu consistência e comparabilidade com as demais técnicas, ainda que seus resultados tenham se mostrado menos satisfatórios.

A comparação dos modelos por meio das métricas MSE, RMSE e MAE revelou diferenças significativas entre as abordagens. O modelo SARIMA obteve os melhores resultados para precipitação, temperatura e umidade, evidenciando sua robustez na captura de padrões sazonais e na maior estabilidade das previsões. No entanto, alguns pressupostos do modelo não foram atendidos, o que pode impactar as inferências, embora ele ainda possa ser utilizado para previsões. O modelo LSTM destacou-se na previsão da velocidade do vento, uma variável caracterizada por maior instabilidade e ruído, demonstrando a flexibilidade dessa arquitetura em lidar com séries temporais mais complexas e não lineares. Por outro lado, o modelo VAR apresentou erros consistentemente mais elevados em todas as variáveis, indicando limitações em sua aplicação neste contexto específico. Portanto, os modelos devem ser utilizados com cautela, especialmente naquelas áreas onde seus pressupostos não são plenamente atendidos.

Portanto, se o interesse principal for nas variáveis precipitação, temperatura ou umidade indica-se utilizar o modelo SARIMA. Estas variáveis podem ser importantes por exemplo, para um agricultor na época do plantio. Por outro lado o modelo LSTM pode ser utilizado por este mesmo agricultor, na cultura do milho, quando o vento for um fator importante.

De modo geral, os resultados confirmam que os modelos estatísticos tradicionais ainda constituem ferramentas poderosas para a previsão de variáveis climáticas, especialmente quando a sazonalidade exerce papel central. As redes neurais LSTM, embora mais exigentes em termos computacionais, mostraram-se competitivas em cenários com maior irregularidade, enquanto o modelo VAR não demonstrou desempenho adequado frente às demais abordagens. Esses achados reforçam a importância de avaliar comparativamente diferentes técnicas, destacando que a escolha do método mais apropriado deve considerar tanto as propriedades estatísticas das séries quanto a natureza dos fenômenos climáticos analisados.

A busca por modelos tradicionais que atendam aos pressupotos inerentes às modelagens, SARIMA e VAR, configura entre os temas para estudos futuros. As funções de busca automática são de grande relevância mas não levam em consideração, na decisão sobre o modelo final, tais pressupostos.

A abordagem da rede neural LSTM por meio de aprendizado bayesiano é outro tema para estudos futuros. Esta abordagem, ao contrário da abordagem convencional permite calcular a incerteza da previsão por meio de intervalos de probabilidade.

Referências Bibliográficas

[1] Amegashie, P. K., Owusu-Sekyere, G., Frempong, E. e Addo, J. O.: *Modelling Trends of Climatic Variability and Malaria in Ghana Using Vector Autoregression*. Advances in Meteorology, 2018:1–12, 2018. https://onlinelibrary.wiley.com/doi/full/10.1155/2018/6124321.

[2] Barros, A. C., Mattos, D. M. de, Oliveira, I. C. L. de e outros: *Análise de Séries Temporais em R: Curso Introdutório*. GEN Atlas, Rio de Janeiro, 2017, ISBN 9788595154902. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788595154902/. Acesso em: 18 jun. 2025.

[3] Bem-Haja, A. K.: *Previsão de nível de bacia hidrográfica utilizando rede neural artificial do tipo LSTM*. https://repositorio.ifsp.edu.br/handle/123456789/709, 2017. Trabalho de Conclusão de Curso, IFSP.

[4] Fernandes, M., Silva, J. e Almeida, C.: *A critical review of RNN and LSTM variants in hydrological time series predictions*. MethodsX, 13:102946, 2024. Open access.

[5] Ferreira, P. G. C. (ed.): *Análise de séries temporais em R: curso introdutório*. Instituto Brasileiro de Economia (IBRE) – FGV, Rio de Janeiro, 2022.

[6] Haykin, S.: *Redes Neurais: Princípios e Prática*. Bookman, Porto Alegre, 2001, ISBN 9788577800865. E-book. Disponível via Minha Biblioteca (integrada.minhabiblioteca). Acesso em: 18jun.2025.

[7] Instituto Nacional de Meteorologia - INMET: *Banco de Dados Meteorológicos para Ensino e Pesquisa – BDMEP*, 2025. https://bdmep.inmet.gov.br/, Acesso em: 05 set. 2025.

[8] Loureiro, A. S.: *Previsão de séries temporais utilizando modelos ARIMA e redes neurais artificiais*. https://imef.furg.br/images/documentos/matematica-aplicada/monografias/2023-Aryel_Soares_Loureiro.pdf, 2023. Trabalho de Conclusão de Curso (Bacharelado em Matemática Aplicada), Universidade Federal do Rio Grande.

[9] Mata, A. C. O. R. da: *Aplicação do Modelo SARIMA em Série do Índice de Calor na Baixada Cuiabana*. https://repositorio.pgsscogna.com.br/bitstream/123456789/23246/1/Ana%20Cristina%20Oliveira%20Ribeiro%20da%20Mata.pdf, 2019. Dissertação de Mestrado, Universidade de Cuiabá (UNIC) — Mestrado em Ciências Ambientais.

[10] Morettin, P. A. e Toloi, C. M. C.: *Análise de Séries Temporais: Modelos Lineares Univariados*. Editora Blucher, São Paulo, 3ª ed., 2018.

[11] Nixtla NeuralForecast Development Team: *NeuralForecast Documentation: AutoLSTM*, 2025. https://github.com/Nixtla/neuralforecast, acesso em 2025-09-17, Version 3.0.2.

[12] Nunes, T.: *Comparison of the Predictive Performance of ARIMA, VAR and VEC Models in Cointegrated Time Series*. https://www.researchgate.net/publication/376409458_Comparison_of_the_Predictive_Performance_of_ARIMA_VAR_and_VEC_Models_in_Cointegrated_Time_Series, 2023. Artigo científico, ResearchGate.

[13] Python Software Foundation: *Python Documentation*, 2025. https://docs.python.org/3/, acesso em 2025-09-05, Version 3.13.7.

