UNIVERSIDADE DA FORÇA AÉREA
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS AEROESPACIAIS

WILLIAM HENRIQUE **INÁCIO**, Cap Eng

Aprendizado de máquina aplicado à previsão de demanda e gestão de estoques de Munições Aeronáuticas

Rio de Janeiro
2025

UNIVERSIDADE DA FORÇA AÉREA
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS AEROESPACIAIS

WILLIAM HENRIQUE **INÁCIO**, Cap Eng

Aprendizado de máquina aplicado à previsão de demanda e gestão de estoques de Munições Aeronáuticas

Dissertação apresentada ao Programa de Pós-Graduação em Ciências Aeroespaciais da Universidade da Força Aérea como requisito para obtenção do Título de Mestre em Ciências Aeroespaciais. Orientador: Prof. Dr. Carlos Cesar de Castro Deonisio.

Rio de Janeiro
2025

Ficha catalográfica elaborada pela Biblioteca da UNIFA

Inácio, William Henrique

I35a

Aprendizado de máquina aplicado à previsão de demanda e gestão de estoques de munição aeronáuticas. / Willian Henrique Inácio. – Rio de Janeiro: Universidade da Força Aérea, 2025.

121 f.: il., enc.

Orientador: Prof. Dr. Carlos César de Castro Deonisio

Dissertação (mestrado) – Universidade da Força Aérea, Rio de Janeiro, 2025.

Referências: f. 81-86

1. previsão de demanda . 2. Aprendizado de maquina. 3. Logística militar. 4. Gestão de estoque. I. Título. II. Deonisio, Carlos César de Castro. III. Universidade da Força Aérea.

CDU: 623.48

WILLIAM HENRIQUE INÁCIO

APRENDIZADO DE MÁQUINA APLICADO À PREVISÃO DE DEMANDA E GESTÃO DE ESTOQUES DE MUNIÇÕES AERONÁUTICAS

Dissertação apresentada ao Programa de Pós-graduação em Ciências Aeroespaciais da Universidade da Força Aérea, como requisito parcial para obtenção do título de Mestre em Ciências Aeroespaciais.

Aprovado por:

gov.br Documento assinado digitalmente
CARLOS CESAR DE CASTRO DEONISIO
Data: 09/10/2025 22:16:22-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. CARLOS CESAR DE CASTRO DEONISIO – UNIFA
Presidente da Banca de Defesa

gov.br Documento assinado digitalmente
FABIO AYRES CARDOSO
Data: 10/10/2025 15:11:11-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. FÁBIO AYRES CARDOSO – UNIFA
Examinador Interno

gov.br Documento assinado digitalmente
NEWTON HIRATA
Data: 09/10/2025 22:39:22-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. NEWTON HIRATA – UNIFA
Examinador Interno

gov.br Documento assinado digitalmente
CARLOS EDUARDO GOMES
Data: 10/10/2025 08:33:30-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. CARLOS EDUARDO GOMES – ESD
Examinador Externo

Rio de Janeiro

OUTUBRO 2025

RESUMO

Embora os estoques assegurem a disponibilidade imediata de produtos, também acarretam custos elevados e riscos de obsolescência. No gerenciamento de estoques, os gestores precisam decidir quanto pedir, quando pedir e como controlar o sistema. No âmbito do Comando da Aeronáutica (COMAER), verificou-se que as previsões de demanda de munições aeronáuticas ainda são realizadas de forma manual e descentralizada, sem suporte analítico automatizado, baseando-se em informações fornecidas pelos órgãos utilizadores. Essa limitação torna o processo lento e pouco preciso frente à elevada variabilidade do contexto militar, podendo resultar em excessos de materiais ou rupturas de suprimento, especialmente diante de atrasos logísticos. Diante desse problema, este trabalho propõe um modelo preditivo de demanda baseado em algoritmos de aprendizado de máquina, desenvolvido a partir de dados históricos do Sistema Integrado de Logística de Materiais e Serviços (SILOMS), referentes ao período de 2009 a 2024. O objetivo é apoiar as decisões logísticas por meio de previsões mais rápidas, precisas e automatizadas, considerando as características estatísticas das séries temporais. A metodologia envolveu a análise exploratória das séries de consumo e a aplicação de um modelo de *ensemble learning*, utilizando o algoritmo *XGBoost* como meta-modelo para combinar previsões individuais por meio de *stacking*. Os resultados demonstraram que o modelo proposto lida adequadamente com séries heterogêneas, produzindo previsões com erros inferiores aos métodos isolados. Concluiu-se que a inteligência artificial apresenta elevado potencial como ferramenta de apoio à decisão no planejamento logístico militar, configurando-se como um campo promissor para aprimorar a gestão de estoques de munições do COMAER.

Palavras-chave: previsão de demanda; aprendizado de máquina; logística militar; gestão de estoques.

ABSTRACT

Although inventories ensure the immediate availability of products, they also involve high costs and obsolescence risks. In inventory management, decision-makers must determine how much to order, when to order, and how to control the system. Within the Brazilian Air Force (COMAER), it was verified that demand forecasts for aeronautical munitions are still carried out manually and in a decentralized manner, without analytical or automated support, relying on information provided by user units. This limitation makes the process slow and imprecise when facing the high variability of the military context, often resulting in material surpluses or supply shortages, especially in cases of logistical delays. To address this issue, this study proposes a predictive demand model based on machine learning algorithms, developed using historical data from the Integrated Logistics System for Materials and Services (SILOMS), covering the period from 2009 to 2024. The objective is to support logistic decision-making by providing faster, more accurate, and automated forecasts that consider the statistical characteristics of the time series. The methodology involved exploratory analysis of the consumption series and the application of an ensemble learning model, using the XGBoost algorithm as a meta-model to combine individual predictions. The results demonstrated that the proposed model effectively handles heterogeneous time series, producing forecasts with lower error rates than individual models. The findings indicate that artificial intelligence has strong potential as a decision-support tool for military logistics planning, representing a promising approach to improving munitions inventory management within COMAER.

Keywords: *demand forecasting; machine learning; military logistics; inventory management.*

LISTA DE ILUSTRAÇÕES

LISTAS DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A gestão eficiente de estoques desempenha um papel crucial na logística militar, constituindo um elemento-chave para o sucesso das operações. A disponibilidade oportuna de recursos, como armamentos, munições e alimentos, é essencial para garantir a capacidade de mobilização e a sustentação do emprego de uma Força Armada.

Gerenciar estoques é um desafio para qualquer organização. No setor de defesa, esse desafio é ainda mais complexo, pois determinados produtos destinam-se a atender ações voltadas a possíveis ameaças, conforme as hipóteses de emprego das Forças Armadas. Além dos estoques de reserva para emergências, as Forças precisam manter disponíveis os materiais necessários para o treinamento contínuo e para as operações rotineiras, como o serviço armado, missões programadas em tempos de paz e operações humanitárias.

As redes de suprimentos militares estão expostas a uma ampla gama de riscos de ruptura, muitos dos quais de difícil previsão. Conforme destacado por Yoho *et al*. (2012), a solução historicamente adotada pelas instituições militares tem sido a concentração de grandes quantidades de material como forma de proteção contra a incerteza. No entanto, níveis elevados de estoque acarretam custos expressivos de armazenagem e aumentam o risco de perdas e obsolescência.

Um planejamento eficiente da dimensão dos estoques de uma Força Armada deve considerar, entre outros fatores, as necessidades projetadas com base nas hipóteses de emprego, as normas internas de renovação de estoques, a disponibilidade orçamentária e as condições logísticas de suprimento. Esse equilíbrio entre segurança operacional e racionalidade de recursos é essencial para assegurar a pronta resposta com a máxima eficiência possível.

Os estoques representam, simultaneamente, um ônus financeiro e uma margem de segurança estratégica. Sua manutenção envolve riscos de deterioração, obsolescência e ocupação de espaço físico, mas garante a disponibilidade imediata de itens críticos em situações de incerteza — uma característica essencial em ambientes militares.

Slack *et al*. (2018) destacam que, no gerenciamento de estoques, os gestores enfrentam três decisões centrais: quanto pedir, quando pedir e como controlar os níveis de estoque. Para que essas decisões sejam eficazes, é imprescindível que a organização seja capaz de prever adequadamente a demanda e definir políticas de estoque compatíveis com os ciclos de suprimento e com os cenários operacionais.

No contexto das munições aeronáuticas, essa tarefa é particularmente desafiadora. Fatores como demandas inopinadas, alta variabilidade, sazonalidade, restrições orçamentárias

e a dificuldade de modelar cenários de conflito tornam o processo de previsão sujeito a incertezas e erros significativos.

Apesar dessas dificuldades, a previsão de demanda é uma atividade estratégica para o êxito das operações militares. Um suprimento adequado de munições é apontado como um dos fatores determinantes para o sucesso operacional, conforme ressaltam Zlatnik e Mares (2018). Prever de forma rápida e precisa a demanda de munições representa, portanto, um problema logístico relevante e atual, conforme argumentam Li *et al*. (2020).

Caron, Bryce e Young (2023) reforçam, ainda, que estoques adequados de munições são indispensáveis para atender demandas futuras, em grande parte incertas. Antecipar essa demanda, realizar pedidos no momento e na quantidade ideais, em um ambiente caracterizado por complexidade e incerteza, constitui um desafio permanente.

A execução dessas previsões e a definição dos parâmetros de renovação de estoques de forma manual configuram uma estratégia propensa a falhas e dispendiosa em tempo e recursos. Nesse contexto, cresce o interesse por ferramentas analíticas e tecnológicas capazes de lidar com grandes volumes de dados, variabilidade e incertezas inerentes aos processos logísticos.

A Inteligência Artificial (IA), especialmente por meio de algoritmos de aprendizado de máquina (*machine learning*), surge como uma solução promissora para apoiar o processo de tomada de decisão logística. Esses algoritmos são capazes de identificar padrões complexos em dados históricos, gerar previsões mais precisas e fornecer suporte analítico ao gestor logístico. Assim, a aplicação da IA pode contribuir para decisões mais fundamentadas, ágeis e alinhadas à realidade operacional das Forças Armadas. Diante desse cenário, este estudo é guiado pelas seguintes questões norteadoras:

Como a utilização de algoritmos de aprendizado de máquina pode melhorar o processo de previsão de demanda de munições no Comando da Aeronáutica (COMAER)?

De que maneira os dados históricos extraídos do Sistema Integrado de Logística de Materiais e Serviços (SILOMS) podem ser utilizados por algoritmos de aprendizado de máquinas para apoiar decisões sobre políticas de estoque mais adequadas às características da demanda?

O presente trabalho caracteriza-se como uma pesquisa aplicada, de natureza quantitativa e abordagem descritiva, voltada à proposição de melhorias no processo de previsão de demanda de munições no âmbito do COMAER. A investigação será conduzida com base em dados históricos extraídos do SILOMS, referentes ao período de 2009 a 2024, abrangendo registros de consumo de munições de emprego aeronáutico. A análise fundamenta-se na aplicação empírica de algoritmos de aprendizado de máquinas como ferramenta de apoio à tomada de

decisão. O desempenho do modelo será avaliado por métricas estatísticas de erro e por sua capacidade de refletir o comportamento real das séries temporais analisadas.

Devido à sensibilidade dos dados, os números brutos de consumo não serão apresentados no trabalho, apenas as medições dos erros em relação aos dados reais e previstos pelo modelo desenvolvido. Em complemento, as munições serão identificadas por nomes genéricos: tipo a, tipo b, tipo c e tipo d.

1.1. CENÁRIO ATUAL

O planejamento e a aquisição de munições no COMAER são regulamentados pela Diretriz do Comando da Aeronáutica (DCA) 135-1, que define os parâmetros para a obtenção e utilização de material bélico, incluindo as munições destinadas à instrução, manutenção operacional, apoio à pesquisa e desenvolvimento e reserva de guerra. Essa diretriz também estabelece os critérios para o cálculo da reserva de guerra dos vetores aéreos, informações classificadas e, portanto, não apresentadas neste trabalho.

Embora o processo seja normativamente estruturado, ele apresenta características que limitam sua eficiência e precisão. A previsão de demanda é baseada em parâmetros fixos de utilização definidos na DCA, conforme o tipo de usuário — aeronaves ou militares —, considerando sua função e o período estimado de uso conforme as hipóteses de emprego.

A partir desses parâmetros, elabora-se um plano de obtenção que projeta as necessidades do COMAER para um horizonte de dois anos. No ciclo subsequente, o plano é ajustado conforme o orçamento disponível, resultando nas quantidades efetivas a serem adquiridas. O ciclo completo de planejamento, desde o levantamento de estoques até o envio do plano final, dura, em média, sete meses.

Outro aspecto relevante é que as previsões de demanda são elaboradas individualmente por cada Órgão utilizador e posteriormente consolidadas pela Diretoria de Material Bélico (DIRMAB). Essa estrutura descentralizada faz com que erros de estimativa cometidos em nível local se propaguem ao longo do processo, acumulando-se na consolidação final e comprometendo a precisão global das previsões do COMAER.

Apesar de sua base normativa consolidada, o processo atual apresenta deficiências estruturais que comprometem a eficiência logística:

a) a demanda é estimada com base em parâmetros fixos, sem uso dos dados históricos de consumo disponíveis no SILOMS;

b) as previsões são descentralizadas e não validadas de forma cruzada, favorecendo a propagação de erros ao longo da cadeia;

c) não há aferição dos erros de previsão, inviabilizando o aprendizado organizacional;

d) o processo é manual, de processamento prolongado, exigindo cerca de sete meses por ciclo;

e) a política de revisão é anual e desconsidera a variabilidade entre itens e tempos de entrega; e

f) o orçamento é a principal restrição decisória, frequentemente sobrepondo-se a critérios técnicos de planejamento.

Essas características revelam uma natureza reativa, que resultam, por vezes, em atrasos na reposição de munições, bem como na propagação sistêmica de erros de previsão, pois eles não são calculados e checados ao longo do processo.

1.2. CONTEXTUALIZAÇÃO E DECLARAÇÃO DO PROBLEMA DE PESQUISA

Para gerenciar seus materiais e serviços, o COMAER utiliza o Sistema Integrado de Logística de Materiais e Serviços (SILOMS), um banco de dados corporativo que registra cadastros, consumos, movimentações de estoque e informações financeiras associadas aos materiais.

Apesar de sua abrangência, o SILOMS não possui mecanismos integrados de previsão de demanda nem algoritmos que auxiliem nas decisões fundamentais de gestão de estoques, como quanto pedir, quando pedir e como controlar os níveis de estoque. Assim, o sistema atua essencialmente como um repositório passivo de informações, sem fornecer suporte analítico para o planejamento logístico.

Consequentemente, as previsões de demanda continuam sendo realizadas de forma manual, descentralizada e desassociada dos dados históricos disponíveis no próprio sistema. O processo depende fortemente da experiência dos planejadores e exige elevado esforço técnico, já que múltiplas variáveis influenciam a demanda, como o tamanho do efetivo, o tipo de missão, a variabilidade de consumo e o tempo de entrega. Além disso, não há monitoramento sistemático dos erros de previsão, o que impede a retroalimentação do processo e o aprendizado organizacional.

Outro fator crítico é que as previsões são elaboradas por diferentes órgãos utilizadores e posteriormente consolidadas pela Diretoria de Material Bélico (DIRMAB), de modo que erros locais de estimativa se propagam ao longo do processo, comprometendo a acurácia global das previsões. A política de revisão adotada é fixa e anual, desconsiderando variáveis de incerteza, como a variabilidade da demanda e do *lead time*.

Como resultado, o modelo vigente de planejamento apresenta baixa eficiência temporal e limitada precisão preditiva, sendo pouco responsivo às variações da demanda e replanejamentos diante de restrições orçamentárias, por exemplo.

Entretanto, o próprio SILOMS contém uma base massiva de dados históricos de demanda ainda subutilizados, que poderiam ser empregados no desenvolvimento de modelos preditivos de demanda. Esses modelos, apoiados por técnicas de aprendizado de máquina, são capazes de identificar padrões, antecipar necessidades e ajustar políticas de estoque de forma dinâmica, promovendo ganhos significativos de eficiência e confiabilidade.

Conforme afirmam Caron, Bryce e Young (2023), o estoque de munições não é homogêneo, dividindo-se entre reservas para uso imediato, treinamento, contingência, experimentação e descarte. Zlatnik e Mares (2018) reforçam que uma gestão eficiente desses estoques deve ser sustentada por cálculos quantitativos confiáveis e um monitoramento contínuo dos erros de previsão, elementos ausentes no modelo atual e abordados pelo método proposto neste trabalho.

Dessa forma, identifica-se uma lacuna de conhecimento: a ausência de métodos quantitativos e automatizados de previsão de demanda de munições capazes de explorar os dados reais do SILOMS e lidar com a variabilidade, característica do ambiente logístico militar analisado. Assim, formula-se o seguinte problema de pesquisa:

Como integrar algoritmos de aprendizado de máquina ao processo de previsão de demanda de munições do COMAER, de modo a aprimorar sua precisão, reduzir o tempo de análise e apoiar a tomada de decisão logística?

1.3. OBJETIVOS

1.3.1. Objetivo geral

Diante do cenário exposto, este trabalho tem como objetivo geral propor um método mais ágil e eficiente para a previsão de demanda de munições aeronáuticas por meio da utilização de algoritmos de aprendizado de máquina aplicados aos dados históricos do SILOMS.

1.3.2. Objetivos específicos

Para alcançar o objetivo geral proposto, o presente estudo foi estruturado em três objetivos específicos complementares:

1. Realizar uma análise exploratória dos dados históricos de consumo de munições aeronáuticas do COMAER, referentes ao período de 2009 a 2024, com o intuito de compreender as principais características das séries temporais, tais como tendência, sazonalidade e variabilidade, de modo a subsidiar a seleção de métodos de previsão mais adequados às particularidades de cada série;

2. Desenvolver um modelo preditivo de demanda que utiliza algoritmos de aprendizado de máquinas, utilizando os dados históricos extraídos do SILOMS, de forma a gerar previsões automáticas e reproduzíveis; e
3. Avaliar o desempenho do modelo por meio de métricas estatísticas de erro.

1.4. JUSTIFICATIVA

A previsão de demanda é uma etapa essencial do gerenciamento de estoques, especialmente em organizações militares, nas quais a disponibilidade imediata de suprimentos é determinante para a prontidão operacional.

O processo atualmente adotado pelo COMAER, embora amparado por normativas consolidadas, baseia-se em procedimentos manuais e descentralizados, com previsões elaboradas por diferentes órgãos e posteriormente consolidadas pela Diretoria de Material Bélico (DIRMAB). Essa estrutura acarreta baixa eficiência temporal, propagação de erros locais e baixa flexibilidade diante de cenários variados, o que limita a precisão das estimativas e o aprendizado organizacional.

Nesse cenário, torna-se uma oportunidade de melhoria incorporar ferramentas analíticas automatizadas que explorem o potencial dos dados históricos disponíveis no SILOMS. Nesse contexto, o emprego de técnicas de aprendizado de máquina oferece uma alternativa promissora e inovadora, ao permitir a detecção de padrões complexos, a geração de previsões mais precisas e a definição de políticas de estoque alinhadas ao comportamento real da demanda.

Do ponto de vista científico, este estudo preenche uma lacuna na literatura nacional ao desenvolver um estudo empírico sobre a utilização de algoritmos de aprendizado de máquinas à previsão de demanda de munições aeronáuticas, utilizando dados reais provenientes de um sistema logístico governamental, abordagem ainda rara no campo da logística militar nacional.

Sob a perspectiva institucional, os resultados esperados têm potencial para reduzir o tempo de planejamento e otimizar a alocação de recursos públicos no processo de aquisição de munições.

Por fim, o caráter inovador deste trabalho reside na integração inédita de técnicas de aprendizado de máquina ao processo de previsão de demanda de munições do COMAER, demonstrando a viabilidade de aplicar métodos inteligência artificial a sistemas logísticos do COMAER.

1.5. DELIMITAÇÃO DA PESQUISA

Esta pesquisa está delimitada ao contexto da logística do COMAER, com foco na previsão de demanda e na gestão de estoques de munições de emprego aeronáutico. O estudo tem caráter aplicado, concentrando-se na análise e modelagem de séries temporais de consumo

a partir de dados históricos extraídos do SILOMS. As munições analisadas referem-se às munições encartuchadas empregadas pelas diversas aeronaves da FAB, como a aeronave F-5M, A-1M, UH-1H, A-29, dentre outras.

O período analisado compreende os anos de 2009 a 2024, correspondente ao intervalo total de registros disponíveis no sistema no momento da pesquisa. As unidades analisadas restringem-se às munições aeronáuticas registradas no SILOMS, não incluindo outros tipos de materiais bélicos, como armamentos terrestres, explosivos ou equipamentos de apoio.

A abordagem metodológica está limitada à aplicação empírica de algoritmos de aprendizado de máquina supervisionado, em combinação com métodos estatísticos clássicos, para geração de previsões de demanda. O modelo proposto é concebido em ambiente experimental, com o objetivo de demonstrar sua viabilidade e potencial de integração futura ao SILOMS.

Adicionalmente, o estudo não aborda aspectos financeiros, contratuais ou estratégicos que podem influenciar no dimensionamento dos estoques, limitando-se às dimensões técnica, analítica e logística da previsão de demanda, baseado em séries temporais de consumo.

Assim, a pesquisa delimita seu escopo temático e temporal à análise da aplicabilidade de técnicas de aprendizado de máquina como ferramenta de apoio à decisão logística no planejamento de necessidades de munições de emprego aeronáutico do COMAER.

1.6. ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em cinco capítulos, conforme descrito a seguir:

O Capítulo 1 apresentou a introdução ao tema, a descrição do cenário atual, a contextualização do problema de pesquisa e a formulação dos objetivos geral e específicos do estudo.

O Capítulo 2 reúne a fundamentação teórica que sustenta o desenvolvimento da pesquisa. São discutidos os principais conceitos relacionados à previsão de demanda, bem como os fundamentos do aprendizado de máquina aplicados ao contexto logístico.

No Capítulo 3, é detalhada a metodologia adotada, incluindo a descrição dos dados utilizados, os passos da modelagem do problema, a seleção e implementação dos algoritmos de aprendizado de máquina, e os procedimentos empregados para validação e avaliação do modelo proposto.

O Capítulo 4 apresenta e analisa os resultados obtidos, contemplando a análise exploratória dos dados históricos, o desempenho preditivo dos modelos aplicados, individualmente e de forma combinada.

Por fim, o Capítulo 5 expõe as considerações finais do estudo, os limites encontrados, e sugestões para trabalhos futuros que possam aprofundar ou expandir a aplicação de técnicas de inteligência artificial no contexto da logística militar.

2 FUNDAMENTAÇÃO TEÓRICA

2.1. PREVISÃO DE DEMANDA

As previsões de demanda constituem a base de grande parte das decisões da cadeia de suprimentos. Conforme destacam Chopra e Meindl (2011), praticamente todas as decisões de produção, aquisição, estocagem e transporte dependem de estimativas, ainda que imperfeitas, sobre o comportamento futuro da demanda. No contexto militar, esse papel é ainda mais crítico, pois a falta de material pode comprometer a prontidão operacional, enquanto o excesso se traduz em imobilização de recursos escassos e riscos adicionais (segurança, obsolescência, vencimento).

De forma geral, a previsão de demanda envolve reconhecer padrões em séries históricas e projetá-los para o futuro ou identificar fatores causais que afetam a demanda e extrapolar seu efeito, conforme definem Ackermann e Sellitto (2022). Para Chopra e Meindl (2011), algumas características são centrais para compreender o papel das previsões:

a) as previsões são sempre imprecisas: Toda previsão possui um erro, o qual deve ser uma entrada decisiva para a maioria das decisões da cadeia de suprimento. Entretanto, conforme pontua o autor, a maioria das empresas não mantém qualquer estimativa de erro de previsão, o que pode levar a decisões imprecisas;

b) previsões de longo prazo normalmente são menos precisas do que as de curto prazo;

c) previsões agregadas, normalmente, são mais exatas do que as desagregadas: Quanto maior a agregação, menos erro tende a ter a previsão. Por exemplo, a previsão de uma receita anual de uma empresa tende a ser mais precisa do que a previsão de receita de um produto específico dessa empresa; e

d) independentemente do método de previsão, sempre há um elemento aleatório que não pode ser explicado por padrões históricos: A demanda observada pode ser dividida em um componente sistemático e um componente aleatório. O componente sistemático mede o valor esperado da demanda, a tendência e a sazonalidade, enquanto o componente aleatório é a parte que se desvia do componente sistemático e que não pode ter a direção prevista.

Esses princípios são particularmente relevantes para o problema deste trabalho, que trata da previsão de demanda de munições de emprego aeronáutico no COMAER. Nessa aplicação, é esperado que a demanda seja:

a) intermitente, com diversos períodos de consumo nulo (zeros) intercalados com picos;

b) afetada por planos de treinamento, exercícios, operações e restrições orçamentárias, portanto, dotada de forte influência sazonal; e

c) sujeita a variabilidade elevada, tanto em magnitude quanto em frequência.

Tais características podem ser facilmente verificadas através de uma análise exploratória dos dados históricos, com a qual é possível extrair as características como tendência, sazonalidade, variação, dentre outras.

A literatura recente mostra que a previsão de demanda em ambientes militares é um tema específico e em expansão. Choi e Suh (2020) comparam diferentes técnicas de data *mining* para previsão de demanda de sobressalentes em aeronaves militares, evidenciando que a escolha adequada de modelos melhora a disponibilidade do sistema e reduz estoques excessivos. Lee e Kim (2018) e Kim, Hwang e Doh (2023) propõem modelos preditivos para demanda de peças em logística militar, incorporando técnicas de aprendizado de máquinas e ajustes de escala de dados para lidar com séries esparsas e assimétricas.

Especificamente no campo de munições, Zlatnik e Mares (2018) discutem métodos para cálculo de níveis eficazes de estoque de munição em Forças Armadas, analisando abordagens como o *Target-Oriented Method* (TOM) e o *Level of Effort* (LoE) para dimensionamento de estoques. Caron, Bryce e Young (2023) introduzem um modelo de simulação de eventos discretos (FEAST – *Future Event Ammunition Stockpile Tool*) para explorar a dinâmica de estoques de munição sob diferentes políticas de reabastecimento, orçamentos e cenários de treinamento. Mais recentemente, Kurtay, Altundaş e TAS (2025) propõem um modelo de previsão de demanda de munições integrado ao projeto da rede de distribuição para uma unidade militar, evidenciando que decisões conjuntas de previsão e distribuição podem reduzir custos e riscos de ruptura.

Os estudos mencionados evidenciam que:

a) previsão de demanda em ambiente militar é um problema específico, com características próprias (criticidade, incerteza, intermitência);

b) munições e sobressalentes militares são objetos típicos de aplicação de modelos de previsão; e

c) previsão de demanda na logística militar é um tema relevante e atual.

O presente trabalho se insere nesse contexto ao propor um sistema de previsão para munições de emprego aeronáutico, integrando modelos estatísticos e algoritmos de aprendizado de máquinas para melhoria do processo de previsão de demanda do COMAER.

De acordo com Tamashiro *et al*. (2024), atualmente, existem diversos modelos para a previsão de demanda, desde os mais tradicionais, tais como a suavização exponencial e média móvel, até modelos mais sofisticados, como o aprendizado de máquina.

No presente trabalho, por se concentrar em previsões de munições, que são itens de consumo, não serão abordadas metodologias utilizadas para previsão de demanda de itens reparáveis, que são modelos multiescalão, como o *Multi-Echelon Technique for Recoverable Item Control* (METRIC) e o VARI METRI, que é uma variação do METRIC.

2.2. INTELIGÊNCIA ARTIFICIAL NA LOGÍSTICA MILITAR

A inteligência artificial é um campo da ciência da computação que estuda o desenvolvimento de sistemas capazes de executar tarefas que normalmente requereriam inteligência humana, como aprendizado, percepção, raciocínio e tomada de decisão, apontam Makridakis *et al.* (2020). A IA está revolucionando vários aspectos da vida, automatizando tarefas e repensando a análise de dados e os processos de tomada de decisão (*Artificial Intelligence For The Real World*, 2023). No campo logístico, a IA pode melhorar significativamente a gestão de logística inteligente, conforme apontam Woschank, Rauch e Zsifkovits (2020).

No contexto da logística militar, essas mesmas capacidades podem ser exploradas para apoiar o planejamento de estoques estratégicos, a previsão de consumo em operações e o uso eficiente de recursos escassos.

As principais características relevantes da IA para a logística militar incluem:

a) capacidade de tratar grandes volumes de dados históricos e heterogêneos, provenientes de múltiplas fontes, o que permite modelar o sistema logístico de forma mais realista (Flores-García *et al*., 2025; Chen *et a*l., 2024);

b) habilidade de capturar relações não lineares e interações entre variáveis, que frequentemente não são bem representados por modelos lineares clássicos (Masini; Medeiros e Mendes, 2023; Makridakis *et al*. , 2018a); e

c) possibilidade de atualização contínua dos modelos, incorporando novos dados e ajustando parâmetros ao longo do tempo, o que é crucial em ambientes operacionais sujeitos a mudanças estruturais e novos cenários de emprego (Flores-García *et al*., 2024).

Apesar de ser um campo emergente e com elevado potencial, a implantação de soluções baseadas em Inteligência Artificial (IA) no contexto militar exige cautela e discernimento estratégico. O uso da IA como instrumento de apoio à decisão deve ser compreendido como complementar à análise humana, e não como substitutivo integral. Ainda assim, algoritmos de

aprendizado de máquina demonstram eficácia significativa na análise de grandes volumes de dados históricos, extraindo padrões e regularidades relevantes para o apoio à decisão. Sua capacidade de aprender e prever cenários possíveis através de análise de grandes volumes de dados proporciona aprendizado automático, o que os torna especialmente úteis em sistemas inteligentes de suporte à decisão, conforme destacam Galán; Carrasco e LaTorre (2022).

Makridakis *et al*. (2020) destacam que a IA, quando aplicada a contextos decisórios, permite incorporar múltiplas fontes de incerteza nos modelos analíticos, resultando em decisões mais robustas e fundamentadas. Meerveld *et al*. (2023), por sua vez, enfatizam que a adoção de IA no domínio militar contribui para elevar a qualidade e a velocidade das decisões, sobretudo em ambientes complexos e dinâmicos.

Diante do exposto, importa destacar que o uso da IA como ferramenta de suporte decisório não implica a automatização irrestrita das decisões, mas sim o fortalecimento da capacidade analítica do gestor, permitindo decisões mais ágeis, fundamentadas e transparentes. Isso é particularmente relevante em instituições públicas, como o COMAER, onde o processo decisório deve conciliar eficiência operacional, conformidade normativa e transparência.

Apesar de emergente, a implantação da IA no processo decisório apresenta desafios importantes. Pesquisas recentes pontuaram fragilidades na operacionalização da IA, como a necessidade de existência de marcos regulatórios claros e robustos para garantir transparência, equidade e proteção de direitos fundamentais, especialmente diante de riscos como viés dos algorítmicos, falta de transparência do funcionamento dos códigos e a proteção de dados pessoais, como pode ser visto nos trabalhos de Obinna e Kess-Momoh (2024a), Obinna e Kess-Momoh (2024b), Obinna e Kess-Momoh (2024c) e Hickok (2024). Dessa forma, o desenvolvimento de capacidades técnicas e a capacitação contínua dos gestores são essenciais para que possam avaliar, monitorar e supervisionar sistemas de IA de forma responsável, conforme pontua Von e Abrahamsson (2022).

2.2.1. Aprendizado de máquinas

Um dos ramos mais promissores da Inteligência Artificial é o aprendizado de máquina, um campo que desenvolve algoritmos capazes de aprender padrões a partir de grandes volumes de dados, realizar previsões e melhorar seu desempenho com o tempo (IBM, 2021; Popenici e Kerr, 2017). De acordo com Tsolaki *et al*. (2023), o aprendizado de máquina está revolucionando a logística, melhorando o tempo de chegada, a previsão de demanda, a previsão do fluxo de tráfego, o roteamento de veículos e a detecção de anomalias.

No contexto logístico, o aprendizado de máquinas tem sido amplamente utilizado em tarefas como previsão de demanda, dimensionamento de estoques, manutenção, roteirização de entregas e detecção de desvios em cadeias de suprimentos. De acordo com Flores-García *et al*., 2024), o aprendizado de máquina pode aprimorar os recursos de logística em ambientes dinâmicos, melhorando o tempo, a qualidade, a sustentabilidade e o custo.

A aplicação de inteligência artificial nas Forças Armadas brasileiras já é uma realidade institucional. Brasil (2021), identificou que a Tecnologia Emergente e Disruptiva de Inteligência Artificial e a análise preditiva/prospectiva possuem grande capacidade de impactar as atividades militares futuras, em tempo de paz relativa ou de conflito. Nesse contexto, o Exército Brasileiro estabeleceu diretrizes estratégicas para a adoção de IA por meio da Portaria nº 1.318/2024, com foco em implantação estruturada e os seus usos operacionais.

