APLICAÇÃO DE MODELOS DE APRENDIZADO DE MÁQUINA E MÉTODOS DE ENSEMBLE PARA A PREVISÃO DE DEMANDA NA INDÚSTRIA TÊXTIL

Rael dos Santos Nehring, Andreza Sartori – Orientadora

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

rsnehring@furb.br, asartori@furb.br

Resumo: A indústria têxtil brasileira movimentou R$ 190 bilhões em 2021, com Santa Catarina concentrando 30% das unidades produtivas. A previsão de demanda é um processo crítico para a eficiência operacional dessas empresas, uma vez que métodos tradicionais apresentam erros de até 79% na previsão de demanda. *Este trabalho tem como objetivo desenvolver um modelo de aprendizado de máquina para aprimorar a previsão de demanda na indústria têxtil. Para isso, foram implementados seis modelos de aprendizado de máquina (CatBoost, XGBoost, ElasticNet, Polynomial Ridge, LSTM e BiLSTM) e dois métodos de ensemble, aplicados a dados reais contendo 13.9 milhões de transações de uma empresa têxtil do Vale do Itajaí, Santa Catarina. A metodologia incluiu a criação de 130 variáveis derivadas (defasagens temporais, médias móveis, tendências e sazonalidade), validação cruzada temporal e otimização Bayesiana via Optuna. O CatBoost apresentou MAE de 0.2524 e MAPE de 9.27%, enquanto o ensemble calibrado alcançou erro na previsão total do volume de vendas de 3.0%. A análise estatística revelou que características temporais representam 65% da relevância preditiva, enquanto indicadores econômicos mensais não contribuíram para o modelo. Conclui-se que modelos de aprendizado de máquina são ferramentas viáveis para previsão de demanda no setor têxtil.*

Palavras-chave: Previsão de Demanda. Aprendizado de Máquina. Indústria têxtil. Ensemble. Séries temporais.

1 INTRODUÇÃO

Em 2021, a indústria têxtil brasileira movimentou cerca de R$ 190 bilhões, respondendo por 5,7% do PIB industrial brasileiro, segundo levantamento feito pela Associação Brasileira da Indústria Têxtil e de Confecção, divulgado em 2022 (Carvalho; Vidal, 2023). Em Santa Catarina, o setor têxtil e de confecção é o maior empregador industrial do estado, responsável por mais de 26% dos postos de trabalho (MILNITZ et al., 2023). Com aproximadamente 5.000 empresas, Santa Catarina representa 30% do total nacional em unidades produtivas têxteis e de vestuário (IEMI, 2024). O Vale do Itajaí concentra o principal polo produtivo, com destaque para Blumenau, Brusque e Itajaí (FACISC, 2024). Esses dados demonstram que a indústria têxtil é uma das mais importantes e competitivas da região do Vale do Itajaí, estado de Santa Catarina.

Em um ambiente de alta competitividade, as indústrias precisam adotar soluções que tornem seus processos mais eficientes, garantindo que consigam manter sua posição no mercado. Um dos processos mais relevantes para a eficiência operacional é a previsão de demanda. Conforme divulgado pelo Febratex Group (2019), altos estoques exigem maior investimento para manutenção, diminuindo o poder de compra da empresa e prejudicando o fluxo de caixa. Candeias et al. (2020), em pesquisa sobre o setor têxtil esportivo brasileiro, demonstraram que métodos tradicionais de previsão apresentam taxas de erro (MAPE) que podem ultrapassar 79% em determinadas famílias de produtos. Segundo os autores, previsões imprecisas levam a perdas por ruptura de estoque ou excesso, impactando negativamente o desempenho econômico. Nesse mesmo contexto, Lorente-Leyva et al. (2020) observaram que métodos tradicionais como Média Móvel Integrada Autorregressiva (ARIMA) e Holt-Winters, embora amplamente utilizados, podem apresentar desempenho inferior em cenários com alto grau de incerteza.

Embora modelos tradicionais como ARIMA e Holt-Winters sejam amplamente utilizados para previsão de demanda em séries temporais, eles apresentam limitações em cenários com alta complexidade e sazonalidade acentuada, características presentes na indústria têxtil (Saadeddin, 2024; Duarte et al., 2014). A sazonalidade é definida como a variação nos padrões de consumo e produção que ocorre em determinados períodos do ano, geralmente influenciada por fatores externos como datas comemorativas, estações do ano, eventos esportivos ou culturais (Wildemberg, 2023). No setor têxtil, essas variações são mais notáveis nas mudanças de coleções conforme as estações (verão/inverno), impactando diretamente na demanda por categorias específicas de roupas. Segundo o Sebrae (2023), a sazonalidade pode ocorrer em escala diária, semanal, mensal ou anual.

Pesquisas recentes apontam ganhos de precisão na previsão de demanda com aprendizado de máquina. Henzel et al. (2022) propuseram um modelo ensemble combinando K-Nearest Neighbor (KNN) e Linear Mixed Model (LMM) para previsão semanal desagregada por produto, cor e tamanho, reduzindo o Erro Médio Absoluto (MAE) em diversas categorias de produtos. Lv et al. (2023) aplicaram Random Forest (RF), Extreme Gradient Boosting (XGB) e Gradient Boosting Decision Tree (GBDT), demonstrando que a inclusão de variáveis climáticas pode diminuir o Erro Quadrático Médio (MSE) em até 86% para roupas sazonais. Yasir et al. (2022) demonstram que a incorporação de variáveis externas como indicadores econômicos (inflação e Índice de Preços ao Consumidor) pode contribuir positivamente para previsão de demanda.

Nesse contexto, o objetivo desse trabalho é desenvolver um modelo de aprendizado de máquina para aprimorar a previsão de demanda, visando reduzir os níveis de estoque e aumentar a margem de contribuição. Os objetivos específicos contemplam: (i) avaliar o desempenho de diferentes modelos de aprendizado de máquina na previsão de demanda na indústria têxtil, utilizando métricas como MAE, MAPE, RMSE e R^2; (ii) analisar o impacto da integração de variáveis externas, como indicadores econômicos e sazonalidade, na acurácia das previsões de demanda geradas pelos modelos de aprendizado de máquina; (iii) validar a eficácia do modelo de aprendizado de máquina com melhor desempenho em um estudo de caso com dados reais de uma indústria têxtil da região do Vale do Itajaí, comparando seus resultados de previsão com métodos tradicionalmente utilizados no setor. Os algoritmos implementados neste trabalho incluem XGBoost, CatBoost, Lasso, Ridge, Polynomial Regression, Bidirectional Recurrent Neural Network (BRNN) e Long Short-Term Memory (LSTM).

Considerando as limitações dos métodos tradicionais e os avanços recentes demonstrados pela aplicação de aprendizado de máquina em contextos similares, espera-se que a abordagem proposta neste trabalho possa superar as taxas de erro observadas nos métodos tradicionais de previsão de demanda. A hipótese central é que a combinação de múltiplos modelos de aprendizado de máquina, aliada à criação de variáveis temporais derivadas e à incorporação de variáveis externas (climáticas e econômicas), permitirá capturar padrões complexos de sazonalidade e tendência presentes na demanda têxtil, resultando em previsões mais precisas que possibilitem melhor planejamento de produção e gestão de estoque na indústria têxtil do Vale do Itajaí.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os fundamentos teóricos que embasam este trabalho. A subseção 2.1 descreve os modelos de aprendizado de máquina utilizados, as métricas de avaliação e o tratamento de séries temporais. A subseção 2.2 caracteriza o processo atual da empresa parceira. Por fim, a subseção 2.3 sintetiza os principais trabalhos correlatos.

2.1 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina (AM) é uma subárea da Inteligência Artificial que se dedica ao desenvolvimento de algoritmos capazes de aprender padrões a partir de dados. Segundo Kelleher et al. (2015, p. 1, tradução nossa), "O aprendizado de máquina é definido como um processo automatizado que extrai padrões dos dados". No contexto do AM, é importante diferenciar os conceitos de algoritmo e modelo, uma vez que essa distinção facilita a compreensão dos processos envolvidos. Conforme Géron (2019), o algoritmo de aprendizado de máquina é um procedimento que processa os dados para gerar um modelo, atuando como uma receita, enquanto o modelo representa o resultado desse processo.

Os modelos de aprendizado de máquina utilizados neste trabalho são classificados em três grupos que se diferenciam pela arquitetura: Algoritmos de Regressão (ElasticNet e Polynomial Ridge), Redes Neurais Recorrentes (LSTM e BiLSTM) e Algoritmos de Gradient Boosting (XGBoost e CatBoost). Cada grupo apresenta características específicas que os tornam adequados para diferentes aspectos da previsão de demanda em séries temporais.

2.1.1 Algoritmos de Regressão

Algoritmos de regressão são modelos de aprendizado supervisionado cujo objetivo é prever um valor numérico contínuo (como preço, demanda ou vendas) a partir de um conjunto de características preditoras, diferenciando-se dos problemas de classificação que preveem categorias discretas (GÉRON, 2019).

Diversas abordagens incorporam técnicas de regularização para melhorar a generalização. A Ridge Regression adiciona regularização L2 (penalização quadrática dos pesos) à função de custo, controlada pelo parâmetro de regularização que mantém os pesos pequenos. Segundo Géron (2019), essa técnica reduz a variância do modelo, mas aumenta ligeiramente o viés, sendo particularmente útil quando há multicolinearidade entre as variáveis preditoras. A Lasso Regression utiliza regularização L1 (penalização absoluta dos pesos), eliminando pesos de *features* menos importantes e realizando seleção automática de *features* (Géron, 2019; Hastie, Tibshirani e Friedman, 2009). Fridgeirsson et al. (2024) compararam técnicas de regularização em 21 problemas de predição, demonstrando que Lasso e ElasticNet apresentam melhor discriminação entre variáveis relevantes e irrelevantes. O ElasticNet combina penalidades L1 e L2, oferecendo equilíbrio entre as abordagens Ridge e Lasso (Beddar-Wiesing et al., 2023). Essa combinação permite que o modelo realize seleção de *features* como o Lasso, mantendo a estabilidade do Ridge quando há *features* correlacionadas.