[14] Rahman, M. M., Islam, M. S. e Chowdhury, M. A. H.: *Modeling and Forecasting of Climatic Parameters: Univariate SARIMA Versus Multivariate Vector Autoregression Approach*. Journal of the Bangladesh Agricultural University, 16(3):492–502, 2018. https://banglajol.info/index.php/JBAU/article/view/36494.

[15] Santos, J. A. A. dos e Chaucoski, Y.: *Previsão do Consumo de Energia Elétrica na Região Sudeste: Um Estudo de Caso Usando SARIMA e LSTM*. https://www.researchgate.net/publication/347638414_Previsao_do_Consumo_de_Energia_Eletrica_na_Regiao_Sudeste_Um_Estudo_de_Caso_Usando_SARIMA_e_LSTM, 2020. Revista CEREUS, Artigo científico.

[16] Silveira Bueno, R. de Losso da: *Econometria de Séries Temporais*. Cengage Learning, São Paulo, 2ª ed., 2018.

[17] Soares, A. C.: *Análise de séries temporais com uso de redes neurais artificiais em dados meteorológicos para previsão de chuva e eventos climáticos severos*. https://repositorio.ufu.br/bitstream/123456789/36857/1/AnaliseSeriesTemporais.pdf, 2022. Dissertação de Mestrado, Universidade Federal de Uberlândia.

[18] Statsmodels Development Team: *Statsmodels: Vector Autoregressive Models (VAR) – Documentação*, 2025. https://www.statsmodels.org/dev/generated/statsmodels.tsa.vector_ar.var_model.VAR.html, acesso em 2025-10-14, Módulo `statsmodels.tsa.vector_ar.var_model.VAR`.

[19] The PMDARIMA Development Team: *PMDARIMA Documentation*, 2025. https://github.com/alkaline-ml/pmdarima, acesso em 2025-09-05, Version 2.0.4.