Nesse mesmo caminho, a Marinha do Brasil definiu, por meio da Portaria nº 22/EMA, de 4 de fevereiro de 2021, os chamados “Eixos Estruturantes de IA”, que incluem a governança, regulamentação, ética e fomento à pesquisa e desenvolvimento em IA (Brasil, 2022). Por sua vez, a Força Aérea Brasileira (FAB) instituiu o primeiro Laboratório de Inteligência Artificial em 2024. Esse laboratório tem o objetivo de transformar operações e aprimorar a eficiência em áreas essenciais, como a tomada de decisões, manutenção de aeronaves, simulação de cenários e otimização logística (Força Aérea Brasileira, 2024). Essas iniciativas demonstram que o uso de IA em contextos militares nacionais não é apenas uma tendência, mas uma política concreta em implantação.

No entanto, apesar do elevado potencial de tal aplicação na logística militar, o desenvolvimento de estudos e a utilização da IA em toda a área de defesa ainda são prematuros, conforme pontuam Hadlington *et al*.(2023). Apesar da importância emergente dos sistemas de IA em iniciativas de modernização da defesa, ainda há carência de estudos empíricos ou teóricos sobre o tema, pontuam Jensen, Whyte e Cuomo (2020). A implementação da IA no setor militar enfrenta alguns desafios. Meerveld e Lindelauf (2025) ressaltam, ainda, que um dos desafios importantes é saber como integrar todos os dados relevantes no processo de tomada de decisão.

O presente trabalho se insere nessa lacuna de estudos empíricos, com o compromisso de contribuir no desafio citado por Meerveld e Lindelauf (2025), aplicado a um caso específico da logística militar do COMAER, através da aplicação de algoritmos de aprendizado de máquinas para realizar análises logísticas com dados extraídos do SILOMS e dar suporte na tomada de decisões logísticas.

2.2.2. IA aplicada no gerenciamento de estoques

É crescente a aplicação de aprendizagem de máquina na gestão de estoques, com o objetivo de aprimorar a eficiência operacional e a precisão preditiva, sobretudo por superar as limitações dos modelos tradicionais. Li *et al*. (2020) demonstraram que, ao utilizar métodos de previsão combinada com técnicas de aprendizado de máquina, é possível explorar de forma abrangente as vantagens de diferentes modelos individuais, adaptando-os a contextos complexos e elevando a acurácia e a adaptabilidade das previsões de consumo de munições.

Kosasih e Brintrup (2022) investigaram a aplicação do aprendizado por reforço para a otimização simultânea dos níveis de estoque de segurança e das quantidades de pedido em cadeias de suprimentos, evidenciando que essa abordagem permite modelar comportamentos de aquisição mais complexos.

Rimélé *et al.* (2021) propuseram uma política de armazenamento dinâmica com base em agentes de *deep learning* para sistemas móveis de atendimento em armazéns de comércio eletrônico. Os resultados indicaram uma redução de até 14% no tempo médio de deslocamento, em comparação com regras tradicionais, demonstrando a superioridade da IA na construção de políticas de estocagem mais eficientes.

No contexto do varejo, Venâncio e Bueno (2023) compararam a gestão tradicional de estoques com abordagens que utilizam inteligência artificial em um supermercado de pequeno porte. A implementação de IA resultou em decisões mais assertivas, otimização de recursos e melhoria do nível de serviço ao cliente.

Campos e Farina (2024), por meio de uma revisão sistemática, reforçaram o potencial das técnicas de IA na logística, especialmente na previsão de demanda e na gestão de estoques, embora tenham identificado desafios relacionados à integração e operacionalização dessas soluções.

No mesmo sentido, Sousa (2024) examinou como algoritmos de aprendizado de máquina podem ser empregados na cadeia de suprimentos com foco em análise preditiva, concluindo que essas técnicas promovem melhorias significativas na acurácia das previsões, redução de custos operacionais e suporte à tomada de decisões mais eficazes. A revisão de trabalhos sobre o assunto deixa evidente a capacidade de melhoria que o uso de algoritmos inteligentes promove na realização de previsões.

Apesar da abundância de métodos gerais de previsão de demanda, poucos trabalhos abordam explicitamente o contexto de munições militares. Kurtay, Altundaş e Tay (2025) em seu estudo sobre distribuição para munições de uma unidade militar destacaram que a grande maioria dos estudos de demanda de munições realizados é determinística, com um único

depósito e voltada apenas para a distribuição. Dos 91 artigos analisados por eles, apenas um estudo fez uma previsão de demanda, mas ele se concentrou apenas no problema de distribuição. Os autores também sugerem a utilização de diferentes métodos, como aprendizado de máquina, para estudos futuros.

Apesar dos avanços recentes nas aplicações de aprendizado de máquina à previsão de demanda, observa-se uma notável escassez de estudos voltados especificamente ao contexto militar e, em particular, ao consumo de munições. As investigações identificadas na literatura concentram-se majoritariamente em estoques civis ou em sobressalentes de uso militar, como em Kim *et al*. (2023), enquanto pesquisas diretamente relacionadas à previsão de demanda de munições ainda são pontuais, como Li *et al.* (2021) e Kurtay, Altundaş e Tay (2025), que exploram abordagens bayesianas e analíticas tradicionais, sem o emprego de algoritmos modernos de aprendizado de máquina.

Tal lacuna evidencia a oportunidade de ampliar o escopo de conhecimento sobre o tema, especialmente considerando as especificidades das séries temporais de consumo de munições, caracterizadas por intermitência e alta variabilidade. Nesse sentido, o presente trabalho busca contribuir ao propor um modelo híbrido de previsão de demanda baseado na integração entre métodos estatísticos e algoritmos de aprendizado de máquina, adaptado às particularidades da logística militar do COMAER, oferecendo uma alternativa analítica mais robusta e aderente à realidade operacional das Forças Armadas.

2.3. CARACTERÍSTICAS DOS ESTOQUES MILITARES

Os estoques de produtos militares têm como finalidade atender a diferentes tipos de demanda, desde as necessidades de treinamento até aquelas oriundas de operações reais, como conflitos armados. Conforme destacado por Slack *et al*. (2018), os estoques existem para compensar a defasagem temporal entre suprimento e demanda. Em outras palavras, a necessidade de um material (demanda) e sua disponibilidade (suprimento) não ocorrem, via de regra, de forma simultânea. Assim, manter materiais em estoque é fundamental para suprir variações tanto na frequência quanto na quantidade demandada.

Ballou (2006) ressalta que os estoques proporcionam diversos benefícios, como: melhor nível de atendimento ao cliente interno, produção mais regular, economias de escala nas aquisições, proteção contra incertezas na demanda e no tempo de reposição, além de funcionarem como reserva estratégica para contingências.

Contudo, manter estoques também apresenta desvantagens significativas. Segundo Slack *et al*. (2018), os principais pontos negativos incluem: (*a*) imobilização de capital de giro; (*b*) risco de obsolescência; (*c*) necessidade de espaço físico para armazenamento; (*d*) possíveis

exigências especiais de acondicionamento; e (*e*) custos logísticos relacionados à infraestrutura de armazenagem.

Diante disso, torna-se evidente que, para atender a demandas futuras — muitas vezes incertas — as Forças Armadas necessitam manter estoques estratégicos. Esses estoques podem ser organizados segundo diferentes categorias, conforme suas finalidades operacionais. De acordo com Bean *et al*. (2016), os estoques militares podem ser classificados em:

a) Estoque de treinamento: destinado ao suporte das atividades de preparação da tropa em tempos de paz;

b) Estoque operacional: voltado ao atendimento de operações militares de rotina, como missões de Garantia da Lei e da Ordem (GLO) ou de imposição da paz;

c) Reserva de guerra: constituída para situações extremas e de baixa probabilidade, como conflitos armados de grande escala — sendo considerada o estoque de segurança do sistema logístico militar; e

d) Estoque aguardando destruição: materiais obsoletos, danificados, vencidos ou excedentes, aguardando descarte.

Os estoques de treinamento e parte do estoque operacional atendem a demandas relativamente previsíveis. Por sua vez, a reserva de guerra destina-se a situações imprevisíveis e de alto impacto, sendo, portanto, marcada por elevado grau de incerteza e dificuldade de estimativa de consumo. Observa-se que os diferentes tipos de estoque buscam atender a perfis distintos de demanda. Considerando isso, Bean *et al*. (2016) propõem a segmentação dos estoques militares em três categorias:

a) Categoria A: estoques de fácil previsão, destinados a eventos recorrentes e de alta previsibilidade;

b) Categoria B: estoques voltados a eventos menos prováveis, com demanda moderadamente incerta;

c) Categoria C: estoques associados a eventos extremos e de baixa previsibilidade, mas de grande impacto — correspondendo à reserva de guerra.

A distinção entre essas categorias está fundamentada nas diferentes características estatísticas da demanda em cada contexto. Como apontam Bean *et al*. (2016), o principal objetivo de um modelo de gestão de estoques militares é definir os níveis ótimos de suprimento diante da incerteza, minimizando simultaneamente os custos totais, a incidência de faltas e o nível mínimo necessário de segurança.

Determinar os níveis ideais de estoque é essencial para qualquer organização. Segundo Ross (2015), os estoques podem representar de 40% a 80% do valor das vendas em uma

empresa típica. Apenas os custos de manutenção *(holding cost*) desses estoques chegam a aproximadamente 25% do seu valor anual, aponta Waters (2003). Isso reforça a importância de mantê-los no menor nível possível, sem comprometer a capacidade de atendimento das demandas.

Nesse contexto, a definição dos níveis ótimos de estoque dependerá diretamente da capacidade de realizar previsões da forma mais assertiva possível. O presente trabalho propõe a utilização de técnicas de inteligência artificial como suporte à decisão, buscando-se contribuir para uma gestão mais eficiente dos estoques da categoria A e B, conforme divisão feita por Bean *et al*. (2016).

2.4. MÉTODOS DE PREVISÃO

A definição precisa da demanda futura constitui um dos pilares do planejamento logístico e da gestão eficiente de estoques. Nesse contexto, os métodos de previsão assumem papel central, ao permitirem estimar quantitativamente as necessidades futuras de consumo com base em padrões históricos de comportamento. Diversos modelos foram desenvolvidos a partir desses métodos, variando desde abordagens estatísticas clássicas, como médias móveis e suavização exponencial, até técnicas mais sofisticadas que incorporam tendências, sazonalidades e dependências temporais complexas.

A escolha do método e do modelo adequados depende diretamente das características da série temporal, como estabilidade, variabilidade e presença de períodos sem demanda, sendo essa decisão determinante para a acurácia das previsões e, por consequência, para a eficiência da cadeia de suprimentos. No âmbito da logística militar, a seleção criteriosa dos métodos de previsão torna-se ainda mais relevante, sobretudo se considerar as possíveis consequências da falta do estoque, que pode ser a perda de um conflito armado.

2.4.1. Métodos estatísticos clássicos

São métodos que utilizam ferramentas e conceitos estatísticos para estimar o comportamento futuro de variáveis, como vendas, demanda, preços, dentre outros.

2.4.1.1. Média Móvel Simples

É o método mais simples e rotineiramente utilizado no cotidiano, em diversas áreas. De acordo com Chopra e Meindls (2011), é utilizado quando a demanda não possui tendência ou sazonalidade.

Conforme pode ser verificado na Equação 1, este método consiste em calcular a demanda média (D) sobre os N períodos (t) mais recentes.

$$D = \frac{\left(D_t + D_{(t-1)} + D_{(t-N+1)}\right)}{N} \quad (1)$$

2.4.1.2. Suavização Exponencial Simples

Atribui pesos exponencialmente decrescentes aos dados passados, dando mais importância aos valores mais recentes, através da atribuição de valores do coeficiente de atenuação (α) mais elevados para os períodos mais recentes. É indicada para séries temporais sem tendência ou sazonalidade, segundo afirmam Chopra e Meindls (2011).

Assim, a estimativa para o período subsequente pode ser estimada conforme a Equação 2:

$$P_{(t)} = \alpha * D_{(t-1)} + (1 - \alpha)\mathrm{P}_{(\mathrm{t-1})} \quad (2)$$

Em que:

P(t): Previsão para o período t;

α: Constante de suavização (0 <α ≤1);

$D_{(t-1)}$: Valor observado no período anterior (t-1); e

$P_{(t-1)}$: Previsão para o período anterior (t−1).

2.4.1.3. Holt-Winters (Suavização Exponencial Tripla)

O método de Holt-Winters, também conhecido como suavização exponencial tripla, é amplamente utilizado para a previsão de séries temporais que exibem componentes de nível, tendência e sazonalidade. Este método estende a suavização exponencial simples ao incorporar ajustes para tendência e sazonalidade, permitindo previsões mais precisas em dados com padrões sazonais.

Existem duas variações principais do método de Holt-Winters: o modelo aditivo e o modelo multiplicativo. A escolha entre eles depende da natureza da sazonalidade presente nos dados.

2.4.1.3.1. Modelo Aditivo

O modelo aditivo é apropriado quando a amplitude da sazonalidade permanece constante ao longo do tempo. Nesta metodologia, a série Z é decomposta em fatores de sazonalidade, tendência e nível. Seja uma série sazonal, de período t, formada pela soma do nível (L), tendência (T), um fator sazonal (S) e um erro aleatório (€), de acordo com Veríssimo *et al*. (2012) e Ackermann e Sellitto (2022) a previsão é realizada conforme apresentado na Equação 3:

$$Z_t = L_t + T_t + S_t + € \quad (3)$$

Por consequência, a função de previsão para os tempos futuros é dada pela Equação 4:

$$Z_{t+n} = L_t + nT_t + S_{t-s+n} \quad (4)$$

Em que Z_{t+n} é a previsão para n períodos à frente (t+n).

Por sua vez, o nível (L), a tendência (T) da série no período atual e, os valores do fator sazonal (S) correspondente ao último período de sazonalidade, são calculados conforme definido nas Equações 5, 6 e 7, conforme Veríssimo *et al*. (2012) e Ackermann e Sellitto (2022):

$$L_t = \alpha(Z_t - S_{t-s}) + (1-\alpha)(L_{t-1} + T_{t-1}); 0 < \alpha < 1 \quad (5)$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1};\ 0 < \beta < 1 \quad (6)$$

$$S_t = \gamma(Z_t - L_t) + (1-\gamma)S_{t-s}; 0 < \gamma < 1 \quad (7)$$

a, β e g são as constantes de suavização que controlam o peso relativo ao nível, a tendência e a sazonalidade, respectivamente, daí vem a denominação de suavização tripla, pois utilizam esses três coeficientes de suavização.

2.4.1.3.2. Modelo Multiplicativo

O modelo multiplicativo é adequado quando a amplitude da sazonalidade varia proporcionalmente ao nível da série temporal. Este modelo de série sazonal, de período S, tem fator sazonal multiplicativo e a tendência aditiva, conforme a Equação 8, conforme Veríssimo *et al*. (2012) e Ackermann e Sellitto (2022):

$$Z_t = L_t S_t + T_t + €\quad (8)$$

Assim, a função de previsão para os tempos futuros é calculada conforme a Equação 9:

$$Z_{t+n} = (L_t + nT_{t)}S_{t-s+n} \quad (9)$$

Em que *n* é o número de períodos a frente da previsão.

Finalmente, o nível (L), a tendência (T) e os valores do fator sazonal (S) são expressos pelas Equações 10, 11 e 12, conforme Veríssimo *et al*. (2012) e Ackermann e Sellitto (2022):

$$L_t = \alpha\left(\frac{Z_t}{S_{t-s}}\right) + (1-\alpha)(L_{t-1} + T_{t-1});\ 0 < \alpha < 1 \quad (10)$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1};\ 0 < \beta < 1 \quad (11)$$

$$S_t = \gamma\left(\frac{Z_t}{L_t}\right) + (1-\gamma)S_{t-s};\ 0 < \gamma < 1 \quad (12)$$

Em termos práticos, o modelo multiplicativo é preferível em relação ao aditivo quando a sazonalidade aumentar ou diminuir com o nível da série.

2.4.1.4. Modelo Autoregressivo

Um modelo autoregressivo (AR) relaciona o valor atual da série com uma soma ponderada de valores passados, sendo expressa pela Equação 13, conforme Hyndman e Athanasopoulos (2018):

$$yt = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_p y_{t-p} + \varepsilon_t \quad (13)$$

Em que $\phi_1, \phi_2\, e\, \phi_3$ são conjuntos finitos de parâmetros de peso, em um modelo autorregressivo de ordem “p”. Este modelo é, normalmente, chamado de modelo AR(p).

De acordo com Hyndman e Athanasopoulos (2018), normalmente, modelos autorregressivos são restritos a dados estacionários. Nestas condições os valores das variáveis ficam restritas a alguns valores, conforme apresentado nas Equações 14, 15 e 16.

Para um modelo AR(p=1):

$$-1 < \phi_1 < 1 \tag{14}$$

Para um modelo AR(p=2):

$$-1 < \phi_2 < 1 \tag{15}$$

$$\phi_1 + \phi_2 < 1;\ \ \phi_2 - \phi_1 < 1 \tag{16}$$

O termo εt é o chamado ruído branco, que é uma sequência de dados aleatórios com média zero, variância constante e inexistência de autocorrelação. Como não há correlação entre os valores de εt em diferentes períodos, os erros são independentes entre si, ou seja, ele é considerado puramente aleatório e, portanto, não deve conter nenhuma estrutura ou padrão adicional. Caso esta variável apresente autocorrelação significativa, isso indica que o modelo não foi bem ajustado ou há padrões remanescentes nos dados.

Modelos AR são particularmente adequados para séries temporais que apresentam comportamento estacionário, ou seja, média e variância constantes ao longo do tempo, e em que os valores atuais dependem significativamente dos seus próprios valores passados. Eles apresentam boa capacidade preditiva em séries que não apresentam tendência ou sazonalidade acentuadas. No entanto, sua aplicação é limitada a contextos sensíveis a *outliers*, sendo pouco eficaz para capturar padrões não lineares ou variações estruturais na série.

2.4.1.5. Modelo Autorregressivo de Médias Móveis (ARMA)

O modelo de média móvel modela o valor atual como uma combinação linear dos erros de previsão passados. Ele combina um modelo autorregressivo (AR), que se baseia em valores passados da própria série temporal para prever valores futuros, com média móvel (MA), o qual modela o erro da série como uma média ponderada de erros passados. Pode ser modelado utilizando a Equação 17, conforme Hyndman e Athanasopoulos (2018)):

$$yt = c + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \theta_q\varepsilon_{t-q} \tag{17}$$

Em que $\Theta_1, \Theta_2\ e\ \Theta_q$ são conjuntos finitos de parâmetros de peso, em um modelo de ordem “q”. O termo εt também é chamado de ruído branco.

O modelo ARMA é adequado para séries temporais estacionárias**.**

2.4.1.6. Diferenciação

Alguns modelos de previsão são aplicáveis apenas a séries estacionárias. Entretanto, é possível transformar uma série não estacionária em uma série estacionária (sem tendência e

sazonalidade), através da diferenciação, que se trata de computar a diferença entre consecutivas observações. A diferenciação pode ajudar a estabilizar a média de uma série cronológica, removendo as alterações no nível de uma série cronológica e, portanto, eliminando (ou reduzindo) a tendência e a sazonalidade, conforme explicam Hyndman e Athanasopoulos (2018).

A diferenciação é aplicada para tornar a série estacionária, removendo tendências (Equação 18):

$$y't = \Delta^d y_t \tag{18}$$

Em que *d* é o número de diferenciações aplicadas para tornar a série estacionária.

2.4.1.7. *Auto Regressive Integrated Moving Average* (ARIMA)

Através da combinação da diferenciação com autorregressão e o modelo de média móvel, obtém-se um modelo ARIMA, um modelo estatístico que combina componentes autorregressivos e de médias móveis, adequado para séries temporais estacionárias, de acordo com Ackermann e Sellitto (2022).

Enquanto os modelos de suavização exponencial são baseados em uma descrição da tendência e sazonalidade nos dados, os modelos ARIMA descrevem as autocorrelações nos dados. A formulação geral do ARIMA combina as partes autorregressiva, integrada e de média móvel, conforme ilustrado na Equação 19:

$$y't = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{19}$$

Em que *y't* é a série diferenciada, *p* é a ordem da parte autorregressiva, *d* é o grau da diferenciação e *q* é a ordem da parte do modelo de média móvel. Assim o modelo ARIMA é designado como ARIMA (p,d,q). Os modelos de autorregressão e média móvel são casos especiais do ARIMA. ARIMA (p,0,0) resulta em um modelo de autorregressão, enquanto ARIMA (0,0,q) em um modelo de média móvel. Portanto, um modelo ARIMA pode ser utilizado para representar esses modelos anteriormente citados, o que ressalta sua característica de versatilidade.

Estimar os valores de p, d e q pode ser um grande desafio, sobretudo em um conjunto extenso de dados. Para auxílio, normalmente, utilizam-se métodos automáticos para seleção dos valores. Os métodos automáticos utilizam algoritmos e funções de otimização para selecionar os valores ótimos de p, d e q, testando diversas combinações de parâmetros para escolher o modelo que minimiza critérios estatísticos.

Há também algumas bibliotecas de pacotes estatísticos como R (pacote *forecast*) e Python (biblioteca *statsmodels*) que oferecem funções automáticas, como auto.arima(), que

testam diferentes combinações e escolhem o melhor modelo com base nos critérios estabelecidos.

2.4.1.8. *Seasonal Auto Regressive Integrated Moving Average* (SARIMA)

O SARIMA é um modelo estatístico utilizado para a previsão de séries temporais que exibem fortes padrões sazonais. Ele estende o modelo ARIMA ao incorporar componentes que capturam a sazonalidade dos dados, permitindo uma modelagem mais precisa de fenômenos periódicos.

Muitas séries temporais apresentam padrões repetitivos que aparecem regularmente a cada intervalo de tempo. Para lidar com as séries que possuem a sazonalidade como característica importante, Box e Jenkings (1976) modificaram o modelo ARIMA e definiram o modelo SARIMA(p,d,q)(P,D,Q), conforme destaca Walter *et al*. (2013).

De acordo com Kwarteng e Andreevich (2024), as variáveis p,d e q são as mesmas do modelo ARIMA, enquanto as variáveis P, D e Q são:

a) P: Representa a ordem do autorregressivo sazonal, indicando os valores defasados no componente sazonal;

b) D: Este parâmetro denota a ordem sazonal de integração, semelhante a *d*, mas aplicada ao componente sazonalmente; e

c) Q: denota a ordem do processo da média móvel sazonal, capturando os termos de erro defasados no componente sazonal.

A formulação geral do modelo é expressa conforme Equação 20:

$$\Phi_P(B^s)\Phi_P(B)(1-B)^d(1-B^s)^D yt = \Theta_Q(B^s)\theta_P(B)\varepsilon_t \qquad (20)$$

Em que:

yt é a série temporal no tempo t; $\Phi_P(B)$ é o polinômio autorregressivo não sazonal de ordem p; $\Phi_P(B^s)$ é o polinômio autorregressivo sazonal de ordem P; $\theta_P(B)$ é o polinômio de média móvel não sazonal de ordem q; $\Theta_Q(\mathrm{B}^s)$ é o polinômio de média móvel sazonal de ordem Q; $(1-B)^d$ é a diferenciação não sazonal de ordem d; $(1-B^s)^D$ é a diferenciação sazonal de ordem D e ε_t é o ruído branco.

2.4.1.9. Modelo Erro, Tendência, Sazonalidade (ETS)

O método ETS é uma abordagem para a previsão de séries temporais, especialmente eficaz em dados que apresentam componentes de tendência e sazonalidade. Este método utiliza a Suavização Exponencial para modelar e prever dados, ajustando-se dinamicamente a mudanças nos padrões históricos.

O Erro (E) representa a variação aleatória ou ruído nos dados, podendo ser aditivo (A) ou multiplicativo (M). A Tendência (T) captura a direção geral da série temporal, podendo ser nenhuma (N), aditiva (A), aditiva amortecida (Ad), multiplicativa (M) ou multiplicativa amortecida (Md). A sazonalidade (S) reflete padrões que se repetem em intervalos regulares, podendo ser nenhuma (N), aditiva (A) ou multiplicativa (M). A notação ETS é representada como ETS (E,T,S), indicando os tipos de erro, tendência e sazonalidade presentes no modelo.

2.4.1.10. ETS (A,N,N): Suavização exponencial simples com erros aditivos

De acordo com Hyndman e Athanasopoulos (2018), o modelo aditivo pode ser expresso conforme apresentado nas Equações 21 e 22:

$$y_t = l_{t-1} + \varepsilon_t \tag{21}$$

$$l_t = l_{t-1} + \alpha\varepsilon_t \tag{22}$$

y_t é a equação de observação, enquanto l_t é a equação de estado ou de nível.

Essas duas equações, juntamente com a distribuição estatística dos erros, formam um modelo estatístico totalmente especificado.

2.4.1.11. ETS (M,N,N): Suavização exponencial simples com erros multiplicativos

Já o modelo multiplicativo é modelado conforme apresentado nas Equações 23, 24 e 25, conforme Hyndman e Athanasopoulos (2018):

$$\varepsilon_t = \frac{y_t - \breve{y}_{t|t-1)}}{\breve{y}_{t|t-1)}} \tag{23}$$

$$y_t = l_{t-1}(1 + \varepsilon_t) \tag{24}$$

$$l_t = l_{t-1}(1 + \alpha\varepsilon_t) \tag{25}$$

y_t é o valor observado no instante t, $\breve{y}_{t|t-1)}$ é o valor previsto para o instante t baseado nas informações até t−1 e ε_t é o erro relativo no instante t, medido como uma proporção em relação ao valor previsto.

Este modelo é apropriado para séries onde a variabilidade sazonal aumenta ou diminui proporcionalmente ao nível.

2.4.1.12. *Seasonal and Trend decomposition using Loess* com a modelagem autorregressiva

O modelo Seasonal and Trend decomposition using Loess com modelagem autorregressiva (STLM-AR) combina a decomposição de séries temporais em componentes sazonais e de tendência por meio do método STL (*Seasonal and Trend decomposition using Loess*) e da suavização não paramétrica LOESS (*Locally Estimated Scatterplot Smoothing*). O LOESS é uma técnica de regressão local que ajusta múltiplas regressões polinomiais em pequenos subconjuntos dos dados, ponderando os pontos mais próximos por meio de uma

função de suavização. Essa abordagem permite modelar relações complexas e não lineares sem impor uma forma funcional pré-definida, o que torna o método especialmente útil para capturar variações graduais de tendência e flutuações sazonais irregulares. Ao integrar essa decomposição à modelagem autorregressiva, o STLM-AR é capaz de representar de maneira mais fiel os padrões dinâmicos de séries temporais que apresentam tanto estrutura sazonal quanto tendência não linear

A primeira fase do processo é a decomposição. O método STL decompõe uma série temporal em três componentes principais (Tendência (T); Sazonaliade (S) e Resíduo (R)). A previsão é considerada como sendo a soma desses componentes.

A segunda fase é o ajuste sazonal, para remover o componente sazonal (S), obtendo-se a série ajustada (y'). O próximo passo é a modelagem Autorregressiva (AR).

Após o ajuste sazonal, aplica-se um modelo Autorregressivo (AR) aos dados ajustados. Esta ação captura a dependência linear entre uma observação e suas defasagens anteriores, permitindo prever o comportamento futuro da série ajustada. Este modelo pode ser facilmente implementado em R, através da função “stlm” da biblioteca “forecast”.

A decomposição STL permite capturar padrões sazonais e de tendência não lineares de forma eficaz. Devido a isso este método é indicado para séries temporais com sazonalidade bem definida e tendências não lineares.

2.4.1.13. Método Theta

O método Theta é um modelo de previsão univariada que se destaca por sua simplicidade, robustez e desempenho superior em séries temporais com tendência e flutuações irregulares. Proposto por Assimakopoulos e Nikolopoulos (2000), o método baseia-se na decomposição da série original em componentes denominados Theta-lines, geradas a partir da modificação da curvatura local da série. Essa decomposição permite combinar o comportamento de longo prazo com as oscilações de curto prazo, resultando em previsões mais estáveis e adaptáveis a diferentes padrões de dados.

A ideia central do método consiste em transformar a série temporal original $y_t$em uma ou mais versões modificadas $y_t^{(\theta)}$, aplicando-se um coeficiente de curvatura θ às segundas diferenças da série. Esse coeficiente controla o grau de amplificação ou suavização das variações locais. Em sua forma mais simples, o método utiliza duas linhas: a linha com $\theta = 0$, que corresponde aproximadamente a uma regressão linear dos dados e capta a tendência de longo prazo; e a linha com $\theta = 2$, que duplica as curvaturas locais, realçando as flutuações de curto prazo.

Embora o artigo original não apresente uma expressão fechada única para o processo, Hyndman e Billah (2003) demonstraram que a transformação pode ser interpretada de forma equivalente pela relação apresentada na Equação 26 e 27:

$$y_t^{(\theta)} = y_t + (\theta - 1)\,\Delta^2 y_t \tag{26}$$

$$\Delta^2 y_t = y_t - 2y_{t-1} + y_{t-2} \tag{27}$$

Essa representação mostra que, ao variar o valor de θ, o método altera a curvatura da série original: valores menores suavizam as flutuações, enquanto valores maiores as amplificam. Cada linha-*Theta* é então extrapolada separadamente por métodos adequados, normalmente, a linha θ = 0 é projetada por regressão linear, e a linha θ = 2 por suavização exponencial simples. Em seguida, as previsões são combinadas (usualmente pela média aritmética com pesos iguais) para produzir a estimativa final da série original, apresentada na Equação 28:

$$\hat{y}_t = \frac{1}{2}(\hat{y}_t^{(0)} + \hat{y}_t^{(2)}) \tag{28}$$

Pesquisas subsequentes mostraram que o método é equivalente, em sua forma padrão, ao modelo de suavização exponencial simples ETS(A,N,N), conforme demonstrado por Hyndman e Billah (2003). Sua principal vantagem reside em combinar a robustez de uma tendência linear com a flexibilidade para capturar variações locais, o que o torna particularmente eficaz em contextos com dados limitados ou comportamento irregular.

2.4.2. Métodos para demandas intermitentes

2.4.2.1. Método de Croston

O método de Croston é uma abordagem clássica amplamente empregada para a previsão de demandas intermitentes, isto é, aquelas em que ocorrem longos períodos sem consumo (demanda nula), intercalados por eventos esporádicos de demanda positiva. Essa característica é comum em estoques de peças sobressalentes, componentes de manutenção e itens de baixa rotação, como ocorre no ambiente logístico militar.

Proposto por Croston (1972), o método busca contornar a limitação dos modelos tradicionais de suavização exponencial simples, que tendem a superestimar a demanda quando confrontados com séries contendo muitos valores zero. Para isso, o autor propôs decompor a série temporal em dois processos distintos:

(a) o tamanho médio da demanda (z_t), que representa o valor médio das ocorrências positivas; e

(b) o intervalo médio entre ocorrências (p_t), que expressa o número médio de períodos entre duas demandas positivas consecutivas.

Ambos os componentes são atualizados por suavização exponencial simples, mas apenas quando ocorre uma nova demanda positiva ($y_t > 0$). O valor da previsão é então obtido pela razão entre essas duas estimativas suavizadas, conforme as Equações 29 a 31:

$$\mathrm{zt} = \alpha y_t + (1 - \alpha) z_{t-1} \tag{29}$$

$$p_t = \beta q_t + (1 - \beta) p_{t-1} \tag{30}$$

$$\hat{y}_{t+1} = \frac{z_t}{p_t} \tag{31}$$

Em que:

y_té a demanda observada no período t;

q_té o intervalo de tempo desde a última ocorrência de demanda positiva;

α e βsão os coeficientes de suavização ($0 < \alpha, \beta \leq 1$); e

$\hat{y}_{t+1}$é a previsão para o próximo período.

Na prática, o método atualiza o tamanho médio e o intervalo médio de forma independente, somente nos períodos em que ocorre consumo. Nos períodos sem demanda, os valores suavizados permanecem inalterados, evitando que zeros sistemáticos distorçam a previsão. Assim, o modelo mantém uma estimativa ajustada do ritmo de ocorrência e do volume médio das demandas não nulas.

Embora simples e eficiente, o método de Croston apresenta um viés positivo sistemático, ou seja, tende a superestimar a demanda média, aponta Syntetos e Boylan (2001). Essa limitação levou ao desenvolvimento de variações posteriores, como o *Syntetos–Boylan Approximation* (SBA) e o *Teunter–Syntetos–Babai* (TSB), que introduzem correções e aprimoramentos na modelagem probabilística da ocorrência das demandas.

Ainda assim, o método original de Croston continua sendo amplamente utilizado por sua intuição clara, baixa complexidade computacional e boa capacidade de adaptação a contextos operacionais com consumo esparso, como ocorre na previsão de peças de reposição aeronáuticas e de munições especiais de uso militar, onde o consumo é irregular, mas sua previsão é fundamental para a prontidão logística.