A Polynomial Regression adiciona potências de *features* como novas variáveis, permitindo ajustar dados não lineares e encontrar relações mais complexas entre *features* (Géron, 2019). Entretanto, Géron (2019) ressalta que um grau polinomial muito alto pode causar *overfitting* severo, prejudicando a capacidade de generalização do modelo para novos dados.

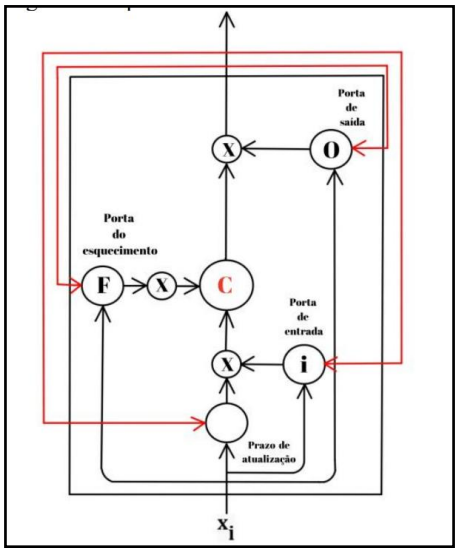
2.1.2 Redes Neurais Recorrentes e Séries Temporais

As Redes Neurais Recorrentes (*Recurrent Neural Network* - RNN) são arquiteturas especializadas em processar dados sequenciais, sendo particularmente adequadas para aplicações com séries temporais. Hyndman e Athanasopoulos (2018) destacam que, em modelos voltados à previsão, a ordem temporal dos dados deve ser preservada, pois os valores passados podem influenciar diretamente os valores futuros. Os autores também ressaltam que séries temporais podem ser decompostas em tendência, sazonalidade e ruído, o que permite identificar padrões cíclicos e projetar a demanda com maior precisão. Enquanto modelos tradicionais, como o ARIMA, são indicados para séries estacionárias e utilizam componentes autorregressivos, de média móvel e de diferenciação, para séries com múltiplas variáveis ou não linearidades, redes neurais como o LSTM são amplamente utilizadas.

De acordo com Ribeiro et al. (2024), a arquitetura Long Short-Term Memory (LSTM), é uma evolução das redes recorrentes tradicionais capaz de captar dependências de longo prazo em dados sequenciais. Cada célula LSTM é composta por uma célula de memória central e três portões distintos: o portão de entrada, que regula a entrada de novas informações relevantes; o portão de esquecimento, responsável por eliminar informações que não devem mais ser consideradas; e o portão de saída, que seleciona os dados que gerarão a saída da célula para a próxima etapa da rede. Essas estruturas trabalham com funções de ativação que determinam, em cada momento, quais informações serão atualizadas, mantidas ou descartadas. Segundo Suresh, et al. (2022), isso permite ao modelo preservar e manipular informações importantes em sequências extensas, reduzindo impactos daqueles eventos que ocorrem com intervalos de tempo variados e superando limitações das RNNs convencionais, como o desaparecimento do gradiente.

A Figura 1 apresenta o diagrama básico de uma célula LSTM. Nela, é possível visualizar a célula de memória central e os três portões principais que atuam controlando o fluxo de informações. O portão de esquecimento determina se um valor deve ou não ser mantido na célula de memória. O portão de entrada define se novos valores de entrada devem ser memorizados. O portão de saída decide qual informação será propagada adiante na rede. Cada portão atua com base nas entradas atuais e nos estados armazenados, o que permite a modelagem de dependências temporais complexas (Ribeiro et al., 2024).

Figura 1 – Diagrama da célula LSTM



Fonte: Ribeiro et al. (2024).

Abbasimehr et al. (2020) demonstraram que LSTM multicamadas com otimização de hiperparâmetros supera métodos tradicionais como ARIMA e RNN em dados reais de manufatura. Albeladi et al. (2023) também evidenciaram que a arquitetura LSTM é eficaz para capturar dependências temporais de longo prazo em padrões não-lineares, tornando o LSTM mais adequado para problemas de previsão de demanda.

Uma extensão da arquitetura LSTM são as Redes Neurais Recorrentes Bidirecionais (BRNN), que representam uma evolução das RNNs tradicionais ao ser capaz de processar informações tanto na ordem cronológica quando na ordem inversa. Schuster e Paliwal (1997) destacam que a BRNN utiliza dois conjuntos de neurônios, um processando a sequência da origem ao fim e outro no sentido contrário. Na etapa de previsão, os resultados desses dois fluxos são combinados,

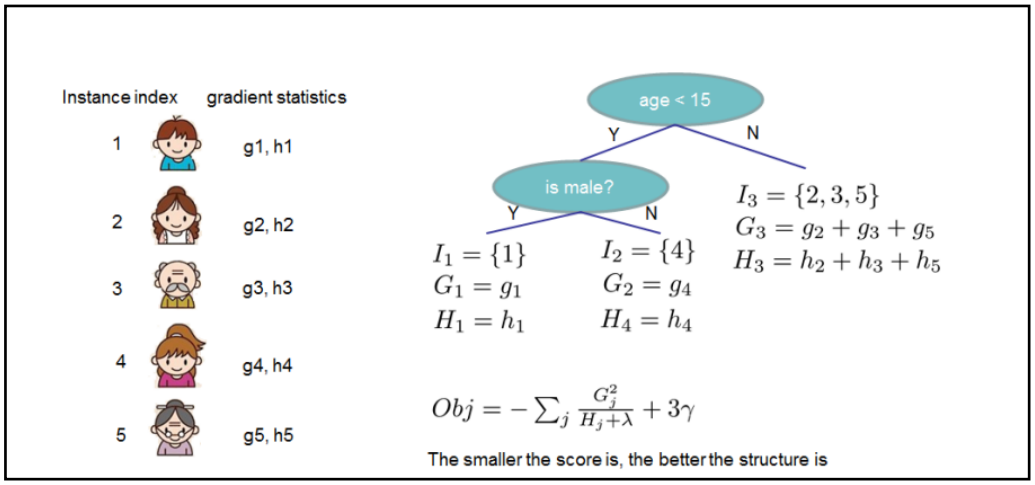
permitindo que cada elemento da sequência avalie tanto os contextos anteriores quanto posteriores. Essa arquitetura amplia a compreensão do contexto temporal, o que a torna útil em situações em que entender relações passadas e futuras é importante para aumentar a precisão das previsões (Lee et al., 2022; Schuster; Paliwal, 1997).

2.1.3 Algoritmos de Gradient Boosting

De acordo com Pinheiro (2023), o Gradient Boosting é uma técnica de aprendizado supervisionado que combina sequencialmente múltiplas árvores de decisão para formar um modelo preditivo robusto. Cada nova árvore é treinada para corrigir os erros das árvores anteriores, formando um modelo aditivo. O algoritmo calcula o gradiente da função de perda em relação às predições atuais, gerando os pseudo-resíduos. Em seguida, ajusta uma nova árvore para prever esses pseudo-resíduos. O modelo final resulta da soma ponderada de todas as árvores, onde cada contribuição é multiplicada por uma taxa de aprendizado que define sua influência no resultado.

O Extreme Gradient Boosting (XGBoost), implementa otimizações significativas dessa abordagem. A Figura 2 ilustra um exemplo de árvore construída pelo XGBoost, no qual um conjunto de cinco instâncias, cada uma associada ao seu gradiente (g_i) e hessiano (h_i), é dividido por meio de regras de decisão sucessivas. Para cada folha resultante, são calculados os somatórios dos gradientes e hessianos (G_j e H_j), que são então utilizados na função objetivo (Obj) apresentada na Figura 2 . Essa função representa o score da estrutura da árvore e permite avaliar o quanto cada divisão contribui para reduzir a perda regularizada. Dessa forma, a figura demonstra de maneira prática como o algoritmo aplica os cálculos estatísticos para determinar o melhor split em cada nível.

Figura 2 – Estrutura de uma árvore do XGBoost



Fonte: Chen e Guestrin (2016).

Chen e Guestrin (2016) descrevem que a função objetivo do XGBoost combina a função de perda com termos de regularização L1 e L2 aplicados aos pesos das folhas, controlando a complexidade do modelo. Pinheiro (2023) detalha que, para dividir cada nó, o algoritmo calcula um score de ganho baseado nos gradientes e hessianos da função de perda. As árvores crescem até uma profundidade máxima definida e depois passam por poda, removendo divisões que não melhoram a predição. Diferentemente de árvores balanceadas tradicionais, o XGBoost permite crescimento assimétrico, adaptando a estrutura da árvore aos padrões presentes nos dados.

O CatBoost, desenvolvido por Prokhorenkova et al. (2018), implementa uma variação da arquitetura de Gradient Boosting com ênfase no tratamento de variáveis categóricas. Pinheiro (2023) descreve que o CatBoost constrói árvores simétricas, nas quais cada nível da árvore aplica a mesma regra de divisão em todos os nós daquele nível. Essa estrutura simétrica garante que a árvore cresça de forma balanceada, facilitando a implementação eficiente em processadores e reduzindo o risco de *overfitting*. Para variáveis categóricas, o algoritmo utiliza técnicas de codificação ordenada que consideram a ordenação das amostras no conjunto de treinamento, evitando vazamento de informação que poderia comprometer a capacidade de generalização do modelo.

Para avaliar a qualidade desses modelos de aprendizado de máquina em problemas de regressão, quatro métricas são amplamente utilizadas. Segundo Géron (2019), o Erro Médio Absoluto (MAE) calcula a média das diferenças absolutas entre valores reais e previstos, enquanto o Erro Percentual Absoluto Médio (MAPE) expressa o erro médio em termos percentuais, facilitando a interpretação em contextos práticos. O Erro Quadrático Médio (MSE) penaliza erros maiores ao elevá-los ao quadrado, tornando-se sensível a outliers. A Raiz do Erro Quadrático Médio (RMSE) representa a raiz quadrada do MSE, mantendo a escala original dos dados e permitindo análises mais intuitivas. Além das métricas tradicionais, outras abordagens têm sido propostas na literatura para contextos específicos. Hyndman e Koehler (2006) propõem o MASE (Mean Absolute Scaled Error) como métrica independente de escala, especialmente útil para comparar

previsões entre diferentes séries temporais. Chai e Draxler (2014) recomendam MAE quando a distribuição de erros é heterogênea e RMSE para erros com distribuição normal, ressaltando que a escolha da métrica deve considerar as características dos dados. O Coeficiente de Determinação (R^2) mede a proporção da variação explicada pelo modelo, variando entre 0 e 1, sendo que valores mais próximos de 1 indicam melhor ajuste. Dodge (2008) afirma que valores altos de R^2 são preferíveis para previsões, indicando menor variação não explicada pelo modelo.