A. Código SARIMA

Apêndice A.1: Código para implementação do modelo SARIMA.

```

!pip uninstall -y numpy pmdarima pandas
!pip install numpy==1.26.4
!pip install pandas==2.3.1
!pip install pmdarima==2.0.4
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
%matplotlib inline
from pmdarima.arima import auto_arima
from statsmodels.tsa.statespace.sarimax import SARIMAX
from pmdarima import arima
from pmdarima.utils import tsdisplay, autocorr_plot, decomposed_plot, plot_acf, plot_pacf
from matplotlib.pyplot import rcParams
rcParams['figure.figsize']=15,6
from pmdarima.arima import ADFTest

df=pd.read_csv("/content/dados_diarios.txt",sep="\t")
df['Precipitacao'] = pd.to_numeric(df['Precipitacao'], errors='coerce')
df['Umidade'] = pd.to_numeric(df['Umidade'], errors='coerce')
df['Temperatura'] = pd.to_numeric(df['Temperatura'], errors='coerce')
df['Vento'] = pd.to_numeric(df['Vento'], errors='coerce')
df['Data'] = pd.to_datetime(df['Data'], format='%d/%m/%Y')
df['Precipitacao'] = np.where(df['Precipitacao'] < 0, 0, df['Precipitacao'])
df['Mes'] = df['Data'].dt.to_period('M')
display(df)
medias_mensais = df[df['Umidade'] > 0].groupby('Mes')['Umidade'].mean()
df['Umidade'] = df.apply(lambda row: medias_mensais[row['Mes']] if row['Umidade'] < 0 else row
    ['Umidade'], axis=1)

display(df)
df2_m = df.groupby("Mes").agg({
    "Precipitacao": "mean",   # Soma da precipita  o
    "Temperatura": "mean",    # M dia da temperatura
    "Umidade": "mean",        # M dia da umidade
    "Vento": "mean"           # M dia do vento
}).reset_index()


# Exibir as primeiras linhas do resultado
print(df2_m.tail(n=6))
data_corte=df2_m.tail(3).iloc[0,0]
print(data_corte)
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Precipitacao'], color='blue', linewidth=1.5)
plt.title('S rie Temporal da Precipita  o', fontsize=14)
```

```
plt.xlabel('Data', fontsize=12)
plt.ylabel('Precipita  o (mm)', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Umidade'], color='green', linewidth=1.5)
plt.title('S rie Temporal da Umidade', fontsize=14)
plt.xlabel('Data', fontsize=12)
plt.ylabel('Umidade (%)', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Temperatura'], color='red', linewidth=1.5)
plt.title('S rie Temporal da Temperatura', fontsize=14)
plt.xlabel('Data', fontsize=12)
plt.ylabel('Temperatura ( C )', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Vento'], color='orange', linewidth=1.5)
plt.title('S rie Temporal da Velocidade do Vento', fontsize=14)
plt.xlabel('Data', fontsize=12)
plt.ylabel('Velocidade do Vento (m/s)', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()
df2_m.tail(3)
df_train = df2_m[df2_m['Mes'] < data_corte].copy()
df_train = df_train.drop(['Mes'],axis=1)
df_test  = df2_m[df2_m['Mes'] >= data_corte].copy()
# Teste de estacionariedade (ADF)
# valor p menor que 0,05 indica que a s rie
# estacion ria
from statsmodels.tsa.stattools import adfuller

def check_stationarity_statsmodels(series):
    result = adfuller(series.dropna())
    p_value = result[1]  # posi  o 1 = p-valor
    return p_value

p_values = df_train.apply(check_stationarity_statsmodels)
print("P-valores do teste ADF:")
print(p_values)
df_diff=df_train.copy()
for coluna in df_train.columns:
  if (p_values[coluna] > 0.05):
    df_diff[coluna] = df_train[coluna].diff()
  else:
    df_diff[coluna] = df_train[coluna]
def check_stationarity(series):
    result = adfuller(series.dropna())
    return result[1] # Return the p-value

p_values2 = df_diff.apply(check_stationarity)
print("P-valores do teste ADF:")
```

```
print(p_values2)
df_diff=df_diff.dropna(axis=0)
df_diff.head()
# Sele  o do melhor modelo.
import time
from pmdarima import auto_arima
# Precipita  o
start_time = time.time()
model_precipitacao = auto_arima(df_diff['Precipitacao'],
                                start_p=0, start_q=0, d=None,
                                max_p=4, max_q=4, max_d=4,
                                start_P=0, start_Q=0, D=0,
                                max_P=3, max_D=3, max_Q=3, max_order=14,
                                m=12, seasonal=True, trace=True,
                                error_action='ignore', suppress_warnings=True,
                                stepwise=True)
tempo = (time.time() - start_time)/60
print(f"Tempo de execu  o (Precipita  o): {tempo:.2f} minutos")
# Temperatura
start_time = time.time()
model_temperatura = auto_arima(df_diff['Temperatura'],
                               start_p=0, start_q=0, d=None,
                               max_p=4, max_q=4, max_d=4,
                               start_P=0, start_Q=0, D=0,
                               max_P=3, max_D=3, max_Q=3, max_order=14,
                               m=12, seasonal=True, trace=True,
                               error_action='ignore', suppress_warnings=True,
                               stepwise=True)
tempo = (time.time() - start_time)/60
print(f"Tempo de execu  o (Temperatura): {tempo:.2f} minutos")


# Vento
start_time = time.time()
model_vento = auto_arima(df_diff['Vento'],
                         start_p=0, start_q=0, d=None,
                         max_p=4, max_q=4, max_d=4,
                         start_P=0, start_Q=0, D=0,
                         max_P=3, max_D=3, max_Q=3, max_order=14,
                         m=12, seasonal=True, trace=True,
                         error_action='ignore', suppress_warnings=True,
                         stepwise=True)
tempo = (time.time() - start_time)/60
print(f"Tempo de execu  o (Vento): {tempo:.2f} minutos")
# Umidade
start_time = time.time()
model_umidade = auto_arima(df_diff['Umidade'],
                           start_p=0, start_q=0, d=None,
                           max_p=6, max_q=6, max_d=6,
                           start_P=0, start_Q=0, D=0,
                           max_P=3, max_D=3, max_Q=3, max_order=18,
                           m=12, seasonal=True, trace=True,
                           error_action='ignore', suppress_warnings=True,
                           stepwise=True)
tempo = (time.time() - start_time)/60
print(f"Tempo de execu  o (Umidade): {tempo:.2f} minutos")