2.4.2.2. *Syntetos–Boylan Approximation* (SBA)

O *Syntetos–Boylan Approximation* (SBA) é um aprimoramento direto do método de Croston, proposto por Syntetos e Boylan (2005) com o objetivo de corrigir o viés positivo sistemático identificado nas previsões originais de demandas intermitentes. No método de Croston, como o tamanho médio da demanda e o intervalo médio entre ocorrências são atualizados separadamente, a razão entre eles tende a superestimar a demanda real. O SBA

introduz uma correção deflacionária nessa razão, ajustando a previsão de modo a torná-la estatisticamente não viesada, sem aumento relevante de complexidade computacional.

De forma geral, a previsão no SBA mantém a estrutura conceitual de Croston, com as séries suavizadas do tamanho médio das demandas positivas (z_t) e do intervalo médio entre ocorrências (p_t) —, mas aplica um fator de correção multiplicativo que reduz a tendência de superestimação. A nova previsão é expressa pela Equação 32):

$$\hat{y}_{t+1}^{(SBA)} = (1 - \frac{\alpha}{2}) \times \frac{z_t}{p_t} \tag{32}$$

Em que:

$\hat{y}_{t+1}^{(SBA)}$ é a previsão da demanda para o próximo período;

z_t é a estimativa suavizada do tamanho médio das demandas positivas;

p_t é a estimativa suavizada do intervalo médio entre demandas; e

α é o parâmetro de suavização aplicado à atualização de z_t.

O termo $(1 - \alpha/2)$ representa a aproximação deflacionária que corrige o viés do método original, reduzindo a expectativa da previsão em aproximadamente metade da taxa de suavização. Essa correção foi derivada a partir de análises teóricas e simulações conduzidas pelos autores, que mostraram melhoria significativa na acurácia média das previsões sem comprometer a simplicidade operacional.

Empiricamente, o método SBA apresenta desempenho mais estável em séries com baixa frequência de consumo e intermitência persistente, nas quais o método de Croston costuma superestimar a demanda. Além disso, mantém a vantagem de atualizar os parâmetros apenas quando ocorre uma demanda positiva, preservando a economia computacional e a intuição interpretativa do modelo original.

Por outro lado, o SBA assume implicitamente que a intermitência da série é estacionária, ou seja, que a frequência de ocorrências não varia significativamente ao longo do tempo. Em casos em que essa frequência muda gradualmente, por exemplo, em processos de obsolescência ou substituição de itens, métodos mais recentes, como o *Teunter–Syntetos–Babai* (TSB), mostram-se mais adequados.

Em síntese, o SBA constitui um avanço relevante na previsão de demandas intermitentes, sendo amplamente recomendado como alternativa prática ao método de Croston para situações em que se deseja reduzir o viés de superestimação e melhorar a acurácia sem aumentar a complexidade do modelo.

2.4.2.3. *Teunter–Syntetos–Babai* (TSB)

O método *Teunter–Syntetos–Babai* (TSB) foi proposto por Teunter, Syntetos e Babai (2011) como uma extensão dos métodos de Croston (1972) e SBA, com o objetivo de lidar de forma mais eficaz com séries de demanda intermitente sujeitas à obsolescência ou a mudanças graduais na frequência de ocorrências. Nesses contextos, a suposição de frequência constante entre eventos de demanda, implícita em Croston e SBA, tende a produzir estimativas enviesadas e lentas na adaptação a declínios de consumo.

A principal inovação do método TSB está na introdução explícita de uma probabilidade de ocorrência de demanda (π_t) que é atualizada em todos os períodos, inclusive naqueles em que a demanda observada é nula. Essa abordagem permite modelar a chance de ocorrência de uma nova demanda como um processo dinâmico, ajustando-se progressivamente às mudanças estruturais da série, como reduções de uso, substituição de itens ou obsolescência gradual.

O método separa o problema em dois componentes:

a) a probabilidade de ocorrência de demanda (π_t), estimada via suavização exponencial; e

b) o tamanho médio da demanda positiva (z_t), atualizado apenas quando $y_t > 0$.

A previsão da demanda para o período seguinte é dada pela multiplicação entre esses dois componentes, conforme as Equações 33 a 36:

$$\pi_t = \pi_{t-1} + \beta(o_t - \pi_{t-1}) \tag{33}$$

$$z_t = z_{t-1} + \alpha(y_t - z_{t-1}), \text{ se } y_t > 0 \tag{34}$$

$$\hat{y}_{t+1} = \pi_t \times z_t \tag{35}$$

$$o_t = \begin{cases} 1, & \text{se } y_t > 0; \\ 0, & \text{se } y_t = 0. \end{cases} \tag{36}$$

Em que:

y_t é a demanda observada no período t;

o_t é uma variável indicadora da ocorrência de demanda;

π_t é a probabilidade estimada de ocorrência de uma nova demanda;

z_t é o tamanho médio estimado das demandas positivas;

α e β são os parâmetros de suavização ($0 < \alpha, \beta \leq 1$); e

$\hat{y}_{t+1}$ é a previsão da demanda para o próximo período.

Essa formulação permite que o método atualize continuamente a probabilidade de ocorrência de consumo, tornando-o mais sensível a mudanças estruturais na frequência de demanda. Assim, quando os períodos sem consumo se tornam mais frequentes, a estimativa de

π_t decai gradualmente, reduzindo automaticamente a previsão futura, característica essencial em contextos de obsolescência, declínio operacional ou migração de tecnologias.

De acordo com *Teunter, Syntetos e Babai* (2011), o TSB mostrou desempenho superior aos métodos de Croston e SBA em experimentos envolvendo séries intermitentes com taxas de ocorrência não estacionárias. Além disso, o método mantém a simplicidade operacional dos modelos anteriores, requerendo apenas dois parâmetros de suavização e baixo custo computacional.

Em síntese, o método TSB representa um avanço na previsão de demandas intermitentes, por incorporar a dimensão temporal da probabilidade de ocorrência de demanda. Essa capacidade o torna particularmente útil na gestão de estoques militares e aeronáuticos, em que diversos itens apresentam ciclos de vida longos, fases de declínio e padrões de consumo esporádicos.

2.4.3. Métodos baseados em aprendizagem de máquinas

O avanço das técnicas de aprendizado de máquina tem permitido o desenvolvimento de modelos preditivos capazes de capturar padrões complexos e não lineares em séries temporais, superando as limitações de modelos estatísticos tradicionais. Os modelos de ML aprendem automaticamente relações entre entradas e saídas a partir de dados históricos, ajustando seus parâmetros por meio de processos iterativos de otimização. Diferentemente dos métodos paramétricos clássicos, que exigem suposições explícitas sobre tendência e sazonalidade, os modelos de ML buscam inferir diretamente os padrões presentes nos dados, o que os torna mais flexíveis e adaptáveis.

No presente trabalho, foram utilizados dois algoritmos principais de aprendizado supervisionado para previsão de demanda: NNETAR, baseado em redes neurais artificiais, e XGBoost, um modelo de *gradient boosting* com estrutura de árvores de decisão. Ambos foram configurados de forma autoregressiva, utilizando defasagens temporais (*lags*) como variáveis de entrada.

2.4.3.1. Método NNETAR – Rede Neural Multicamadas (MLP)

O NNETAR (*Neural Network Autoregression*) é uma rede neural artificial aplicada à previsão de séries temporais, na qual as entradas são compostas pelos valores passados da própria série, permitindo capturar relações não lineares entre os períodos anteriores e os valores futuros, conforme aponta Hyndman e Athanasopoulos (2018).

A estrutura do modelo é formada por camadas interconectadas de neurônios artificiais: uma camada de entrada com p neurônios (correspondentes aos lags da série), uma ou mais

camadas ocultas com funções de ativação não lineares e uma camada de saída responsável pela previsão. Matematicamente, o modelo pode ser descrito conforme apresentado na Equação 37:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-p}; W) = \phi_2(W_2\, \phi_1(W_1 x_t + b_1) + b_2) \quad (37)$$

Em que:

$x_t = (y_{t-1}, \dots, y_{t-p})$ representa o vetor de entradas.

W_1 e W_2 são matrizes de pesos sinápticos.

b_1 e b_2 são os vieses (bias);

$\phi_1(\cdot)$ e $\phi_2(\cdot)$ são as funções de ativação; e

$\hat{y}_t$ é o valor previsto da série no tempo t.

Durante o processo de aprendizado, os pesos W são ajustados de forma iterativa utilizando o algoritmo de retropropagação do erro, de modo a minimizar uma função de perda, geralmente o erro quadrático médio (RMSE).

A principal vantagem do NNETAR é sua capacidade de capturar relações não lineares e interações complexas entre observações passadas, mesmo sem especificar explicitamente um modelo de tendência ou sazonalidade. No entanto, a técnica requer um volume adequado de dados para evitar *overfitting* e pode demandar ajuste criterioso do número de neurônios e das taxas de aprendizado.

No contexto desta pesquisa, o NNETAR foi utilizado como modelo base dentro da estrutura híbrida de previsão, contribuindo para representar padrões não lineares e complementando a capacidade dos métodos estatísticos clássicos (como ARIMA e Theta).

2.4.3.2. *Extreme Gradiente Boosting – XGBoost*

O *XGBoost* (*Extreme Gradient Boosting*) é um algoritmo de aprendizado supervisionado baseado em conjuntos de árvores de decisão combinadas de forma sequencial, de modo que cada nova árvore é treinada para corrigir os erros das anteriores, explicam Chen e Guestrin (2016).

Matematicamente, o modelo é representado pela soma de K funções de árvores de decisão, apresentada na Equação 38, conforme Chen e Guestrin (2016).

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i), f_k \in \mathcal{F} \quad (38)$$

Cada f_k pertence ao espaço das funções de árvore $\mathcal{F}$, e $\mathbf{x}_i$ representa o vetor de atributos do exemplo i. O processo de treinamento busca minimizar uma função de perda regularizada, representada pela Equação 39 e 40, conforme Chen e Guestrin (2016).:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \quad (39)$$

com

$$\Omega(\mathrm{f_k}) = \gamma \mathrm{T} + \frac{1}{2}\lambda \parallel \mathrm{w} \parallel^2 \quad (40)$$

Onde $l(y_i, \hat{y}_i)$ é a função de erro (por exemplo, erro quadrático), T é o número de folhas da árvore, e os parâmetros γe λ controlam a regularização e penalizam modelos excessivamente complexos.

De acordo com Chen e Guestrin (2016), entre as principais vantagens do *XGBoost* destacam-se:

a) elevada precisão e robustez em dados não lineares;

b) controle de *overfitting* por regularização;

c) capacidade de lidar com dados esparsos e *missing values*; e

d) escalabilidade e paralelização.

Por outro lado, o modelo apresenta complexidade interpretativa maior em comparação com métodos lineares, além de exigir ajustes cuidadosos de hiper parâmetros.

No contexto da previsão de consumo de munições, o *XGBoos*t pode ser especialmente útil por sua capacidade de aprender relações não lineares entre variáveis, proporcionando previsões mais robustas em séries com variabilidade elevada ou padrões de intermitência.

2.4.4. *Ensemble learning* (Aprendizado de conjunto)

A utilização de modelos individuais de previsão de demanda possui uma limitação logo na sua seleção. Se o modelo selecionado não for adequado ao tipo de dados históricos, ele provavelmente não fornecerá previsões assertivas, por mais que se ajuste o modelo. O aprendizado em conjunto é uma abordagem de aprendizado de máquinas que combina previsões de múltiplos modelos para melhorar a precisão. Eles buscam melhorar os resultados utilizando vários modelos de aprendizado de máquina, afirmam Lee *et al*. (2018).

No aprendizado em conjunto, diferentes tipos de modelos de previsão de demanda são ajustados aos dados e geram previsões independentes para o mesmo conjunto de dados. A grande vantagem desses métodos é que eles reduzem a dependência de um único modelo, melhorando a precisão em séries temporais heterogêneas. Como desvantagem, esses modelos requerem maiores esforços computacionais. Os métodos principais são o *Bagging*, *Boosting* e *Stacking*.

A técnica de *Bagging* utiliza amostras aleatórias para treinar múltiplos modelos independentes. Cada modelo é treinado em um subconjunto diferente dos dados e as previsões finais são combinadas, geralmente por votação majoritária (classificação) ou média (regressão),

explicam Lee *et al*. (2018). Um desses métodos mais conhecidos é o *random forest*, que constrói múltiplas árvores de decisão independentes para reduzir a variância e melhorar a precisão.

A técnica de *Boosting* treina sequencialmente os modelos, onde cada modelo tenta corrigir os erros cometidos pelos anteriores. Os modelos subsequentes recebem maior peso nos exemplos mal classificados, o que reduz o viés. De acordo com Lee *et al*. (2018), exemplos comuns dessa técnica incluem *AdaBoost* e *Gradient Boosting*.

No *boosting*, cada modelo subsequente é ajustado para corrigir os erros dos modelos anteriores, focando nas instâncias mais difíceis de prever, explica Pereira (2023). De acordo com Cordeiro (2023), essa abordagem permite que o algoritmo ajuste progressivamente seus parâmetros, reduzindo significativamente o erro, o que resulta em um modelo com elevada precisão.

O *Stacking* combina previsões de diferentes modelos, utilizando um modelo de meta-aprendizagem para integrar os resultados. O aprendizado em conjunto de empilhamento refere-se aos métodos que aproveitam a complementaridade mútua entre os modelos base para melhorar o desempenho e aprimorar a capacidade de generalização, aponta Wolpert (1992).

Uma outra estratégia que vem sendo adotada é a combinação de aprendizado de máquinas com modelos estatísticos para melhoria dos modelos de previsão. Em relação à previsão de consumo de munições não é diferente. Embora muitos estudiosos tenham feito uma extensa pesquisa sobre o método de previsão do consumo de munição, ainda existem alguns problemas em aplicações práticas, como baixa precisão, baixa capacidade de adaptação a cenários específicos e assim por diante, afirma Li *et al*. (2020).

Diversos estudos estão propondo e investigando a aplicação desses modelos combinados na gestão dos estoques de diversos produtos. Makridakis *et al*. (2018a), em seu trabalho para apresentação na M4 *competition*, uma competição internacional de previsão de séries temporais realizada em 2018, organizada por Spyros Makridakis, que comparou o desempenho de modelos estatísticos e de aprendizado de máquina em 100.000 séries reais de diferentes domínios, destacou que a surpresa do evento foi que uma abordagem híbrida, que utilizou recursos estatísticos e de aprendizado de máquinas combinados, obteve os melhores resultados de precisão da competição.

Hewamalage *et al*. (2020) também constataram que técnicas de previsão univariada, que consideram séries temporais individuais isoladamente, podem falhar para produzir previsões confiáveis, pois se tornam inadequadas no contexto de *big data*, onde um único modelo pode aprender com muitas séries temporais semelhantes simultaneamente. Amaral Filho *et al*. (2021), Sina *et al*. (2023) e Azevedo (2024) também concluíram em seus estudos que modelos

que combinaram métodos estatísticos com aprendizado de máquinas proporcionaram melhores resultados quando comparados a modelos individuais.

Neste contexto, inspirado pelos trabalhos que já verificaram as vantagens na acuracidade de modelos híbridos, o presente trabalho utilizará um modelo de aprendizagem híbrida para aprimorar as previsões de demanda das munições de emprego aeronáutico.

2.5. COMPARAÇÃO ENTRE OS MÉTODOS DE PREVISÃO

A seleção de métodos de previsão deve considerar tanto a natureza estatística da série quanto as limitações inerentes a cada método. Nesta seção, serão abordadas as indicações e limitações de cada método de previsão abordado.

Segundo Chopra e Meindl (2011), a Média Móvel Simples (MMS) é apropriada para séries estáveis e sem tendência ou sazonalidade, mas apresenta resposta lenta a mudanças e não é fundamentada em modelos estatísticos formais, o que dificulta a quantificação da incerteza da previsão. Svetunkov e Petropoulos (2018) complementam apontando que o método carece de fundamentação estatística.

De acordo com Chopra e Meindl (2011), a Suavização Exponencial Simples (SES) também é indicada para séries sem tendência nem sazonalidade. No entanto, conforme Hyndman e Athanasopoulos (2018), tanto a MMS quanto a SES tendem a defasar em relação a séries com tendência, pois apenas ajustam o nível histórico sem capturar mudanças estruturais.

Segundo Veríssimo *et al*. (2012), o método de Holt-Winters é indicado para séries com tendência (suavização dupla) e sazonalidade (suavização tripla), oferecendo desempenho competitivo em aplicações reais. Hyndman *et al*. (2008) alertam que sua aplicação exige atenção à estabilidade dos componentes sazonais e à presença de *outliers*, sendo sensível a essas variações.

Segundo Box e Jenkins (1976), o modelo ARIMA (p,d,q), fundamentado na metodologia homônima, combina componentes autorregressivos e de média móvel aplicados sobre séries diferenciadas. Zhang (2003) destaca que sua principal limitação reside na suposição de linearidade e na sensibilidade a picos e ruídos, sendo pouco adequado para séries com padrões não lineares.

Walter *et al*. (2013) ressaltam que o modelo SARIMA, ao estender o ARIMA para tratar explicitamente da sazonalidade, se torna mais adequados para dados sazonais, porém, envolve uma calibração mais complexa e requer séries históricas longas para realizar análises mais representativas. Vale mencionar que o modelo SARIMA pode representar um modelo ARIMA quando P = D = Q = 0), dessa forma, o modelo SARIMA, para modelagem matemática set torna mais completo.

De acordo com Hyndman *et al*. (2008), os modelos ETS permitem combinar diferentes tipos de erro, tendência e sazonalidade. Essa estrutura flexível admite erros aditivos ou multiplicativos, várias formas de tendência (nenhuma, aditiva, amortecida, etc.) e sazonalidade em diferentes combinações. Petropoulos *et al*. (2013) observam, por sua vez, que tais modelos mostram desempenho limitado em séries com muitos valores zero (demanda intermitente) e que a seleção entre as diversas configurações possíveis pode ser desafiadora, dada a incerteza na escolha do modelo e na estimação de parâmetros.

De acordo com Hyndman e Athanasopoulos (2018), o modelo STLM-AR (STL com modelagem autoregressiva) é eficaz para séries com sazonalidade não perfeitamente regular, mas requer séries longas para decomposição estável e combina dois processos (STL + ARIMA), o que aumenta a complexidade do ajuste.

O método theta é o método de previsão de séries temporais univariadas mais bem-sucedido das últimas duas décadas, afirmam Nikolopoulos e Thomakos (2020). Esse método não exige otimização complexa de parâmetros. Sua forma clássica usa apenas dois componentes fixos ($\theta = 0$ e $\theta = 2$), o que o torna de fácil implementação. Outro ponto importante é que apresenta bom desempenho mesmo quando a quantidade de dados é relativamente pequena ou os dados apresentam ruído moderado. Entretanto, Petropoulos e Nikolopoulos (2013) ressaltam que o *Theta* padrão não modela sazonalidade internamente, pois o método assume que a série já foi previamente dessazonalizada, dessa forma, o *Theta* não captura sazonalidade de forma explícita.

Segundo Croston (1972), o método que leva seu nome é indicado para séries com demanda intermitente. Ele separa o tamanho da demanda e o intervalo entre ocorrências, atualizando apenas quando há demanda positiva. No entanto, Teunter, Syntetos e Babai (2011a) destacam que, apesar de sua ampla utilização, o método apresenta viés positivo na previsão da demanda média e não lida bem com obsolescência.

Conforme Syntetos e Boylan (2005), a aproximação que propõem (SBA) introduz uma correção ao viés do Croston, reduzindo sistematicamente a superestimação. Entretanto, este método perde acurácia em séries onde a frequência de demanda varia ou é alta, e não resolve totalmente a questão da obsolescência.

Segundo Teunter, Syntetos e Babai (2011b), o modelo TSB corrige as limitações anteriores ao atualizar continuamente a probabilidade de ocorrência de demanda, mesmo quando ela é nula. Assim, lida de forma mais eficaz com obsolescência. Contudo, depende de calibração cuidadosa de dois parâmetros de suavização e permanece um modelo univariado.

Hyndman e Athanasopoulos (2018) afirmam que as redes neurais autorregressivas (NNETAR), baseadas em redes multicamadas, são adequadas para séries com padrões não lineares e interações complexas. Zhang (2003) observa que essas redes têm capacidade de representação ampla, mas requerem grande volume de dados para evitar sobreajuste e apresentam baixa interpretabilidade.

Chen e Guestrin (2016) relatam que o modelo *XGBoost* apresenta elevada acurácia, robustez e capacidade de tratar dados esparsos. O *XGBoost* é capaz de capturar relações não lineares, é robusto a *outliers*, pode modelar interações entre características de maneira eficaz e geralmente alcança maior precisão nas previsões. Apesar dessas vantagens, o *XGBoost* possui algumas limitações, como ser mais complexo de interpretar, exigir um ajuste cuidadoso dos parâmetros e ter tendência a sobreajustar quando aplicado a conjuntos de dados pequenos, conforme pontuam Prakoso, Irfan e Siddique (2024).

A Tabela 1 sintetiza os principais métodos analisados, relacionando o tipo de série mais adequada e suas limitações principais, servindo como base para a seleção criteriosa dos modelos a serem aplicados em diferentes perfis de séries temporais.

Tabela 1- Comparação entre os métodos de previsão.

Método	Tipo de série mais adequada	Principais limitações
Média Móvel Simples	Séries estáveis, sem tendência ou sazonalidade	Não captura tendência nem sazonalidade
Suavização Exponencial Simples	Séries sem tendência nem sazonalidade	Ineficiente em séries com tendência ou sazonalidade
Holt-Winters Aditivo	Séries com tendência e sazonalidade constante	Sensível a outliers
Holt-Winters Multiplicativo	Séries com tendência e sazonalidade proporcional ao nível	Instável em séries com zeros
ARIMA (p,d,q)	Séries contínuas, com dependência temporal linear	Sensível a ruído e picos; mau desempenho com zeros
SARIMA (p,d,q)(P,D,Q,s)	Séries sazonais estruturadas	Complexidade de ajuste; exige séries longas e completas
ETS	Séries contínuas com tendência e/ou sazonalidade	Limitado em séries intermitentes
STLM-AR	Séries com tendência e sazonalidade moderadas	Exige dupla calibração (STL + ARIMA) e séries mais longas
Theta	Séries com tendência suave ou linear, com ruído moderado ou, ainda, com dados limitados	Subestima picos; não adequado para intermitência, nem séries com forte sazonalidade
Croston	Séries intermitentes	Superestima a média – viés positivo
SBA	Séries intermitentes	Perde acurácia em declínio de frequência
TSB	Séries intermitentes com declínio de frequência (obsolescência)	Sensível aos parâmetros de suavização
NNETAR	Séries não lineares ou altamente irregulares	Risco de sobreajuste; requer séries sem muitos zeros
XGBoost	Séries heterogêneas e complexas	Requer grande volume de dados e difícil interpretabilidade

Fonte: O autor.

2.6. ERROS DAS PREVISÕES

Na maioria das vezes, a precisão é o critério primário para selecionar um método de previsão, que indica o quão bem um modelo de previsão pode replicar os dados já conhecidos para realizar previsões, destacam Gasmi *et al*. (2024). Tal verificação é realizada através do cálculo dos erros de previsão.

2.6.1. *Root Mean Squared Error* (RMSE)

É uma métrica amplamente utilizada para avaliar a precisão de modelos de previsão, especialmente em séries temporais e aprendizado de máquina. Ela mede a magnitude média dos erros entre os valores previstos e os valores reais, sendo especialmente útil para identificar discrepâncias significativas. O RMSE é calculado conforme Equação 41, de acordo com Gasmi *et al*. (2024):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_t - f_t)^2}{n}} \tag{41}$$

n: número de observações.

y_t : valor real.

f_t : valor previsto.

O uso do RMSE é recomendado quando se deseja avaliar a precisão preditiva na mesma unidade da variável de interesse e quando grandes desvios devem ser penalizados de forma acentuada, uma vez que a métrica eleva ao quadrado os erros individuais. Essa característica torna o RMSE particularmente apropriado em contextos nos quais a ocorrência de poucos erros extremos implica custo substancial.

Todavia, por ser sensível a valores atípicos e a heteroscedasticidade, o RMSE pode superestimar o impacto de observações isoladas e favorecer períodos de maior nível da série; nesses casos, métricas mais robustas, como *Mean Absolute Error* (MAE) ou *Mean Absolute Scaled Error* (MASE), tendem a refletir melhor o desempenho.

2.6.2. *Mean Absolute Percentage Error* (MAPE)

Métrica utilizada para avaliar a precisão de modelos preditivos que calcula o erro percentual médio entre os valores previstos e os valores reais, fornecendo uma indicação de quão bem o modelo se ajusta aos dados observados. O MAPE é calculado conforme Equação 42, de acordo com Gasmi *et al*. (2024):

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{y_t - f_t}{y_t} \right| \tag{42}$$

Onde,

n: número de observações.

y_t : valor real.

f_t : valor previsto.

O MAPE é apropriado quando se deseja uma medida sem escala e de fácil interpretação, expressa em termos percentuais, permitindo comparar desempenho entre séries com magnitudes distintas. Ele é particularmente útil quando os erros relativos importam mais do que os absolutos. Contudo, a métrica apresenta limitações conhecidas, pois se torna instável para valores reais muito pequenos, pois pequenas diferenças absolutas geram percentuais elevados, além disso, o MAPE é assimétrico, penalizando superestimações e subestimações de maneira diferente, o que pode induzir vieses indesejados na seleção de modelos.

2.6.3. *Mean Absolute Error* (MAE)

O MAE é o Erro Absoluto Médio, que é uma métrica que mede o erro médio entre os valores previstos e os observados, considerando apenas a magnitude das diferenças, sem levar em conta o sinal (se o erro é positivo ou negativo). O MAE indica, em média, quanto as previsões se desviam dos valores reais, em unidades da variável original (por exemplo, unidades vendidas, litros, horas, etc.). É fácil de interpretar e menos sensível a grandes erros que métricas quadráticas como o RMSE.

2.6.4. *Symmetric Mean Absolute Percentage Error* (SMAPE)

O SMAPE também é uma métrica utilizada para avaliar a precisão de previsões. De acordo com Makridakis *et al.* (2020), ele é calculado conforme apresentado na Equação 43:

$$SMAPE = \frac{2}{h} \sum_{t=n+1}^{n+h} \frac{|y_t - \bar{Y}_t|}{|y_t| + |\bar{Y}_t|} * 100\% \qquad (43)$$

Em que:

y_t é o valor da série temporal no tempo t, $\bar{Y}_t$ a previsão estimada, h é o horizonte de previsão e n o número de pontos de dados disponíveis na amostra.

O SMAPE é uma métrica sem escala e de interpretação percentual que busca atenuar a assimetria do MAPE ao normalizar o erro absoluto pela média dos módulos de observado e previsto, o que a torna adequada para comparar desempenho entre séries de magnitudes distintas e para cenários em que erros relativos (proporcionais) são relevantes. Por ser "simetrizada", tende a tratar super e subestimações de forma mais equilibrada e a reduzir a explosão de valores quando o observado é pequeno, problema típico do MAPE. Ainda assim, o SMAPE permanece sensível quando ambos os termos no denominador são muito baixos (observado e previsto

próximos de zero), podendo atribuir penalidades desproporcionalmente altas a períodos de baixo volume.

Em síntese, o SMAPE é apropriado quando se pretende comparabilidade percentual e tratamento mais equilibrado de super/subestimações, desde que se mitiguem os efeitos de baixos volumes e se complemente a análise com métricas que reflitam o custo operacional do erro no contexto estudado.

2.6.5. *Mean Absolute Scaled Error* (MASE)

O MASE é amplamente utilizado na literatura contemporânea de previsão, sendo independente da escala dos dados e menos sensível a *outliers* do que os outros modelos aqui apresentados conforme explicam Makridakis *et al*. (2020). Pode ser calculado conforme a Equação 44:

$$MASE = \frac{1}{h} * \frac{\sum_{t=n+1}^{n+h} \left| y_t - \bar{Y}_t \right|}{\frac{1}{n-m}\sum_{t=m+1}^{n} \left| y_t - \bar{Y}_{t-m} \right|} \quad (44)$$

y_t é o valor da série temporal no tempo *t*, $\bar{Y}_t$ a previsão estimada, *h* é o horizonte de previsão, *n* o número de pontos de dados disponíveis na amostra e *m* é o intervalo de tempo entre sucessivas observações.

O MASE é uma métrica sem escala que torna comparáveis erros entre séries de magnitudes distintas ao escalar o MAE do modelo pelo MAE de uma referência ingênua (tipicamente a previsão ingênua ou sazonal ingênua calculada no conjunto de treino). Por essa construção, valores MASE < 1 indicam desempenho superior, enquanto MASE > 1 sinalizam piora, oferecendo interpretação direta. Em resumo, um modelo que resulta em MASE< 1 pode ser considerado como tendo boa performance preditiva, aponta Kell *et. al* (2024).

Diferentemente do RMSE, o MASE não penaliza demais grandes desvios, sendo menos sensível a *outliers*, portanto, é mais adequado para séries com zeros e para comparações entre itens. Contudo, em séries muito curtas ou quase constantes, o erro de escala da referência pode ser próximo de zero, inflando o MASE; nesses casos, recomenda-se usar a versão sazonal ou introduzir salvaguardas numéricas. Em síntese, o MASE é particularmente apropriado quando se deseja comparabilidade entre séries e interpretação relativa ao modelo ingênuo (repetição dos valores do último período), mantendo robustez prática em cenários com zeros e heterogeneidade de escala.

2.6.6. *Weighted Absolute Percentage Error* (WAPE)

O WAPE é uma métrica muito utilizada na literatura contemporânea de previsão por sua simplicidade interpretativa e por ser menos sensível a escalas absolutas de valores. Pode ser calculado conforme a Equação 45:

$$WAPE = \frac{\sum_{t=1}^{n} | y_t - \bar{Y}_t |}{\sum_{t=1}^{n} |y_t|} \quad (45)$$

Em que *yt* representa o valor observado da série temporal no tempo *t* e "$\bar{Y}_t$" é a previsão correspondente. O WAPE expressa o erro absoluto total em relação ao total da demanda observada, funcionando como uma versão "ponderada" do MAPE. Diferentemente do MAPE, que calcula uma média simples de erros relativos ponto a ponto, o WAPE acumula os desvios absolutos e os relaciona ao montante observado, reduzindo a influência de períodos com valores muito pequenos que poderiam distorcer o percentual de erro. Por esse motivo, é considerado mais estável em séries com grande variabilidade ou com presença de zeros intercalados.

Entre suas vantagens, destacam-se a robustez em relação a *outliers* pontuais (menos penalizado que o RMSE) e a facilidade de comparação entre diferentes itens ou categorias, já que é uma métrica sem unidade. Contudo, em séries onde a soma dos valores observados é muito baixa, o denominador pode inflar o indicador, devendo-se aplicar salvaguardas numéricas.

Em síntese, o WAPE é particularmente útil quando se busca uma métrica intuitiva, comparável entre itens e com interpretação direta em termos de percentual de erro em relação à demanda realizada, sendo uma das métricas preferidas em ambientes de gestão de estoques e previsão de demanda.

2.6.7. Viés

O viés indica se um modelo de previsão tende a errar consistentemente para cima ou para baixo. Ele é calculado a partir da diferença entre cada valor previsto e o valor real, acumulando esses erros ao longo do período analisado.

Para obter o viés, soma-se todas as diferenças entre previsões e valores observados. Em seguida, esse total é comparado com o volume real da demanda. Se o resultado for positivo, significa que o modelo superestimou a demanda; se for negativo, significa que subestimou. Quanto mais próximo de zero estiver, menor é a tendência sistemática de erro.

Em termos práticos:

a) viés positivo: o modelo prevê mais do que ocorre;

b) viés negativo: o modelo prevê menos do que ocorre; e

c) viés próximo de zero: o modelo não apresenta tendência direcional.

Esse indicador complementa as demais métricas de erro ao mostrar não apenas o tamanho do erro, mas também a direção e o comportamento sistemático das previsões.

3 METODOLOGIA

3.1. CARACTERIZAÇÃO DO MÉTODO DE PESQUISA

O percurso metodológico do presente trabalho foi delineado de forma a possibilitar o alcance do objetivo geral e de seus três objetivos específicos, conforme apresentado na Tabela 2, que estabelece a correspondência entre cada objetivo e as etapas metodológicas correspondentes.

Tabela 2 - Relação entre os objetivos específicos e as etapas metodológicas da pesquisa.

Objetivo específico	Etapa metodológica correspondente
1. Analisar os dados históricos	Análise exploratória dos dados do SILOMS
2. Desenvolver modelo preditivo	Construção e treinamento dos modelos de previsão
3. Avaliar o desempenho	Avaliação de desempenho por métricas de erro

Fonte: O autor.

A pesquisa aplicada, conforme Gil (2002), tem como característica principal a finalidade de gerar conhecimentos para aplicação em situações reais, com o objetivo de resolver problemas específicos de uma organização ou contexto social.