2.2 PROCESSO ATUAL DA EMPRESA ANALISADA

A empresa analisada neste estudo é uma indústria têxtil localizada na região do Vale do Itajaí que atua em diversas frentes produtivas e comerciais. Suas atividades incluem produção própria de malhas, beneficiamento (tingimento e acabamento), confecção de roupas para diferentes públicos (infantil, juvenil e adulto) e uso de vários canais de venda para que ampliam seu alcance e reduzem a dependência de um único mercado. Os canais de venda abrangem e-commerce, grandes varejistas, representantes comerciais e uma plataforma B2B, demonstrando a abrangência de sua atuação no mercado. Essa diversidade de canais exige processos de previsão de demanda robustos para atender adequadamente a cada segmento.

Atualmente, o processo de previsão de demanda da empresa é baseado em uma análise histórica e na experiência dos gestores, seguindo um fluxo estruturado entre diferentes setores. O fluxo se inicia no setor de Desenvolvimento de Produto, onde a equipe realiza um estudo de mercado para definir o mix de produtos a serem produzidos. Nessa etapa, os produtos são caracterizados conforme seu perfil estratégico, como "convencionais" ou "aposta em tendência", entre outras categorias que auxiliam na compreensão da motivação estratégica por trás da escolha de cada produto. Com o mix definido e as fichas técnicas cadastradas, realiza-se uma simulação dos custos de produção, que serve como base para a definição do preço de venda de cada item.

Com os produtos e preços estabelecidos, o relatório é encaminhado à equipe comercial, que avalia de forma descritiva o potencial de venda de cada item com base em sua experiência de mercado. Em seguida, as informações são direcionadas à equipe de demanda, responsável por uma análise mais quantitativa dos dados. A empresa optou por não detalhar os métodos e ferramentas utilizados nessa etapa por questões estratégicas e de confidencialidade. Esse setor atua em conjunto com o financeiro, ajustando as projeções de acordo com o orçamento do período e o percentual de crescimento esperado para o ano fiscal. Por fim, essas informações consolidadas são apresentadas ao Diretor de Produção, que apresenta o plano aos demais membros da Diretoria e acionistas para aprovação ou ajustes finais. Embora este processo integre diferentes áreas da empresa e incorpore múltiplas perspectivas, ele apresenta limitações ao se basear fortemente na experiência subjetiva da equipe de vendas.

Quanto à segurança e sigilo dos dados, foi acordado que o nome da empresa não será divulgado, assim como qualquer dado sensível relacionado a faturamento ou identificação de clientes. Os resultados apresentados utilizarão apenas dados agregados e métricas comparativas mencionadas anteriormente. Todas as informações confidenciais foram anonimizadas para preservar a privacidade da organização parceira.

2.3 TRABALHOS CORRELATOS

Nesta subseção, são apresentados os estudos com maior correlação ao trabalho proposto. O Quadro 1 apresenta o estudo de Henzel et al. (2022), que propõe o uso combinado de KNN e LMM para previsão semanal de demanda na indústria da moda, partindo de previsões agrupadas por categoria para estimar a demanda por item considerando cor e tamanho. Embora tenha reduzido o MAE em relação a previsão passada pela empresa, a eficácia depende da qualidade da previsão já realizada anteriormente pela empresa e da similaridade entre os produtos dentro de cada categoria.

Quadro 1 – Demand forecasting in the fashion business – an example of customized nearest neighbour and linear mixed model approaches

Referência	Henzel *et al.* (2022)
Objetivos	Propor e avaliar abordagens de Aprendizado de Máquina (AM) para a previsão de demanda de produtos da indústria da moda, especialmente para situações em que não há histórico de vendas suficiente ou onde produtos apresentam alta volatilidade devido às tendências sazonais.
Principais funcionalidades	Realização de previsões semanais para produtos específicos a partir das previsões agregadas (por categoria) passadas pela empresa parceira (*baseline*) do estudo. O método proposto permite detalhar a previsão de demanda por produto, cor e tamanho.
Ferramentas de desenvolvimento	Métodos utilizados: *Naive*, KNN, LMM e uma abordagem combinada (*ensemble*) entre a média KNN e LMM. Base de dados: Registros semanais de vendas de uma rede varejista de moda, com tipo de produto, tamanho, cor principal, semana de venda, estação e presença de feriados na semana. Possui dados de jan/2016 a nov/2021. O tamanho da base não foi informado. Pré-processamento: Uso do Prophet, uma ferramenta de previsão de séries temporais desenvolvida pelo Facebook (atual Meta), para decomposição de tendência/sazonalidade. Linguagens: R e Python

Resultados e conclusões	A abordagem combinada apresentou uma redução do Erro Absoluto Médio (MAE) em 4 categorias: categoria 3 (*baseline* = 6,86, *ensemble* = 6,71), categoria 6 (*baseline* = 11,55, *ensemble* = 10,70), categoria 7 (*baseline* = 4,17, *ensemble* = 4,07) e categoria 2 (*baseline* = 9,68, LMM = 9,29). Na categoria 5, o método *Naive* teve MAE menor (*baseline* = 20.23, *naive* = 17.48), enquanto as categorias 1 e 4 não apresentaram melhora. Os autores concluem que a abordagem de desagregação é viável, mas a precisão depende da qualidade da previsão agregada inicial e da representatividade dos dados históricos.

Fonte: elaborado pelo autor.

O Quadro 2 apresenta o estudo de Lv et al. (2023), que investiga a influência de variáveis climáticas na previsão de vendas usando empilhamento entre Random Forest (RF), Extreme Gradient Boosting (XGB) e Gradient Boosting Descision Tree (GBDT). Os ganhos foram mais expressivos em produtos sazonais como vestidos e jaquetas, limitando a aplicabilidade em produtos básicos.

Quadro 2 – Clothing Sales Forecast Considering Weather Information: An Empirical Study in Brick-and-Mortar Stores by Machine-Learning

Referência	Lv *et al.* (2023)
Objetivos	Investigar se a inclusão de informações climáticas (dados externos) melhora a precisão da previsão de vendas de vestuário em lojas físicas, separando por categorias de roupas sazonais e básicas.
Principais funcionalidades	Realização de previsões diárias de vendas por categoria de produto em lojas físicas, considerando atributos dos produtos, histórico de vendas e variáveis climáticas. O estudo avalia se o clima influencia a precisão da previsão em diferentes categorias (básico e sazonal).
Ferramentas de desenvolvimento	Métodos utilizados: RF, XGB, GBDT e uma estratégia de empilhamento (*stacking*). Na primeira camada do empilhamento, foram utilizados como base os três algoritmos citados anteriormente. Na segunda, as previsões geradas foram utilizadas como entrada para um modelo de regressão bayesiana, que então gerava a previsão final. Base de dados: Registros diários de vendas de uma rede de lojas em Hangzhou, de 2018 a 2019. Possui mais de 100 mil registros, agrupados por categoria (camisetas, vestidos, jaquetas, etc.). Foram combinados com informações meteorológicas da região (temperatura, umidade, vento, precipitação, entre outras). Ao total, a base contou com 23 variáveis distintas. Pré-processamento: Remoção de valores ausentes e anormais. Linguagem: Python
Resultados e conclusões	A inclusão de variáveis climáticas aumentou o erro na previsão de vendas totais (produtos básicos e sazonais). O Erro Quadrático Médio (MSE) do modelo GBDT foi de 443,21 sem dados climáticos e passou para 944,63 com a inclusão dessas variáveis. No entanto, houve redução no MSE para produtos sazonais, como vestidos (−86,03%), casacos de inverno (−80,14%) e camisas (−41,49%). Os autores concluíram que variáveis climáticas podem ser consideradas na previsão de vendas de roupas sazonais, mas não são recomendadas para categorias básicas, com demanda mais estável.

Fonte: elaborado pelo autor.

O Quadro 3 sintetiza o estudo de Lorente-Leyva et al. (2020), que compara métodos clássicos (ARIMA, Holt-Winters e STL) com algoritmos de AM (Multilayer Perceptron, Support Vector Machine, Random Forest e Redes Bayesianas) em uma indústria têxtil no Equador. Os modelos de AM apresentaram maior acurácia, porém o estudo se restringe a séries históricas sem explorar variáveis externas.

Quadro 3 – A Comparison of Machine Learning and Classical Demand Forecasting Methods: A case Study of Ecuadorian Textile Industry

Referência	Lorente-Leyva *et al.* (2020)
Objetivos	Comparar métodos clássicos (ARIMA, STL, *Holt-Winters*) e de AM para previsão de demanda no setor têxtil equatoriano.
Principais funcionalidades	Realização de previsões mensais para produtos específicos da indústria têxtil (uniformes sublimados, camisetas esportivas, camisetas sublimadas e camisetas polo).
Ferramentas de desenvolvimento	Métodos utilizados: ARIMA, *Holt-Winters*, STL *Decomposition*, *Naive*, MLP, SVM, RF e Redes Bayesianas (BNs). Base de dados: Uso de dados históricos de vendas mensais de 2016 a 2019, por *Stock Keeping Unit* (SKU). Os métodos consideraram apenas o tempo como variável independente. O tamanho exato da base não foi informado. Pré-processamento: Não especificado no estudo. Linguagem: R
Resultados e conclusões	O estudo mostrou que o MLP teve o melhor desempenho entre todos os modelos, com MAPE entre cerca de 4 e 9 e RMSE entre 2 e 14. O SVM ficou atrás, com MAPE entre 7 e 11 e RMSE entre 3 e 22. Com base nesses resultados, os autores concluíram que técnicas de Aprendizado de Máquina se ajustam melhor às necessidades de previsão da indústria têxtil, sobretudo quando há mais dados e maior incerteza.

Fonte: elaborado pelo autor.