# Melhor modelo.

# Melhor modelo (formato simples)
print("Precipita  o:", f"ARIMA{model_precipitacao.order}{model_precipitacao.seasonal_order
    }[12]")
```

```
169 print("Temperatura: ", f"ARIMA{model_temperatura.order}{model_temperatura.seasonal_order}[12]"
    )
170 print("Vento:       ", f"ARIMA{model_vento.order}{model_vento.seasonal_order}[12]")
171 print("Umidade:     ", f"ARIMA{model_umidade.order}{model_umidade.seasonal_order}[12]")
172 print("Crit rios de avalia  o dos modelos:\n")
173
174 print("Precipita  o:")
175 print("  AIC:", model_precipitacao.aic())
176 print("  BIC:", model_precipitacao.bic())
177
178 print("\nTemperatura:")
179 print("  AIC:", model_temperatura.aic())
180 print("  BIC:", model_temperatura.bic())
181
182 print("\nVento:")
183 print("  AIC:", model_vento.aic())
184 print("  BIC:", model_vento.bic())
185
186 print("\nUmidade:")
187 print("  AIC:", model_umidade.aic())
188 print("  BIC:", model_umidade.bic())
189
190 # Treinando o model escolhido em step
191
192 # Ajustar modelos para cada vari vel usando os dados de treino
193
194 model_precipitacao.fit(df_diff['Precipitacao'])
195 model_temperatura.fit(df_diff['Temperatura'])
196 model_vento.fit(df_diff['Vento'])
197 model_umidade.fit(df_diff['Umidade'])
198
199 #colocar print dos 4 modelos de cada variavel
200
201 model_precipitacao.summary()
202 model_temperatura.summary()
203 model_vento.summary()
204 model_umidade.summary()
205 ### VERIFIC  O DE PRESSUPOSTOS
206
207 import matplotlib.pyplot as plt
208 import statsmodels.api as sm
209 from statsmodels.stats.diagnostic import acorr_ljungbox, het_arch
210 from scipy.stats import jarque_bera, shapiro
211
212 # Fun  o para diagn stico de res duos
213 def diagnosticar_residuos(modelo, nome):
214     residuos = modelo.resid()
215
216     print(f"\n==== Diagn stico para {nome} ====")
217
218     # 1. Independ ncia dos res duos
219     print("\nTeste de Ljung-Box (autocorrela  o):")
220     lb = acorr_ljungbox(residuos, lags=[12], return_df=True)
221     print(lb)
222
223     fig, ax = plt.subplots(figsize=(5,2))  # define tamanho aqui
224     sm.graphics.tsa.plot_acf(residuos, lags=24, ax=ax)  # usa o mesmo eixo
225     ax.set_title("ACF dos res duos - Precipita  o")
226     plt.show()
227
228     # 2. Normalidade
```

```
    print("\nTeste de Normalidade (Jarque-Bera):")
    jb = jarque_bera(residuos)
    print(f"Estat stica={jb.statistic:.4f}, p-valor={jb.pvalue:.4f}")

    # 2. Normalidade
    #print("\nTeste de Normalidade (Shapiro-Wilk):")
    #sw = jarque_bera(residuos)
    #print(f"Estat stica={sw.statistic:.4f}, p-valor={sw.pvalue:.4f}")

    fig, ax = plt.subplots(figsize=(5,2))  # define tamanho aqui
    sm.qqplot(residuos, line='s', ax=ax)
    plt.title(f"Q-Q Plot - Precipita  o")
    plt.show()

    fig, ax = plt.subplots(figsize=(5,2))  # define tamanho aqui
    plt.hist(residuos, bins=20, edgecolor='k')
    plt.title(f"Histograma dos res duos - {nome}")
    plt.show()

    # 3. Homocedasticidade
    print("\nTeste de Heterocedasticidade (ARCH):")
    arch = het_arch(residuos)
    print(f"Estat stica={arch[0]:.4f}, p-valor={arch[1]:.4f}")
diagnosticar_residuos(model_umidade, "Umidade")
diagnosticar_residuos(model_precipitacao, "Precipita  o")
diagnosticar_residuos(model_temperatura, "Temperatura")
diagnosticar_residuos(model_vento, "Vento")
h=df_test.shape[0]
print(h)
# Fazer previs o e obter intervalo de confian a para cada modelo - escala transformada

previsao_precipitacao, conf_int_precipitacao = model_precipitacao.predict(n_periods=h,
    return_conf_int=True)
previsao_temperatura, conf_int_temperatura = model_temperatura.predict(n_periods=h,
    return_conf_int=True)
previsao_vento, conf_int_vento = model_vento.predict(n_periods=h, return_conf_int=True)
previsao_umidade, conf_int_umidade = model_umidade.predict(n_periods=h, return_conf_int=True)

ultimo_valor = df_train['Temperatura'].iloc[-1] # ok
print(ultimo_valor)
def reverter_diferenciacao(previsao_diff, ultimo_valor):
    """Reverte a diferencia  o acumulando a previs o e somando ao  ltimo  valor real"""
    return np.r_[ultimo_valor, previsao_diff].cumsum()[1:]
# Considera que voc  j  tem: p_values, df_train (log1p, antes do diff), h_*

# Precipita  o
if p_values['Precipitacao'] > 0.05:
    ultimo_valor = df_train['Precipitacao'].iloc[-1]
    previsao_precipitacao = reverter_diferenciacao(previsao_precipitacao, ultimo_valor)
    conf_int_precipitacao = np.column_stack([
        reverter_diferenciacao(conf_int_precipitacao[:, 0], ultimo_valor),
        reverter_diferenciacao(conf_int_precipitacao[:, 1], ultimo_valor)
    ])
else:
    previsao_precipitacao = previsao_precipitacao
    conf_int_precipitacao = conf_int_precipitacao

# Temperatura
if p_values['Temperatura'] > 0.05:
    ultimo_valor = df_train['Temperatura'].iloc[-1]
    previsao_temperatura = reverter_diferenciacao(previsao_temperatura, ultimo_valor)
```

```
    conf_int_temperatura = np.column_stack([
        reverter_diferenciacao(conf_int_temperatura[:, 0], ultimo_valor),
        reverter_diferenciacao(conf_int_temperatura[:, 1], ultimo_valor)
    ])
else:
    previsao_temperatura = previsao_temperatura
    conf_int_temperatura = conf_int_temperatura

# Vento
if p_values['Vento'] > 0.05:
    ultimo_valor = df_train['Vento'].iloc[-1]
    previsao_vento = reverter_diferenciacao(previsao_vento, ultimo_valor)
    conf_int_vento = np.column_stack([
        reverter_diferenciacao(conf_int_vento[:, 0], ultimo_valor),
        reverter_diferenciacao(conf_int_vento[:, 1], ultimo_valor)
    ])
else:
    previsao_vento = previsao_vento
    conf_int_vento = conf_int_vento

# Umidade
if p_values['Umidade'] > 0.05:
    ultimo_valor = df_train['Umidade'].iloc[-1]
    previsao_umidade = reverter_diferenciacao(previsao_umidade, ultimo_valor)
    conf_int_umidade = np.column_stack([
        reverter_diferenciacao(conf_int_umidade[:, 0], ultimo_valor),
        reverter_diferenciacao(conf_int_umidade[:, 1], ultimo_valor)
    ])
else:
    previsao_umidade = previsao_umidade
    conf_int_umidade = conf_int_umidade
## N o  h   mais a necessidade dessa transforma  o inversa

#previsao_precipitacao_original = np.expm1(previsao_precipitacao_log)