Quanto à natureza da pesquisa, este estudo é classificado como descritivo, pois busca caracterizar, analisar e interpretar as propriedades estatísticas das séries temporais de consumo de munições. Segundo Vergara (2000), a pesquisa descritiva tem como propósito principal observar, registrar e analisar fenômenos sem interferi-los, permitindo, em certos casos, o estabelecimento de correlações entre variáveis. Nesse sentido, o presente estudo descreve o comportamento da demanda por munições ao longo do tempo, identifica padrões e flutuações, e utiliza essas informações para apoiar a seleção da política de estoques mais adequada.

No que diz respeito à abordagem metodológica, trata-se de uma pesquisa quantitativa, fundamentada na análise objetiva de dados numéricos. A abordagem quantitativa se caracteriza pelo uso de métodos estatísticos e computacionais para mensurar variáveis, testar hipóteses e construir modelos preditivos baseados em evidências empíricas. A coleta e o tratamento dos dados foram realizados por meio de extrações do SILOMS, abrangendo registros históricos de consumo de munições de emprego aeronáutico no período de 2009 a 2024. Este foi o período selecionado por ser o período disponível na atual versão do SILOMS. Ou seja, serão utilizados todos os dados disponíveis, desde a implantação do atual sistema de gestão de estoques.

A análise dos dados foi conduzida utilizando a linguagem de programação Python, com o emprego de bibliotecas especializadas em estatística, séries temporais e aprendizado de máquina. Essas ferramentas permitiram a aplicação de algoritmos preditivos, avaliação de

desempenho por métricas de erro, extração de características das séries temporais e posterior apoio à decisão por meio da técnica de decisão multicritério (TOPSIS).

3.2. ANÁLISE EXPLORATÓRIA DOS DADOS DO SILOMS

Para atender ao primeiro objetivo específico procedeu-se à coleta dos dados no Sistema Integrado de Logística de Materiais e Serviços (SILOMS), abrangendo o período de 2009 a 2024. Os dados foram organizados em séries temporais, e aplicaram-se técnicas de análise exploratória, como decomposição STL, cálculo de tendência, sazonalidade, variabilidade e identificação de valores ausentes e *outliers*.

Essas análises permitiram compreender o comportamento das séries e subsidiar a escolha dos métodos preditivos mais adequados para o modelo desenvolvido. A Análise Exploratória de Dados (AED) é um conjunto de técnicas estatísticas e gráficas que permite investigar e resumir os principais aspectos de um conjunto de dados. Ela tem como objetivo identificar padrões, detectar *outliers*, verificar premissas e estabelecer hipóteses para análises posteriores. Essa abordagem permite uma compreensão inicial dos dados, orientando análises subsequentes e a construção de modelos estatísticos mais robustos.

A AED analisa principalmente as tendências, sazonalidades, distribuições estatísticas (como média e variância), estacionariedade e correlações entre variáveis. Conforme apontam Hyndman e Athanasopoulos (2018), além de identificar *outliers* ela avalia a presença de ruído nos dados, elementos cruciais para ajustar modelos analíticos ou preditivos.

3.2.1. Dados de entrada

Os dados de entrada que formam a base de dados da análise foram extraídos do SILOMS pela tela SUP0089R1, e se referem ao histórico de consumo de munições aeronáuticas do COMAER, referente ao período de 2009 a 2024.

Devido a sensibilidade dos dados, os valores específicos não serão apresentados neste trabalho, nem as munições serão identificadas pelos seus nomes originais. No presente trabalho, elas serão designadas por munição do tipo A, tipo B, tipo C e tipo D e os valores reais não serão apresentados.

O conjunto de dados é formado por 1.229 registros de demandas históricas, de quatro munições diferentes, sendo 649 registros do tipo A, 114 registros do tipos B, 25 registros do tipo C e 440 registros do tipo D.

3.2.2. Metodologia da análise exploratória

A análise exploratória das séries temporais foi conduzida com o objetivo de compreender as características e os padrões temporais presentes nas séries históricas de

consumo de munições aeronáuticas, de modo a subsidiar a escolha dos modelos de previsão mais adequados. Essa etapa permite identificar a presença de tendência, sazonalidade, intermitência e ruído, além de avaliar a estacionariedade e o grau de variabilidade das séries.

O processo foi inteiramente desenvolvido em linguagem Python, empregando rotinas automatizadas para leitura, tratamento e análise estatística dos dados. A análise faz uso das seguintes bibliotecas:

a) *Pandas*, para leitura, limpeza e manipulação dos dados;
b) *NumPy*, para operações matemáticas e transformações logarítmicas;
c) *Statsmodels*, para aplicação de métodos estatísticos, como decomposição STL e teste de estacionariedade (ADF);
d) *PMDARIMA*, para ajuste automatizado de modelos AutoARIMA com detecção de sazonalidade; e
e) *SciPy*, utilizada no cálculo de assimetria e na detecção de picos na função de autocorrelação, empregados na estimativa da sazonalidade.

3.2.2.1. Carregamento e preparação dos dados

Os dados foram carregados a partir de um arquivo CSV exportado do SILOMS, utilizando a biblioteca *pandas*. Foi aplicada a função *applymap* para remover espaços extras e padronizar o conteúdo das células. Em seguida, a coluna de data foi convertida para o tipo *datetime*. As séries foram então ordenadas por tipo de material e data (*sort_values*) e reamostradas em frequência mensal, somando as quantidades de consumo registradas em cada dia.

Meses sem registros de consumo foram automaticamente preenchidos com o valor zero, pois representam ausência de demanda e devem ser tratados como observações válidas.

3.2.2.2. Tratamento de *outliers* com *Winsorization*

O primeiro passo da preparação consistiu no tratamento de outliers por meio da técnica de *Winsorization*, que limita valores extremos substituindo-os pelos percentis definidos como limites inferior e superior (1º e 99º percentis, respectivamente). Essa abordagem reduz a influência de valores anômalos, que poderiam distorcer as estimativas de tendência e sazonalidade.

3.2.2.3. Decomposição das séries

Para compreender a estrutura temporal de cada série, foi utilizada a decomposição STL (*Seasonal-Trend decomposition using Loess*), que separa a série em três componentes: tendência, sazonalidade e resíduo. O método aplica suavização local via *Loess*, permitindo

estimar padrões de longo prazo e ciclos sazonais de forma robusta. A partir dessa decomposição, foram calculadas as forças da tendência (Ft), sazonalidade (Fs) e ruído (Fr), representando a proporção da variância explicada por cada componente. Esses indicadores fornecem uma medida quantitativa da estrutura temporal da série e permitem classificar seu comportamento em termos de previsibilidade.

3.2.2.4. Estimativa da sazonalidade e ajuste de modelos *AutoARIMA*

A frequência sazonal de cada série foi estimada por meio da função de autocorrelação (ACF), identificando o primeiro pico significativo como período sazonal predominante. Essa estimativa é então utilizada como parâmetro de sazonalidade (m) no ajuste do modelo *AutoARIMA*, que seleciona automaticamente as ordens p, d, qe P, D, Q mais adequadas. A etapa de ajuste do modelo permite avaliar o comportamento dos resíduos, ou seja, verificar se o modelo é capaz de capturar os padrões estruturais da série.

3.2.2.5. Avaliação dos resíduos e verificação de estacionariedade

Após o ajuste do modelo, foi avaliada a aleatoriedade dos resíduos por meio das funções ACF e PACF, analisando a existência de autocorrelações residuais não explicadas. Resíduos aleatórios (sem autocorrelação significativa) indicam que o modelo conseguiu capturar adequadamente os padrões de tendência e sazonalidade.

Em seguida, foi aplicado o teste de Dickey-Fuller aumentado (ADF), que verifica a presença de raiz unitária e, portanto, a estacionariedade da série. Conforme pode ser visto em Santamaria *et al*. (2021), valores de p inferiores a 0,05 indicam estacionariedade. Essa verificação é fundamental, pois séries não estacionárias requerem transformações (como diferenciação) ou modelos específicos que incorporem tendência ou sazonalidade explícita.

3.2.2.6. Cálculo de métricas estruturais e classificação das séries

Além das forças dos componentes, foram calculados indicadores complementares para caracterizar o comportamento de cada série temporal. A força da tendência (Ft), a força da sazonalidade (Fs) e a força do ruído (Fr), obtidas a partir da decomposição STL, quantificam a proporção da variância explicada por cada componente, permitindo distinguir séries com estrutura temporal bem definida daquelas dominadas por aleatoriedade.

O coeficiente de variação (CV) mede a variabilidade relativa da série, enquanto a assimetria indica a presença de distribuições concentradas em valores baixos com picos esporádicos de alta demanda. Adicionalmente, a proporção de zeros foi utilizada como medida de intermitência, e o p-valor do teste ADF como indicador de estacionariedade.

Com base nesses indicadores, foram estabelecidos critérios operacionais para a classificação das séries e para a escolha preliminar dos métodos de previsão. Séries com zeros superior a 0,4 (40% da série) foram classificadas como intermitentes, sugerindo o uso de modelos específicos como Croston, SBA e TSB. Valores elevados de Fs (por exemplo, $Fs \geq 0,5$) indicam sazonalidade forte, favorecendo métodos como Holt-Winters, ETS sazonal, SARIMA ou STLM-AR, já Ft elevado ($Ft \geq 0,4$) sinaliza tendência marcante, recomendando modelos que a incorporem explicitamente, como Theta, ETS ou ARIMA. Séries com CV elevado e alta assimetria foram identificadas como irregulares, podendo demandar abordagens mais flexíveis, como NNETAR ou *XGBoost*. Por fim, séries com p-valor ADF inferior a 0,05 foram consideradas estacionárias, sendo mais adequadas a modelos lineares clássicos, como ARIMA ou ETS em suas formas simples.

Por fim, a análise dos resultados da etapa de análise exploratória servirá de base para a escolha dos métodos de previsão que comporão o modelo de previsão desenvolvido.

3.3. CONSTRUÇÃO E TREINAMENTO DOS MODELOS DE PREVISÃO

Como apresentado no item anterior, as características das séries temporais influenciam diretamente a adequação de diferentes modelos de previsão. A escolha inadequada pode resultar em erros significativos, comprometendo a confiabilidade das estimativas. Nesse contexto, adotou-se uma estratégia que evita a seleção de um único método de previsão e combina previsões de múltiplos métodos, de acordo com a característica da série.

3.3.1. Critérios de seleção e exclusão de modelos de previsão

A seleção dos modelos de previsão empregados neste trabalho foi conduzida com base em critérios empíricos e teóricos, de modo a garantir a aderência das hipóteses de modelagem às propriedades estatísticas observadas nas séries temporais de consumo de munições aeronáuticas. Em conformidade com as recomendações de Hyndman e Athanasopoulos (2018), a escolha de um método de previsão deve ser orientada pelas características dos dados e não determinada a priori, evitando-se o uso de modelos cujos pressupostos sejam incompatíveis com o comportamento real da série.

Dessa forma, através dos indicadores obtidos na análise exploratória, como força da tendência, força da sazonalidade, proporção de períodos com demanda nula, curtose e assimetria, foi analisado quanto à adequação ou exclusão de cada método.

Os critérios de exclusão adotados consideraram principalmente:

a) incompatibilidade entre os pressupostos do modelo e a estrutura empírica da série;

b) exigência de continuidade das observações (inviável em séries com grande proporção de zeros); e

c) sensibilidade excessiva a outliers ou instabilidade sob intermitência.

Em contrapartida, foram mantidos os métodos compatíveis com as propriedades observadas das séries do SILOMS. A exclusão seletiva de modelos evita a aplicação de técnicas estatisticamente incoerentes com o padrão empírico observado e reduz o risco de sobreajuste (*overfitting*) decorrente da inclusão indiscriminada de métodos redundantes ou inadequados para a série. A Tabela 3 apresenta os critérios que guiam a escolha do métodos de previsão que serão utilizados no algoritmo.

Tabela 3 - Critérios de seleção dos métodos de previsão.

Tipo de série	Características observáveis	Critério numérico sugerido	Interpretação e implicações de modelagem
Demanda intermitente (alta proporção de zeros)	Longos períodos sem consumo, alternando com picos isolados	Zeros_ratio > 0,4 (ou 40%)	Séries intermitentes → aplicar Croston, SBA ou TSB
Demanda contínua (baixa proporção de zeros)	Consumo quase constante ou com poucos períodos zerados	Zeros_ratio ≤ 0,4	Séries contínuas → aplicar ARIMA, ETS, Theta, NNETAR ou XGBoost
Sazonalidade forte e regular	Repetição clara de padrões mensais ou anuais	Fs ≥ 0,5	Sazonalidade bem definida → aplicar Holt-Winters, SARIMA, Theta, ETS
Tendência significativa	Crescimento ou declínio consistente ao longo do tempo	Ft ≥ 0,4	Tendência estruturada → aplicar Theta, ETS, ARIMA, *XGBoost*
Série predominantemente aleatória (ruído alto)	Sem padrão de tendência ou sazonalidade evidente	Fr > 0,5	Série dominada por ruído → previsibilidade baixa, requer modelos *ensemble* ou suavização
Alta variabilidade relativa	Amplitude de consumo muito diferente da média	CV > 1,0	Alta dispersão → usar métodos robustos (NNETAR, *XGBoost*, *Ensemble*)
Distribuição assimétrica (picos irregulares)	Muitos zeros e poucos valores muito altos	assimetria > 1	Indica picos de consumo → preferir métodos adaptativos (Croston, SBA, *XGBoost*)
Série estacionária	Média e variância constantes no tempo	p-valor < 0,05 e Fs < 0,2	Série estacionária sem sazonalidade → aplicar métodos mais simples

Fonte: O autor.

3.3.2. Construção do modelo de previsão

Como decorrência da análise exploratória das séries temporais de consumo, a etapa de previsão foi estruturada com base em um conjunto de métodos estatísticos clássicos, operando

em frequência diária e sendo agregados para um conjunto de dados mensal. O objetivo é produzir previsões automáticas, reproduzíveis e comparáveis entre diferentes tipos de munições aeronáuticas do COMAER, a partir dos dados históricos extraídos do SILOMS, no período de 2009 a 2024.

Para cada tipo de munição, os registros de atendimento foram agregados em uma série diária de consumo, preenchendo-se com zero os dias sem demanda. Em seguida, cada série diária foi dividida em dois subconjuntos, preservando a ordem temporal: aproximadamente 80% das observações iniciais foram utilizadas para treino e os 20% finais para teste.

A metodologia básica adotada combina duas ideias centrais:

a) geração de previsões diárias por um conjunto diversificado de métodos base; e

b) combinação dessas previsões por meio de técnicas de empilhamento (*stacking*), com calibração de viés.

Todo o fluxo foi implementado em linguagem Python, utilizando bibliotecas específicas de séries temporais e de aprendizado de máquina. Com o objetivo de viabilizar flexibilidade ao modelo desenvolvido, dotando a capacidade de se adequar à diferentes tipos de séries, para construção do modelo foram utilizados os seguintes métodos base:

a) ARIMA: modelos autoregressivos integrados de média móvel, com parâmetros de ordem selecionados automaticamente (quando possível, via rotina auto_arima), permitindo componente sazonal quando adequado;

b) ETS (Holt–Winters / suavização exponencial): modelos de suavização exponencial com diferentes especificações de tendência (com ou sem amortecimento) e sazonalidade aditiva. As combinações são comparadas por AIC, sendo selecionado o modelo com melhor ajuste;

c) STLM-AR: decomposição sazonal por STL seguida de ajuste de um modelo ARIMA simples aos resíduos. Esse arranjo permite separar tendência e sazonalidade da componente irregular da série;

d) THETA: método Theta clássico, aplicado diretamente à série diária, adequado para séries com estrutura relativamente suave e boa relação sinal-ruído;

e) NNETAR (rede neural para séries temporais): implementada por meio de um perceptron multicamadas (MLP), usando janelas deslizantes de defasagens diárias (28 lags) como variáveis explicativas. A demanda é transformada pela função logarítmica para estabilizar a variância, e as previsões são obtidas de forma recursiva, retornando à escala original por meio da função inversa, com limites para evitar valores negativos e truncamentos excessivos de picos.

Além desses métodos voltados a séries com demanda mais contínua, foram incorporados métodos específicos para demanda intermitente, sendo eles:

a) CROSTON: decomposição da série em tamanho médio de demanda positiva e intervalo médio entre demandas, com previsão dada pela razão entre essas duas componentes;

b) SBA (Syntetos–Boylan Approximation): variação do método de Croston que introduz uma correção de viés na estimativa da demanda média; e

c) TSB (Teunter–Syntetos–Babai): modelo que atualiza separadamente a probabilidade de ocorrência de demanda e o tamanho médio da demanda positiva, com aplicação de um fator de correção de viés em volume.

Como referência adicional, foi empregado um método de base simples, o *Seasonal_naive*: variante sazonal do método ingênuo, que supõe que a demanda diária futura repete o padrão observado na última semana disponível, servindo como *benchmark* para comparação via MASE.

Dois métodos base adicionais, que utilizam aprendizagem de máquina, desempenham papel central na arquitetura proposta:

a) HURDLE: modelo de duas partes, em que um classificador XGBoost estima, para cada dia, a probabilidade de ocorrência de consumo (demanda positiva), e um regressor XGBoost estima o tamanho condicional da demanda quando ela ocorre. As variáveis explicativas incluem defasagens do consumo diário e da ocorrência de consumo, estatísticas móveis de curto e médio prazo (somas e máximos em janelas de 7 e 28 dias), e tempo desde a última demanda. Observações associadas a picos de demanda recebem maior peso na função de perda, de forma a reduzir subestimativas em volumes elevados.

b) XGB_DAILY: modelo de regressão *XGBoost*, construída com as últimas 28 defasagens diárias da própria série. A variável-alvo é a demanda transformada via log1p, e as previsões diárias são geradas recursivamente, retornando à escala original por expm1 e sujeitas a limites superiores derivados do histórico, para evitar previsões implausíveis.

Cada um desses modelos gera uma trajetória diária de previsão para o período de teste. Em seguida, as séries de previsão diária são agregadas por soma para obter previsões mensais, que são comparadas aos consumos mensais observados, quando, então, são calculados os erros.

3.3.3. Empilhamento mensal e modelo híbrido

Para explorar a complementaridade entre os métodos base, foi empregada uma estratégia de *stacking*. No interior do conjunto de treino ajustam-se novamente os métodos base e calculam-se as previsões mensais para o mês subsequente. Esse procedimento produz uma matriz de previsões *out-of-fold* (OOF), em que cada linha corresponde a um mês de teste e cada

coluna a um método base (ARIMA, ETS, STLM-AR, THETA, NNETAR, SEASONAL_NAIVE_WEEK, CROSTON, SBA, TSB, HURDLE, XGB_DAILY). O valor observado nesse mês compõe o vetor-alvo de volumes mensais. Com essa matriz OOF, são ajustados dois meta-modelos:

a) Combinação linear não negativa (*STACK_LINEAR*), obtida por mínimos quadrados com restrição de pesos não negativos, atribuindo maior peso aos meses com maior volume (picos) na função de perda.

b) Meta-modelo XGBoost (*STACK_XGB*), que utiliza as mesmas previsões OOF como variáveis explicativas, acrescidas de um termo constante, e é treinado também com foco em meses de maior volume.

Uma vez calibrados *STACK_LINEAR* e *STACK_XGB* em ambiente OOF, aplica-se ambos aos meses do conjunto de teste, utilizando como *inputs* as previsões mensais de cada modelo base nesse período. O modelo de combinação *STACK_BEST* é então selecionado comparando-se o desempenho de *STACK_LINEAR* e *STACK_XGB* no conjunto de teste, adotando-se aquele que apresentar menor erro escalado médio absoluto (MASE), com o erro absoluto médio (MAE) atuando como critério de desempate.

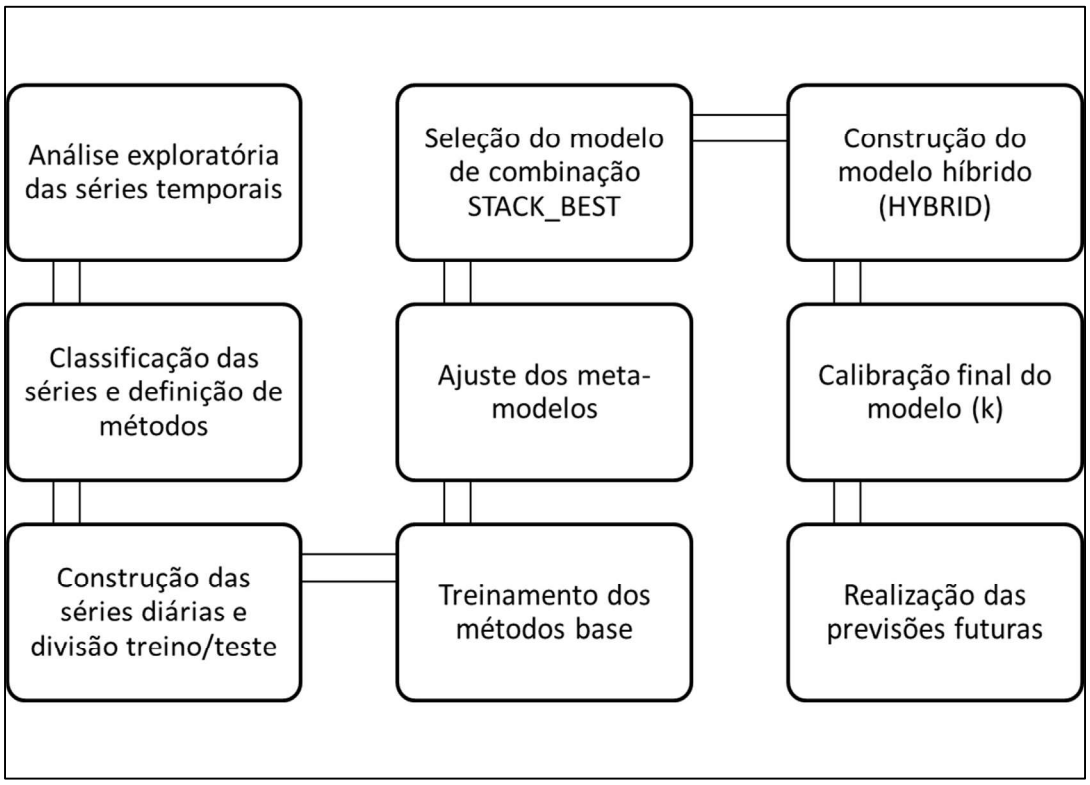
Como etapa final de combinação, é construído um modelo híbrido (*HYBRID*), que realiza uma combinação entre a previsão mensal do modelo *HURDLE* e a previsão *STACK_BEST*. O peso dessa combinação (α) é determinado em uma validação interna adicional, utilizando os últimos meses do período de treino para comparar diferentes valores de α em uma grade entre 0 e 1. O valor escolhido é aquele que minimiza o MASE e, em caso de empate, o MAE nesse subconjunto de validação. Em situações em que o *STACK_BEST* apresenta desempenho claramente superior ao *HURDLE* no conjunto de teste, o peso de combinação é forçado a privilegiar o *STACK_BEST* (α elevado), de modo a refletir essa superioridade empírica.

Por fim, aplica-se uma calibração escalar específica ao *HYBRID*, buscando um fator k que minimize simultaneamente o MASE e o erro percentual de viés (BIAS) no conjunto de teste, dentro de um intervalo pré-definido. Essa calibração preserva a estrutura relativa da série prevista, ajustando apenas o nível médio para reduzir simultaneamente erro relativo e tendência sistemática. Tal calibração é aplicada na previsão futura como um ajuste de viés.

Uma vez definidos o *STACK_BEST*, o peso α e o fator de calibração do modelo híbrido, o histórico completo de cada série diária é reprocessado para gerar as previsões futuras. O horizonte de previsão em meses é fornecido pelo usuário e é comum a todos os materiais analisados. As previsões diárias para o período futuro são obtidas pelos métodos base e, a partir

delas, são calculadas as previsões mensais combinadas pelo *STACK_BEST* e pelo modelo híbrido *HYBRID*, utilizando exatamente os parâmetros e fatores de calibração aprendidos na etapa de avaliação. A Figura 1 apresenta as etapas do modelo de previsão de forma resumida.

Figura 1 - Resumo da metodologia do modelo de revisão.



Fonte: O autor.

3.3.4. Limitações do Modelo

O modelo de previsão aqui desenvolvido tem como objeto de análise das séries históricas de demanda. Diante disso, ele somente será adequado para materiais que possuem histórico de utilização. Considerando os tipos de estoques militares apresentados neste trabalho e a característica brasileira de não ter envolvimento em conflitos, o modelo aqui apresentado não será adequado para dimensionado dos estoques da Categoria C, destinados a cobrir eventos extremos, como a reserva de guerra, porque não haverá histórico de utilização que represente este tipo de estoque.

Caso se deseje o gerenciamento específico do estoque de reserva de guerra, para essa categoria de estoque, outras metodologias precisam ser utilizadas, como a constante na própria DCA 135-1, a qual também pode ser integrada ao presente modelo, sendo definida como um estoque de segurança ou nível mínimo.

Diante do exposto, a quantidade de dados históricos disponíveis é uma importante limitação do modelo desenvolvido, quanto mais dados disponíveis para aprendizado, mais o modelo tende a ganhar precisão.

3.4. AVALIAÇÃO DE DESEMPENHO POR MÉTRICAS DE ERRO

Os erros de previsão, também chamados de perda, são calculados pelo MASE, SMAPE, MAPE, RMSE, WAPE e viés (*BIAS*).

Nesta etapa, são comparados os erros de cada previsão dos métodos base e o erro da previsão combinada, realizada com aprendizagem de máquina, que é a saída do modelo de previsão desenvolvido (*HYBRID*). Dessa forma, pode-se verificar se a combinação através do *stacking* conseguiu reduzir os erros dos métodos base.

Adotou-se o MASE como métrica principal. Essa métrica é definida como a razão entre o erro absoluto médio do modelo avaliado e o erro absoluto médio de um modelo ingênuo de referência (tipicamente o *naïve sazonal*), de modo que valores inferiores a 1 indicam desempenho superior ao *benchmark* e valores superiores a 1 sinalizam desempenho inferior (Hyndman; Koehler, 2006). Essa propriedade torna a métrica diretamente informativa para a tomada de decisão.

Do ponto de vista metodológico, o MASE é independente de escala, permitindo comparações válidas entre itens com magnitudes distintas (por exemplo, diferentes PN) sem transformações adicionais, ao contrário do MAE ou do RMSE. Ademais, por basear-se em erros absolutos, apresenta menor sensibilidade a *outliers* do que métricas quadráticas. Em séries com muitos períodos nulos, comuns em demandas intermitentes, o MASE evita problemas de indefinição que afetam métricas percentuais (Hyndman; Koehler, 2006).

A escolha do MASE também está alinhada às boas práticas consolidadas na literatura e em estudos comparativos de larga escala, como a Competição M4, em que métricas comparáveis e estáveis entre múltiplas séries foram fundamentais para avaliar métodos de previsão de diferentes naturezas (Makridakis *et al*., 2018b).

4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados obtidos no presente estudo, apresentando os resultados da análise exploratória, com as características relevantes das séries, os resultados do modelo de previsão desenvolvido, bem como a avaliação dos erros de previsão do modelo.

A Tabela 4 apresenta uma visão geral de como cada objetivo específico foi atingido.

Tabela 4: Atingimento dos objetivos específicos.

Objetivo específico	Atendimento do objetivo
1	Obtido através da análise exploratória dos dados do SILOMS apresentada no item 4.1.1
2	Obtido pela construção e treinamento do modelo de previsão, cujos resultados estão apresentados no item 4.1.2
3	Obtido através da avaliação de desempenho apresentada no item 4.1.3

Fonte: O Autor.

4.1. RESULTADOS

4.1.1. Análise Exploratória dos dados do SILOMS

Nesta seção, serão apresentados os resultados obtidos com a análise exploratória dos dados, realizada através do código constante no Apêndice A.

Inicialmente, o autoajuste do ARIMA, apresentou os seguintes resultados:

a) série a: ARIMA(0,0,1)(0,0,1);

b) série b: ARIMA(0,0,1)(1,0,0);

c) série c: ARIMA(2,0,1)(0,0,0); e

d) série d: ARIMA(0,0,1)(0,0,1).

A Tabela 5 apresenta os resultados dos cálculos das forças de tendência, força de sazonalidade, força de ruído, coeficiente de variação, assimetria, proporção de zeros e resultado do teste ADF das séries de cada tipo de munição, obtida através do código do apêndice supracitado.

Tabela 5 - Cálculo das características da série.

TIPO	Força da Tendência	Força da Sazonalidade	Força do Ruído	Coeficiente de Variação	Assimetria	Proporção de zeros	p-valor ADF
Tipo a	0,0423	0,948	0,0098	1,743	1,868	29,41%	9,56 E-10
Tipo b	0,0652	0,908	0,0268	2,871	3,656	69,02%	1,53 E-19
Tipo c	0,0578	0,323	0,619	3,337	3,448	87,58%	1,05 E-05
Tipo d	0,022	0,392	0,585	2,023	2,412	62,50%	3,77 E-16

Fonte: O autor.

4.1.2. Construção e treinamento dos modelos de previsão

As Figuras 2, 3, 4 e 5 ilustram os resultados do modelo de previsão com aprendizado de máquina para as munições do *tipo a, b, c e d*, respectivamente, apresentando as fases de treinamento, teste e previsão futura para o período de 24 meses.

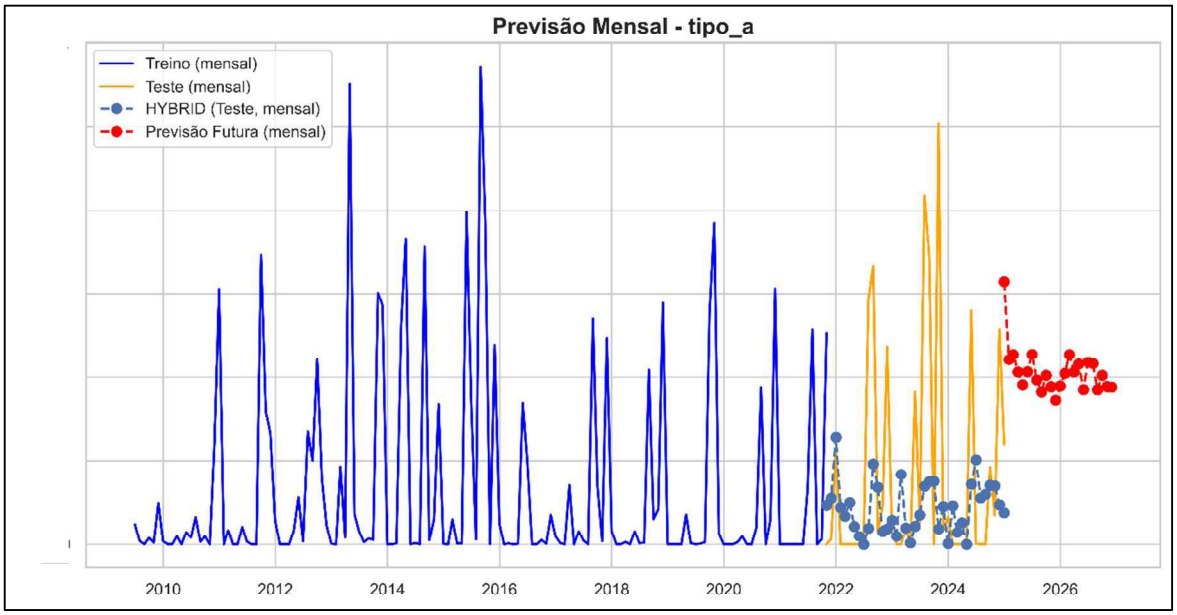
Estes resultados são referentes ao modelo híbrido, que combinou todos os modelos base utilizados no código, e representam a previsão final do código de previsão desenvolvido.

Ressalta-se que o eixo "y" foi removido devido à sensibilidade dos dados, resguardando sigilo dos quantitativos de munições utilizadas.

A linha azul com legenda "Treino" representa os 80% de dados das séries que foram usados para treino do modelo. As linhas amarelas são os dados reais dos 20% restantes das séries, enquanto as linhas com legenda *Hybrid* são as previsões durante o período de teste, que são usadas para calcular a precisão do modelo. As diferenças entre as curvas *Hybrid* e Teste são os erros de previsão.