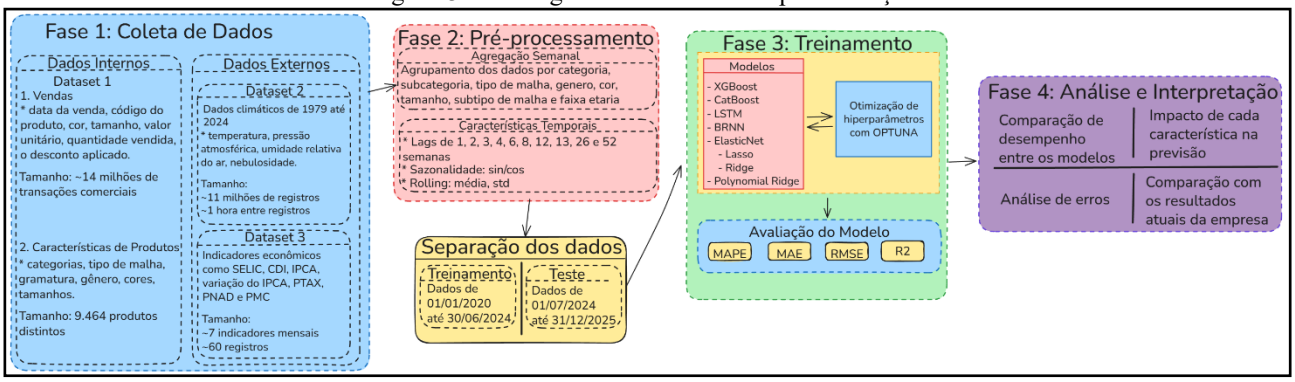
O presente trabalho se diferencia ao propor uma abordagem mais abrangente que integra esses elementos em um único estudo. Foram desenvolvidos e comparados seis modelos distintos (XGBoost, CatBoost, ElasticNet, Polynomial Ridge, LSTM e BRNN), incorporando 122 *features* que incluem *lags* temporais, médias móveis, tendências, sazonalidade cíclica, variáveis climáticas e indicadores macroeconômicos (SELIC, IPCA, câmbio e índice de varejo). A estratégia de

target deslocado permite previsões de demanda com 12 semanas de antecedência, horizonte significativamente superior aos estudos correlatos que focam em previsões semanais ou mensais de curto prazo. Conforme descrito na PROCESSO ATUAL DA EMPRESA ANALISADA, a aplicação foi realizada em contexto de múltiplas frentes de atuação (produção, beneficiamento e comercialização), representando um caso de uso mais complexo que os contextos de varejo ou manufatura isolada explorados nos trabalhos anteriores.

3 DESCRIÇÃO DO MODELO

Este capítulo descreve os procedimentos metodológicos adotados para o desenvolvimento deste trabalho. A Figura 3 apresenta uma visão geral do pipeline de implementação, organizado em quatro fases sequenciais. A primeira fase consiste na Coleta de Dados, integrando registros de vendas da empresa parceira, características dos produtos, indicadores econômicos nacionais e variáveis climáticas. A segunda fase realiza o pré-processamento, transformando os dados para frequência semanal e criando variáveis temporais derivadas (defasagens, médias móveis tendências e sazonalidade), que servem para capturar padrões históricos, tendências e sazonalidade. A terceira fase corresponde ao treinamento, implementando seis modelos com otimização de hiperparâmetros. A quarta fase, análise e interpretação, é onde ocorre a comparação de desempenho entre os modelos e a identificação das características mais relevantes para a previsão de demanda. As seções seguintes detalham cada etapa: a seção 3.1 apresenta a descrição dos dados utilizados, a seção 3.2 detalha o pré-processamento e a criação de variáveis derivadas, e a seção 3.3 descreve a implementação e treinamento dos modelos.

Figura 3 – Fluxograma da Fase de Implementação



Fonte: elaborado pelo autor.

Todos os testes foram executados em um computador equipado com processador Intel Core i7-13620H, 32 GB de RAM, GPU NVIDIA GeForce RTX 4050 e sistema operacional Windows 11. A GPU foi utilizada para acelerar o treinamento dos modelos de aprendizado profundo (LSTM e BRNN) por meio do TensorFlow com suporte a CUDA.

3.1 BASE DE DADOS

Os dados utilizados neste trabalho dividem-se em dados internos, fornecidos pela empresa parceira, e dados externos, obtidos em fontes públicas. O conjunto de dados abrange o período de janeiro de 2020 a dezembro de 2024, totalizando 5 anos de registros históricos. Os dados estão organizados em quatro categorias principais: dados de vendas; características dos produtos; indicadores econômicos; e dados climáticos.

Os dados de vendas foram extraídos do sistema de gestão da empresa parceira. Cada registro representa uma transação comercial e inclui o número da nota fiscal, a data da venda, o código do produto, a cor, o tamanho, o valor unitário, a quantidade vendida e o desconto aplicado. Também são registrados o período de faturamento (em semanas), o CNPJ do cliente, sua localização (cidade e estado), o CNPJ do representante comercial, a região de vendas e a coleção do produto (verão ou inverno). No total, foram registradas aproximadamente 14 milhões de transações distribuídas ao longo de 257 semanas, abrangendo 26 estados brasileiros. A Figura 4 apresenta a distribuição geográfica das vendas da empresa analisada destacando os dez estados com maior volume comercial no período analisado.

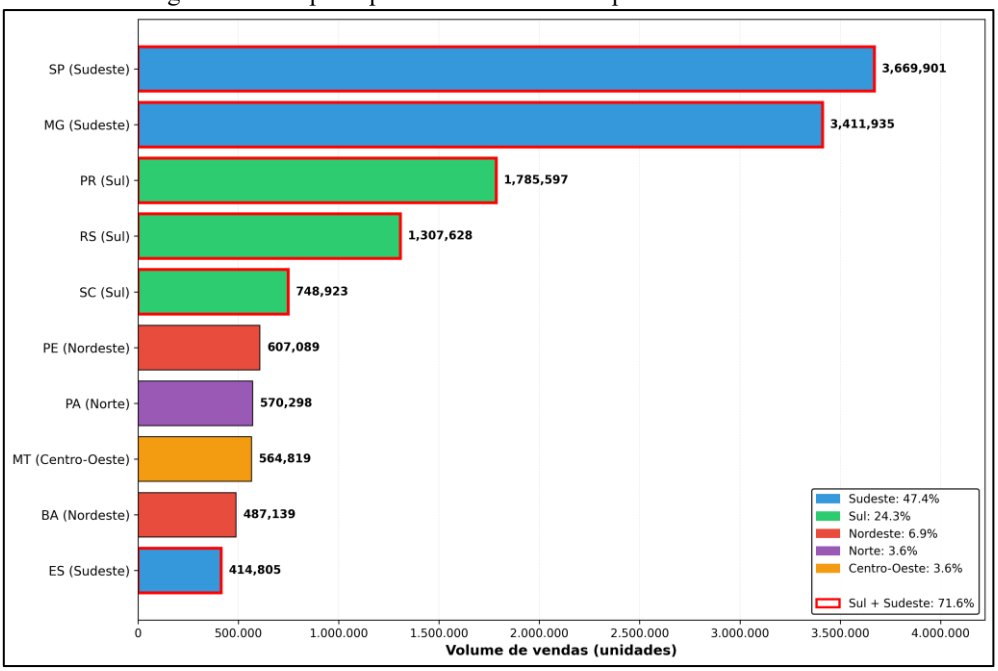
Conforme ilustrado na Figura 4 , há uma concentração significativa de 71,7% nas regiões Sul e Sudeste, refletindo a distribuição econômica e demográfica do país. Os dois principais estados, São Paulo (3.669.901 unidades) e Minas Gerais (3.411.935 unidades), correspondem sozinhos a mais da metade do volume comercializado. A região Sul apresenta participação expressiva, com Paraná (1.785.597), Rio Grande do Sul (1.307.628) e Santa Catarina (748.923) entre os cinco maiores volumes. As regiões Nordeste, Norte e Centro-Oeste possuem representatividade menor, mas contribuem para a diversificação geográfica do negócio, com destaque para Pernambuco (607.089), Pará (570.298) e Mato Grosso (564.819). Essa distribuição geográfica desigual é um fator relevante para a modelagem, pois diferentes regiões podem apresentar padrões de sazonalidade e preferências de produtos distintas, que precisam ser capturados pelas características regionais do modelo.

O conjunto de dados de características dos produtos contém informações técnicas e descritivas de cada produto

comercializado pela empresa, utilizadas como variáveis explicativas nos modelos de aprendizado de máquina. Cada produto é identificado por um código e nome únicos. São classificados em categorias como blusa, camiseta, vestido, calça, bermuda, short, casaco e saia, além de uma subcategoria que identifica se o produto possui ou não insumos importados. As informações técnicas incluem o tipo de malha (tecido ou malha), o subtipo de malha (meia malha, algodão, viscose, favo, moletom, gorgurão, ribana, canelado, entre outros) e a gramatura (peso do tecido em g/m²). Indicadores binários informam se o produto possui estampa ou aplicações. Além disso, são registradas todas as cores e tamanhos disponíveis (PP, P, M, G, GG, EG ou numerações específicas), o gênero (masculino, feminino ou relançamento) e a faixa etária (adulto, juvenil, infantil, primeiros passos ou *plus size*). Essas informações permitem caracterizar os produtos de forma detalhada, capturando aspectos técnicos e de design que podem influenciar a demanda.

Para capturar o impacto de variáveis macroeconômicas na demanda por produtos têxteis, foram coletados indicadores econômicos brasileiros de fontes públicas oficiais, especificamente do Instituto Brasileiro de Geografia e Estatística (IBGE) e do Banco Central do Brasil (BACEN), com periodicidade mensal. Os indicadores selecionados incluem a taxa Selic (taxa básica de juros da economia brasileira), a taxa CDI (Certificado de Depósito Interbancário), o IPCA mensal (Índice Nacional de Preços ao Consumidor Amplo), a taxa de câmbio PTAX (média de venda do dólar americano em relação ao real), a variação mensal do IPCA, a taxa de desocupação da PNAD (Pesquisa Nacional por Amostra de Domicílios Contínua) e o índice PMC (Pesquisa Mensal de Comércio) para o setor varejista. Esses indicadores foram escolhidos por refletirem aspectos relevantes do ambiente econômico que podem afetar o consumo de vestuário, como poder de compra, inflação, desemprego e atividade do varejo.