#conf_int_precipitacao_original = np.expm1(conf_int_precipitacao_log)

#previsao_temperatura_original = np.expm1(previsao_temperatura_log)
#conf_int_temperatura_original = np.expm1(conf_int_temperatura_log)

#previsao_vento_original = np.expm1(previsao_vento_log)
#conf_int_vento_original = np.expm1(conf_int_vento_log)

#previsao_umidade_original = np.expm1(previsao_umidade_log)
#conf_int_umidade_original = np.expm1(conf_int_umidade_log)

previsao_precipitacao
plt.figure(figsize=(12, 6))

# Plotting the historical data
plt.plot(df2_m.index, df2_m['Precipitacao'], label='Hist rico', color='blue')
plt.plot(df_test.index,previsao_precipitacao,label='Previs o',color='red')
plt.fill_between(
    df_test.index,
    conf_int_precipitacao[:, 0],  # limite inferior
    conf_int_precipitacao[:, 1],  # limite superior
    color='gray',
    alpha=0.3,
    label='Intervalo de Previs o'
)
# Est tica
plt.title('S rie Temporal - Precipita  o')
```

```
plt.xlabel(' ndice ')
plt.ylabel('Precipita  o')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
plt.figure(figsize=(12, 6))

# Plotting the historical data
plt.plot(df2_m.index, df2_m['Temperatura'], label='Hist rico', color='blue')
plt.plot(df_test.index,previsao_temperatura,label='Previs o',color='red')
plt.fill_between(
    df_test.index,
    conf_int_temperatura[:, 0],  # limite inferior
    conf_int_temperatura[:, 1],  # limite superior
    color='gray',
    alpha=0.3,
    label='Intervalo de Previs o'
)
# Est tica
plt.title('S rie Temporal - Temperatura')
plt.xlabel(' ndice ')
plt.ylabel('Umidade')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
plt.figure(figsize=(12, 6))

# Plotting the historical data
plt.plot(df2_m.index, df2_m['Vento'], label='Hist rico', color='blue')
plt.plot(df_test.index,previsao_vento,label='Previs o',color='red')
plt.fill_between(
    df_test.index,
    conf_int_vento[:, 0],  # limite inferior
    conf_int_vento[:, 1],  # limite superior
    color='gray',
    alpha=0.3,
    label='Intervalo de Previs o'
)
# Est tica
plt.title('S rie Temporal - Vento')
plt.xlabel(' ndice ')
plt.ylabel('Vento')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
plt.figure(figsize=(12, 6))

# Plotting the historical data
plt.plot(df2_m.index, df2_m['Umidade'], label='Hist rico', color='blue')
plt.plot(df_test.index,previsao_umidade,label='Previs o',color='red')
plt.fill_between(
    df_test.index,
    conf_int_umidade[:, 0],  # limite inferior
    conf_int_umidade[:, 1],  # limite superior
    color='gray',
    alpha=0.3,
    label='Intervalo de Previs o'
)
```

```
# Est tica
plt.title('S rie Temporal - Umidade')
plt.xlabel(' ndice ')
plt.ylabel('Umidade')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
import matplotlib.pyplot as plt

# Configura  o global
plt.rcParams.update({
    "figure.figsize": (12, 6),
    "axes.titlesize": 16,
    "axes.labelsize": 14,
    "xtick.labelsize": 12,
    "ytick.labelsize": 12,
    "legend.fontsize": 12,
    "grid.linestyle": "--",
    "grid.alpha": 0.7,
})

def plot_series(df_hist, df_test, previsao, conf_int, var_name, pred_color):
    """
    Gr fico em preto para hist rico, linha vertical vermelha,
    apenas a previs o em cor destacada.
    """
    plt.figure()

    # Hist rico em preto
    plt.plot(df_hist.index, df_hist[var_name], label='Hist rico', color='black')

    # Previs o colorida
    plt.plot(df_test.index, previsao, label='Previs o', color=pred_color)

    # Intervalo de confian a em mesma cor da previs o, transparente
    plt.fill_between(df_test.index, conf_int[:,0], conf_int[:,1], color=pred_color, alpha=0.3)

    # Linha vertical vermelha pontilhada (n o entra na legenda)
    plt.axvline(x=df_test.index[0], color='red', linestyle=':', linewidth=2)

    # Est tica
    plt.title(f"S rie Temporal - {var_name}")
    plt.xlabel(" ndice ")
    plt.ylabel(var_name)
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()

# Chamadas para cada vari vel
plot_series(df2_m, df_test, previsao_precipitacao, conf_int_precipitacao, "Precipitacao", "
    blue")
plot_series(df2_m, df_test, previsao_temperatura, conf_int_temperatura, "Temperatura", "
    darkorange")
plot_series(df2_m, df_test, previsao_umidade, conf_int_umidade, "Umidade", "green")
plot_series(df2_m, df_test, previsao_vento, conf_int_vento, "Vento", "purple")

forecast = pd.DataFrame({
    "Precipitacao": previsao_precipitacao,
    "Umidade": previsao_umidade,
```

```
    "Temperatura": previsao_temperatura,
    "Vento": previsao_vento
})
forecast #valores preditos
# intervalos de confian a
print("Previs o precipita  o:")
print(previsao_precipitacao)
print("\nIntervalo de confian a precipita  o:")
print(conf_int_precipitacao)

print("\nPrevis o temperatura:")
print(previsao_temperatura)
print("\nIntervalo de confian a temperatura:")
print(conf_int_temperatura)

print("\nPrevis o vento:")
print(previsao_vento)
print("\nIntervalo de confian a vento:")
print(conf_int_vento)

print("\nPrevis o umidade:")
print(previsao_umidade)
print("\nIntervalo de confian a umidade:")
print(conf_int_umidade)
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
import pandas as pd

# DataFrames: df_test (valores observados), forecast (valores previstos)
# Assegure-se de que ambos est o alinhados temporalmente (mesmo  ndice )

# Dicion rios para armazenar os resultados
rmse_resultados = {}
mae_resultados = {}
mse_resultados = {}

for col in forecast.columns:
    # Remove valores nulos, se houver
    y_true = df_test[col].dropna()
    y_pred = forecast[col].dropna()

    # Alinhar os  ndices  antes de comparar
    y_true, y_pred = y_true.align(y_pred, join='inner')

    # C lculo das m tricas
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)

    # Armazenar resultados
    mse_resultados[col] = mse
    rmse_resultados[col] = rmse
    mae_resultados[col] = mae

# Mostrar os resultados
print("\nMSE por vari vel:")
for col, val in mse_resultados.items():
    print(f"{col}: {val:.4f}")

print("\nRMSE por vari vel:")
for col, val in rmse_resultados.items():
```

```
530     print(f"{col}: {val:.4f}")
531
532 print("\nMAE por vari vel:")
533 for col, val in mae_resultados.items():
534     print(f"{col}: {val:.4f}")
```