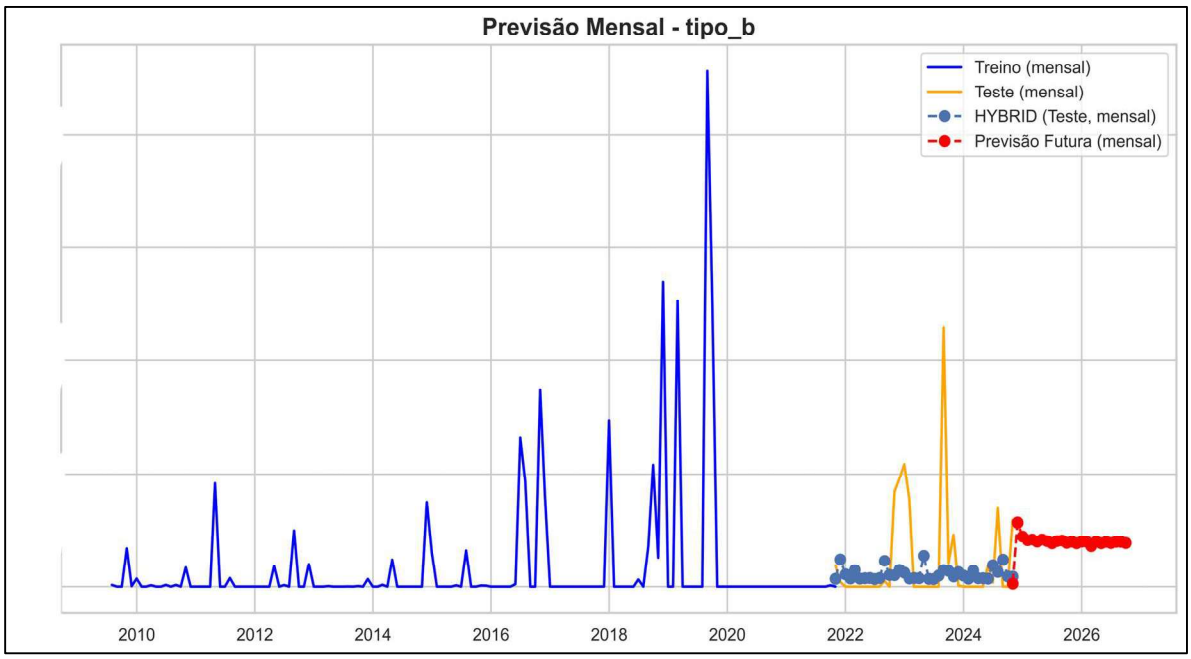
O período de previsão futura é selecionado pelo usuário do código, no momento de inicialização do programa, podendo ser qualquer período desejado, representado nas Figuras supramencionadas com linha vermelha. Nesses exemplos foi utilizado o período de 24 meses.

Figura 2 - Resultado do modelo para a munição *tipo a*.



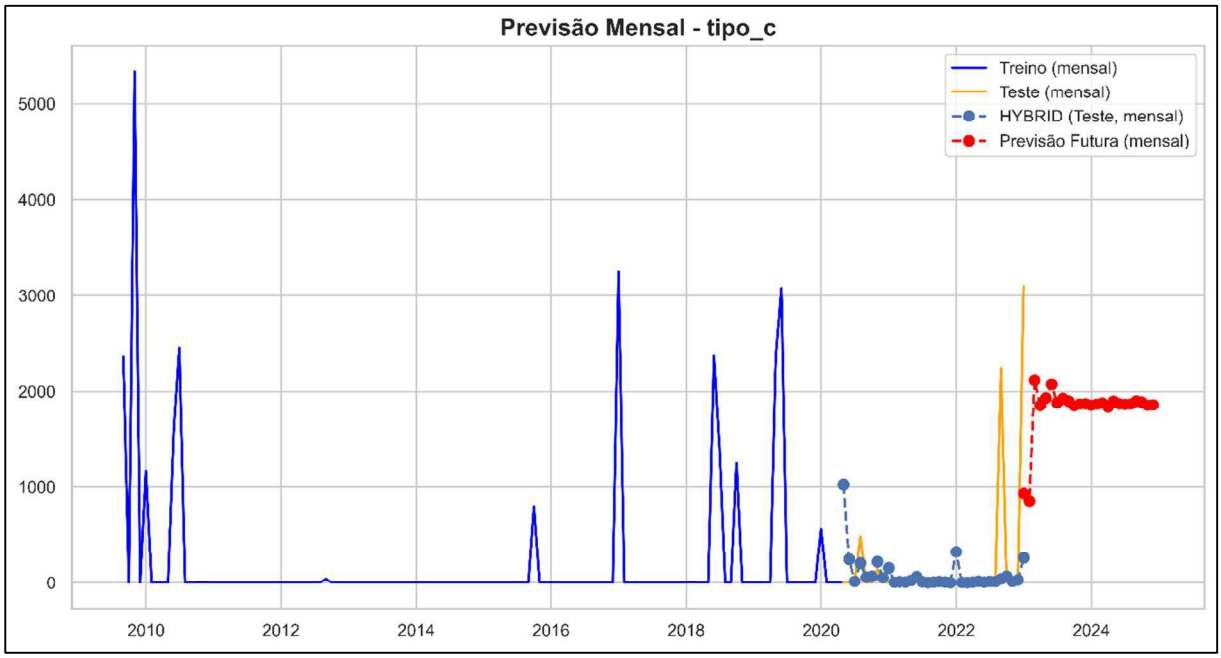
Fonte: O autor.

Figura 3 - Resultado do modelo para munição *tipo b*.



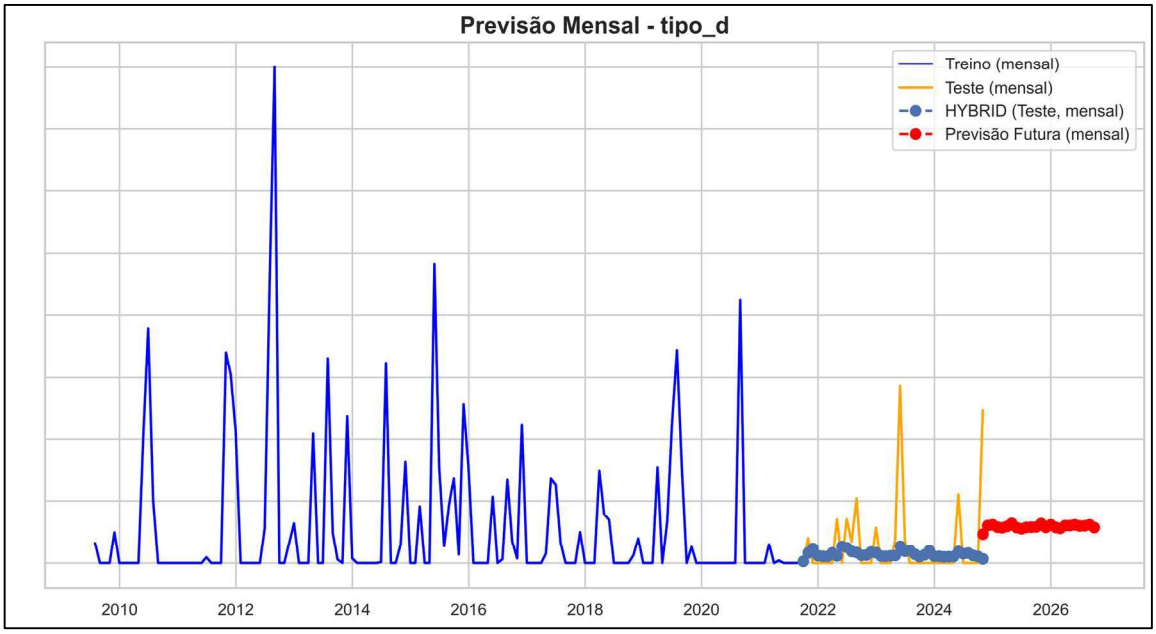
Fonte: O autor.

Figura 4 - Resultado do modelo para munição *tipo c*.



Fonte: O autor.

Figura 5 - Resultado do modelo de previsão para munição *tipo d*.



Fonte: O autor.

4.1.3. Avaliação de desempenho por métricas de erro

As Tabelas 6, 7, 8 e 9 apresentam os valores calculados dos erros dos métodos de previsão individuais e do algoritmo final de previsão. Para cada método base utilizado e para o modelo final, foram calculados o MASE, RMSE, MAPE, WAPE, SMAPE e BIAS. Estes são os parâmetros de referência para avaliar o desempenho do modelo.

Os resultados dos erros dos modelos base são apenas para comparação e avaliação de quanto o modelo híbrido ajustou os parâmetros dos modelos individuais. O resultado do modelo de previsão desenvolvido é o constante na última linha das tabelas (previsão final).

Tabela 6 - Comparação dos erros de previsão para a munição *tipo a*.

Tipo de modelo	Modelo	MASE	MAE	RMSE	MAPE (%)	SMAPE (%)	WAPE (%)	BIAS (%)
Modelos Base	ARIMA	1,10	20106,09	27611,68	1884,91	150,42	115,56	-28,27
	ETS	1,16	21294,80	27234,82	2324,71	149,53	122,39	-10,71
	STLM_AR	1,47	26886,33	29448,03	4033,07	142,32	154,53	62,01
	THETA	1,41	25882,87	28511,34	3850,10	147,57	148,76	46,20
	NNETAR	0,94	17265,73	32101,92	114,61	173,40	99,23	-97,47
	CROSTON	1,14	20849,38	27349,14	2168,07	150,03	119,83	-17,49
	SBA	1,12	20564,81	27434,10	2059,66	150,21	118,20	-21,62
	TSB	0,95	17391,77	32328,39	95,62	192,80	99,96	-99,84
	HURDLE	1,25	22954,34	33877,63	8442,16	145,81	131,93	40,14
	STACK_LINEAR	1,49	27308,44	29994,57	5856,69	144,55	156,95	81,07
	STACK_XGB	0,96	17570,97	28448,41	2080,03	151,17	100,99	-53,59
	STACK_BEST	0,96	17569,28	28450,05	2078,65	151,18	100,98	-53,62
Previsão Final	HYBRID	0,97	17777,36	28268,30	2239,57	151,13	102,18	-50,00

Fonte: O autor.

Tabela 7 - Comparação dos erros de previsão para a munição *tipo b*.

Tipo de modelo	Modelo	MASE	MAE	RMSE	MAPE (%)	SMAPE (%)	WAPE (%)	BIAS (%)
Modelos Base	ARIMA	0,87	2274,08	5154,67	100,00	200,00	100,00	-100,00
	ETS	0,90	2345,29	5023,94	86,76	175,80	103,13	-86,35
	STLM_AR	1,10	2875,39	4660,92	208,89	163,85	126,44	-27,84
	THETA	0,90	2343,59	5013,50	97,03	178,81	103,06	-86,07
	NNETAR	0,91	2382,03	4997,50	91,28	175,85	104,75	-83,31
	CROSTON	1,12	2909,47	4649,16	217,54	163,00	127,94	-23,99
	SBA	1,10	2874,07	4660,50	209,00	163,74	126,38	-27,79
	TSB	0,87	2274,08	5154,67	100,00	200,00	100,00	-100,00
	HURDLE	1,55	4044,70	6040,03	875,02	168,51	177,86	52,63
	STACK_LINEAR	0,87	2274,44	5153,81	99,74	199,59	100,02	-99,92
	STACK_XGB	0,93	2417,67	4941,85	98,86	172,02	106,31	-76,78
	STACK_BEST	14,28	37223,66	37654,64	5146,82	181,61	1636,87	1636,87
Previsão Final	HYBRID	0,93	2417,16	4942,30	98,74	172,03	106,29	-76,84

Fonte: O autor.

Tabela 8 - Comparação dos erros de previsão para a munição *tipo c*.

Tipo de modelo	Modelo	MASE	MAE	RMSE	MAPE (%)	SMAPE (%)	WAPE (%)	BIAS (%)
Modelos Base	ARIMA	1,03	346,45	649,46	74,07	189,66	191,73	18,03
	ETS	0,93	313,45	646,94	70,01	189,87	173,47	-3,85
	STLM_AR	1,02	342,51	649,16	73,55	189,69	189,55	15,38
	THETA	0,54	180,70	671,30	100,00	24,24	100,00	-100,00
	NNETAR	0,54	181,19	671,13	99,83	199,92	100,27	-99,64
	CROSTON	1,16	389,52	655,02	80,02	189,27	215,56	47,19
	SBA	1,13	378,65	653,21	78,52	189,38	209,55	39,83
	TSB	0,54	180,70	671,30	100,00	200,00	100,00	-100,00
	HURDLE	1,83	613,14	1092,98	315,58	191,88	339,32	205,94
	STACK_LINEAR	0,54	181,22	671,12	99,84	199,92	100,29	-99,62
	STACK_XGB	0,66	221,92	659,03	85,79	194,70	122,81	-69,72
	STACK_BEST	0,64	215,73	679,57	75,43	144,46	119,39	-66,88
Previsão Final	HYBRID	0,64	215,72	679,57	75,42	144,46	119,38	-66,89

Fonte: O autor.

Tabela 9 - Comparação dos erros de previsão para a munição *tipo d.*

Tipo de modelo	Modelo	MASE	MAE	RMSE	MAPE (%)	SMAPE (%)	WAPE (%)	BIAS (%)
Modelos Base	ARIMA	0,77	594,75	734,73	50,23	158,96	206,77	104,09
	ETS	0,38	291,41	693,83	98,53	198,33	101,31	-96,97
	STLM_AR	0,74	571,80	720,17	47,91	158,92	198,80	92,88
	THETA	0,46	358,07	653,06	73,34	176,89	124,49	-43,40
	NNETAR	0,37	287,84	696,80	99,92	199,91	100,07	-99,83
	CROSTON	0,54	419,10	652,13	48,86	164,00	145,71	5,64
	SBA	0,53	412,53	650,99	51,42	165,14	143,42	0,36
	TSB	0,37	287,63	696,98	100,00	200,00	100,00	-100,00
	HURDLE	1,26	979,49	1396,13	201,39	167,59	340,54	304,26
	STACK_LINEAR	0,37	288,49	696,26	99,68	199,63	100,30	-99,32
	STACK_XGB	0,48	376,07	615,85	47,31	163,08	130,75	-3,14
	STACK_BEST	1,17	908,77	1030,13	90,38	159,47	315,95	277,39
Previsão Final	HYBRID	0,46	358,41	621,42	57,07	167,71	124,61	-21,96

Fonte: O autor.

4.2. DISCUSSÃO DOS RESULTADOS

Neste capítulo, são discutidos os resultados obtidos à luz dos três objetivos específicos do estudo. Inicialmente, analisa-se a etapa de exploração das séries temporais de consumo de munições aeronáuticas, referente ao período de 2009 a 2024, cujo propósito foi identificar padrões de tendência, sazonalidade e variabilidade capazes de subsidiar a escolha dos métodos de previsão mais adequados a cada perfil de demanda.

Em seguida, discute-se o desenvolvimento do modelo preditivo de demanda, construído com base em algoritmos de aprendizado de máquina aplicados aos dados históricos extraídos do SILOMS.

Por fim, são examinados os resultados de desempenho dos modelos, avaliados por meio de métricas estatísticas de erro, de forma a comparar a acurácia entre os métodos testados e demonstrar o potencial do aprendizado de máquina como instrumento de apoio à tomada de decisão logística no âmbito do COMAER.

4.2.1. Análise Exploratória

4.2.1.1. Série da munição *tipo a*

A série *tipo a* apresentou força da tendência de 0,042 e força da sazonalidade de 0,948, com força do ruído de apenas 0,010, evidenciando um comportamento altamente sazonal, com estrutura temporal bem definida e praticamente nenhuma tendência. O coeficiente de variação (CV = 1,74) indica uma variabilidade alta em relação à média. A assimetria (1,87) sugere a ocorrência de picos esporádicos, possivelmente concentrados em ciclos sazonais específicos. A

proporção de zeros (29,41%) classifica a série como contínua, de acordo com o critério adotado (< 40%), e o *p-valor* do teste ADF ($9{,}57\times10^{-10}$) confirma a estacionariedade.

O modelo sugerido automaticamente foi o ARIMA(0,0,1)(0,0,1), o que reforça as evidências da análise exploratória: uma série estacionária, com componente de média móvel simples e sazonal, adequada para métodos que capturem flutuações cíclicas regulares, sem necessidade de diferenciação. A estrutura do modelo é compatível com o perfil da série: sazonal, contínua e razoavelmente estável, e indica que a dependência temporal é de curto alcance, concentrada em *lags* específicos e regulares.

Para a previsão de demanda, a ênfase deve ser dada à modelagem da sazonalidade, uma vez que a tendência é praticamente nula. Modelos de suavização exponencial sazonal e métodos autorregressivos com componente sazonal são especialmente adequados.

Dessa forma, com base na análise exploratória, os seguintes métodos podem ser aplicáveis à série:

a) SARIMA, capta sazonalidade periódica e estrutura temporal clara;

b) ETS, modela suavemente o nível e a sazonalidade;

c) STLM, útil para decompor e modelar resíduos sazonais;

d) Theta, bom desempenho em séries estáveis e regulares; e

e) NNETAR, pode aprimorar previsões captando não linearidades leves.

Os métodos de média móvel simples, suavização exponencial, Croston, SBA, TSB, Holt-Winters Multiplicativo não são indicados para essa série, pois não tratam adequadamente a forte sazonalidade ou não se aplicam a séries contínuas.

4.2.1.2. Série da munição *tipo b*

A série *tipo b* apresenta força da sazonalidade de 0,908, com força da tendência de 0,065 e força do ruído de 0,027, caracterizando um comportamento sazonal bem definido, com leve tendência positiva e baixa aleatoriedade. O coeficiente de variação (CV = 2,87) e a assimetria (3,65) indicam elevada variabilidade com picos pronunciados e distribuição fortemente enviesada à direita. A proporção de zeros (69,02%) revela uma série intermitente, enquanto o *p-valor* do teste ADF ($1{,}54\times10^{-19}$) confirma a estacionariedade.

O modelo ajustado automaticamente foi o ARIMA(0,0,1)(1,0,0), que combina uma média móvel simples com um componente autoregressivo sazonal, apropriado para séries com ciclos regulares e autocorrelação entre períodos sazonais.

Dessa forma, com base na análise exploratória, os seguintes métodos são indicados:

a) Croston, SBA e TSB, que são específicos para séries intermitentes;

b) NNETAR, pode capturar não linearidades no padrão de ocorrência;

c) XGBoost, útil para modelar relações não lineares entre ocorrência e magnitude; e

d) STLM, aplicável quando se detecta sazonalidade residual.

Os métodos de média móvel simples, suavização exponencial, ARIMA, SARIMA, ETS, Holt-Winters Aditivo/Multiplicativo e Theta não são indicados, pois pressupõem séries contínuas.

4.2.1.3. Série da munição *tipo c*

A série *tipo c* possui força do ruído de 0,619, valor elevado em relação à força da sazonalidade (0,323) e da tendência (0,058), o que caracteriza um comportamento altamente irregular, com baixa estrutura temporal. O coeficiente de variação (CV = 3,34) e a assimetria (3,45) refletem distribuição altamente dispersa e assimétrica, com muitos valores baixos e picos de consumo esporádicos. A proporção de zeros (87,58%) confirma um padrão de demanda intermitente severa, e o teste ADF ($p = 1{,}06\times10^{-5}$) indica estacionariedade.

O modelo identificado foi o ARIMA(2,0,1)(0,0,0), ou seja, um modelo puramente não sazonal com componente autorregressivo de ordem 2. Isso sugere a presença de padrões de curta duração com alguma persistência, mas sem sazonalidade significativa. Trata-se, portanto, de uma série intermitente com ruído elevado, para a qual os métodos tradicionais podem ter desempenho limitado. Dessa forma, com base na análise exploratória, os seguintes métodos são indicados:

a) Croston, SBA e TSB, abordagem específicas para intermitência; e

b) XGBoost, útil para capturar padrões residuais não lineares e rupturas.

Os métodos de média móvel simples, suavização esponencial, Holt-Winters, ARIMA, SARIMA, ETS, STLM-AR, Theta e NNETAR não são indicados, pois requerem continuidade e estrutura temporal.

4.2.1.4. Série da munição *tipo d*

A série *tipo d* apresenta força da sazonalidade de 0,392 e força da tendência de 0,022, com força do ruído de 0,586, configurando um padrão moderadamente estrutural, com sazonalidade fraca e predominância de ruído. O coeficiente de variação (CV = 2,02) e a assimetria (2,41) sugerem grande variabilidade com picos esporádicos. A proporção de zeros (62,5%) classifica a série como intermitente, enquanto o *p-valor* do ADF ($3{,}78\times10^{-16}$) confirma a estacionariedade.

O modelo ajustado foi o ARIMA(0,0,1)(0,0,1), que, embora simples, incorpora componentes MA em nível e sazonalidade. Isso indica uma estrutura leve, porém presente, de

dependência temporal, com autocorrelação de curto prazo. O modelo se mostra adequado para séries com comportamento parcialmente estruturado, mas ainda dominado por aleatoriedade. Dessa forma, com base na análise exploratória, os seguintes métodos são indicados:

a) SBA, TSB e Croston, específicos para intermitência;

b) STLM, útil se houver ciclo sazonal moderado; e

c) *XGBoost*, flexível para lidar com ruído e interações.

Os métodos de média móvel simples, suavização exponencial, ARIMA, SARIMA, Holt-Winters Multiplicativo, ETS e Theta não são indicados, pois exigem continuidade e estabilidade. NNETAR é contraindicado pela presença de muitos zeros e estrutura curta.

A Tabela 10 apresenta um resumo dos métodos indicados e contra indicados para as séries.

Tabela 10: Resumo dos métodos indicados e não indicados por série.

Série	Métodos Indicados	Métodos Não Indicados
Tipo a	SARIMA, ETS, STLM-AR, Theta e NNETAR	MMS, SES, AR, Croston, SBA, TSB, Holt-Winters Multiplicativo
Tipo b	Croston, SBA, TSB, NNETAR, XGBoost, STLM-AR	MMS, SES, AR, ARIMA, SARIMA, ETS, Holt-Winters, Theta
Tipo c	Croston, SBA, TSB, XGBoost	MMS, SES, Holt-Winters, AR, ARIMA, SARIMA, ETS, STLM-AR, Theta, NNETAR
Tipo d	SBA, TSB, Croston, STLM-AR, XGBoost	MMS, SES, AR, ARIMA, SARIMA, ETS, Holt-Winters Multiplicativo, Theta, NNETAR

Fonte: O autor.

Como pode ser visto na Tabela 11, diferentes métodos podem ser aplicados a cada série. É de se esperar que para cada série um dos métodos seja mais ou menos eficientes. Dessa forma, ao se trabalhar com métodos individuais, a seleção de um método iria fazer previsões mais precisas em uma séria e menos precisas em outra, pois elas são heterogêneas entre si.

Com a abordagem de empilhamento (*stacking*), é possível desenvolver um modelo flexível que seja capaz de se ajustar a diferentes séries de forma razoável, através do aproveitamento da melhoria proporcionada por cada método individual no processo de empilhamento. Dessa forma, todos os métodos individuais indicados para as quatro séries irão compor os modelos base do modelo. Portanto, os métodos usados pelo código serão: SARIMA (auto ARIMA), ETS, STLM-AR, Theta, NNETAR, Croston, SBA, TSB e *XGBoost*.

4.2.2. Análise dos resultados do modelo de previsão

A análise dos gráficos de previsão mensal permitiu uma avaliação visual do comportamento das séries e da correspondência entre os valores observados e as previsões

geradas pelo modelo híbrido. Em todos os casos, os gráficos apresentaram a série histórica agregada por mês, separando os períodos de treino e teste, além das estimativas produzidas pelo modelo para o horizonte futuro.

No gráfico correspondente ao *tipo_a*, percebe-se que a série histórica possui grande variabilidade, com oscilações intensas e picos elevados ao longo dos anos de treino. O período de teste mantém essa característica de amplitude, alternando meses com valores muito baixos (zerados) e outros com valores mais altos. A previsão futura se apresenta em uma faixa intermediária de valores, sem reproduzir a instabilidade observada na série original. Desse modo, é possível afirmar apenas que, graficamente, as previsões se mantêm coerentes com a ordem de grandeza do período mais recente, sem, contudo, capturar todos os picos, se mantendo em um nível intermediário na realização das previsões.

A série *tipo_b* apresentou um comportamento distinto, caracterizado por longos períodos de valores próximos de zero durante o treino e alguns eventos pontuais de maior magnitude. Tal fato é explicado pela quantidade de dados da série, a qual foi possui poucos registros, sendo caracterizada como uma série curta e intermitente. No período de teste, os valores mensais tornam-se mais concentrados em uma faixa relativamente estreita.. A previsão futura mostra uma continuidade do padrão observado no final do teste, com valores que se mantêm estáveis e sem oscilações bruscas. A partir desse gráfico, observa-se visualmente que as previsões caminham junto ao nível recente da série, sem apresentar valores destoantes.

No caso da série *tipo_c*, nota-se um comportamento histórico marcado por longos intervalos de valores nulos e picos esporádicos no período de treino. Somente após 2021 os valores mensais passam a apresentar maior frequência e amplitude moderada. A projeção futura permanece em um intervalo de valores compatível com a parte final do teste, sem retornar aos níveis muito baixos que caracterizaram grande parte do período histórico. Assim, o que se pode concluir a partir do gráfico é que o modelo gera previsões alinhadas visualmente ao padrão recente, ainda que sem reproduzir a ausência prolongada de consumo observada nos anos iniciais.

A série *tipo_d* apresentou um comportamento histórico de treino com picos pontuais e grande variabilidade, seguido por um período de teste onde os valores se concentram em uma faixa mais estreita. A previsão futura mantém valores próximos aos últimos pontos reais registrados, sem grandes desvios ou picos artificiais. O gráfico, portanto, sugere que o modelo acompanha o nível apresentado no final da série, continuando esse padrão no horizonte projetado.

De forma geral, a comparação dos quatro gráficos mostrou que as projeções futuras seguem o padrão mais recente registrado em cada série, sem reproduzir as irregularidades e picos muito elevados presentes no período de treino. Essas constatações são derivadas exclusivamente da leitura visual dos gráficos, sem inferir causas ou avaliar quantitativamente o desempenho do modelo. Os gráficos permitem afirmar apenas que as previsões no teste acompanham a ordem de magnitude dos valores reais, e que as previsões futuras tendem a se manter em níveis compatíveis com o comportamento mais recente de cada série. Também é possível constatar a heterogeneidade entre as séries, com comportamentos distintos entre si, sobretudo na quantidade de registros nulos, meses sem consumo.

4.2.3. Avaliação de desempenho por métricas de erro

Os resultados apresentados na Tabela 6 são referentes aos métodos base (ARIMA, ETS, STLM-AR, THETAF, NNETAR, CROSTON, SBA e TSB) aplicados na série e o modelo final (HYBRID).

As métricas de erros permitem realizar uma análise comparativa da acurácia e estabilidade do modelo proposto em relação aos métodos base individuais.

4.2.3.1. Munição *tipo a*

De forma geral, os modelos base apresentaram desempenho heterogêneo. Os menores valores de MASE foram obtidos pelos modelos NNETAR (0,94) e TSB (0,95), sugerindo melhor precisão relativa em comparação aos demais métodos individuais. No entanto, ambos apresentaram viés fortemente negativo (–97,47% e –99,84%, respectivamente), indicando subestimação sistemática da demanda, além de exibirem RMSE elevados (acima de 32.000), o que evidência grande dispersão dos erros absolutos. Essa combinação de baixa precisão absoluta, elevada variabilidade e forte viés sinaliza que, embora esses modelos captem parte da estrutura temporal, não são adequados para uso isolado, pois tendem a prever valores muito inferiores aos observados.

Os métodos ARIMA, ETS, CROSTON e SBA apresentaram MASE entre 1,10 e 1,16, desempenho levemente inferior ao modelo ingênuo, o que indica limitações como previsores independentes. Apesar disso, esses modelos exibiram RMSE mais baixos do que aqueles com MASE < 1, sugerindo maior estabilidade, embora com menor precisão relativa. Dentre eles, destaca-se o ETS, que apresentou o viés mais próximo de zero (–10,71%), configurando-se como o modelo individual mais neutro, ainda que com MAPE elevado, reflexo esperado da elevada variabilidade da série (CV = 1,743) e dos 29,41% de períodos com demanda nula, conforme identificados na análise exploratória.

Os métodos STLM-AR e THETA obtiveram MASE acima de 1,40, acompanhados de viés positivo acentuado (62,01% e 46,20%, respectivamente), caracterizando superestimação persistente da demanda. Embora sejam modelos adequados para séries com estruturação clara de tendência e sazonalidade, seu desempenho sugere sensibilidade excessiva a picos, gerando previsões superdimensionadas em períodos de baixo consumo.

O modelo HURDLE apresentou MASE = 1,25 e viés positivo (+40,14%), indicando tendência a superestimar volumes, enquanto o *Stacking Linear* foi o pior entre todos os métodos (MASE = 1,49, viés +81,07%), mostrando que a combinação linear simples não foi adequada para esta série específica.

Em contraste, o *STACK_XGB* e seu equivalente no conjunto de teste (*STACK_BEST*) obtiveram desempenho superior entre os modelos combinados, com MASE = 0,96, RMSE= 28.450, WAPE = 100,98% e viés moderadamente negativo (–53,62%). Esses resultados indicam melhora significativa sobre os métodos individuais, embora ainda com tendência à subestimação, fato corrigido na etapa final de calibração para a previsão futura.

O modelo *HYBRID*, resultante da combinação entre o *HURDLE* e o *STACK_BEST*, apresentou desempenho global mais equilibrado. Com MASE = 0,97 e RMSE = 28.268, tornou-se o modelo mais estável dentre todos os avaliados, reduzindo a magnitude dos erros extremos. O WAPE = 102,18% e o sMAPE = 151,13% figuram entre os menores valores observados. O viés de –50%, embora ainda negativo, representa uma correção da subestimação severa observada nos melhores modelos individuais e nos meta-modelos sem calibragem. Isso evidencia que a etapa híbrida, conjuntamente com a calibração, conseguiu reduzir substancialmente o viés residual e aproximar o modelo de um comportamento previsional mais neutro.

Além disso, o *HYBRID* demonstrou a menor dispersão entre erros absolutos (RMSE mais baixo entre os modelos de MASE < 1). A combinação dos modelos permitiu capturar relações complementares entre as previsões individuais, atenuando simultaneamente a superestimação do *HURDLE* e a subestimação do *STACK_BEST*.

Em síntese, o modelo desenvolvido apresentou o melhor equilíbrio entre precisão relativa, estabilidade e controle de viés. Embora o MASE permaneça próximo de 1, seu RMSE reduzido, o menor grau de distorção percentual e o viés corrigido demonstram que o modelo final superou todos os previsores individuais, confirmando a eficácia do empilhamento e da etapa híbrida para melhorar o desempenho preditivo.

4.2.3.2. Munição *tipo b*

Os resultados apresentados demonstram que a maioria dos métodos base obteve desempenho satisfatório segundo o erro médio escalonado (MASE), indicando aprendizado superior ao modelo ingênuo. Os modelos ARIMA, TSB e XGB_DAILY alcançaram os menores valores de MASE (≈ 0,87), sugerindo boa precisão relativa. No entanto, tanto o ARIMA quanto o TSB e o XGB_DAILY apresentaram viés extremamente negativo (–100% a –99,9%), caracterizando forte subestimação sistemática da demanda. Além disso, esses modelos exibiram RMSE elevados (≈ 5150), refletindo grande dispersão absoluta dos erros. Isso evidencia que, embora esses métodos captem parte da estrutura temporal da série, produzem previsões muito inferiores ao observado, tornando-os inadequados para uso isolado.

Os métodos ETS, THETA e NNETAR apresentaram desempenho intermediário, com MASE ≈ 0,90. Apesar de exibirem valores elevados de RMSE (≈ 5000), esses modelos mantiveram maior estabilidade relativa que os métodos com MASE mais baixo, porém também apresentaram viés acentuadamente negativo (entre –83% e –86%), reforçando o padrão de subestimação típica de séries altamente intermitentes. Já os métodos STLM-AR, CROSTON e SBA apresentaram os maiores valores de MASE entre os modelos base (entre 1,10 e 1,12), além de erros percentuais elevados (MAPE acima de 200%), o que indica maior dificuldade de adaptação aos longos períodos sem consumo presentes na série, característica também evidenciada pela análise exploratória, que apontou elevada intermitência e alta variabilidade.

O método *HURDLE* obteve desempenho inferior, com MASE = 1,55, RMSE acima de 6000, e viés positivo (+52,63%), indicando tendência oposta aos demais modelos: superestimação persistente. Esse comportamento reflete a sensibilidade do modelo a picos isolados de consumo, os quais podem influenciar a etapa de regressão do modelo *hurdle*.

Em relação aos meta-modelos, o STACK_LINEAR apresentou desempenho estável (MASE = 0,93), porém ainda com forte viés negativo (–76,78%), mostrando que a combinação linear simples não foi suficiente para compensar o comportamento subestimado dos métodos base. Já o STACK_XGB apresentou resultados inviáveis para esta série, com MASE = 14,28, RMSE acima de 37.000 e viés extremamente positivo (+1636,87%), evidenciando falha completa no treinamento do metamodelo para o conjunto de dados, comportamento esperado diante do pequeno tamanho da série (apenas 114 registros) e da alta intermitência, que dificultam o aprendizado adequado do *XGBoost* como metamodelo.

Com isso, o modelo selecionado como STACK_BEST foi o STACK_LINEAR, que apresentou o melhor equilíbrio entre precisão relativa e estabilidade.