Figura 4 - Dez principais estados do Brasil por volume de vendas



Fonte: elaborado pelo autor.

Os dados climáticos foram coletados para todos os estados brasileiros a partir da plataforma OpenWeather (openweathermap.org). Foram fornecidas variáveis como data e hora da medição, localização (cidade, latitude e longitude), temperatura (atual, mínima, máxima, sensação térmica e ponto de orvalho), pressão atmosférica (ao nível do mar e do solo), umidade relativa do ar, vento (velocidade, direção e rajadas), precipitação (quantidade de chuva em 1h e 3h), nebulosidade (cobertura de nuvens) e condições climáticas gerais (classificação como céu limpo, chuva, nublado, entre outras). Embora a plataforma disponibilize dados desde 1979, foram utilizados apenas os dados correspondentes ao período de vendas analisado neste trabalho (janeiro de 2020 a dezembro de 2024).

3.2 PRÉ-PROCESSAMENTO E CRIAÇÃO DE VARIÁVEIS TEMPORAIS

O pré-processamento dos dados foi estruturado em três etapas sequenciais. A subseção 3.2.1 descreve o procedimento de carregamento e integração dos dados de diferentes fontes, que são vendas, produtos, clima e indicadores econômicos, em uma base unificada com agregação semanal. A subseção 3.2.2 detalha a criação de variáveis temporais derivadas, incluindo defasagens (valores de vendas passadas), médias móveis, sazonalidade e tendências. Por fim, a subseção 3.2.3 descreve a preparação do *target* deslocado (*shifted target*), utilizado para viabilizar previsões com horizonte de 12 semanas. A base de dados final, após o pré-processamento completo e aplicação dos critérios de exclusão, continha 735 mil registros e 130 características. Cada observação corresponde a uma combinação única de semana e grupo de características de produto. O período analisado abrange de janeiro de 2020 a dezembro de 2024.

3.2.1 Carregamento e Integração dos Dados

A primeira etapa do pré-processamento consistiu no carregamento e transformação de cada fonte de dados separadamente, seguida pela integração em uma base unificada com frequência semanal. O pré-processamento iniciou pelos dados de vendas. Os registros de vendas foram carregados e concatenados para garantir compatibilidade entre os formatos de diferentes períodos. Como as vendas estavam registradas em transações diárias individuais, foi necessário agregá-las para frequência semanal. As características dos produtos foram associadas aos registros de vendas usando o código do produto como chave de relacionamento. Isso permitiu incorporar variáveis descritivas como categoria, subcategoria, tipo de malha, gênero, cor, tamanho, subtipo de malha e faixa etária. Em seguida, os dados foram agrupados por essas características de produto e por semana. Para cada grupo semanal, foram calculadas quatro métricas agregadas. A primeira foi a quantidade total vendida. A segunda foi o preço médio unitário praticado. A terceira foi o número de pedidos únicos realizados e a quarta foi o número de produtos distintos vendidos. Com isso, os dados de vendas, originalmente em transações diárias, foram transformados em séries temporais semanais agregadas por características de produto.

O pré-processamento dos dados climáticos envolveu a agregação temporal dos dados coletados em frequência horária para todos os estados brasileiros. Para tornar esses dados compatíveis com a periodicidade semanal das vendas, foi calculada a média de cada variável meteorológica por estado e por semana. As variáveis meteorológicas incluem temperatura, umidade relativa do ar, pressão atmosférica e velocidade do vento. Para a precipitação, foi calculado o acumulado semanal ao invés da média. Esse processo reduziu a granularidade dos dados de horária para semanal, preservando as informações relevantes sobre as condições climáticas de cada período.

Os indicadores econômicos foram coletados em periodicidade mensal das fontes oficiais do IBGE e BACEN. Esses indicadores incluem Selic, CDI, IPCA, PTAX, variação do IPCA, taxa de desocupação da PNAD e PMC. Para compatibilizar com a frequência semanal, foi aplicada uma reamostragem replicando o valor mensal para todas as semanas daquele mês.

Depois de transformar todos os conjuntos de dados para frequência semanal, foi realizada a integração em uma base unificada. A junção foi feita usando chaves temporais (semana e ano) e espaciais (estado) para combinar os dados de vendas agregados por características de produto com as variáveis climáticas e econômicas correspondentes. Essa integração resultou em uma base de dados multivariada onde cada linha representa uma combinação única de semana, características de produto e localização geográfica.

Durante o processo de integração, foram aplicados critérios para garantir a qualidade dos dados. Apenas as características de produto com menos de 60% de valores ausentes foram mantidas na base final. Isso evitou o uso de variáveis com informações muito incompletas. Para os valores faltantes nas variáveis climáticas, foi utilizada a imputação pela média da variável no respectivo estado e período do ano. Para as variáveis econômicas, eventuais valores faltantes foram preenchidos usando interpolação temporal dos períodos adjacentes. Além disso, séries temporais de produtos com menos de 52 semanas de histórico foram removidas da base. Essas séries não possuíam dados suficientes para criar as variáveis de defasagem de longo prazo necessárias para a modelagem.

3.2.2 Engenharia de Características Temporais

A criação de variáveis temporais derivadas foi projetada para capturar padrões históricos, tendências e sazonalidade presentes nos dados. Uma variável de defasagem representa o valor de uma variável em um momento anterior no tempo. Por exemplo, uma defasagem de 4 semanas nas vendas significa utilizar o valor de vendas de 4 semanas atrás como informação para prever a semana atual. Essas variáveis são fundamentais para modelos de séries temporais porque permitem ao modelo aprender com padrões passados e identificar ciclos de comportamento. O Quadro 4 apresenta o código utilizado para criação das variáveis de defasagem. Para vendas, foram aplicados períodos de 1, 2, 3, 4, 6, 8, 12, 13, 26 e 52 semanas (linhas 1 e 2). Para o preço médio e o número de pedidos foram criadas defasagens de até 12 semanas (linhas 4 e 5). Além disso, foram aplicadas defasagens em algumas variáveis climáticas como velocidade do vento e umidade. Os períodos selecionados permitem capturar diferentes escalas temporais, como curto prazo (1-4 semanas), médio prazo (8-13 semanas), sazonalidade semestral (26 semanas) e sazonalidade anual (52 semanas).

Quadro 4 – Geração de variáveis de defasagem

```
1  for lag in [1, 2, 3, 4, 6, 8, 12, 13, 26, 52]:
2      df_group[f"y_lag_{lag}w"] = df_group["y"].shift(lag)
3      if "preco_medio" in df_group.columns and lag <= 12:
4          df_group[f"preco_lag_{lag}w"] = df_group["preco_medio"].shift(lag)
5          df_group[f"pedidos_lag_{lag}w"] = df_group["num_pedidos"].shift(lag)
```

Fonte: Elaborado pelo autor.

O Quadro 5 apresenta o código para criação de variáveis de janela móvel. Essas variáveis calculam estatísticas (média, desvio padrão, mínimo e máximo) sobre períodos deslizantes de tempo, permitindo ao modelo capturar tendências

e volatilidade recentes nas vendas. Para a criação dessas variáveis, foram utilizadas as janelas de 2, 4, 8, 13, 26 e 52 semanas (linha 1). As funções `mean()`, `std()`, `min()` e `max()` são aplicadas nas linhas 2 a 4, 5 a 7, 8 a 10 e 11 a 13, respectivamente, para calcular a média móvel, o desvio padrão, o valor mínimo e o valor máximo em cada janela. A função `shift(1)` aplicada nas linhas 2, 5, 8 e 11 garante que apenas dados históricos sejam utilizados, evitando vazamento de informação futura.

Quadro 5 – Geração das variáveis de janela móvel

```
1  for window in [2, 4, 8, 13, 26, 52]:
2      df_group[f"y_ma_{window}w"] = df_group["y"].shift(1).rolling(
3          window=window, min_periods=1
4      ).mean()
5      df_group[f"y_std_{window}w"] = df_group["y"].shift(1).rolling(
6          window=window, min_periods=1
7      ).std()
8      df_group[f"y_min_{window}w"] = df_group["y"].shift(1).rolling(
9          window=window, min_periods=1
10     ).min()
11     df_group[f"y_max_{window}w"] = df_group["y"].shift(1).rolling(
12         window=window, min_periods=1
13     ).max()
```

Fonte: Elaborado pelo autor.

O Quadro 6 apresenta parte do código utilizado para criação das características de sazonalidade. A sazonalidade foi capturada por meio de duas abordagens complementares. Primeiro, foram aplicadas transformações trigonométricas (seno e cosseno) às variáveis semana do ano e mês (linhas 1-4), o que permite representar padrões cíclicos de forma contínua, garantindo que períodos como a última semana de dezembro e a primeira de janeiro sejam reconhecidos como próximos. Também foram criadas variáveis binárias indicando as estações do ano (verão na linha 5, inverno na linha 6, primavera na linha 7 e outono na linha 8), permitindo que o modelo identifique comportamentos sazonais específicos de cada período, como maior demanda de roupas leves no verão ou agasalhos no inverno.