B. Código VAR

Apêndice B.1: Código para implementação do modelo VAR.

```
1
2  import numpy as np
3  import pandas as pd
4  import matplotlib.pyplot as plt
5  import statsmodels.api as sm
6  from statsmodels.tsa.api import VAR
7  from statsmodels.tsa.stattools import adfuller
8
9
10 df=pd.read_csv('dados_diarios.txt',sep='\t')
11 df['Precipitacao'] = pd.to_numeric(df['Precipitacao'], errors='coerce')
12 df['Umidade'] = pd.to_numeric(df['Umidade'], errors='coerce')
13 df['Temperatura'] = pd.to_numeric(df['Temperatura'], errors='coerce')
14 df['Vento'] = pd.to_numeric(df['Vento'], errors='coerce')
15 df['Data'] = pd.to_datetime(df['Data'], format='%d/%m/%Y')
16
17 # Plotando as s ries temporais
18 plt.figure(figsize=(10, 5))
19 for column in df[df.columns.difference(['Data'])].columns:
20   plt.plot(df['Data'].index, df[column], label=column)
21 plt.legend()
22 plt.title('S ries Temporais Multivariadas')
23 plt.show()
24
25 df['Precipitacao'] = np.where(df['Precipitacao'] < 0, 0, df['Precipitacao'])
26 df['Mes'] = df['Data'].dt.to_period('M')
27 medias_mensais = df[df['Umidade'] > 0].groupby('Mes')['Umidade'].mean()
28 df['Umidade'] = df.apply(lambda row: medias_mensais[row['Mes']] if row['Umidade'] < 0 else row
       ['Umidade'], axis=1)
29 plt.figure(figsize=(10, 5))
30 plt.plot(df.index, df['Precipitacao'], color='blue', linewidth=1.5)
31 plt.title('S rie Temporal da Precipita  o', fontsize=14)
32 plt.xlabel('Data', fontsize=12)
33 plt.ylabel('Precipita  o (mm)', fontsize=12)
34 plt.grid(True)
35 plt.tight_layout()
36 plt.show()
37
38 plt.figure(figsize=(10, 5))
39 plt.plot(df.index, df['Umidade'], color='green', linewidth=1.5)
40 plt.title('S rie Temporal da Umidade', fontsize=14)
41 plt.xlabel('Data', fontsize=12)
42 plt.ylabel('Umidade (%)', fontsize=12)
43 plt.grid(True)
44 plt.tight_layout()
45 plt.savefig('Umidade.png', dpi=300, bbox_inches='tight')
46
47 plt.figure(figsize=(10, 5))
```

```
plt.plot(df.index, df['Temperatura'], color='orange', linewidth=1.5)
plt.title('S rie Temporal da Temperatura', fontsize=14)
plt.xlabel('Data', fontsize=12)
plt.ylabel('Temperatura ( C )', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.savefig('Temperatura.png', dpi=300, bbox_inches='tight')

plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Vento'], color='purple', linewidth=1.5)
plt.title('S rie Temporal da Velocidade do Vento', fontsize=14)
plt.xlabel('Data', fontsize=12)
plt.ylabel('Velocidade do Vento (m/s)', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.savefig('Vento.png', dpi=300, bbox_inches='tight')
# Plotando as s ries temporais
plt.figure(figsize=(10, 5))
for column in df[df.columns.difference(['Data','Mes'])].columns:
  plt.plot(df['Data'].index, df[column], label=column)
plt.legend()
plt.title('S ries Temporais Multivariadas')
plt.show()

# Agregar os dados por m s
# Ver se melhor dados di rios ou mensais
df_m = df.groupby("Mes").agg({
    "Precipitacao": "mean",   # Soma da precipita  o
    "Temperatura": "mean",   # M dia da temperatura
    "Umidade": "mean",       # M dia da umidade
    "Vento": "mean"          # M dia do vento
}).reset_index()


# Exibir as primeiras linhas do resultado
print(df_m.tail(n=6))
# Separa  o entre treino e teste
n_test=3
df_train = df_m.iloc[:-n_test]
print(df_train.tail())
# seleciona todas as linhas exceto as n_test  ltimas

df_test = df_m.iloc[-n_test:]
# seleciona as  ltimas  n_test linhas
print(df_test)

# Teste de estacionariedade (ADF)
# valor p menor que 0,05 indica que a s rie
# estacion ria
def check_stationarity(series):
    result = adfuller(series)
    return result[1]  # Retorna o p-valor

p_values = df_train.drop('Mes', axis=1).apply(check_stationarity)
print("P-valores do teste ADF:")
print(p_values)
# Se alguma s rie  n o for estacion ria, diferenciar
df_diff = df_train.drop('Mes', axis=1).copy()

for col in df_train.drop('Mes', axis=1).columns:
    if p_values[col] > 0.05:
```

```
        # Se n o estacion ria     diferenciar
        df_diff[col] = df_train[col].diff()
    else:
        # Se estacion ria     mant m como est
        df_diff[col] = df_train[col]

# Remover primeira linha que ter  NaN em pelo menos uma coluna
df_diff = df_diff.dropna()

#diferencia apenas a serie que tem p valor maior que 0,05

# Teste de estacionariedade (ADF)
def check_stationarity(series):
    result = adfuller(series)
    return result[1]  # Retorna o p-valor

p_values = df_diff.apply(check_stationarity)
print("P-valores do teste ADF:")
print(p_values)
from statsmodels.graphics.tsaplots import plot_pacf

# Verificar como interpretar os gr ficos
for column in df_diff.columns:
  plot_pacf(df_diff[column], lags=30)
  plt.show()
model=VAR(df_diff)
maxlag=20
order_results = model.select_order(maxlags=maxlag)
print(order_results.summary())
lag_order = order_results.aic

#essa fun ao (model.select_order) procura a melhor ordem de acordo com qual metrica, aic, bic
    , fpe??? ate o lag maximo de 20 escolhido arbitrariamente
from statsmodels.tsa.api import VAR

model = VAR(df_diff)
maxlag = 20

# Seleciona a melhor ordem de lag usando diferentes crit rios
order_results = model.select_order(maxlags=maxlag)

# Mostra o resumo com todos os crit rios
print(order_results.summary())

# Pega o lag de cada crit rio
lag_order_aic = order_results.aic
lag_order_bic = order_results.bic
lag_order_fpe = order_results.fpe
lag_order_hqic = order_results.hqic

print(f"Lag escolhido pelo AIC: {lag_order_aic}")
print(f"Lag escolhido pelo BIC: {lag_order_bic}")
print(f"Lag escolhido pelo FPE: {lag_order_fpe}")
print(f"Lag escolhido pelo HQIC: {lag_order_hqic}")

print(lag_order)
var_model = model.fit(lag_order)
print(var_model.summary())

#colocar a equa  o do modelo VAR na metodologia "resumida"
#colocar tabelas das equa  es no latex
```

```
# Ver como interpretar o teste a seguir

variaveis=['Vento','Precipitacao', 'Temperatura','Umidade']

for var in variaveis:
  causality_test = var_model.test_causality(caused=var,