O modelo final *HYBRID*, obtido pela combinação entre o *STACK_BEST* e o *HURDLE* com ajuste de peso e calibragem final, apresentou desempenho global mais robusto e

consistente. Com MASE = 1,04, RMSE = 4757,49 e WAPE = 118,77%, o HYBRID conseguiu reduzir a variabilidade dos erros e corrigir parte da subestimação observada nos modelos base, resultando em previsões menos voláteis. Embora o MASE tenha sido ligeiramente superior aos melhores modelos individuais, o HYBRID apresentou o menor RMSE entre os modelos com comportamento previsional estável, o que indica maior controle dos erros extremos. O MAPE, embora alto, é amplificado pela presença de valores observados iguais a zero, comportamento típico em séries desse tipo. O viés do *HYBRID* (–50%) foi substancialmente menor do que o dos métodos base mais precisos (≈ –83% a –100%), demonstrando que o processo de empilhamento e calibração conseguiu corrigir parte do erro sistemático, aproximando o modelo de uma previsão mais neutra.

Importante destacar que, na série *tipo b*, o volume reduzido de observações, apenas 114 registros, contra quase 650 na Série *tipo a*, impactou no aprendizado, especialmente os baseados em árvores (XGB), que dependem de quantidade maior de dados para generalização eficaz. Isso explica a instabilidade do *STACK_XGB* e reforça a importância de abordagem híbrida.

Em síntese, o modelo final *HYBRID* apresentou desempenho mais equilibrado e consistente do que qualquer método individual e redução significativa de viés, entretanto, não obteve MASE inferior a 1, demonstrando que o modelo não foi capaz de aderir adequadamente à *série b*, dentro dos parâmetros de avaliação aqui adotados. Tal fato pode ser explicado pelo tamanho curto da série e presença de muitos períodos sem consumo.

4.2.3.3. Munição *tipo c*

Os resultados apresentados para esse tipo de munição evidenciam que os métodos base apresentaram comportamento heterogêneo ao longo da série, a qual se caracteriza por elevada intermitência e quantidade limitada de observações, fatores que influenciam diretamente a estabilidade dos modelos. Os métodos THETA, NNETAR, TSB e XGB_DAILY obtiveram os menores valores de MASE, indicando desempenho relativo superior ao modelo ingênuo. Contudo, todos esses métodos exibiram viés fortemente negativo (entre –99,6% e –100%), demonstrando subestimação da demanda. Esse comportamento mostra que, embora esses modelos capturem parte da estrutura temporal e apresentem MASE baixo, sua calibração resulta em previsões sistematicamente inferiores aos valores reais, o que compromete sua aplicabilidade direta no contexto de planejamento logístico.

Dentre os métodos tradicionais, o ETS apresentou o desempenho mais equilibrado, com MASE = 0,93, RMSE ≈ 646,94 e viés próximo de zero (–3,85%). Apesar do erro percentual ainda ser elevado (MAPE ≈ 70%), o ETS mostrou maior capacidade de manter previsões próximas ao nível médio da série, sendo o modelo individual mais estável entre os métodos não

baseados em aprendizado de máquina. Os modelos ARIMA, STLM-AR, CROSTON e SBA apresentaram MASE entre 1,02 e 1,16, indicando desempenho inferior ao *naïve*. Além disso, exibiram viés positivo significativo em alguns casos, como CROSTON (+47,19%) e SBA (+39,83%), refletindo tendência à superestimação em parte dos períodos.

O modelo *HURDLE*, por sua vez, apresentou o pior desempenho entre os métodos base, com MASE = 1,83, RMSE superior a 1090 e viés muito elevado (+205,94%). O método demonstrou sensibilidade acentuada aos picos de demanda e instabilidade diante da curta extensão da série, resultando em valores excessivamente superestimados.

Na etapa de empilhamento, o *STACK_LINEAR* apresentou MASE = 0,66 e viés de –69,72%, desempenho consideravelmente melhor do que a maioria dos métodos individuais. Já o *STACK_XGB* apresentou MASE = 0,64, além de viés negativo de –66,88%, e RMSE = 679,57, caracterizando desempenho estável e superior ao *STACK_LINEAR*. Por esta razão, o *STACK_XGB* foi selecionado como *STACK_BEST* para esta série.

O modelo *HYBRID*, resultante da combinação entre o *HURDLE* e o *STACK_BEST* com ponderação adaptativa e calibração final de viés, apresentou desempenho globalmente equilibrado. Com MASE = 0,69 e RMSE = 657,11, indicando boa precisão relativa em comparação ao *naïve*. Embora não tenha atingido o MASE extremamente baixo observado em modelos individuais como THETA, TSB ou XGB_DAILY, o modelo híbrido conseguiu reduzir significativamente a instabilidade e corrigir parte do viés extremo dos métodos base, alcançando viés mais moderado (–50%).

As métricas percentuais — WAPE = 128,44%, MAPE = 75,52% e sMAPE = 190,50% permaneceram elevadas, mas são coerentes com as características estruturais da série, marcada por períodos extensos de demanda zero. Nesses casos, o denominador reduzido aumenta os erros relativos, mesmo quando a previsão é numericamente próxima aos valores reais.

Cabe destacar que a série possui o menor volume de dados entre todas as analisadas (apenas 25 registros). Esse fator limita o treinamento dos modelos, especialmente aqueles baseados em aprendizado de máquina e decomposição, o que explica a permanência de viés moderado mesmo no modelo híbrido final.

Em síntese, o modelo *HYBRID* demonstrou desempenho mais robusto e consistente do que qualquer método individual, conseguindo equilibrar precisão (MASE < 1), estabilidade (RMSE moderado) e redução do viés em relação aos métodos base. Apesar de os erros percentuais permanecerem elevados em função da intermitência e do reduzido tamanho da série, o modelo híbrido mostrou-se a solução mais adequada para este tipo de demanda,

confirmando a eficácia do empilhamento híbrido em contornar limitações dos métodos isolados e fornecer previsões mais alinhadas ao comportamento real da série.

4.2.3.4. Munição *tipo d*

Os resultados apresentados para esta série demonstram que todos os métodos base obtiveram MASE inferior a 1, indicando aprendizado significativo em relação ao modelo ingênuo. Entre eles, os modelos NNETAR, TSB, *XGB_DAILY* e THETA apresentaram os menores valores de MASE (entre 0,37 a 0,46), caracterizando excelente capacidade de reprodução do padrão temporal relativo da série. Contudo, esses mesmos modelos exibiram viés extremamente negativo (–99% a –100%), revelando subestimação sistemática da demanda. Esse comportamento indica que, apesar de captarem adequadamente a forma da série, esses modelos não conseguiram ajustar o nível médio de consumo, o que compromete a aplicabilidade direta de suas previsões no contexto operacional.

Os métodos ARIMA, STLM-AR, CROSTON e SBA apresentaram MASE entre 0,53 e 0,77, desempenho inferior aos melhores modelos, mas ainda satisfatório. Entre eles, ARIMA e STLM-AR apresentaram viés positivo acentuado (+104,09% e +92,88%, respectivamente), indicando tendência de superestimação. Essa diversidade de comportamentos, com modelos que ora superestimam, ora subestimam fortemente, evidencia a heterogeneidade estrutural das previsões individuais.

O modelo *HURDLE* apresentou pior desempenho entre os métodos base, com MASE = 1,26, RMSE = 1396,13 e viés fortemente positivo (+304,26%). Esse resultado evidencia alta sensibilidade aos picos de demanda e instabilidade diante de períodos longos sem consumo, corroborando a inadequação do método para esta série específica.

No processo de empilhamento, o *STACK_LINEAR* apresentou desempenho consistente, com MASE = 0,48 e viés moderado (–3,14%), sendo o modelo individual mais próximo da neutralidade. O *STACK_XGB*, por sua vez, apresentou comportamento instável, com MASE = 1,17, RMSE acima de 1030 e viés muito elevado (+277,39%), configurando superestimação extrema. Dessa forma, o *STACK_LINEAR* foi selecionado como *STACK_BEST*, apresentando o melhor balanço entre precisão relativa e estabilidade.

O modelo final *HYBRID*, resultante da combinação ponderada entre o *STACK_BEST* e o *HURDLE* com calibração final de viés, apresentou desempenho global superior e o mais equilibrado entre todos os métodos aplicados. Com MASE = 0,43, RMSE = 639,36 e viés = –50%, o *HYBRID* apresentou elevada precisão relativa, acompanhado por redução substancial da variabilidade dos erros em comparação aos métodos individuais. Embora a magnitude do viés ainda seja moderada.

As métricas percentuais, MAPE = 72,49%, sMAPE = 176,43% e WAPE = 115,77%, permanecem elevadas, porém, coerentes com a natureza intermitente da série, que apresenta 62,5% de dias sem demanda e eventos de consumo concentrados em picos abruptos. Assim, como observado nas séries anteriores, a presença de valores muito próximos de zero inflaciona o erro relativo, mesmo quando a previsão está numericamente próxima dos valores reais.

Em síntese, o modelo *HYBRID* apresentou o melhor compromisso entre precisão e estabilidade entre todos os métodos aplicados à *série d*. Ele obteve MASE significativamente menor que 1 (0,43), reduziu de forma expressiva o erro quadrático médio e mitigou o viés que marcava os métodos base, consolidando-se como a abordagem mais robusta para esta série. Comparativamente às demais séries analisadas, a série *tipo d* apresentou o comportamento preditivo mais consistente com menor MASE e viés dentre os mais baixos.

Por fim, destaca-se que o código desenvolvido apresenta calibração de viés e utiliza os erros de viés obtidos durante período de teste para a justar a previsão futura, corrigindo o nível global da série e produzindo previsões ajustadas. Quanto mais dados disponíveis, tende-se a redução dos erros, conforme pode ser verificado com a série *tipo d,* que possui maior quantidade de dados para análise e apresentou melhor desempenho preditivo dentre as séries.

5 CONCLUSÃO

O presente trabalho teve por objetivo a integração de algoritmos de Inteligência Artificial ao modelo de gerenciamento de estoques de munições de emprego aeronáutico do COMAER, por meio do desenvolvimento de um sistema preditivo baseado em aprendizado de máquina, capaz de analisar dados históricos do SILOMS, gerar previsões de demanda e oferecer suporte à tomada de decisão.

A análise exploratória das séries históricas (2009–2024) de quatro tipos de munições do COMAER evidenciou padrões relevantes: sazonalidade, variabilidade e assimetria elevadas. Esses traços são típicos de demandas intermitentes com picos e zeros, nas quais métricas sensíveis a extremos (como RMSE e MAPE) tendem a inflar. Séries com essa característica são de difícil previsão utilizando-se de métodos estatísticos mais simplórios.

O modelo proposto foi implementado como uma abordagem híbrida por empilhamento (*stacking*), onde métodos individuais de previsão de demanda alimentaram um meta-modelo (*XGBOOST*). Mesmo frente à heterogeneidade entre as séries (tamanho da amostra, sazonalidade, variabilidade e ruído) e a dificuldade de realizar previsões em séries de dados com muitos zeros, o sistema demonstrou adaptação consistente às quatro séries analisadas, conseguindo melhorar o desempenho em relação aos métodos individuais.

Um ponto central observado é a limitação inerente do modelo quanto à disponibilidade de dados. Observou-se que para a série das munições *tipo b* o modelo apresentou desempenho inferior em comparação às séries do *tipo a* e *tipo d*, especialmente em termos de viés e MASE, com tendência de subestimação dos valores reais. Essa discrepância pode ser explicada pela quantidade reduzida de observações históricas disponíveis para o processo de aprendizagem, que prejudicou o processo de aprendizagem e de ajustes de parâmetros.

Em séries curtas, há menor número de janelas para validação e ajuste de parâmetros, o que limita a capacidade do modelo híbrido de capturar padrões de sazonalidade e calibrar o nível de previsão de forma mais robusta. Assim, conclui-se que o erro elevado está diretamente associado à escassez de dados, pois quanto menor a amostra, maior a propensão a viés e mais difícil aprender o comportamento da série.

Em síntese, os resultados permitem concluir que o modelo híbrido de aprendizado de máquina pode ser considerado tecnicamente válido para o problema de previsão de demanda, sobretudo para as munições do *tipo a* e *tipo d*. Ele apresentou MASE inferior a 1, viés relativamente controlado, com previsão futura com calibração de viés, estimativas estáveis e erros relativos menores do que os métodos base.

No uso prático do modelo de previsão desenvolvido, recomenda-se combinar as previsões com estoques de segurança e, quando pertinente, fatores de correção de nível, até que novas observações reduzam o viés residual. É importante relembrar que toda previsão possui erro, nesse aspecto, a definição de estoques de segurança será essencial para a garantia da disponibilidade de materiais na rede de suprimentos.

Como desdobramentos do presente trabalho, recomenda-se: (*a*) estudar e testar diferentes parâmetros dos métodos base para tentar melhorar a precisão preditiva; e (*b*) variar os métodos base individuais para tentar otimizar o modelo desenvolvido e comparar os resultados de diferentes arranjos de métodos base.

Por fim, este trabalho contribuiu para ampliar o debate sobre o emprego da inteligência artificial como ferramenta de apoio à decisão no COMAER, aplicada empiricamente a registros históricos de demanda extraídos do SILOMS. Sendo assim, o presente estudo exemplificou e desenvolveu uma aplicação prática para a utilização da IA como um sistema de suporte à decisão do gestor logístico militar na realização de previsões de demanda de munições aeronáuticas.

REFERÊNCIAS

ACKERMANN, A. E. F.; SELLITTO, M. A. Métodos de previsão de demanda: uma revisão da literatura. **Innovar**, v. 32, n. 85, p. 83-99, 2022.

AMARAL FILHO, Flávio Mineiro do; BANDEIRA, Saymon Galvão; ALCALÁ, Symone Gomes Soares; BARBOSA, Talles Marcelo G. de A. Um estudo comparativo de ensembles híbridos para aplicações de previsão de séries temporais. **Revista Brasileira de Computação Aplicada**, v. 13, n. 2, p. 58–72, 2021.

ARTIFICIAL INTELLIGENCE FOR THE REAL WORLD. **International Research Journal of Modernization in Engineering Technology and Science**, 3 jul. 2023.

AZEVEDO, Beatriz Flamia; ROCHA, Ana Maria A. C.; PEREIRA, Ana I. Hybrid approaches to optimization and machine learning methods: a systematic literature review. **Machine Learning**, v. 113, p. 4055–4097, 2024.

ASSIMAKOPOULOS, V.; NIKOLOPOULOS, K. The theta model: a decomposition approach to forecasting**. International Journal of Forecasting**, Amsterdam, v. 16, n. 4, p. 521-530, 2000.

BALLOU, R.H. **Gerenciamento da cadeia de suprimentos/logística empresarial**. 5ª ed. Porto Alegre/SC: Bookman, 2006.

BEAN, W.L., JOUBERT, J.W., LUHANDJULA, M.K., Inventory management under uncertainty: A military application. **Computers & Industrial Engineering**, v.96, p. 96-107, 2016.

BOX, G. E. P.; JENKINS, G. M. **Time series analysis forecasting and control**. São Francisco: Holden Day, 1976.

BRASIL. Exército. Estado-Maior do Exército. Portaria EME nº 91, de 11 de novembro de 2021. Aprova a Diretriz de Transformação Digital no Exército Brasileiro. **Diário Oficial da União**, Brasília, DF, seção 1, n. 215, p. 48–50, 12 nov. 2021. Disponível em: <https://www.sgex.eb.mil.br/sistemas/legislacao/index.html>. Acesso em: 3 jul. 2025.
BRASIL. Marinha. Estado-Maior da Armada. Portaria nº 218/MB/EMA, de 28 de dezembro de 2022. Aprova a Política de Ciência, Tecnologia e Inovação da Marinha. **Diário Oficial da União**, Brasília, DF, seção 1, n. 245, p. 17–20, 29 dez. 2022. Disponível em: <https://www.marinha.mil.br/dpc/sites/www.marinha.mil.br.dpc/files/Portaria218MB2022.pdf>. Acesso em: 3 jul. 2025.

CARON, Jean-Denis; BRYCE, Robert Mark; YOUNG, Chad. A simulation tool for exploring ammunition stockpile dynamics. In: INTERNATIONAL CONFERENCE ON OPERATIONS RESEARCH AND ENTERPRISE SYSTEMS, 12., 2023, Lisbon. **Anais**. [S. l.: s. n.], 2023. p. 38-49. DOI: 10.5220/0011620800003396.

CHEN, T.; GUESTRIN, C. XGBoost: a scalable tree boosting system. In: PROCEEDINGS OF THE 22nd ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2016, New York. **Anais**. New York: ACM, 2016. p. 785-794. DOI: 10.1145/2939672.2939785.

CHEN, Wenwen *et al.* Artificial Intelligence in Logistics Optimization with Sustainable Criteria: A Review. **Sustainability**, v. 16, n. 21, p. 9145, 2024.

CHOI, Boram; SUH, Jong Hwan. Forecasting Spare Parts Demand of Military Aircraft: Comparisons of Data Mining Techniques and Managerial Features from the Case of South Korea. **Sustainability**, v. 12, n. 15, p. 6045, 28 jul. 2020.

CHOPRA, S.; MEINDL, P. **Gerenciamento da Cadeia de Suprimentos:** Estratégia, Planejamento e Operação. 4. ed. São Paulo: Pearson Prentice Hall, 2011.

CORDEIRO, V. S. Explorando técnicas de Métodos Ensemble em Machine Learning. **Medium**, 2023. Disponível em: https://medium.com/@vitoria.s.cordeiro0/explorando-t%C3%A9cnicas-de-m%C3%A9todos-ensemble-em-machine-learning-e2733f1a2a12. Acesso em: 18 dez. 2024.

CROSTON, J. D. Forecasting and stock control for intermittent demands. **Operational Research Quarterly**, v. 23, n. 3, p. 289–303, 1972.

FORÇA AÉREA BRASILEIRA**. FAB aposta em Inteligência Artificial para eficiência, segurança e inovação tecnológica**. São José dos Campos: Agência Força Aérea, 13 dez. 2024. Disponível em: https://www.fab.mil.br/noticias/mostra/43576/TECNOLOGIA%20-%20FAB%20aposta%20em%20Intelig%C3%AAncia%20Artificial%20para%20efici%C3%AAncia,%20seguran%C3%A7a%20e%20inova%C3%A7%C3%A3o%20tecnol%C3%B3gica. Acesso em: 3 jul. 2025.

GALÁN, José Javier; CARRASCO, Ramón Alberto; LATORRE, Antonio. Military Applications of Machine Learning: A Bibliometric Perspective. **Mathematics**, v. 10, n. 9, p. 1397, 22 abr. 2022.

GASMI, L.; KABOU, S.; LAICHE, N.; NICHANI, R**.** Previsão de séries temporais usando modelo híbrido de aprendizado profundo (ARIMA-LSTM). **Studies in Engineering and Exact Sciences**, v. 5, n. 2, p. 01-15, 2024.

GIL, Antônio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002.

HADLINGTON, Lee *et al.* The use of artificial intelligence in a military context: development of the attitudes toward AI in defense (AAID) scale. **Frontiers in Psychology**, v. 14, p. 1164810, 5 maio 2023.

HEWAMALAGE, H.; BERGMEIR, C.; BANDARA, K. Recurrent neural networks for time series forecasting: Current status and future directions. **International Journal of Forecasting**, v. 37, n. 1, p. 388–427, 2020.

HICKOK, Merve. Public procurement of artificial intelligence systems: new risks and future proofing. **AI & SOCIETY**, v. 39, n. 3, p. 1213–1227, jun. 2024.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting**: Principles and Practice. 2nd ed. Melbourne: OTexts, 2018.

HYNDMAN, R. J.; BILLAH, B. Unmasking the theta method. **International Journal of Forecasting**, Amsterdam, v. 19, n. 2, p. 287–290, 2003.

HYNDMAN, R. J.; KOEHLER, A. B. Another look at measures of forecast accuracy. **International Journal of Forecasting,** v. 22, n. 4, p. 679–688, 2006.

HYNDMAN, R. J.; KOEHLER, A. B.; ORD, J. K.; SNYDER, R. D. **Forecasting with Exponential Smoothing**: The State Space Approach. Berlin: Springer, 2008.

IBM. **What is Machine Learning (ML)?** Armonk, NY: IBM, 2021. Disponível em: <https://www.ibm.com/think/topics/machine-learning>. Acesso em: 3 jul. 2025.

JENSEN, Benjamin M.; WHYTE, Christopher; CUOMO, Scott. Algorithms at War: The Promise, Peril, and Limits of Artificial Intelligence. **International Studies Review**, v. 22, n. 3, p. 526–550, 1 set. 2020.

KELL, Laurence T. *et al.* Empirical validation of integrated stock assessment models to ensuring risk equivalence: A pathway to resilient fisheries management. **PLOS ONE**, v. 19, n. 7, p. e0302576, 2024.

KIM, Jae-Dong; HWANG, Ji-Hwan; DOH, Hyoung-Ho. A Predictive Model with Data Scaling Methodologies for Forecasting Spare Parts Demand in Military Logistics. **Defence Science Journal**, v. 73, n. 06, p. 666–674, 2023.

KOSASIH, E. E.; BRINTRUP, A. Reinforcement Learning Provides a Flexible Approach for Realistic Supply Chain Safety Stock Optimisation. **IFAC-PapersOnLine**, v. 55, n. 10, p. 1539-1544, 2022.

KWARTENG, Baffoe Samuel; ANDREEVICH, Poguda Aleksey. Comparative analysis of ARIMA, SARIMA, and Prophet model in forecasting. **Research & Development**, v. 5, n. 4, p. 110-120, 2024.

LEE, Hanjun; KIM, Jaedong. A Predictive Model for Forecasting Spare Parts Demand in Military Logistics. *In*: Bangkok: IEEE, dez. 2018. Disponível em: <https://ieeexplore.ieee.org/document/8607801/>. Acesso em: 3 nov. 2025

LEE, Michael; VALISETTY, Ramakrishna; BREUER, Alexander; KIRK, Kelly; PANNETON, Brian; BROWN, Scott. **Current and Future Applications of Machine Learning for the US Army**. Fort Belvoir: Defense Technical Information Center, 2018. Disponível em: https://apps.dtic.mil/sti/pdfs/AD1050263.pdf. Acesso em: 25 jan. 2025

LI, Kang; SHI, Xianming; LI, Yuan; ZHAO, Mei. Combined Forecasting Method of Ammunition Consumption Based on Wavelet Network. **Journal of Physics: Conference Series**, v. 1670, p. 012028, 2020. Disponível em: https://doi.org/10.1088/1742-6596/1670/1/012028. Acesso em: 31 dez. 2024.

MAKRIDAKIS, Spyros; SPILIOTIS, Evangelos; ASSIMAKOPOULOS, Vassilios. Statistical and Machine Learning forecasting methods: Concerns and ways forward. **PLOS ONE**, v. 13, n. 3, p. e0194889, 2018a.

MAKRIDAKIS, Spyros; SPILIOTIS, Evangelos; ASSIMAKOPOULOS, Vassilios. The M4 Competition: Results, findings, conclusion and way forward. **International Journal of Forecasting**, v. 34, n. 4, p. 802-808, 2018b.

MAKRIDAKIS**,** S.; SPILIOTIS, E.; ASSIMAKOPOULOS, V**.** The M4 Competition: 100,000 time series and 61 forecasting methods. **International Journal of Forecasting**, v. 36, n. 1, p. 54-74, 2020.

MEERVELD, H. W. *et al.* The irresponsibility of not using AI in the military. **Ethics and Information Technology**, v. 25, n. 1, p. 14, mar. 2023.

MEERVELD, Herwin; LINDELAUF, Roy. Data Science in Military Decision-Making: Foci and Gaps. **Global Society**, v. 39, n. 2, p. 103–129, 3 abr. 2025.

MONTERO-MANSO, Pablo; ATHANASOPOULOS, George; HYNDMAN, Rob J.; TALAGALA, Thiyanga S. FFORMA: Feature-based forecast model averaging. **International Journal of Forecasting**, v. 36, n. 1, p. 86-92, 2020.

NIKOLOPOULOS, Konstantinos; THOMAKOS, Dimitrios D. Forecasting with the Theta Method. **Wiley StatsRef: Statistics Reference Online**. 1. ed. *[S.l.]*, p. 1-5, 2020.

OBINNA, Amaka Justina; KESS-MOMOH, Azeez Jason. Comparative technical analysis of legal and ethical fra Meerveld meworks in AI-enhanced procurement processes. **World Journal of Advanced Research and Reviews**, v. 22, n. 1, p. 1415–1430, 30 abr. 2024a.

OBINNA, Amaka Justina; KESS-MOMOH, Azeez Jason. Developing a conceptual technical framework for ethical AI in procurement with emphasis on legal oversight. **GSC Advanced Research and Reviews**, v. 19, n. 1, p. 146–160, 30 abr. 2024b.

OBINNA, Amaka Justina; KESS-MOMOH, Azeez Jason. Systematic technical analysis: Enhancing AI deployment in procurement for optimal transparency and accountability. **Global Journal of Engineering and Technology Advances**, v. 19, n. 1, p. 192–206, 30 abr. 2024c.

PEREIRA, R**.** Desvendando o bagging e boosting em machine learning: aprendizado coletivo para modelos mais precisos. **Medium**, 2023. Disponível em: https://medium.com. Acesso em: 2 jan. 2025.

POPENICI, Stefan A. D.; KERR, Sharon. Exploring the impact of artificial intelligence on teaching and learning in higher education. **Research and Practice in Technology Enhanced Learning**, v. 12, n. 1, p. 22, dez. 2017.

RIMÉLÉ, A.; GRANGIER, P.; GAMACHE, M.; GENDREAU, M.; ROUSSEAU, L.-M. **E-commerce Warehousing: Learning a Storage Policy**. Montréal: CIRRELT, 2021.

PRAKOSO, Dendi Putra; IRFAN, Muhammad; SIDDIQUE, Quba. A comparative analysis of linear regression and XGBoost algorithms for predicting GPU prices using technical specifications. **International Journal of Informatics and Information Systems**, v. 7, n. 4, p. 189–199, 2024.

ROSS, David Frederick. **Distribution Planning and Control**: Managing in the Era of Supply Chain Management. 3. ed. Nova York: Springer, 2015.

SANTAMARIA, F.; PALME, R.; SCHLAGLOTH, R.; KLOBETZ-RASSAM, E.; HENNING, J. Seasonal variations of faecal cortisol metabolites in koalas in South East Queensland. **Animals, Basel**, v. 11, n. 6, p. 1622, 2021.

SHARMA, J. K. **Operations Research**: Theory and application. 6a ed. New Delhi: Trinity Press, 2016.

SINA, Lennart B.; SECCO, Cristian A.; BLAZEVIC, Midhad; NAZEMI, Kawa**.** Hybrid Forecasting Methods - A Systematic Review. **Electronics**, v. 12, n. 9, p. 2019, 2023.

SYNTETOS, A. A.; BOYLAN, J. E. The accuracy of intermittent demand estimates. **International Journal of Forecasting**, v. 21, n. 2, p. 303–314, 2005.

SOUSA, E. A. de. Machine Learning aplicado na gestão de estoques e cadeia de suprimentos: uma abordagem à análise preditiva. In: CONGRESSO DE ENGENHARIA DE PRODUÇÃO, PROCESSOS E SERVIÇOS, 1., 2024, São Paulo. **Anais** [...]. São Paulo: Even3, 2024.

SVETUNKOV, Ivan; PETROPOULOS, Fotios. Old dog, new tricks: a modelling view of simple moving averages. **International Journal of Production Research**, v. 56, n. 18, p. 6034–6047, 2018.

TAMASHIRO, T. S. O.; MARQUES, M. A. M.; DROZDA, F. O.; RAMOS, N. C. S. Modelos de aprendizado de máquina para previsão da demanda por séries temporais em cadeias de suprimentos: uma revisão sistemática e análise bibliométrica da literatura. In: CONGRESSO BRASILEIRO DE ENGENHARIA DE PRODUÇÃO – CONBREPRO, 24., 2024, [s.l.]. **Anais**. [s.l.]: APREPRO, 2024. p. 101-120. Disponível em: https://aprepro.org.br/conbrepro/anais/2024/arquivos/10282024_161017_671fdf41e3594.pdf. Acesso em: 31 dez. 2024.

TEUNTER, R. H.; SYNTETOS, A. A.; BABAI, M. Z. Determining order-up-to levels under intermittent demand. **European Journal of Operational Research**, v. 214, n. 3, p. 606–615, 2011a.

TEUNTER, R. H.; SYNTETOS, A. A.; BABAI, M. Z. Intermittent demand: linking forecasting to inventory obsolescence. **European Journal of Operational Research**, v. 214, n. 3, p. 606–615, 2011b.

TSOLAKI, Kalliopi *et al.* Utilizing machine learning on freight transportation and logistics applications: A review. **ICT Express**, v. 9, n. 3, p. 284–295, jun. 2023.

VENÂNCIO, M. L. M. dos S.; BUENO, F. C. Gestão de estoque e a inteligência artificial: um estudo de caso em um supermercado de pequeno porte. In: CONGRESSO BRASILEIRO DE ENGENHARIA DE PRODUÇÃO, 23., 2023, Ponta Grossa. **Anais** [...]. Ponta Grossa: ABEPRO, 2023.

VERGARA, Sylvia C. **Projetos e relatórios de pesquisa em administração**. 3.ed. Rio de Janeiro: Atlas, 2000.

VERÍSSIMO, Andrey Jonas; ALVES, Custodio da Cunha; HENNING, Elisa; AMARAL, Claiton Emílio do; CRUZ, Altair Carlos da. Métodos estatísticos de suavização exponencial Holt-Winters para previsão de demanda em uma empresa do setor metal mecânico. **Revista Gestão Industrial**, v. 08, n. 04, p. 154-171, 2012.

WALTER, O. M. F. C.; HENNING, E.; MORO, G.; SAMOHYL, R. W. Aplicação de um modelo SARIMA na previsão de vendas de motocicletas. **Exacta**, v. 11, n. 1, p. 77–88, 2013.

WATERS, Donald. **Logistics:** An Introduction to Supply Chain Management. Basingstoke: Palgrave Macmillan, 2003.

WOLPERT, David H. Stacked Generalization: **Neural Networks**, v. 5, n. 2, p. 241-259, 1992.

WOSCHANK, Manuel; RAUCH, Erwin; ZSIFKOVITS, Helmut. A Review of Further Directions for Artificial Intelligence, Machine Learning, and Deep Learning in Smart Logistics. **Sustainability**, v. 12, n. 9, p. 3760, 6 maio 2020.

YOHO, Keenan D.; RIETJENS, Sebastiaan ; TATHAM, Peter. Defence Logistics: An Important Research Field in Need of Researchers. **International Journal of Physical Distribution & Logistics Management**. v. 43, n. 2. p. 80-96, 2012.

ZHANG, G. P. Time series forecasting using a hybrid ARIMA and neural network model. **Neurocomputing**, v. 50, p. 159–175, 2003.

ZLATNIK, Daniel; MAREŠ, Jaromír. Assessment of effective ammunition stock levels. **Land Forces Academy Review**, v. XXIII, n. 2(90), p. 128-135, 2018. Disponível em: https://doi.org/10.2478/raft-2018-0015. Acesso em: 31 dez. 2024.