Quadro 6 – Geração de variáveis de sazonalidade

```
1  df_group["sin_week"] = np.sin(2 * np.pi * df_group["week"] / 52)
2  df_group["cos_week"] = np.cos(2 * np.pi * df_group["week"] / 52)
3  df_group["sin_month"] = np.sin(2 * np.pi * df_group["month"] / 12)
4  df_group["cos_month"] = np.cos(2 * np.pi * df_group["month"] / 12)
5
6  df_group["is_summer"] = df_group["month"].isin([12, 1, 2]).astype(int)
7  df_group["is_winter"] = df_group["month"].isin([6, 7, 8]).astype(int)
8  df_group["is_spring"] = df_group["month"].isin([9, 10, 11]).astype(int)
9  df_group["is_autumn"] = df_group["month"].isin([3, 4, 5]).astype(int)
```

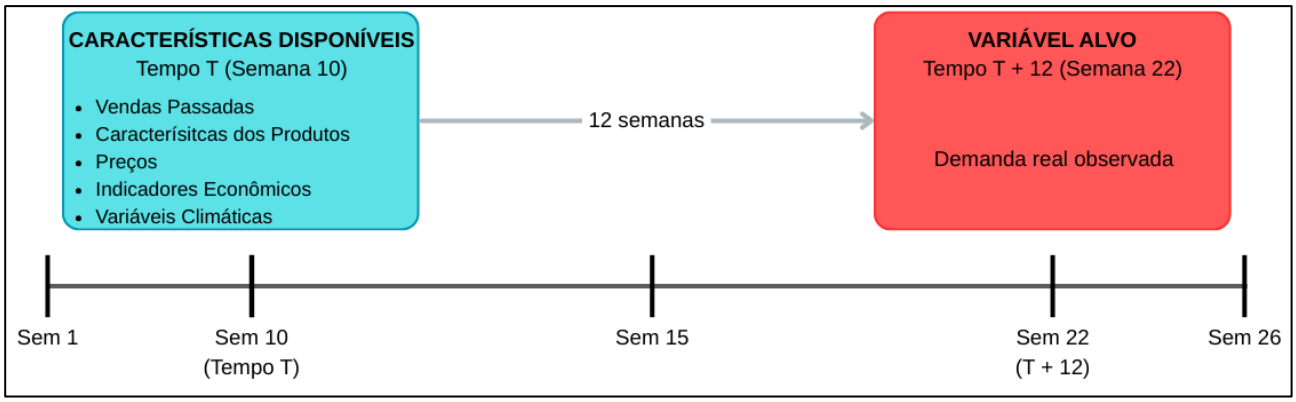
Fonte: Elaborado pelo autor.

A tendência temporal foi modelada criando um contador sequencial de semanas, que é normalizado dividindo cada valor pelo número total de semanas no histórico do produto. Essa operação resulta em uma escala de 0 a 1. Adicionalmente, foi criada uma versão elevada ao quadrado, permitindo que o modelo capture tanto tendências lineares (crescimento constante) quanto não lineares (crescimento acelerado ou desacelerado). Para garantir a integridade temporal dos dados e evitar vazamento de informação, todas as variáveis de defasagem e janela móvel utilizam apenas dados históricos. A função `shift()` aplicada no código desloca os valores no tempo, assegurando que, ao prever a demanda da semana T, o modelo utilize apenas informações disponíveis até a semana T-1. Essa prática é realizada para simular o cenário real de previsão, onde decisões devem ser tomadas com base exclusivamente em informações passadas.

3.2.3 Preparação da Variável Alvo para Previsão

Para possibilitar a previsão de demanda em horizontes futuros, foi implementada uma estratégia de *target* deslocado, também denominada de *shifted target*. A Figura 5 ilustra o conceito central dessa abordagem, mostrando como as características disponíveis no tempo T são utilizadas para prever a demanda no tempo T+12. O modelo é treinado para aprender a relação entre as características dos produtos e variáveis disponíveis no tempo T (caixa azul) e a demanda real observada no tempo T+12 (caixa vermelha). Por exemplo, quando o modelo recebe dados da Semana 10, contendo vendas passadas, preços e indicadores econômicos daquela semana, ele é treinado para prever a demanda que será efetivamente observada na Semana 22, ou seja, 12 semanas no futuro. Essa estratégia permite que o modelo forneça previsões de longo prazo úteis para planejamento de produção, gestão de estoque e alocação de recursos.

Figura 5 - Esquema temporal da variável alvo para previsão



Fonte: elaborado pelo autor.

É importante ressaltar que apenas variáveis que estarão efetivamente disponíveis no momento da previsão podem ser utilizadas como características. Por exemplo, ao prever a semana T+12, não é possível utilizar dados climáticos da própria semana nem indicadores econômicos não publicados. A divisão entre treinamento e teste foi realizada de forma temporal, utilizando dados até 30 de junho de 2024 para treinamento e o período subsequente para a validação. Dessa forma, é possível garantir a preservação da ordem cronológica dos eventos e evitar qualquer forma de vazamento de dados.

3.3 IMPLEMENTAÇÃO E TREINAMENTO DOS MODELOS

Neste trabalho foram testados seis modelos de aprendizado: CatBoost, XGBoost, ElasticNet, Polynomial Ridge, LSTM e BiLSTM. Todos os modelos foram implementados em linguagem Python e otimizados utilizando a biblioteca Optuna para busca de hiperparâmetros. A escolha dos modelos foi baseada na revisão bibliográfica, considerando tanto métodos tradicionais de regressão quanto abordagens modernas de Gradient Boosting e Aprendizado Profundo. O processo de implementação seguiu uma metodologia sistemática, iniciando com a preparação dos dados conforme descrito na Seção 3.2, seguida pela otimização de hiperparâmetros, treinamento final e validação. Cada modelo foi avaliado tanto em termos de métricas de precisão pontual (MAE, RMSE, MAPE, R^2) quanto de acurácia de volume agregado, métrica crítica para o planejamento de produção na indústria têxtil. Todos os modelos foram estruturados de forma consistente, utilizando o mesmo conjunto de características, a mesma divisão temporal de dados e as mesmas métricas de avaliação.

O Quadro 7 apresenta a estrutura do processo de otimização de hiperparâmetros implementado utilizando a biblioteca Optuna. Para garantir evitar *overfitting*, foi adotada uma estratégia de validação cruzada temporal com classe `TimeSeriesSplit` com 3 folds (linha 1), respeitando a ordem cronológica dos dados. A classe é utilizada na linha 23, através da variável tscv. Segundo Hyndman e Athanasopoulos (2018), essa abordagem é fundamental em séries temporais, pois garante que o modelo seja treinado apenas com dados passados e validado com dados futuros, simulando o cenário real de produção.

Cada modelo foi otimizado por 100 execuções, conforme especificado na linha 40, por meio do parâmetro `n_trials=100`. A métrica de otimização utilizada foi o Erro Médio Absoluto (MAE), calculada sobre o conjunto de validação na linha 34. O Optuna utiliza a classe `TPESampler` (linha 6) para sugerir configurações promissoras de hiperparâmetros com base nos resultados de execuções anteriores. Além disso, foi implementada a interrupção antecipada através da classe `MediaPruner` (linha 7), que interrompe execuções com desempenho inferior antecipadamente. A métrica de otimização do MAE é calculada como a média dos erros nas 3 divisões de validação, conforme implementado nas linhas 23 a 35. A execução da otimização é iniciada na linha 42, por meio do método `study.optimize()`.

Quadro 7 – Otimização de hiperparâmetros com Optuna

```
1  tscv = TimeSeriesSplit(n_splits=3)
2
3  # Configuração do estudo Optuna
4  study = optuna.create_study(
5      direction='minimize',
6      sampler=TPESampler(seed=RANDOM_STATE),
7      pruner=MedianPruner(n_startup_trials=5, n_warmup_steps=1)
8  )
9
10 # Definição da função objetivo
11 def objective(trial):
12     params = {
13         'iterations': trial.suggest_int('iterations', 100, 1000, step=50),
14         'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3, log=True),
15         'depth': trial.suggest_int('depth', 4, 10),
16         'l2_leaf_reg': trial.suggest_float('l2_leaf_reg', 1, 10),
17         'random_state': RANDOM_STATE,
18         'verbose': False,
19         'loss_function': 'MAE'
20     }
21
22     scores = []
23     for fold_idx, (train_idx, val_idx) in enumerate(tscv.split(X_train)):
24         X_fold_train = X_train.iloc[train_idx]
25         y_fold_train = y_train.iloc[train_idx]
26         X_fold_val = X_train.iloc[val_idx]
27         y_fold_val = y_train.iloc[val_idx]
28
29         model = CatBoostRegressor(**params)
30         model.fit(X_fold_train, y_fold_train, eval_set=(X_fold_val, y_fold_val),
31 verbose=False)
32
33         y_fold_pred = model.predict(X_fold_val)
34         fold_mae = mean_absolute_error(y_fold_val, y_fold_pred)
35         scores.append(fold_mae)
36
37     return np.mean(scores)
38
39 # Execução da otimização
40 study.optimize(objective, n_trials=100)
```

Fonte: Elaborado pelo autor.

3.3.1 Modelos de Regressão Linear

Dois modelos de regressão linear foram implementados usando a biblioteca `scikit-learn` versão 1.3+. Ambos utilizam o módulo `StandardScaler` para normalização prévia das características, aplicado separadamente ao conjunto de treino e teste para evitar vazamento de dados. Esta abordagem garante que informações do conjunto de teste não influenciem o processo de normalização. Os Quadro 8 e Quadro 9 no APÊNDICE A apresentam o código fonte dos modelos Polynomial Ridge e ElasticNet.

O modelo Polynomial Ridge foi implementado através de *pipeline* que combina `PolynomialFeatures` e Ridge do `scikit-learn`, conforme apresentado no Quadro 8 do APÊNDICE A (linhas 11-20). A implementação utilizou o parâmetro `include_bias=False` (linha 17) para evitar colinearidade com o termo de intercepto do Ridge. O espaço de busca para o grau polinomial foi limitado ao intervalo [1, 3] (linha 7) baseando-se em Géron (2019), que recomenda graus baixos para conjuntos com muitas *features*. O parâmetro *alpha* foi explorado no intervalo [0.001, 100] em escala logarítmica (linha 9), permitindo testar desde regularização mínima até forte. O parâmetro `interaction_only` foi testado como booleano (linha 8), determinando se o modelo deve incluir apenas termos de interação entre características originais.

O modelo ElasticNet foi implementado usando a classe `ElasticNet` do `scikit-learn` com parâmetro `max_iter=10000` (Quadro 9 do APÊNDICE A , linha 14) para garantir convergência. O espaço de busca para *alpha* foi definido no intervalo [0.0001, 10.0] em escala logarítmica (linha 12). O parâmetro `l1_ratio` foi explorado no intervalo [0.0, 1.0] (linha 13), controlando o equilíbrio entre as regularizações L1 e L2, onde valores próximos a 0 favorecem o comportamento do modelo Ridge e valores próximos a 1 favorecem o comportamento do modelo Lasso. O parâmetro `selection` foi testado entre as opções `cyclic` e `Random` (linha 16).