        causing = [v for v in variaveis if v != var], kind='f')
  print(causality_test.summary())
## Diagn stico - res duos e estabilidade do modelo

# res duos n o correlacionados
# H0: n o h  autocorrela  o significativa nos res duos
from statsmodels.stats.diagnostic import acorr_ljungbox

residuos = var_model.resid
residuos.columns

for col in residuos.columns:
    # Teste de Portmanteau (PT.asymptotic e PT.adjusted)
    # Por que lags=[10] ???? Esse valor testa se lags n o considerados
    # no modelo pode estar causando autocorrela  o. Isto  , autocorrela  o
    # pela falta de lags maiores. Ent o , se a modelagem
    # foi feita com lag_order, deve ser testado, por exempo
    # lags=[lag_order+10]
    #
    pt_asymp = acorr_ljungbox(residuos[col], lags=[lag_order+25], return_df=True)
    pt_adjusted = acorr_ljungbox(residuos[col], lags=[lag_order+25], model_df=2, return_df=
    True)  # Ajustado

    print(f"Teste de Portmanteau (Asymptotic) {col}:", pt_asymp)
    print("-"*40)
    print(f"Teste de Portmanteau (Adjusted) {col}:", pt_adjusted)
    print("-"*40)

# teste de heterocedasticidade
# H0: Os erros tem vari ncia cte ao longo do tempo
from statsmodels.stats.diagnostic import het_arch

for col in residuos.columns:
    print(f'Teste ARCH para {col}:')
    arch_test = het_arch(residuos[col])
    print(f'LM Statistic: {arch_test[0]:.4f}, p-valor: {arch_test[1]:.4f}')
    print('-' * 40)
# teste de normalidade
# H0: normalidade

from scipy.stats import jarque_bera

for col in residuos.columns:
    jb_test = jarque_bera(residuos[col])  # Teste de Jarque-Bera
    print(f"Teste de Jarque-Bera {col}:\n", jb_test)
    print('-' * 40)

#colocar tabela com p valores
print(residuos)
# Normalidade multivariada
!pip install pingouin
from pingouin import multivariate_normality
mardia_test = multivariate_normality(residuos, alpha=0.05)
print(mardia_test)
# Fazendo previs es
```

```
# df_train.values[-lag_order:] => lag_order    a quantidade de obs passadas
# necess rias para fazer a pr xima previs o. A primeira previs o    feita
# com lag_order obs passadas todas dos daddos coletados ("dados reais")
# Para a segunda previs o, usa-se as  ltimas  lag_order-1 "reais" + a primeira
# prevista na etapa anterior. O comando var_model.forecast faz isso
# automaticamente.

forecast = var_model.forecast(df_diff.values[-lag_order:],
                              steps=n_test)
pred_index = df_diff.index
forecast_diff = pd.DataFrame(forecast,columns=df_train.drop(['Mes'],axis=1).columns)
print(forecast_diff)
print(df_test) #valores observados
irf = var_model.forecast_interval(df_diff.values[-lag_order:], steps=n_test, alpha=0.05)
lower_bound = pd.DataFrame(irf[1], index=df_test.drop(['Mes'],axis=1).index, columns=df_train.
    drop(['Mes'],axis=1).columns)
upper_bound = pd.DataFrame(irf[2], index=df_test.drop(['Mes'],axis=1).index, columns=df_train.
    drop(['Mes'],axis=1).columns)
print(lower_bound)
# Revertendo a diferencia  o para a escala antes da diferencia ao
forecast_df = forecast_diff.copy()

# Lista das colunas que devem ser revertidas (todas menos precipita  o)
cols_reverter = [c for c in forecast_diff.columns if c != "Precipitacao"]

# Obt m o  ltimo  valor real de cada coluna antes da diferencia  o
last_values = df_train[cols_reverter].iloc[-1]

# Aplica revers o s  nessas colunas
forecast_df[cols_reverter] = forecast_diff[cols_reverter].cumsum().add(last_values[
    cols_reverter], axis=1)
print(forecast_diff)
print(forecast_df) #valores preditos
# Revertendo os intervalos para a escala antes da diferencia ao
lower_bound_df = lower_bound.copy()
upper_bound_df = upper_bound.copy()

lower_bound_df[cols_reverter] = lower_bound[cols_reverter].cumsum().add(last_values[
    cols_reverter], axis=1)
upper_bound_df[cols_reverter] = upper_bound[cols_reverter].cumsum().add(last_values[
    cols_reverter], axis=1)

print(lower_bound_df) #limite inferior
print(upper_bound_df) #limite superior
#ver sobre umidade do ar se pode passar de 100%, se nao puder falar que o modelo nao limita o
    limite superior a 100
df_m.columns
# Plotando os resultados na escala original
# Definindo cores espec ficas para cada vari vel
cores = {
    'Precipitacao': 'blue',
    'Temperatura': 'orange',
    'Umidade': 'green',
    'Vento': 'purple'
}

for col in forecast_df.columns:
    fig, ax = plt.subplots(figsize=(10, 4))

    # S rie observada (linha preta)
    ax.plot(df_m.drop(['Mes'], axis=1, errors='ignore').index,
```

```
            df_m.drop(['Mes'], axis=1, errors='ignore')[col],
            label='Hist rico', color='black')

    # S rie prevista
    cor = cores.get(col, 'blue')  # padr o azul se n o estiver no dicion rio
    ax.plot(df_test.index, forecast_df[col],
            label='Previs o', color=cor)

    # Intervalo de confian a
    ax.fill_between(df_test.index, lower_bound_df[col], upper_bound_df[col],
                    color=cor, alpha=0.2)

    # Linha vertical divis o treino/teste
    ax.axvline(df_train.index[-1], color='red', linestyle='dotted')

    # Ajustes visuais
    ax.set_title(f'S rie Temporal - {col}')
    ax.set_ylabel(col)
    ax.set_xlabel('Data')
    ax.legend(loc='upper left')

    plt.show()

  #deixar todos os graficos de todos os modelos padronizados
print(df_m.tail(n=3)) #observado
print(forecast_df) #previsto
print(lower_bound_df)
print(upper_bound_df)
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
import pandas as pd

# Lista para armazenar os resultados
metricas = []

# Loop para cada vari vel prevista
for col in forecast_df.columns:
    y_true = df_test[col]
    y_pred = forecast_df[col]

    # C lculo das m tricas
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)

    # Adicionar ao dicion rio
    metricas.append({
        "Vari vel": col,
        "MSE": mse,
        "RMSE": rmse,
        "MAE": mae
    })

# Criar DataFrame com as m tricas
df_metricas = pd.DataFrame(metricas)

# Exibir tabela
print(df_metricas)
```