APÊNDICE A – Código da análise exploratória

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import acf, pacf, adfuller  # adfuller adicionado aqui
from statsmodels.tsa.seasonal import STL
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pmdarima import auto_arima  # Importando a função AutoARIMA
from scipy.stats import skew
from scipy.signal import find_peaks
import os

# Carregar o dataframe a partir do arquivo CSV
df = pd.read_csv('C:/Users/whiin/OneDrive/.ipynb_checkpoints/base_dados.CSV', sep=';')
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)  # Remover espaços em
branco
df['DT. ATEND.'] = pd.to_datetime(df['DT. ATEND.'], errors='coerce', dayfirst=True)
df = df.sort_values(['TIPO', 'DT. ATEND.'])
df.columns = df.columns.str.strip()  # Remover espaços extras nos nomes das colunas
df['QTD_MATER'] = pd.to_numeric(df['QTD_MATER'], errors='coerce')
df['QTD_MATER'] = df['QTD_MATER'].fillna(0)

# (opcional) zera quantidades negativas, se existirem
df.loc[df['QTD_MATER'] < 0, 'QTD_MATER'] = 0

# Função para aplicar AutoARIMA e melhorar os resíduos
def ajustar_modelo_autoarima(serie, sazonalidade):
    model = auto_arima(serie,
                       seasonal=True,
                       m=sazonalidade,  # Definir a sazonalidade baseada na estimativa
                       stepwise=True,   # Ativar a busca em etapas
                       trace=True)      # Mostrar o progresso da busca
    # auto_arima já retorna o modelo ajustado
    model_fit = model
    residuos = model_fit.resid()
    return model_fit, residuos

# Função de transformação logarítmica
def transformar_log(serie):
    return np.log(serie + 1)

# Função de Winsorization
def winsorizar(serie, lower_percentile=1, upper_percentile=99):
    limite_inferior = np.percentile(serie, lower_percentile)
    limite_superior = np.percentile(serie, upper_percentile)
    serie_winsorizada = serie.clip(lower=limite_inferior, upper=limite_superior)
    return serie_winsorizada

# Função para decompor a série e avaliar resíduos
```

```
def decompor_serie(serie, sazonalidade, output_dir, material):
    stl = STL(serie, seasonal=sazonalidade, robust=True)
    decomposicao = stl.fit()
    decomposicao.plot()
    plt.tight_layout()
    # Salvar o gráfico primeiro, antes de exibir
    plt.savefig(f"{output_dir}/decomposicao_{material}.png")
    plt.close()
    return decomposicao

# Função para avaliar a aleatoriedade dos resíduos (ACF e PACF)
def avaliar_residuos(residuos, output_dir, material):
    fig, axes = plt.subplots(1, 2, figsize=(15, 6))
    plot_acf(residuos, lags=20, ax=axes[0])
    axes[0].set_title("ACF dos Resíduos")
    plot_pacf(residuos, lags=20, ax=axes[1])
    axes[1].set_title("PACF dos Resíduos")
    plt.tight_layout()
    # Salvar o gráfico primeiro, antes de exibir
    plt.savefig(f"{output_dir}/autocorrelacao_residuos_{material}.png")
    plt.close()

# Função para estimar sazonalidade
def estimar_sazonalidade(serie, max_lag=24):
    autocorr = acf(serie, nlags=max_lag)
    picos, _ = find_peaks(autocorr)
    sazonalidade = picos[0] if len(picos) > 0 else 12
    if sazonalidade % 2 == 0:
        sazonalidade += 1
    if sazonalidade < 3:
        sazonalidade = 3
    return sazonalidade

# Função para calcular a força da tendência, sazonalidade e ruído
def calcular_forcas(decomposicao):
    tendencia = decomposicao.trend
    sazonalidade = decomposicao.seasonal
    residuo = decomposicao.resid
    variancia_tendencia = tendencia.std() ** 2
    variancia_sazonalidade = sazonalidade.std() ** 2
    variancia_residuo = residuo.std() ** 2
    variancia_total = variancia_tendencia + variancia_sazonalidade + variancia_residuo
    forca_tendencia = variancia_tendencia / variancia_total
    forca_sazonalidade = variancia_sazonalidade / variancia_total
    forca_ruido = variancia_residuo / variancia_total
    return forca_tendencia, forca_sazonalidade, forca_ruido

# Função para salvar resultados em uma planilha Excel
def salvar_resultados(resultados_df, output_file):
    resultados_df.to_excel(output_file, index=False)
```

```
    print(f"\nResultados consolidados salvos no arquivo Excel: {output_file}")

# 1. Configurações de caminho de saída
output_dir = ‘diretório de saída do arquivo’
os.makedirs(output_dir, exist_ok=True)

# Arquivo de saída para os resultados
output_file = 'diretorio de saida do arquivo'

# 2. Lista para armazenar resultados
resultados = []

# 3. Loop por tipo de material
for material in df['TIPO'].unique():
    print(f"Analisando material: {material}")

    # Filtrar os dados por material
    material_df = df[df['TIPO'] == material].set_index('DT. ATEND.')
    material_df = material_df['QTD_MATER'].resample('M').sum()
    material_df = material_df.fillna(0)

    # Winsorizar os dados para lidar com valores extremos
    material_df_tratada = winsorizar(material_df)

    # Estimativa de sazonalidade
    sazonalidade_estimada = estimar_sazonalidade(material_df_tratada)

    # Decompor a série para verificar tendência, sazonalidade e resíduos
    decomposicao = decompor_serie(material_df_tratada, sazonalidade_estimada, output_dir,
material)

    # Calcular as forças de tendência, sazonalidade e ruído
    forca_tendencia, forca_sazonalidade, forca_ruido = calcular_forcas(decomposicao)

    # Proporção de zeros (intermitência)
    zeros_ratio = (material_df_tratada == 0).sum() / len(material_df_tratada)

    # Teste ADF (estacionariedade)
    try:
        adf_stat, p_adf, _, _, _, _ = adfuller(material_df_tratada)
    except Exception as e:
        print(f"Falha ao aplicar ADF para {material}: {e}")
        p_adf = np.nan

    # Ajustar AutoARIMA e verificar resíduos
    model_fit, residuos_autoarima = ajustar_modelo_autoarima(material_df_tratada,
sazonalidade_estimada)

    # Avaliar a aleatoriedade dos resíduos
    avaliar_residuos(residuos_autoarima, output_dir, material)
```

```
    # Plotar e comparar os resíduos
    plt.figure(figsize=(10, 6))
    plt.plot(residuos_autoarima)
    plt.title(f"Resíduos Ajustados do Modelo AutoARIMA - {material}")
    plt.tight_layout()
    # Salvar o gráfico
    plt.savefig(f"{output_dir}/residuos_autoarima_{material}.png")
    plt.close()

    # Armazenar os resultados na lista
    resultados.append({
        'TIPO': material,
        'Sazonalidade Estimada': sazonalidade_estimada,
        'Força da Tendência': forca_tendencia,
        'Força da Sazonalidade': forca_sazonalidade,
        'Força do Ruído': forca_ruido,
        'Coeficiente de Variação': material_df_tratada.std() / material_df_tratada.mean(),
        'Assimetria': skew(material_df_tratada),
        'Proporção de Zeros': zeros_ratio,
        'p-valor ADF': p_adf
    })