3.3.2 Modelos de Gradient Boosting

Dois algoritmos de Gradient Boosting foram implementados para capturar relações não-lineares nos dados de demanda. Os Quadro 10 e Quadro 11 no APÊNDICE B apresenta o código fonte dos modelos XGBoost e CatBoost. Ambos foram configurados com `random_state=42` para reprodutibilidade dos experimentos. O parâmetro `verbose=0` foi utilizado durante a otimização para reduzir a quantidade de logs gerados.

O modelo XGBoost foi implementado usando a biblioteca `xgboost` versão 2.0+ através da classe `XGBRegressor`, conforme apresentado no Quadro 10 do APÊNDICE BAPÊNDICE BAPÊNDICE B. A implementação utilizou `objective='reg:squarederror'` (linha 6) como função de perda e `eval_metric='mae'` (linha 7) como métrica de avaliação alinhada ao critério de otimização do `Optuna`. O parâmetro `early_stopping_rounds=50` (linha 33) foi configurado para interromper o treinamento quando não houver melhoria no conjunto de validação. O espaço de busca incluiu `n_estimators` no intervalo [100, 1000] com incrementos de 50 (linha 10), `max_depth` em [3, 10] (linha 11), e `learning_rate` em [0.01, 0.3] em escala logarítmica (linha 12). Os parâmetros `subsample` e `colsample_bytree` foram explorados em [0.6, 1.0] para controlar aleatoriedade no treinamento (linhas 13 e 14). As regularizações `reg_alpha` (`L1`) e `reg_lambda` (`L2`) foram testadas no intervalo [0, 1] (linhas 17e 18).

O modelo CatBoost foi implementado usando a biblioteca `catboost` versão 1.2+ através da classe `CatBoostRegressor`, conforme apresentado no Quadro 11 do APÊNDICE B. A implementação utilizou `loss_function='MAE'` (linha 21) alinhada à métrica de otimização. Uma característica específica do CatBoost é o tratamento nativo de variáveis categóricas através da classe `Pool` (linhas 26-35), que recebe o parâmetro `cat_features` para identificar informações categóricas. O espaço de busca incluiu `iterations` em [100, 1000] com incrementos de 50 (linha 11), `learning_rate` em [0.01, 0.3] em escala logarítmica (linha 12), e `dept` em [4, 10] (linha 13). Parâmetros específicos do CatBoost incluem `l2_leaf_reg` em [1, 10] (linha 14), `min_data_in_leaf` em [1, 100] (linha 15), `random_strength` em [0, 10] (linha 16), `bagging_temperature` em [0, 1] (linha 17), e `border_count` em [32, 255] (linha 18). Durante o treinamento final (linhas 42-61), o parâmetro `early_stopping_rounds=50` (linha 47) foi configurado para detecção automática de *overfitting*, utilizando os objetos `Pool` para treino e validação com as variáveis categóricas devidamente identificadas.

3.3.3 Modelos de Redes Neurais Recorrentes

Dois modelos de Redes Neurais Recorrentes foram implementados usando `TensorFlow` 2.15+ com `backend Keras`. A GPU NVIDIA GeForce RTX 4050 foi utilizada através de CUDA para acelerar o processo de treinamento. A configuração de GPU permitiu reduzir o tempo de otimização de hiperparâmetros das RNAs. Os dados foram preparados em duas etapas para adequação ao formato requerido pelas RNAs, conforme apresentando no Quadro 12 do APÊNDICE C.

A arquitetura é construída de forma condicional baseada no hiperparâmetro `use_second_lstm` (linha 24), que determina se o modelo terá uma ou duas camadas. Quando duas camadas são utilizadas, a primeira camada recebe parâmetro `return_sequences=True` (linha 32) para passar sequências completas à segunda camada, enquanto a segunda camada utiliza `return_sequences=False` implicitamente para retornar apenas o último estado oculto. Camadas *Dropout* são inseridas após cada camada LSTM com taxa variável para regularização (linhas 40 e 46). Opcionalmente, camadas `BatchNormalization` (linhas 39 e 45) podem ser incluídas baseadas no hiperparâmetro `use_batch_norm` (linha 23). O espaço de busca incluiu `lstm_units_1` em [32, 256] com incrementos de 32 (linha 21), `dropout_rate` em [0.1, 0.5] (linha 22), `learning_rate` em [0.0001, 0.01] em escala logarítmica para (linha 25), e `batch_size` em [32, 64, 128, 256] (linha 26). O modelo é compilado com otimizador `Adam` incluindo `clipnorm=1.0` (linha 51) para estabilizar o treinamento.

O modelo BiLSTM foi implementado usando *wrapper* Bidirectional do `Keras` aplicado às camadas LSTM, conforme apresentado no Quadro 13 do APÊNDICE C. A implementação segue o mesmo padrão do LSTM, com camadas bidirecionais substituindo as unidirecionais através do *wrapper* Bidirecional (linhas 15 e 19). A arquitetura também é condicional, baseada no hiperparâmetro `use_second_lstm` (linha 8), podendo ter uma ou duas camadas BiLSTM. O espaço de busca é idêntico ao LSTM, exceto pelos nomes dos parâmetros (`brnn_units_1` na linha 5 e implicitamente `brnn_units_2` para a segunda camada). O *wrapper* Bidirectional concatena automaticamente as saídas das direções direta e reversa, dobrando a dimensionalidade da saída de cada camada. Os *callbacks* de treinamento seguem a mesma configuração do modelo LSTM.

3.3.4 Estratégia de Ensemble

Neste trabalho, foram implementadas duas estratégias de ensemble por meio de funções Python para combinação de previsões, conforme apresentado no Quadro 14 do APÊNDICE D. A implementação utiliza a função `ensemble_weighted_average` (linhas 4 a 10) para calcular a média ponderada de previsões, e

`ensemble_calibrated` (linhas 12 a 21) para aplicar fatores de calibração antes da combinação. As previsões dos modelos CatBoost, XGBoost e ElasticNet são combinadas utilizando pesos definidos no dicionário apresentado nas linhas 24 a 40.

A primeira configuração implementa média ponderada por meio da função `ensemble_weighted_average` (linha 44), que combina as previsões com pesos de 50% para CatBoost (linha 27), 30% para XGBoost (linha 32) e 20% para ElasticNet (linha 37). Os pesos foram definidos empiricamente com base em análise de MAE de validação e tempo de inferência.

A segunda configuração aplica o fator de calibração antes da combinação ponderada. O fator é calculado como a razão entre a soma dos valores reais e a soma das previsões no conjunto de treino. Neste caso, o modelo previu 21.185 unidades mas o valor real foi 26.401, resultando no `calibration_factor` de 1.246 (26.401 / 21.185) para o CatBoost (linha 28). A função `ensemble_calibrated` (linhas 12-21) multiplica as previsões de cada modelo por seus respectivos fatores de calibração (linhas 47 e 48), e em seguida a função `ensemble_weighted_average` é aplicada às previsões calibradas (linha 49). Esta flexibilidade permite adaptar o *ensemble* conforme o contexto de uso, seja para previsões pontuais ou volumes agregados.

4 RESULTADOS

Esta seção discute os resultados obtidos pelos modelos de aprendizado de máquina desenvolvidos para previsão de demanda na indústria têxtil. A subseção 4.1 descreve os hiperparâmetros otimizados pelo framework Optuna para cada modelo implementado. A subseção 4.2 apresenta a análise comparativa entre todos os modelos implementados. A subseção 4.3 detalha os resultados da estratégia de ensemble. A subseção 4.4 analisa a importância das características e o impacto das variáveis externas.

4.1 OTIMIZAÇÃO DE HIPERPARÂMETROS

A otimização de hiperparâmetros foi realizada utilizando o framework `Optuna`, conforme descrito na Seção 3.3.5. O processo envolveu 100 execuções para cada modelo, explorando sistematicamente os espaços de busca definidos e selecionando as configurações que minimizassem o erro médio absoluto no conjunto de validação. A Figura 6 do APÊNDICE E apresenta os intervalos de busca explorados pelo `Optuna` e os valores finais selecionados para cada hiperparâmetro dos seis modelos implementados. A visualização permite compreender o posicionamento das configurações ótimas dentro do espaço de busca, revelando padrões de otimização específicos para cada algoritmo.

Para os modelos de gradient boosting, o `Optuna` selecionou configurações distintas. O XGBoost convergiu para configuração conservadora. O `learning_rate` de 0.0101 permite aprendizado gradual, enquanto o `max_depth` de 3 níveis limita a complexidade das árvores para evitar *overfitting*. O `n_estimators` de 150 árvores busca equilibrar poder preditivo e custo computacional. Os parâmetros de regularização foram `gamma de 1.4`, `reg_alpha de 0.24` e `reg_lambda de 0.92` penalizam complexidade desnecessária. Os parâmetros `subsample de 0.79` e `colsample_bytree de 0.82` controlam amostragem de observações e características, introduzindo diversidade entre as árvores e reduzindo *overfitting*. O CatBoost utilizou 850 iterações com `learning_rate` de 0.103 e `depth` de 5 níveis, construindo árvores levemente mais profundas que o XGBoost. O `min_data_in_leaf` de 89 exige uma quantidade mínima de amostras por folha, prevenindo divisões excessivamente específicas. Os parâmetros l2_leaf_reg de 5.88, `random_strength` de 2.13 e `bagging_temperature` de 0.19 proporcionam regularização e estocasticidade controlada.

Os modelos lineares apresentaram configurações mais simples. O ElasticNet convergiu para `alpha` de 0.0152 e `l1_ratio` de 0.18, indicando regularização leve com predomínio de Ridge sobre Lasso, utilizando seleção aleatória de características. O Polynomial Ridge utilizou `degree` de 1, equivalente a modelo linear simples, com `alpha` de 91.24 indicando forte regularização. As configurações sugerem que transformações polinomiais não trouxeram benefícios e que regularização forte é necessária para estabilizar coeficientes em dados de alta dimensionalidade.