C. Código LSTM

Apêndice C.1: Código para implementação do modelo LSTM para a variável precipitação.

```
!pip install --upgrade neuralforecast
!pip install torch --upgrade
!pip install openpyxl
import pandas as pd
from neuralforecast.auto import AutoLSTM
from neuralforecast.core import NeuralForecast
df=pd.read_csv("/content/dados_diarios.txt",sep="\t")
df['Precipitacao'] = pd.to_numeric(df['Precipitacao'], errors='coerce')
df['Umidade'] = pd.to_numeric(df['Umidade'], errors='coerce')
df['Temperatura'] = pd.to_numeric(df['Temperatura'], errors='coerce')
df['Vento'] = pd.to_numeric(df['Vento'], errors='coerce')
df['Data'] = pd.to_datetime(df['Data'], format='%d/%m/%Y')
import numpy as np
df['Precipitacao'] = np.where(df['Precipitacao'] < 0, 0, df['Precipitacao'])
df['Mes'] = df['Data'].dt.to_period('M')
medias_mensais = df[df['Umidade'] > 0].groupby('Mes')['Umidade'].mean()
# Cada linha do df receber  medias_mensais[row['Mes']] se row['Umidade'] < 0 e,
# caso contr rio, continuar  com o valor antigo (row['Umidade'])
df['Umidade'] = df.apply(lambda row: medias_mensais[row['Mes']] if row['Umidade'] < 0 else row
    ['Umidade'], axis=1)
# Agregar os dados por m s
# Ver se melhor dados di rios ou mensais
df_m = df.groupby("Mes").agg({
    "Precipitacao": "mean",   # Soma da precipita  o
    "Temperatura": "mean",    # M dia da temperatura
    "Umidade": "mean",        # M dia da umidade
    "Vento": "mean"           # M dia do vento
}).reset_index()


# Exibir as primeiras linhas do resultado
print(df_m.tail(n=6))
df_m=df_m.rename(columns={'Mes': 'Data'})
print(df_m.tail)
# Definir tamanho do conjunto de treino (80% dos dados)
data_limite=df_m.tail(3).iloc[0,0]
print(data_limite)

train = df_m.loc[df_m['Data'] < data_limite]
valid = df_m.loc[df_m['Data'] >= data_limite]

h = valid.shape[0]
print(h)
import matplotlib.pyplot as plt
from neuralforecast.auto import AutoLSTM
from neuralforecast.losses.pytorch import MAE
import torch.nn as nn
```

```
48
49 #modelo = AutoLSTM(h=h, loss=MAE())
50 auto_lstm_model = AutoLSTM(
51     h=h,               # prever 3 meses    frente
52     loss=MAE()
53 )
54 model = NeuralForecast(models=[auto_lstm_model], freq='M')
55 # Renomeando para 'ds'
56 train = train.rename(columns={'Data': 'ds'})
57 valid = valid.rename(columns={'Data': 'ds'})
58 train['ds'] = train['ds'].dt.to_timestamp()
59 valid['ds'] = valid['ds'].dt.to_timestamp()
60
61 # Renomeando para 'y'
62 train = train.rename(columns={'Precipitacao': 'y'})
63 valid = valid.rename(columns={'Precipitacao': 'y'})
64
65
66 # O dataset pode ter mais que uma  s rie
67 # Neste caso     s   uma
68 train['unique_id'] = 'serie1'
69 valid['unique_id'] = 'serie1'
70
71 train=train[['ds','y','unique_id']]
72 valid=valid[['ds','y','unique_id']]
73
74 train['y']=train['y']
75 # Ajuste do modelo:
76
77 import time # para medir o tempo gasto
78
79 start_time = time.time() # tempo inicial
80 model.fit(train, val_size=3*h)  # Ajusta o modelo
81 tempo = (time.time() - start_time)/60
82 print(f"Tempo gasto: {tempo} minutos")
83
84 def wmape(y_true, y_pred):
85     return np.abs(y_true - y_pred).sum() / np.abs(y_true).sum()
86 p = model.predict().reset_index()
87 print(p)
88
89 p['AutoLSTM'].values
90 valid['AutoLSTM']=p['AutoLSTM'].values
91 auto_lstm_model = model.models[0]
92 results = auto_lstm_model.model.state_dict()
93 # Obter state_dict do modelo
94 state_dict = auto_lstm_model.model.state_dict()
95
96 # Quantidade de camadas LSTM
97 num_layers = len([k for k in state_dict.keys() if 'weight_ih_l' in k])
98
99 # Quantidade de neur nios ocultos por camada
100 hidden_size = state_dict['hist_encoder.weight_hh_l0'].shape[0] // 4  # Divide por 4 devido  s
       gates LSTM
101
102 print(f"N mero de camadas LSTM: {num_layers}")
103 print(f"N mero de neur nios por camada: {hidden_size}")
104 print(state_dict.keys())
105 # Plotando os resultados
106 plt.figure(figsize=(10, 4))
107 # Lembrar de mudar a vari vel AQUI conforme quem    y ######################
```

```
#plt.plot(df_m['Data'].dt.to_timestamp(), df_m['Precipitacao'], label='S rie Temporal Real', linestyle='dashed')
plt.plot(df_m['Data'].dt.to_timestamp(), df_m['Precipitacao'], label='S rie Temporal Real', color='black')
plt.plot(valid['ds'], valid['AutoLSTM'], label='Previs o AutoLSTM', color='blue')
plt.axvline(train['ds'].iloc[-1], color='red', linestyle='dotted')
plt.xlabel('Data')
plt.ylabel('Precipitacao') # E AQUI TAMB M ############
plt.title('Previs o com AutoLSTM')
plt.legend(loc='upper left')
plt.show()
from sklearn.metrics import mean_squared_error, mean_absolute_error

rmse = np.sqrt(mean_squared_error(valid['y'],valid['AutoLSTM']))
mae = mean_absolute_error(valid['y'], valid['AutoLSTM'])
mse = mean_squared_error(valid['y'],valid['AutoLSTM'])
[print(rmse),print(mse),print(mae)]
```