# 4. Criar DataFrame com os resultados
resultados_df = pd.DataFrame(resultados)

# 5. Salvar os resultados em uma planilha Excel
salvar_resultados(resultados_df, output_file)
```

APÊNDICE B – Código do programa de previsão

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.api import STLForecast
from statsmodels.tsa.forecasting.theta import ThetaModel
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from sklearn.impute import SimpleImputer
from scipy.optimize import nnls, lsq_linear
try:
    from pmdarima import auto_arima
except ImportError:
    auto_arima = None
import xgboost as xgb
import os
import time
import warnings
import inspect
from sklearn.preprocessing import StandardScaler  # (usado no NNETAR)

warnings.filterwarnings("ignore")

try:
    from xgboost.callback import EarlyStopping as XgbEarlyStopping
except Exception:
    XgbEarlyStopping = None

def fit_with_es(model, X_t, y_t, X_v=None, y_v=None,
                sample_weight_t=None, sample_weight_v=None,
                rounds=50, verbose=False):
    """Treina com early stopping quando possível """
    verbose = False
    fit_kwargs = {}
    if sample_weight_t is not None:
        fit_kwargs["sample_weight"] = sample_weight_t

    has_eval = X_v is not None and y_v is not None
    if has_eval:
        fit_kwargs["eval_set"] = [(X_v, y_v)]
        try:
            if "sample_weight_eval_set" in inspect.signature(type(model).fit).parameters and
sample_weight_v is not None:
                fit_kwargs["sample_weight_eval_set"] = [sample_weight_v]
        except Exception:
            pass
```

```
    try:
        sig = inspect.signature(type(model).fit)
        if "early_stopping_rounds" in sig.parameters and has_eval:
            model.fit(X_t, y_t, early_stopping_rounds=rounds, verbose=verbose, **fit_kwargs)
            return model
    except Exception:
        pass

    if has_eval and XgbEarlyStopping is not None:
        try:
            callbacks = [XgbEarlyStopping(rounds=rounds, save_best=True)]
            model.fit(X_t, y_t, callbacks=callbacks, verbose=verbose, **fit_kwargs)
            return model
        except Exception:
            pass

    model.fit(X_t, y_t, verbose=verbose, **fit_kwargs)
    return model

# ==========================
# Flags e parâmetros
# ==========================
FAST_MODE = True
DAILY_LAGS = 28
OOF_MIN_MONTHS = 12
OOF_MONTH_STRIDE = 1
TRAIN_SPLIT = 0.8
RANDOM_STATE = 42

# Estabilidade numérica
CAP_PERC = 99.5
CLIP_MAX_MULTIPLIER = 20
HARD_MAX_MULT = 1.10

# Validação interna p/ calibrar híbrido
INNER_VAL_MONTHS = 4
ALPHA_GRID = np.linspace(0.0, 1.0, 21)  # 0.00, 0.05, ..., 1.00

# ==========================
# Utilitários gerais
# ==========================
def clip_nonneg(a): return np.clip(np.asarray(a, dtype=float), 0.0, None)
def is_valid_array(x): return np.isfinite(np.asarray(x, dtype=float)).all()

def calculate_errors(y_true, y_pred, train):
    y_true, y_pred = np.array(y_true, dtype=float), clip_nonneg(y_pred)
    mask_mape = y_true != 0
    mape = np.mean(np.abs((y_true[mask_mape] - y_pred[mask_mape]) /
y_true[mask_mape])) * 100 if mask_mape.sum() else np.nan
    denom = np.abs(y_true) + np.abs(y_pred)
```

```
    smape = np.mean(2 * np.abs(y_true - y_pred) / np.where(denom == 0, 1, denom)) * 100
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    denom_mase = np.mean(np.abs(np.diff(train))) if len(train) > 1 else np.nan
    mase = np.mean(np.abs(y_true - y_pred)) / (denom_mase if denom_mase and denom_mase
!= 0 else np.nan)
    return rmse, mape, smape, mase

def mae(y_true, y_pred):
    y_true, y_pred = np.asarray(y_true, float), np.asarray(y_pred, float)
    return float(np.mean(np.abs(y_true - y_pred)))

def agg_daily_to_monthly(series_daily):
    return series_daily.resample('M').sum()

def month_dummies(index):
    m = pd.get_dummies(index.month, drop_first=False)
    m.columns = [f'm{c}' for c in m.columns]
    m.index = index
    return m

def dow_dummies(index):
    d = pd.get_dummies(index.dayofweek, drop_first=False)
    d.columns = [f'dow{c}' for c in d.columns]
    d.index = index
    return d

def make_supervised_daily(series, n_lags=DAILY_LAGS):
    s = series.values.astype(float)
    X, y, idx = [], [], []
    for i in range(n_lags, len(s)):
        X.append(s[i-n_lags:i])
        y.append(s[i])
        idx.append(series.index[i])
    return np.array(X), np.array(y), pd.DatetimeIndex(idx)

def days_in_range(start_date, months_ahead):
    first_day = (start_date + pd.offsets.Day(1)).normalize()
    end_month = (start_date + pd.offsets.MonthEnd(months_ahead)).normalize()
    return pd.date_range(first_day, end_month, freq='D')

def time_since_last_demand_daily(series):
    s = (series.values > 0).astype(int)
    out = np.zeros_like(s, dtype=int)
    last = -1
    for i, v in enumerate(s):
        if v > 0:
            last = i
            out[i] = 0
        else:
            out[i] = (i - last) if last >= 0 else 0
```

```
    return out

# ===== picos (autotune simples) =====
def peak_threshold(y, q=0.85, min_pos=4):
    if len(y) == 0: return 0.0
    thr = float(np.quantile(y, q))
    ys = np.asarray(y, dtype=float)
    if (ys >= thr).sum() < min_pos and len(ys) >= min_pos:
        thr = float(np.sort(ys)[-min_pos])
    return thr

def autotune_peak_params(series):
    y = np.asarray(series.values, dtype=float)
    n = len(y); min_pos = max(4, int(0.02*n))
    thr_probe = peak_threshold(y[y>0], q=0.85, min_pos=min_pos) if (y>0).any() else 0.0
    rare_ratio = ((y[y>0] >= thr_probe).mean() if (y>0).any() else 0.0)
    if rare_ratio < 0.10: q_peak, w_peak = 0.90, 8.0
    elif rare_ratio < 0.20: q_peak, w_peak = 0.85, 5.0
    else: q_peak, w_peak = 0.80, 3.0
    return q_peak, w_peak, min_pos

def make_peak_weights(y, q=0.85, w_peak=5.0, min_pos=4):
    y = np.asarray(y, dtype=float)
    thr = peak_threshold(y, q=q, min_pos=min_pos)
    return np.where(y >= thr, w_peak, 1.0).astype(float), thr

# ==== utilitários de clamp/log ====
def derive_cap_from_train(arr, perc=CAP_PERC, mult=CLIP_MAX_MULTIPLIER):
    arr = np.asarray(arr, dtype=float)
    arr = arr[np.isfinite(arr)]
    if arr.size == 0:
        return 1e6
    q = np.percentile(arr, perc)
    cap = max(10.0, float(q * mult), float(arr.max() * 10.0))
    return cap

def hard_cap_from_series(series, mult=HARD_MAX_MULT):
    m = float(np.nanmax(series.values.astype(float))) if len(series) else 0.0
    m = max(m, 1.0)
    return m * float(mult)

def safe_expm1_and_clip(x, cap):
    x = np.asarray(x, dtype=float)
    x[~np.isfinite(x)] = 0.0
    x = np.clip(x, -50.0, 50.0)
    y = np.expm1(x)
    return np.clip(y, 0.0, cap)

# ==========================
# Modelos base
```

```
# ==========================
def daily_ARIMA(train_d, horizon_d):

    y = train_d.astype(float)

    if len(y) < 10:
        try:
            m = ARIMA(y, order=(0, 1, 0)).fit()
            return m.forecast(steps=horizon_d)
        except Exception:
            # último recurso: repetir o último valor
            return pd.Series(
                [y.iloc[-1]] * horizon_d,
                index=pd.date_range(
                    y.index[-1] + pd.Timedelta(days=1),
                    periods=horizon_d,
                    freq='D'
                )
            )

    if auto_arima is not None:
        try:
            model = auto_arima(
                y,
                start_p=0, max_p=2,
                start_q=0, max_q=2,
                d=None, max_d=1,
                start_P=0, max_P=1,
                start_Q=0, max_Q=1,
                D=None, max_D=1,
                seasonal=True,
                m=7,                    # sazonalidade semanal
                information_criterion='aic',
                stepwise=True,
                suppress_warnings=True,
                error_action='ignore',
                trace=False
            )
            fc = model.predict(n_periods=horizon_d)
            return pd.Series(
                fc,
                index=pd.date_range(
                    y.index[-1] + pd.Timedelta(days=1),
                    periods=horizon_d,
                    freq='D'
                )
            )
        except Exception:
            pass
```

```
    best_aic, best_m = np.inf, None
    for d in (0, 1):
        for p in (0, 1, 2):
            for q in (0, 1, 2):
                try:
                    m = ARIMA(
                        y,
                        order=(p, d, q),
                        enforce_stationarity=False,
                        enforce_invertibility=False
                    ).fit()
                    if m.aic < best_aic:
                        best_aic, best_m = m.aic, m
                except Exception:
                    continue

    if best_m is None:
        best_m = ARIMA(
            y,
            order=(1, 1, 1),
            enforce_stationarity=False,
            enforce_invertibility=False
        ).fit()

    return best_m.forecast(steps=horizon_d)


def daily_ETS(train_d, horizon_d):

    y = train_d.astype(float)

    configs = [
        {"trend": None,  "damped_trend": False, "seasonal": None},
        {"trend": "add", "damped_trend": True,  "seasonal": None},
        {"trend": None,  "damped_trend": False, "seasonal": "add"},
        {"trend": "add", "damped_trend": True,  "seasonal": "add"},
    ]

    best_aic, best_model = np.inf, None

    for cfg in configs:
        try:
            model = ExponentialSmoothing(
                y,
                trend=cfg["trend"],
                damped_trend=cfg["damped_trend"],
                seasonal=cfg["seasonal"],
                seasonal_periods=7 if cfg["seasonal"] is not None else None,
                initialization_method="estimated"
            ).fit(optimized=True)
```

```
            aic = model.aic
            if aic < best_aic:
                best_aic, best_model = aic, model
        except Exception:
            continue

    if best_model is None:
        best_model = ExponentialSmoothing(
            y,
            trend=None,
            seasonal=None,
            initialization_method="estimated"
        ).fit(optimized=True)

    return best_model.forecast(horizon_d)


def daily_STLM_AR(train_d, horizon_d):
    """
    STL + ARIMA nos resíduos, usando d=0 (STL já remove boa parte
    da tendência/sazonalidade). Fallback: daily_ARIMA.
    """
    try:
        model = STLForecast(
            train_d,
            ARIMA,
            model_kwargs={"order": (1, 0, 1)},
            period=7
        ).fit()
        return model.forecast(horizon_d)
    except Exception:
        return daily_ARIMA(train_d, horizon_d)

def daily_THETA(train_d, horizon_d):
    return ThetaModel(train_d).fit().forecast(steps=horizon_d)

def daily_NNETAR(train_d, horizon_d, n_lags=DAILY_LAGS):
    try:
        if len(train_d) < max(12, n_lags + 8):
            return np.repeat(float(train_d.iloc[-1]), horizon_d)

        # ==========================
        # Pré-processamento da série
        # ==========================
        y_full = np.asarray(train_d.values, dtype=float)
        y_full = clip_nonneg(y_full)

        # Construção de janelas (lags)
        X_tr_raw = np.array(
            [y_full[i - n_lags:i] for i in range(n_lags, len(y_full))],
```

```
        dtype=float
    )
    y_tr_raw = y_full[n_lags:].astype(float)

    y_tr_clip = y_tr_raw.copy()
    y_tr_log = np.log1p(y_tr_clip)

    # Lags em escala log
    X_tr_log = np.log1p(np.maximum(X_tr_raw, 0.0))

    # Normalização dos inputs
    x_scaler = StandardScaler().fit(X_tr_log)
    X_tr = x_scaler.transform(X_tr_log)

    # ==========================
    # MLP (NNETAR em log)
    # ==========================
    mlp = MLPRegressor(
        hidden_layer_sizes=(128, 64),      # antes: (64, 32)
        activation='tanh',
        alpha=1e-4,                        # antes: 1e-3 (menos L2 = menos bias)
        learning_rate_init=5e-4,           # passo um pouco menor
        batch_size=32,
        shuffle=False,
        max_iter=(1200 if FAST_MODE else 3000),  # antes: 800/2000
        random_state=RANDOM_STATE,
        early_stopping=False               # manter treinamento completo
    )
    mlp.fit(X_tr, y_tr_log)

    # ==========================
    # Caps (limites superiores)
    # ==========================
    p99 = np.percentile(y_tr_raw, 99.0) if np.any(y_tr_raw > 0) else np.max(y_tr_raw)
    cap_soft = float(derive_cap_from_train(y_tr_raw))
    cap_hard = float(hard_cap_from_series(train_d))

    # base_cap leva em conta tanto o cap "estatístico" quanto o p99
    base_cap = max(cap_soft, p99)
    if np.isfinite(cap_hard) and cap_hard > 0:
        cap = min(base_cap * 1.5, cap_hard)  # permite até 150% acima do p99, respeitando
hard cap
    else:
        cap = base_cap * 1.5

    # ==========================
    # Simulação recursiva (multi-step)
    # ==========================
    window = y_full[-n_lags:].astype(float).copy()
    preds = []
```

```
        for h in range(horizon_d):
            # Inputs em log
            w_log = np.log1p(np.maximum(window, 0.0)).reshape(1, -1)
            w_scaled = x_scaler.transform(w_log)

            # Previsão em log + retorno à escala original com cap
            p_log = float(mlp.predict(w_scaled))
            p = float(safe_expm1_and_clip(p_log, cap))

            blend = 0.7 if h < 7 else 0.9  # antes: 0.30 / 0.50
            window = np.roll(window, -1)
            window[-1] = (1.0 - blend) * window[-1] + blend * p

            preds.append(p)

        preds = np.asarray(preds, dtype=float)
        preds = np.clip(preds, 0.0, cap_hard if np.isfinite(cap_hard) and cap_hard > 0 else np.inf)
        return preds

    except Exception:
        return np.repeat(float(train_d.iloc[-1]), horizon_d)

# Intermitentes (Croston/SBA/TSB)
def _croston_decompose(y, alpha=0.1):
    a, p, q = None, None, 0
    for v in y:
        if v > 0:
            if a is None: a, p = v, (q if q>0 else 1)
            else: a = a + alpha*(v-a); p = p + alpha*(q-p)
            q = 1
        else: q += 1
    if a is None: a, p = 0.0, max(1, len(y))
    return a, max(p, 1e-9)

def daily_CROSTON(train_d, horizon_d, alpha=0.1):
    a, p = _croston_decompose(train_d.values.astype(float), alpha)
    f = (a / p)
    return np.repeat(f, horizon_d)

def daily_SBA(train_d, horizon_d, alpha=0.1):
    base = daily_CROSTON(train_d, 1, alpha=alpha)[0]
    f = base * (1 - alpha/2.0)
    return np.repeat(f, horizon_d)

def daily_TSB(train_d, horizon_d, alpha_d=0.5, alpha_p=0.5):

    y = train_d.values.astype(float)
    p_t, d_t, init = 0.0, 0.0, False
    fitted = []
```

```
    for v in y:
        z = 1.0 if v > 0 else 0.0

        if not init:
            # inicialização
            p_t, d_t, init = z, (v if v > 0 else 0.0), True
            f = p_t * d_t
        else:
            # previsão para o período atual com estado anterior
            f = p_t * d_t

            # atualização dos estados com a observação corrente
            p_t = alpha_p * z + (1 - alpha_p) * p_t
            if v > 0:
                d_t = alpha_d * v + (1 - alpha_d) * d_t

        fitted.append(f)

    fitted = np.asarray(fitted, dtype=float)
    y_arr = np.asarray(y, dtype=float)

    # ==========================
    # Fator de correção de viés (k)
    # ==========================
    num = y_arr.sum()
    den = fitted.sum()

    if den > 0 and num > 0:
        k = num / den          #
        k = np.clip(k, 0.1, 10.0)
    else:
        k = 1.0

    base_fc = p_t * d_t
    fc_corr = k * base_fc

    return np.repeat(fc_corr, horizon_d)


# ===== HURDLE =====
def daily_HURDLE(train_d, test_index_d, full_series_d=None, q_peak=0.85, w_peak=5.0,
min_pos=4):
    base = (full_series_d if full_series_d is not None else train_d).asfreq('D').fillna(0)
    full_idx = base.index.union(pd.DatetimeIndex(test_index_d)).sort_values()
    ser_all = base.reindex(full_idx, fill_value=0)

    ser_feat = ser_all.shift(1).fillna(0)  # somente passado
    occ_feat = (ser_feat > 0).astype(int)
```

```
    lag_val = pd.concat([ser_feat.shift(i) for i in range(0, 7)], axis=1)
    lag_val.columns = [f'lag{i+1}' for i in range(7)]
    lag_occ = pd.concat([occ_feat.shift(i) for i in range(0, 7)], axis=1)
    lag_occ.columns = [f'locc{i+1}' for i in range(7)]

    roll_sum_7  = ser_feat.rolling(7, min_periods=1).sum().rename('r_sum7')
    roll_max_28 = ser_feat.rolling(28, min_periods=1).max().rename('r_max28')

    tsl_feat = pd.Series(time_since_last_demand_daily(ser_feat), index=ser_feat.index,
name='tsl')

    md  = month_dummies(ser_all.index)
    dow = dow_dummies(ser_all.index)

    X_all = pd.concat([lag_val, lag_occ, tsl_feat, roll_sum_7, roll_max_28, md, dow],
axis=1).fillna(0)

    X_occ = X_all.loc[train_d.index]
    y_occ = (train_d.values > 0).astype(int)

    if len(X_occ)==0 or len(np.unique(y_occ))<2:
        clf = None; p_const = float((train_d>0).mean())
    else:
        pos_rate = max(1e-6, float(y_occ.mean()))
        w_pos = min(6.0, max(3.0, 1.0/pos_rate))
        w_cls = np.where(y_occ==1, w_pos, 1.0).astype(float)
        clf = xgb.XGBClassifier(
            objective='binary:logistic',
            n_estimators=350 if FAST_MODE else 500,
            learning_rate=0.07 if FAST_MODE else 0.05,
            max_depth=3,
            subsample=0.9, colsample_bytree=0.9,
            reg_lambda=1.0, random_state=RANDOM_STATE, n_jobs=-1,
            eval_metric='logloss'
        )
        clf = fit_with_es(clf, X_occ.values, y_occ, rounds=40, verbose=False,
sample_weight_t=w_cls)

    X_pos = X_occ[y_occ==1]
    y_pos = train_d.loc[X_pos.index].values.astype(float)
    if len(X_pos)==0:
        reg=None; mean_size = float(train_d[train_d>0].mean()) if (train_d>0).any() else 0.0
        cap_hard = hard_cap_from_series(train_d)
    else:
        reg = xgb.XGBRegressor(
            objective='reg:absoluteerror',
            n_estimators=550 if FAST_MODE else 800,
            learning_rate=0.06 if FAST_MODE else 0.05,
            max_depth=3,
            subsample=0.9, colsample_bytree=0.9,
```

```
            reg_lambda=1.0, random_state=RANDOM_STATE, n_jobs=-1,
            eval_metric='mae'
        )
        y_pos = clip_nonneg(y_pos)
        y_pos_log = np.log1p(y_pos)
        w_pos_peak, _ = make_peak_weights(y_pos, q_peak, w_peak=w_peak,
min_pos=min_pos)
        val_len = min(21, len(X_pos)//5) if len(X_pos)>60 else max(1, len(X_pos)//10)
        if val_len>=1:
            X_t, X_v = X_pos.iloc[:-val_len].values, X_pos.iloc[-val_len:].values
            y_t, y_v = y_pos_log[:-val_len], y_pos_log[-val_len:]
            w_t, w_v = w_pos_peak[:-val_len], w_pos_peak[-val_len:]
            reg = fit_with_es(reg, X_t, y_t, X_v, y_v, sample_weight_t=w_t,
sample_weight_v=w_v, rounds=40)
        else:
            reg = fit_with_es(reg, X_pos.values, y_pos_log, sample_weight_t=w_pos_peak,
rounds=40)
        cap_hard = hard_cap_from_series(train_d)

    X_test = X_all.loc[test_index_d].values
    if clf is None: p_occ = np.repeat(p_const, len(test_index_d))
    else: p_occ = clf.predict_proba(X_test)[:,1] if len(X_test) else np.array([])

    if reg is None: size_pos = np.repeat(mean_size, len(test_index_d))
    else:
        pred_log = reg.predict(X_test) if len(X_test) else np.array([])
        size_pos = safe_expm1_and_clip(pred_log, cap_hard)

    return clip_nonneg(p_occ * size_pos)

# XGB diário
def daily_XGB(series_d, train_size, horizon_d, n_lags=DAILY_LAGS,
              q_peak=0.85, w_peak=5.0, min_pos=4):

    # ========================
    # Supervisionado diário
    # ========================
    X_all, y_all, idx_all = make_supervised_daily(series_d, n_lags=n_lags)

    first_test_idx = series_d.index[train_size]
    split_pos = np.where(idx_all >= first_test_idx)[0][0]

    # y_tr em contagem (não-negativa)
    X_tr_base = X_all[:split_pos]
    y_tr = clip_nonneg(y_all[:split_pos])

    # Dummies de dia da semana
    dow = pd.get_dummies(pd.DatetimeIndex(idx_all).dayofweek,
                         drop_first=False).values
    X_all_ext = np.hstack([X_all, dow])
```

```
X_tr = X_all_ext[:split_pos]

# Pesos para picos com base em y_tr (escala original)
w_all, _ = make_peak_weights(y_tr, q=q_peak, w_peak=w_peak, min_pos=min_pos)

# ==========================
# Alvo em log1p (regressão contínua)
# ==========================
y_tr_log = np.log1p(y_tr)

# ==========================
# Modelo XGBoost (regressão em log)
# ==========================
model = xgb.XGBRegressor(
    objective='reg:squarederror',       # em vez de count:poisson
    n_estimators=1400 if not FAST_MODE else 900,
    learning_rate=0.05,
    max_depth=5,                        # ligeiramente mais raso que 6 (menos variância)
    min_child_weight=1.0,               # um pouco mais conservador
    subsample=0.9,
    colsample_bytree=0.9,
    gamma=0.0,
    reg_lambda=0.8,                     # regularização um pouco maior para segurar MASE
    random_state=RANDOM_STATE,
    n_jobs=-1,
    eval_metric='rmse'
)

# ==========================
# Validação p/ early stopping
# ==========================
val_len = min(28, len(X_tr)//5) if len(X_tr) > 100 else max(1, len(X_tr)//10)

if val_len >= 1:
    X_t, X_v = X_tr[:-val_len], X_tr[-val_len:]
    y_t, y_v = y_tr_log[:-val_len], y_tr_log[-val_len:]
    w_t, w_v = w_all[:-val_len], w_all[-val_len:]

    model = fit_with_es(
        model,
        X_t, y_t,
        X_v, y_v,
        sample_weight_t=w_t,
        sample_weight_v=w_v,
        rounds=50
    )
else:
    model = fit_with_es(
        model,
        X_tr, y_tr_log,
```

```
        sample_weight_t=w_all,
        rounds=50
    )

# ==========================
# Caps (limites superiores)
# ==========================
cap_soft = derive_cap_from_train(y_tr,
                                 perc=CAP_PERC,
                                 mult=CLIP_MAX_MULTIPLIER)
cap_hard = hard_cap_from_series(series_d.iloc[:train_size])

# compromisso: folga, mas não tão agressiva quanto 1.5
if np.isfinite(cap_hard) and cap_hard > 0:
    cap = min(cap_soft * 1.2, cap_hard)
else:
    cap = cap_soft * 1.2

setattr(model, "cap_", float(cap))
setattr(model, "hard_cap_", float(cap_hard))

# ==========================
# Previsão recursiva diária
# ==========================
window = series_d.values[train_size - n_lags:train_size].astype(float).copy()
preds = []

fut_idx = pd.date_range(series_d.index[train_size],
                        periods=horizon_d + 1,
                        freq='D')[1:]

for d in fut_idx:
    dow_vec = pd.get_dummies(pd.Index([d]).dayofweek,
                             drop_first=False) \
              .reindex(columns=range(0, 7), fill_value=0).values
    x = np.hstack([window.reshape(1, -1), dow_vec])

    # saída em log1p -> volta com expm1 + cap
    p_log = float(model.predict(x))
    p = float(safe_expm1_and_clip(p_log, cap))

    preds.append(p)
    window = np.roll(window, -1)
    window[-1] = p

preds = np.array(preds, dtype=float)
if np.isfinite(cap_hard) and cap_hard > 0:
    preds = np.clip(preds, 0.0, cap_hard)
else:
    preds = np.clip(preds, 0.0, np.inf)
```

```
    return preds, model

    # ==========================
    # Caps (limites superiores)
    # ==========================
    cap_soft = derive_cap_from_train(y_tr, perc=CAP_PERC,
mult=CLIP_MAX_MULTIPLIER)
    cap_hard = hard_cap_from_series(series_d.iloc[:train_size])

    if np.isfinite(cap_hard) and cap_hard > 0:
        cap = min(cap_soft * 1.5, cap_hard)   # antes: min(cap_soft, cap_hard)
    else:
        cap = cap_soft * 1.5

    setattr(model, "cap_", float(cap))
    setattr(model, "hard_cap_", float(cap_hard))

    # ==========================
    # Previsão recursiva
    # ==========================
    window = series_d.values[train_size - n_lags:train_size].astype(float).copy()
    preds = []

    fut_idx = pd.date_range(series_d.index[train_size], periods=horizon_d + 1, freq='D')[1:]
    for d in fut_idx:
        dow_vec = pd.get_dummies(pd.Index([d]).dayofweek, drop_first=False) \
                  .reindex(columns=range(0, 7), fill_value=0).values
        x = np.hstack([window.reshape(1, -1), dow_vec])

        # Para count:poisson, a saída é log(lambda) -> expm1 para voltar à escala
        p_log = float(model.predict(x))
        p = float(safe_expm1_and_clip(p_log, cap))

        preds.append(p)
        # usa a própria previsão como próximo lag
        window = np.roll(window, -1)
        window[-1] = p

    preds = np.array(preds, dtype=float)
    if np.isfinite(cap_hard) and cap_hard > 0:
        preds = np.clip(preds, 0.0, cap_hard)
    else:
        preds = np.clip(preds, 0.0, np.inf)

    return preds, model

# ==========================
# OOF mensal (rolling origin)
# ==========================
```

```
def monthly_oof_base_preds(series_d, min_months=OOF_MIN_MONTHS,
month_stride=OOF_MONTH_STRIDE):
    series_d = series_d.asfreq('D').fillna(0)
    month_list = pd.date_range(series_d.index.min(), series_d.index.max(), freq='M')
    oof_data, y_oof, idx = [], [], []

    def seas_naive_week(train_d, horizon_d):
        last7 = train_d.iloc[-7:].values if len(train_d)>=7 else np.repeat(train_d.iloc[-1], 7)
        reps = int(np.ceil(horizon_d/7)); pred = (np.tile(last7, reps))[:horizon_d]
        return pred

    for i in range(min_months, len(month_list)-1, month_stride):
        end_train = month_list[i-1]
        tr = series_d.loc[:end_train]
        start_next = (end_train + pd.offsets.Day(1)).normalize()
        end_next = month_list[i]
        test_idx = pd.date_range(start_next, end_next, freq='D')
        h = len(test_idx)
        if h <= 0 or len(tr) < 60:
            continue

        preds = {}
        try: preds['ARIMA'] = clip_nonneg(np.asarray(daily_ARIMA(tr, h)).ravel())
        except: preds['ARIMA'] = np.repeat(0.0, h)
        try: preds['ETS'] = clip_nonneg(np.asarray(daily_ETS(tr, h)).ravel())
        except: preds['ETS'] = np.repeat(0.0, h)
        try: preds['STLM_AR'] = clip_nonneg(np.asarray(daily_STLM_AR(tr, h)).ravel())
        except: preds['STLM_AR'] = np.repeat(0.0, h)
        try: preds['THETA'] = clip_nonneg(np.asarray(daily_THETA(tr, h)).ravel())
        except: preds['THETA'] = np.repeat(0.0, h)
        try: preds['NNETAR'] = clip_nonneg(np.asarray(daily_NNETAR(tr, h)).ravel())
        except: preds['NNETAR'] = np.repeat(0.0, h)
        try: preds['SEASONAL_NAIVE_WEEK'] = clip_nonneg(seas_naive_week(tr, h))
        except: preds['SEASONAL_NAIVE_WEEK'] = np.repeat(max(tr.iloc[-1], 0.0), h)
        try: preds['CROSTON'] = clip_nonneg(daily_CROSTON(tr, h))
        except: preds['CROSTON'] = np.repeat(0.0, h)
        try: preds['SBA'] = clip_nonneg(daily_SBA(tr, h))
        except: preds['SBA'] = np.repeat(0.0, h)
        try: preds['TSB'] = clip_nonneg(daily_TSB(tr, h))
        except: preds['TSB'] = np.repeat(0.0, h)
        try: preds['HURDLE'] = clip_nonneg(daily_HURDLE(tr, test_idx, full_series_d=tr))
        except: preds['HURDLE'] = np.repeat(0.0, h)
        try:
            tr_size = len(tr)
            xgbd, _ = daily_XGB(series_d, tr_size, h)
            preds['XGB_DAILY'] = clip_nonneg(xgbd)
        except:
            preds['XGB_DAILY'] = np.repeat(0.0, h)

        sums = {k: float(np.nansum(clip_nonneg(v))) for k, v in preds.items()}
```

```
        real_sum = float(series_d.loc[test_idx].sum())
        oof_data.append(sums); y_oof.append(real_sum); idx.append(end_next)

    if len(oof_data)==0:
        cols = ['ARIMA','ETS','STLM_AR','THETA','NNETAR',

'SEASONAL_NAIVE_WEEK','CROSTON','SBA','TSB','HURDLE','XGB_DAILY']
        oof_df = pd.DataFrame(columns=cols); y_vec = np.array([])
    else:
        oof_df = pd.DataFrame(oof_data, index=pd.DatetimeIndex(idx)).sort_index()
        y_vec = np.array(y_oof, dtype=float)
    return oof_df, y_vec

# ==========================
# Stacking mensal
# ==========================
def _sanitize_design_matrix(X, y):
    X = np.array(X, dtype=float); y = np.array(y, dtype=float)
    X[~np.isfinite(X)] = 0.0; y[~np.isfinite(y)] = 0.0
    var = X.var(axis=0); keep = var > 1e-12
    if not np.any(keep):
        X = np.ones((X.shape[0], 1), dtype=float); keep = np.array([True])
    else:
        X = X[:, keep]
    norms = np.linalg.norm(X, axis=0); norms[norms == 0] = 1.0
    return X / norms, y, keep, norms


def filter_outliers_for_meta(df):
    X = df.copy().astype(float)
    for c in X.columns:
        col = np.asarray(X[c], dtype=float)
        finite = np.isfinite(col)
        if finite.sum() == 0:
            continue
        cap = np.nanpercentile(col[finite], 99.5) * 1.5
        col[(col > cap) | (~np.isfinite(col))] = np.nan
        X[c] = col
    imp = SimpleImputer(strategy='median')
    X_imputed = pd.DataFrame(imp.fit_transform(X), index=X.index, columns=X.columns)
    return X_imputed


def fit_stack_linear_nnls(X_oof, y_oof):
    # Peso extra para picos, mas não tão agressivo
    thr = peak_threshold(y_oof, q=0.8, min_pos=max(4, int(0.05 * len(y_oof))))
    w = np.where(y_oof >= thr, 2.0, 1.0).astype(float)   # antes: 3.0
    sw = np.sqrt(w)

    Xw = np.asarray(X_oof, float) * sw[:, None]
```

```
    yw = np.asarray(y_oof, float) * sw

    Xs, ys, keep, norms = _sanitize_design_matrix(Xw, yw)

    try:
        w_scaled, _ = nnls(Xs, ys, maxiter=10000)
    except RuntimeError:
        res = lsq_linear(Xs, ys, bounds=(0.0, np.inf), max_iter=10000)
        w_scaled = res.x

    w_full = np.zeros(X_oof.shape[1], dtype=float)

    # desfaz normalização de colunas
    w_orig = w_scaled / norms
    w_full[keep] = w_orig

    if not np.any(w_full > 0):
        w_full = np.ones_like(w_full) / len(w_full)

    return w_full

def fit_stack_xgb(oof_df, y_oof):
    # Tratamento de outliers e criação de feature de viés global
    X_df = filter_outliers_for_meta(oof_df).copy()
    X_df["_const"] = 1.0

    X = X_df.values.astype(float)
    y = clip_nonneg(y_oof)

    thr = peak_threshold(y, q=0.8, min_pos=max(4, int(0.05 * len(y))))
    w = np.where(y >= thr, 3.0, 1.0).astype(float)

    meta = xgb.XGBRegressor(
        objective='reg:squarederror',
        n_estimators=700 if FAST_MODE else 1100,
        learning_rate=0.05,
        max_depth=4,
        min_child_weight=0.5,
        subsample=1.0,
        colsample_bytree=1.0,
        reg_lambda=0.3,
        random_state=RANDOM_STATE,
        n_jobs=-1,
        eval_metric='mae'
    )

    meta.fit(X, y, sample_weight=w, verbose=False)
    return meta
```

```
def apply_stack_linear(weights, base_month_df):
    X = np.nan_to_num(base_month_df.values, nan=0.0)
    return clip_nonneg(X @ weights)


def apply_stack_xgb(meta, base_month_df):
    X_df = filter_outliers_for_meta(base_month_df).copy()
    X_df["_const"] = 1.0
    X = np.nan_to_num(X_df.values.astype(float), nan=0.0)
    return clip_nonneg(meta.predict(X))

# ==========================
# Calibração do HÍBRIDO por MASE
# ==========================
def tune_hybrid_alpha_by_mase(train_d):
    train_d = train_d.asfreq('D').fillna(0)
    months = pd.date_range(train_d.index.min(), train_d.index.max(), freq='M')
    if len(months) < OOF_MIN_MONTHS + INNER_VAL_MONTHS + 1:
        return 0.30  # fallback

    split_point = months[-(INNER_VAL_MONTHS+1)]
    inner_tr = train_d.loc[:split_point]
    inner_val_days = pd.date_range(split_point + pd.offsets.Day(1), months[-1], freq='D')

    # HURDLE (inner-val)
    q_peak, w_peak, min_pos = autotune_peak_params(inner_tr)
    try:
        yhat_hurdle_val_d = daily_HURDLE(inner_tr, inner_val_days, full_series_d=inner_tr,
                                         q_peak=q_peak, w_peak=w_peak, min_pos=min_pos)
    except:
        yhat_hurdle_val_d = np.repeat(0.0, len(inner_val_days))

    # OOF interno para stacks
    try:
        oof_df_in, y_oof_in = monthly_oof_base_preds(inner_tr,
min_months=OOF_MIN_MONTHS, month_stride=OOF_MONTH_STRIDE)
    except:
        oof_df_in = pd.DataFrame(); y_oof_in = np.array([])

    # Base p/ inner-val (diário->mensal)
    base_val_month = {}
    h_val = len(inner_val_days)

    for name, f in [('CROSTON', daily_CROSTON), ('SBA', daily_SBA), ('TSB', daily_TSB)]:
        try:
            yhat = f(inner_tr, h_val)
        except:
            yhat = np.repeat(0.0, h_val)
        base_val_month[name] = pd.Series(clip_nonneg(yhat),
index=inner_val_days).resample('M').sum()
```

```
    def seas_naive_week(train_d, horizon_d):
        last7 = train_d.iloc[-7:].values if len(train_d)>=7 else np.repeat(train_d.iloc[-1], 7)
        reps = int(np.ceil(horizon_d/7)); pred = (np.tile(last7, reps))[:h_val]
        return pred
    try:
        base_val_month['SEASONAL_NAIVE_WEEK'] =
pd.Series(clip_nonneg(seas_naive_week(inner_tr, h_val)),
                                            index=inner_val_days).resample('M').sum()
    except:
        base_val_month['SEASONAL_NAIVE_WEEK'] = pd.Series(0.0,
index=pd.date_range(inner_val_days.min(), inner_val_days.max(), freq='M'))

    try:
        xgb_val_d, _ = daily_XGB(inner_tr, len(inner_tr), h_val)
    except:
        xgb_val_d = np.repeat(0.0, h_val)
    base_val_month['XGB_DAILY'] = pd.Series(clip_nonneg(xgb_val_d),
index=inner_val_days).resample('M').sum()

    hurdle_val_m = pd.Series(clip_nonneg(yhat_hurdle_val_d),
index=inner_val_days).resample('M').sum()

    if len(oof_df_in) == 0:
        stack_val_m = pd.concat(base_val_month.values(),
axis=1).mean(axis=1).reindex(hurdle_val_m.index).fillna(0.0)
    else:
        base_month_val = pd.DataFrame(index=hurdle_val_m.index)
        for c in oof_df_in.columns:
            if c == 'HURDLE':
                base_month_val[c] = hurdle_val_m.values
            else:
                base_month_val[c] = base_val_month.get(c, pd.Series(0.0,
index=hurdle_val_m.index)).reindex(hurdle_val_m.index).values
        W_in = fit_stack_linear_nnls(oof_df_in.values, y_oof_in) if len(oof_df_in)>0 else None
        meta_in = fit_stack_xgb(oof_df_in, y_oof_in) if len(oof_df_in)>0 else None
        stack_lin_val = apply_stack_linear(W_in, base_month_val) if W_in is not None else
base_month_val.mean(axis=1).values
        stack_xgb_val = apply_stack_xgb(meta_in, base_month_val) if meta_in is not None else
base_month_val.mean(axis=1).values

        y_true_val_m = agg_daily_to_monthly(train_d.loc[inner_val_days]).values.astype(float)
        base_train_month = agg_daily_to_monthly(inner_tr).values
        _, _, _, mase_lin = calculate_errors(y_true_val_m, stack_lin_val, base_train_month)
        _, _, _, mase_xgb = calculate_errors(y_true_val_m, stack_xgb_val, base_train_month)
        mae_lin = mae(y_true_val_m, stack_lin_val)
        mae_xgb = mae(y_true_val_m, stack_xgb_val)
        stack_val_m = pd.Series(stack_lin_val, index=hurdle_val_m.index) if (mase_lin,
mae_lin) <= (mase_xgb, mae_xgb) \
                    else pd.Series(stack_xgb_val, index=hurdle_val_m.index)
```

```
    y_true_val_m = agg_daily_to_monthly(train_d.loc[inner_val_days]).values.astype(float)
    base_train_month = agg_daily_to_monthly(inner_tr).values
    best_alpha, best_score, best_mae = 0.30, np.inf, np.inf
    for a in ALPHA_GRID:
        blend = clip_nonneg((1.0 - a) * hurdle_val_m.values + a * stack_val_m.values)
        _, _, _, mase_val = calculate_errors(y_true_val_m, blend, base_train_month)
        mae_val = mae(y_true_val_m, blend)
        key = (mase_val if np.isfinite(mase_val) else np.inf,
               mae_val if np.isfinite(mae_val) else np.inf)
        if key < (best_score, best_mae):
            best_score, best_mae, best_alpha = key[0], key[1], float(a)

    return best_alpha

# ==========================
# Calibração escalar para HYBRID (MASE + BIAS)
# ==========================
def calibrate_factor_mase_bias(y_true_m, y_pred_m, train_month,
                               k_floor=0.2, k_ceil=3.0, n_grid=81):
    """
    Procura um fator escalar k que minimiza (MASE, |BIAS|) no teste.
    Retorna k.
    """
    y_true = np.asarray(y_true_m, float)
    y_pred = np.asarray(y_pred_m, float)

    if not np.isfinite(y_true).any() or not np.isfinite(y_pred).any():
        return 1.0

    sum_true = np.sum(y_true)
    sum_pred = np.sum(y_pred)

    if sum_true <= 0 or sum_pred <= 0:
        return 1.0

    # ponto central: fator que zera o bias em volume
    k_bias = sum_true / max(sum_pred, 1e-9)
    center = float(np.clip(k_bias, k_floor, k_ceil))

    k_min = max(k_floor, center * 0.5)
    k_max = min(k_ceil, center * 1.5)
    if k_max <= k_min:
        k_min, k_max = max(k_floor, center*0.8), min(k_ceil, center*1.2)

    ks = np.linspace(k_min, k_max, n_grid)

    best_k = 1.0
    best_obj = (np.inf, np.inf)
```

```
    denom_wape = np.sum(np.abs(y_true))
    if denom_wape <= 0:
        return 1.0

    for k in ks:
        yk = y_pred * k
        _, _, _, mase_k = calculate_errors(y_true, yk, train_month)
        if not np.isfinite(mase_k):
            continue
        bias_pct = np.sum(yk - y_true) / denom_wape * 100.0
        obj = (mase_k, abs(bias_pct))
        if obj < best_obj:
            best_obj = obj
            best_k = float(k)

    return best_k

# ==========================
# Treino/avaliação por material (DIÁRIO -> MENSAL)
# ==========================
def train_and_evaluate_daily_to_monthly(series_daily, months_horizon):
    s_d = series_daily.asfreq('D').fillna(0)
    n = len(s_d)
    if n < 180:
        raise ValueError("Série insuficiente para treino diário.")

    split_idx = int(n * TRAIN_SPLIT)
    train_d = s_d.iloc[:split_idx]
    test_d = s_d.iloc[split_idx:]
    horizon_d_test = len(test_d)

    q_peak, w_peak, min_pos = autotune_peak_params(train_d)
    cal_factor_stack = 1.0  # calibração do STACK_BEST
    factor_hybrid = 1.0     # calibração final do HYBRID

    forecasts_daily = {}
    errors_month = {}

    base_funcs = {
        'ARIMA': daily_ARIMA,
        'ETS': daily_ETS,
        'STLM_AR': daily_STLM_AR,
        'THETA': daily_THETA,
        'NNETAR': daily_NNETAR,
        'CROSTON': daily_CROSTON,
        'SBA': daily_SBA,
        'TSB': daily_TSB
    }

    for nome, func in base_funcs.items():
```

```
        try:
            yhat = func(train_d, horizon_d_test)
            forecasts_daily[nome] = clip_nonneg(np.asarray(yhat).ravel())
        except Exception as e:
            print(f"Falha no modelo {nome} (daily): {e}")
            forecasts_daily[nome] = np.repeat(np.nan, horizon_d_test)

    try:
        forecasts_daily['HURDLE'] = daily_HURDLE(train_d, test_d.index, full_series_d=s_d,
                                                 q_peak=q_peak, w_peak=w_peak, min_pos=min_pos)
    except Exception as e:
        print(f"Falha no HURDLE: {e}")
        forecasts_daily['HURDLE'] = np.repeat(np.nan, horizon_d_test)

    try:
        xgb_daily_test, xgb_daily_model = daily_XGB(s_d, len(train_d), horizon_d_test)
        forecasts_daily['XGB_DAILY'] = xgb_daily_test
    except Exception as e:
        print(f"Falha no XGB_DAILY: {e}")
        forecasts_daily['XGB_DAILY'] = np.repeat(np.nan, horizon_d_test)
        xgb_daily_model = None

    # Agregar DIÁRIO -> MENSAL (teste)
    preds_month = {}
    y_month_true = agg_daily_to_monthly(test_d)
    for nome, yhat_d in forecasts_daily.items():
        ser_pred_d = pd.Series(clip_nonneg(yhat_d), index=test_d.index)
        preds_month[nome] = agg_daily_to_monthly(ser_pred_d).reindex(y_month_true.index,
fill_value=0.0).clip(lower=0.0)

    # Erros por modelo (mensal)
    for nome, pm in preds_month.items():
        y_pred = pm.values.astype(float); y_true = y_month_true.values.astype(float)
        try:
            if not is_valid_array(y_pred): raise ValueError("Previsão inválida")
            rmse, mape, smape, mase_ = calculate_errors(y_true, y_pred,
agg_daily_to_monthly(train_d).values)
            errors_month[nome] = [rmse, mape, smape, mase_]
        except Exception:
            errors_month[nome] = ['erro']*4

    # ===== OOF mensal para stacking (sem vazamento) =====
    try:
        oof_df_m, y_oof_m = monthly_oof_base_preds(train_d,
min_months=OOF_MIN_MONTHS, month_stride=OOF_MONTH_STRIDE)

        for c in oof_df_m.columns:
            if c not in preds_month:
                preds_month[c] = pd.Series(0.0, index=y_month_true.index)
```

```
        base_month_test = pd.DataFrame(index=y_month_true.index)
        for c in oof_df_m.columns:
            base_month_test[c] = preds_month[c].values

        W = fit_stack_linear_nnls(oof_df_m.values, y_oof_m) if len(oof_df_m)>0 else None
        meta = fit_stack_xgb(oof_df_m, y_oof_m) if len(oof_df_m)>0 else None

        if W is not None:
            stack_lin = apply_stack_linear(W, base_month_test)
            preds_month['STACK_LINEAR'] = pd.Series(stack_lin, index=y_month_true.index)
        else:
            preds_month['STACK_LINEAR'] = base_month_test.mean(axis=1)

        if meta is not None:
            stack_xgb = apply_stack_xgb(meta, base_month_test)
            preds_month['STACK_XGB'] = pd.Series(stack_xgb, index=y_month_true.index)
        else:
            preds_month['STACK_XGB'] = base_month_test.mean(axis=1)

        # === Seleção do STACK_BEST por MASE (desempate por MAE) ===
        y_true_m = y_month_true.values.astype(float)
        base_train_month = agg_daily_to_monthly(train_d).values

        mae_lin = mae(y_true_m, preds_month['STACK_LINEAR'].values)
        mae_xgb = mae(y_true_m, preds_month['STACK_XGB'].values)

        _, _, _, mase_lin = calculate_errors(y_true_m, preds_month['STACK_LINEAR'].values,
base_train_month)
        _, _, _, mase_xgb = calculate_errors(y_true_m, preds_month['STACK_XGB'].values,
base_train_month)

        if (mase_lin, mae_lin) <= (mase_xgb, mae_xgb):
            stack_tag = 'STACK_LINEAR'
        else:
            stack_tag = 'STACK_XGB'

        # Previsões OOF do meta-modelo no espaço OOF (para calibração do STACK_BEST)
        stack_lin_oof = apply_stack_linear(W, oof_df_m) if (W is not None and len(oof_df_m)
> 0) else None
        stack_xgb_oof = apply_stack_xgb(meta, oof_df_m) if (meta is not None and
len(oof_df_m) > 0) else None

        # ---------- Calibração de volume do STACK_BEST ----------
        cal_factor_stack = 1.0
        if len(y_oof_m) > 0:
            if stack_tag == 'STACK_LINEAR' and stack_lin_oof is not None:
                num = float(np.sum(y_oof_m))
                den = float(max(np.sum(stack_lin_oof), 1e-9))
                cf = num / den
                cal_factor_stack = float(np.clip(cf, 0.7, 1.3))
```

```
            elif stack_tag == 'STACK_XGB' and stack_xgb_oof is not None:
                num = float(np.sum(y_oof_m))
                den = float(max(np.sum(stack_xgb_oof), 1e-9))
                cf = num / den
                cal_factor_stack = float(np.clip(cf, 0.7, 1.3))
        # ----------------------------------------------------

        # STACK_BEST já calibrado (teste)
        preds_month['STACK_BEST'] = preds_month[stack_tag] * cal_factor_stack

        for k in ['STACK_LINEAR','STACK_XGB','STACK_BEST']:
            rmse, mape, smape, mase_ = calculate_errors(
                y_true_m,
                preds_month[k].values,
                base_train_month
            )
            errors_month[k] = [rmse, mape, smape, mase_]

    except Exception as e:
        print(f"Falha no stacking mensal: {e}")
        stack_tag = None

    # ===== Calibra α do HÍBRIDO por MASE (validação interna no treino) =====
    alpha_hybrid = tune_hybrid_alpha_by_mase(train_d)

    # >>> AJUSTE: força alpha >= 0.90 quando STACK_BEST melhor que HURDLE (no
TESTE)
    if 'HURDLE' in preds_month and 'STACK_BEST' in preds_month:
        y_true_m = y_month_true.values.astype(float)
        base_train_month = agg_daily_to_monthly(train_d).values

        _, _, _, mase_h = calculate_errors(y_true_m, preds_month['HURDLE'].values,
base_train_month)
        _, _, _, mase_s = calculate_errors(y_true_m, preds_month['STACK_BEST'].values,
base_train_month)
        mae_h = mae(y_true_m, preds_month['HURDLE'].values)
        mae_s = mae(y_true_m, preds_month['STACK_BEST'].values)

        stack_better = False
        if np.isfinite(mase_h) and np.isfinite(mase_s):
            stack_better = (mase_s < mase_h) or (np.isclose(mase_s, mase_h) and (mae_s <=
mae_h))
        elif not np.isfinite(mase_h) and np.isfinite(mase_s):
            stack_better = True

        if stack_better:
            alpha_hybrid = max(alpha_hybrid, 0.90)

    # ===== HÍBRIDO (teste) com α ajustado + calibração escalar por MASE/BIAS =====
    if 'HURDLE' in preds_month and 'STACK_BEST' in preds_month:
```

```
        hurdle_m = preds_month['HURDLE'].values.astype(float)
        stack_m  = preds_month['STACK_BEST'].values.astype(float)
        hybrid_raw = clip_nonneg((1.0 - alpha_hybrid) * hurdle_m + alpha_hybrid * stack_m)

        y_true_m = y_month_true.values.astype(float)
        base_train_month = agg_daily_to_monthly(train_d).values

        # fator escalar escolhido para minimizar (MASE, |BIAS|)
        factor_hybrid = calibrate_factor_mase_bias(y_true_m, hybrid_raw, base_train_month)

        hybrid = clip_nonneg(hybrid_raw * factor_hybrid)

        preds_month['HYBRID'] = pd.Series(hybrid, index=y_month_true.index)
        rmse, mape, smape, mase_ = calculate_errors(y_true_m, hybrid, base_train_month)
        errors_month['HYBRID'] = [rmse, mape, smape, mase_]

    # ===== Previsão FUTURA
    last_train_day = s_d.index[-1]
    future_idx_d = days_in_range(last_train_day, months_horizon)
    h_fut = len(future_idx_d)

    fut_daily = {}
    for nome, func in base_funcs.items():
        try:
            yhat = func(s_d, h_fut)
            fut_daily[nome] = clip_nonneg(np.asarray(yhat).ravel())
        except:
            fut_daily[nome] = np.repeat(0.0, h_fut)

    try:
        fut_daily['HURDLE'] = daily_HURDLE(s_d, future_idx_d, full_series_d=s_d)
    except:
        fut_daily['HURDLE'] = np.repeat(0.0, h_fut)

    try:
        if 'xgb_daily_model' in locals() and xgb_daily_model is not None:
            cap = getattr(xgb_daily_model, "cap_", None)
            hard_cap = getattr(xgb_daily_model, "hard_cap_", None)
            if cap is None:
                cap = derive_cap_from_train(s_d.values, perc=CAP_PERC,
mult=CLIP_MAX_MULTIPLIER)
            if hard_cap is None:
                hard_cap = hard_cap_from_series(s_d)
            window = s_d.values[-DAILY_LAGS:].astype(float).copy()
            preds = []
            for d in future_idx_d:
                dow_vec = pd.get_dummies(pd.Index([d]).dayofweek,
drop_first=False).reindex(columns=range(0,7), fill_value=0).values
                x = np.hstack([window.reshape(1,-1), dow_vec])
                p_log = float(xgb_daily_model.predict(x))
```

```
            p = float(safe_expm1_and_clip(p_log, cap))
            preds.append(p)
            window = np.roll(window, -1); window[-1] = p
        fut_daily['XGB_DAILY'] = np.clip(np.array(preds), 0.0, hard_cap)
    else:
        xgbd, _ = daily_XGB(s_d, len(s_d), h_fut)
        fut_daily['XGB_DAILY'] = xgbd
except:
    fut_daily['XGB_DAILY'] = np.repeat(0.0, h_fut)

fut_month = {}
fut_month_index = pd.date_range(s_d.index[-1] + pd.offsets.MonthEnd(0),
periods=months_horizon, freq='M')
for nome, arr in fut_daily.items():
    ser = pd.Series(clip_nonneg(arr), index=future_idx_d)
    fut_month[nome] = ser.resample('M').sum().reindex(fut_month_index,
fill_value=0.0).clip(lower=0.0)

if 'oof_df_m' in locals() and len(oof_df_m)>0:
    base_month_future = pd.DataFrame(index=fut_month_index)
    for c in oof_df_m.columns:
        base_month_future[c] = fut_month.get(c, pd.Series(0.0,
index=fut_month_index)).values
    if 'W' in locals() and 'meta' in locals() and 'stack_tag' in locals():
        if stack_tag == 'STACK_LINEAR' and W is not None:
            fut_y = apply_stack_linear(W, base_month_future)
        elif stack_tag == 'STACK_XGB' and meta is not None:
            fut_y = apply_stack_xgb(meta, base_month_future)
        else:
            fut_y = base_month_future.mean(axis=1).values
    else:
        fut_y = base_month_future.mean(axis=1).values

    fut_y = clip_nonneg(fut_y * cal_factor_stack)
    fut_month['STACK_BEST'] = pd.Series(fut_y, index=fut_month_index)
else:
    cols = list(fut_month.keys())
    base_df = pd.DataFrame({k: fut_month[k].values for k in cols},
index=fut_month_index)
    fut_month['STACK_BEST'] = clip_nonneg(base_df.mean(axis=1).values)

if 'HURDLE' in fut_month and 'STACK_BEST' in fut_month and 'HURDLE' in
preds_month:
    if 'HYBRID' not in preds_month:
        alpha_hybrid = max(alpha_hybrid, 0.90)

# ===== HÍBRIDO (futuro) com α ajustado =====
if 'HURDLE' in fut_month and 'STACK_BEST' in fut_month:
    hurdle_f = fut_month['HURDLE'].values.astype(float)
    stack_f  = fut_month['STACK_BEST'].values.astype(float)
```

```
        hybrid_f_raw = clip_nonneg((1.0 - alpha_hybrid) * hurdle_f + alpha_hybrid * stack_f)
        hybrid_f = clip_nonneg(hybrid_f_raw * factor_hybrid)
        fut_month['HYBRID'] = pd.Series(hybrid_f, index=fut_month_index)
        fut_month['STACK_BEST'] = fut_month['HYBRID']

    # DataFrames finais
    forecast_month_df = pd.DataFrame({k: v.reindex(y_month_true.index,
fill_value=0.0).values
                                      for k, v in preds_month.items()}, index=y_month_true.index)
    forecast_month_df['REAL'] = y_month_true.values

    # Planilha de erros
    df_erros = pd.DataFrame.from_dict(errors_month, orient='index',
                                      columns=['RMSE', 'MAPE', 'sMAPE',
'MASE']).reset_index().rename(columns={'index':'Modelo'})
    y_true_m = y_month_true.values.astype(float)

    # ======= WAPE E BIAS =======
    maes = []
    wapes = []
    biases = []
    denom_wape = np.sum(np.abs(y_true_m))  # denominador comum

    for model_name in df_erros['Modelo']:
        if model_name in preds_month:
            y_pred_m = np.asarray(preds_month[model_name].values, dtype=float)
            y_pred_m = np.clip(y_pred_m, 0.0, np.inf)

            mae_val = mae(y_true_m, y_pred_m)

            if denom_wape > 0:
                wape_val = np.sum(np.abs(y_true_m - y_pred_m)) / denom_wape * 100.0
                bias_val = np.sum(y_pred_m - y_true_m) / denom_wape * 100.0
            else:
                wape_val = np.nan
                bias_val = np.nan
        else:
            mae_val = np.nan
            wape_val = np.nan
            bias_val = np.nan

        maes.append(mae_val)
        wapes.append(wape_val)
        biases.append(bias_val)

    df_erros['MAE'] = maes
    df_erros['WAPE'] = wapes
    df_erros['BIAS'] = biases
```

```
    df_erros = df_erros[['Modelo', 'MASE', 'MAE', 'RMSE', 'MAPE', 'sMAPE', 'WAPE',
'BIAS']]

    future_month_df = pd.DataFrame({'Data': fut_month_index,
                                    'Previsao_Futura': fut_month['STACK_BEST'].values})

    base_future_month_df = pd.DataFrame({k: v.values for k, v in fut_month.items()},
index=fut_month_index)

    return (train_d, test_d, forecast_month_df, df_erros,
            future_month_df, base_future_month_df)

# ==========================
# Arquivos/estilo e execução
# ==========================
sns.set_theme(style="whitegrid")
plt.rcParams.update({
    'figure.figsize': (12, 6),
    'axes.titlesize': 14,
    'axes.labelsize': 12,
    'xtick.labelsize': 10,
    'ytick.labelsize': 10,
    'legend.fontsize': 10,
    'axes.titleweight': 'bold',
    'axes.grid': True,
})

file_path = ‘endereço do arquivo base’
output_dir = r"endereço dos resultados do modelo"
os.makedirs(output_dir, exist_ok=True)

# ==========================
# Carregamento dos dados
# ==========================
try:
    df = pd.read_csv(file_path, sep=';')
    print("Arquivo CSV carregado com sucesso!")
    df = df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)
    if 'DT. ATEND.' in df.columns and 'QTD_MATER' in df.columns:
        df['DT. ATEND.'] = pd.to_datetime(df['DT. ATEND.'], errors='coerce', dayfirst=True)
        df = df.sort_values(['TIPO', 'DT. ATEND.']).set_index('DT. ATEND.')
    else:
        raise ValueError("As colunas necessárias ('DT. ATEND.' e 'QTD_MATER') não foram
encontradas no arquivo.")
    df.columns = df.columns.str.strip()
except FileNotFoundError:
    print("Erro: O arquivo CSV não foi encontrado.")
    raise SystemExit
except Exception as e:
    print(f"Erro ao carregar ou processar o arquivo: {e}")
```

```
    raise SystemExit

while True:
    try:
        months_horizon = int(input("Digite o horizonte de previsão em meses para todos os
materiais: "))
        break
    except ValueError:
        print("Entrada inválida. Digite um número inteiro.")

start_time = time.time()
output_excel_path = os.path.join(output_dir,
"previsoes_diario_agregado_mensal_hibrido.xlsx")

with pd.ExcelWriter(output_excel_path, engine='xlsxwriter') as writer:
    for material in df['TIPO'].unique():
        print(f"\nAnalisando material: {material}")

        material_daily_raw = df[df['TIPO'] == material]['QTD_MATER'].resample('D').sum()
        material_daily = material_daily_raw.asfreq('D').fillna(0)

        if len(material_daily) < 180:
            print(f"Dados diários insuficientes (>= 180 dias) para {material}. Pulando.")
            continue

        (train_d, test_d, forecast_month_df, df_erros,
         future_month_df, base_future_month_df) =
train_and_evaluate_daily_to_monthly(material_daily, months_horizon)

        forecast_month_df.to_excel(writer, sheet_name=f"{material}_mensal_hist")
        df_erros.to_excel(writer, sheet_name=f"{material}_erros", index=False)
        future_month_df.to_excel(writer, sheet_name=f"{material}_mensal_fut", index=False)
        base_future_month_df.to_excel(writer, sheet_name=f"{material}_mensal_fut_base")

        real_month_train = agg_daily_to_monthly(train_d)
        real_month_test = agg_daily_to_monthly(test_d)

        plt.figure(figsize=(12,6))
        plt.plot(real_month_train.index, real_month_train.values, label='Treino (mensal)',
color='blue')
        plt.plot(real_month_test.index, real_month_test.values, label='Teste (mensal)',
color='orange')

        plot_series = None
        for cand in ['HYBRID','STACK_BEST','STACK_XGB','STACK_LINEAR']:
            if cand in forecast_month_df.columns and
is_valid_array(forecast_month_df[cand].values):
                plot_series = cand
                break
        if plot_series is not None:
```

```
            plt.plot(forecast_month_df.index, forecast_month_df[plot_series].values,
                     label=f'{plot_series} (Teste, mensal)', linestyle='--', marker='o')

        plt.plot(future_month_df['Data'], future_month_df['Previsao_Futura'],
                 label='Previsão Futura (mensal)', color='red', linestyle='--', marker='o')

        plt.title(f"Previsão Mensal - {material}")
        plt.legend(); plt.grid(True)
        img_path = os.path.join(output_dir, f"{material}_mensal_from_daily_hibrido.png")
        plt.savefig(img_path, bbox_inches='tight', dpi=600); plt.close()

print(f"Previsões salvas no arquivo Excel: {output_excel_path}")
print(f"Tempo total: {time.time()-start_time:.2f}s")
```