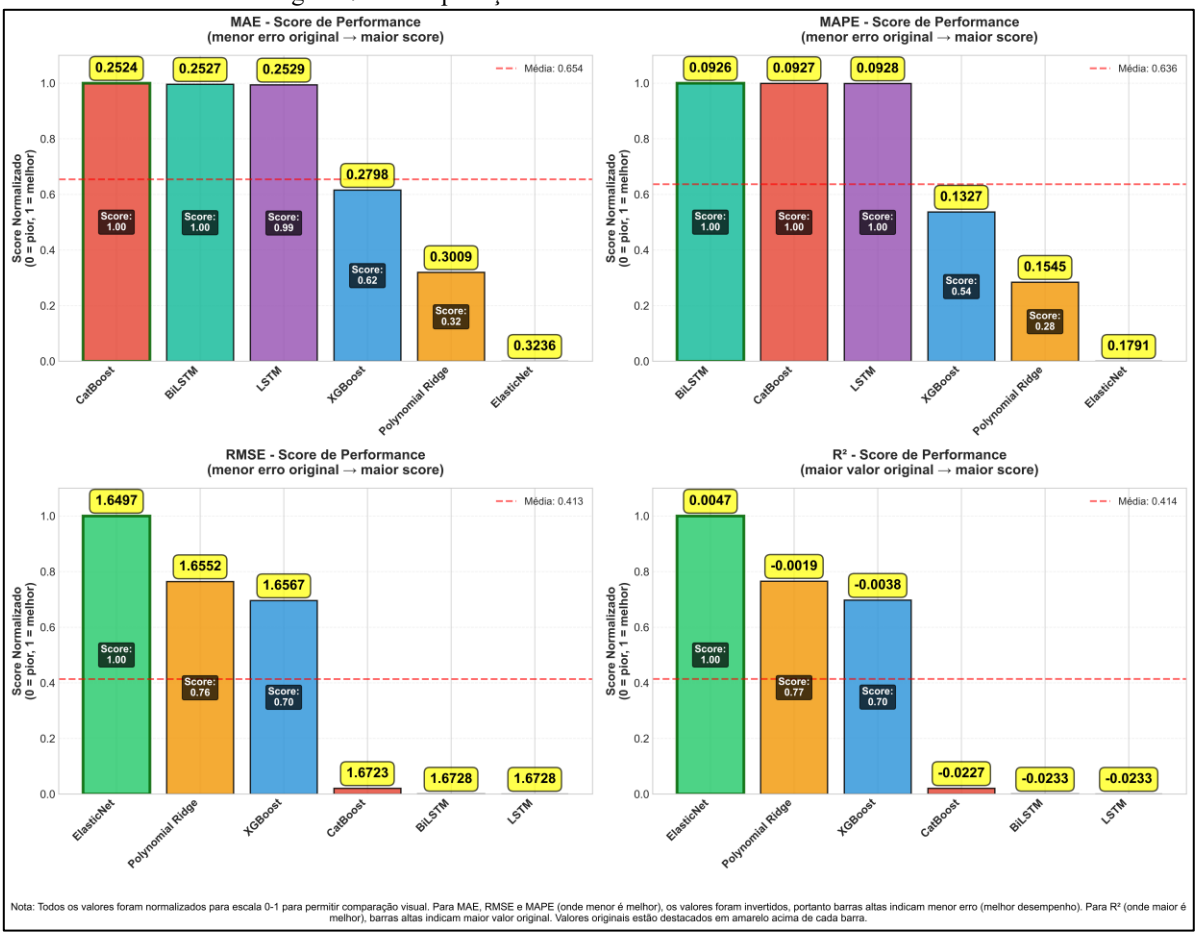
As redes neurais recorrentes convergiram para arquiteturas de complexidade moderada. O LSTM selecionou duas camadas (192 `lstm_units_1`, 64 `lstm_units_2`) com `dropout_rate` de 0.38 e `learning_rate` conservador de 0.0017. O BiLSTM também utilizou duas camadas (224 `brnn_units_1`, 32 `brnn_units_2`) com `dropout_rate` de 0.35 e `learning_rate` superior de 0.0057. Ambos utilizaram `batch_size` reduzidos (64 e 32) sem `batch normalization` (`use_batch_norm=False`), indicando que o pré-processamento foi suficiente para estabilizar treinamento. O gargalo mais acentuado do BiLSTM (224 para 32) pode funcionar como regularização implícita.

4.2 ANÁLISE COMPARATIVA DOS MODELOS

A Figura 7 apresenta as métricas de desempenho dos seis modelos individuais treinados, ordenados em ordem decrescente de desempenho. Para permitir a comparação visual direta entre as métricas de diferentes escalas, os valores foram normalizados na escala 0-1. Para métricas onde menor valor indica melhor desempenho, os valores foram

invertidos, resultando em barras mais altas para modelos com melhor performance. Os valores originais de cada métrica estão destacados acima das barras. Todos os modelos foram avaliados no mesmo conjunto de teste, garantindo comparabilidade direta dos resultados. As métricas foram calculadas considerando apenas previsões não-negativas, pois valores negativos de demanda são fisicamente impossíveis e foram corrigidos para zero durante o pós-processamento. Os resultados revelam dois padrões distintos de desempenho. Modelos mais complexos alcançaram melhor precisão pontual, enquanto modelos mais simples apresentaram melhor acurácia de volume total de vendas. Essa dualidade indica a necessidade de selecionar o modelo conforme o objetivo da previsão.

Figura 7 – Comparação de métricas dos modelos individuais



Fonte: elaborado pelo autor.

O CatBoost alcançou o melhor desempenho em precisão pontual, com MAE de 0.2524 unidades e MAPE de 9.27%. Este resultado indica erro médio de apenas um quarto de unidade por previsão, valor notável considerando a variabilidade da demanda têxtil. Os modelos de aprendizado profundo BiLSTM e LSTM apresentaram desempenho praticamente idêntico, com MAE de 0.2527 e 0.2528 unidades, respectivamente. A diferença mínima entre esses modelos e o CatBoost sugere que as variáveis temporais derivadas, criadas anteriormente, já capturam adequadamente os padrões sequenciais, tornando desnecessária a complexidade adicional das arquiteturas recorrentes. Entretanto, esses três modelos subestimaram o volume total de vendas em aproximadamente 20%.

O XGBoost apresentou desempenho equilibrado entre precisão pontual e volume total de vendas. Com MAE de 0.2798 unidades, o modelo apresentou erro 10.8% superior ao CatBoost, mas subestimou o volume total de vendas em apenas 8.0%, significativamente melhor que a subestimação de 19.8% do CatBoost. Sua configuração conservadora, com learning rate de 0.0101 e profundidade máxima de 3 níveis, resultou em melhor capacidade de generalização.

Os modelos lineares apresentaram o padrão inverso aos modelos complexos. O ElasticNet registrou MAE de 0.3236 unidades e MAPE de 17.91%, valores substancialmente superiores aos modelos de Gradient Boosting. Contudo, o erro de volume foi de apenas -3.0%, o melhor resultado entre todos os modelos individuais. O erro de volume calcula a diferença percentual entre o volume total previsto e o volume total real no período de teste. O Polynomial Ridge apresentou comportamento similar, com MAE de 0.3009 unidades e erro de volume de -3.8%. Essa inversão demonstra que modelos mais simples tendem a preservar melhor as proporções agregadas, mesmo com maior erro nas previsões individuais.

A métrica RMSE, medida na mesma unidade da variável alvo (quantidade de peças), apresentou valores entre 1.6497 e 1.6723 unidades para todos os modelos. O ElasticNet registrou o menor RMSE com 1.6497 unidades, enquanto o CatBoost apresentou 1.6723 unidades, diferença de 0.0226 que não representa distinção relevante entre os modelos. O coeficiente R^2 apresentou valores negativos ou próximos de zero para todos os modelos, com exceção do ElasticNet que alcançou 0.0047. Esses valores indicam que os modelos não conseguiram explicar a variabilidade da demanda. O R^2 negativo revela que as previsões foram menos acuradas que simplesmente utilizar a média histórica como previsão, sugerindo que as variáveis climáticas, econômicas, sazonalidade e tendência possivelmente tiveram contribuição limitada para o modelo. Esse comportamento indica que a demanda em horizonte de 12 semanas pode ser predominantemente estocástica, com fatores não capturados pelas variáveis disponíveis exercendo maior influência sobre as vendas.

4.3 ANÁLISE DA ESTRATÉGIA DE ENSEMBLE

Neste trabalho foram implementadas duas estratégias de *ensemble*. Considerando que os modelos individuais apresentaram contrapartida entre precisão pontual e acurácia na previsão do volume real de vendas, buscou-se avaliar se combinações de modelos poderiam equilibrar essas métricas. A primeira estratégia denominada de Ensemble Ponderado Simples, atribui pesos diferentes para CatBoost, XGBoost e ElasticNet e combina suas previsões por média ponderada. A segunda estratégia, de Ensemble Calibrado, aplica um fator de correção às previsões antes de realizar a combinação ponderada. A Tabela 1 apresenta a comparação entre Catboost Individual, o Ensemble Ponderado e o Ensemble Calibrado.

Tabela 1 - Comparação das estratégias de ensemble

Estratégia	MAE	RMSE	R^2	MAPE	Erro Volume	Tempo Inferência
CatBoost Individual	0.2524	1.6723	-0.0227	9.27%	-18.8%	~100ms
Ensemble Ponderado Simples	0.2625	1.6599	-0.0076	11.11%	-12.9%	~185ms
Ensemble Calibrado	0.3199	1.6526	0.0013	17.48%	-3.0%	~185ms

Fonte: elaborado pelo autor.

O CatBoost Individual serve como referência de comparação, apresentando MAE de 0.2524 unidades e MAPE de 9.27%. O tempo de inferência de aproximadamente 100ms torna esta estratégia adequada para aplicações em tempo real. Entretanto o principal ponto fraco é a subestimação de -18.8% no volume total de vendas, o que representa uma limitação para o planejamento total de produção.

Na primeira estratégia (Ensemble Ponderado Simples) o MAE aumentou para 0.2625 unidades, mas o erro de volume de vendas reduziu para -12.9%, representando uma redução de 5.9 pontos percentuais em relação ao CatBoost. O R^2 melhorou de -0.0227 para -0.0076, se aproximando de zero e indicando maior capacidade de explicar a variância em relação à média simples. O tempo de inferência aumentou para aproximadamente 185ms devido à necessidade de executar três modelos.

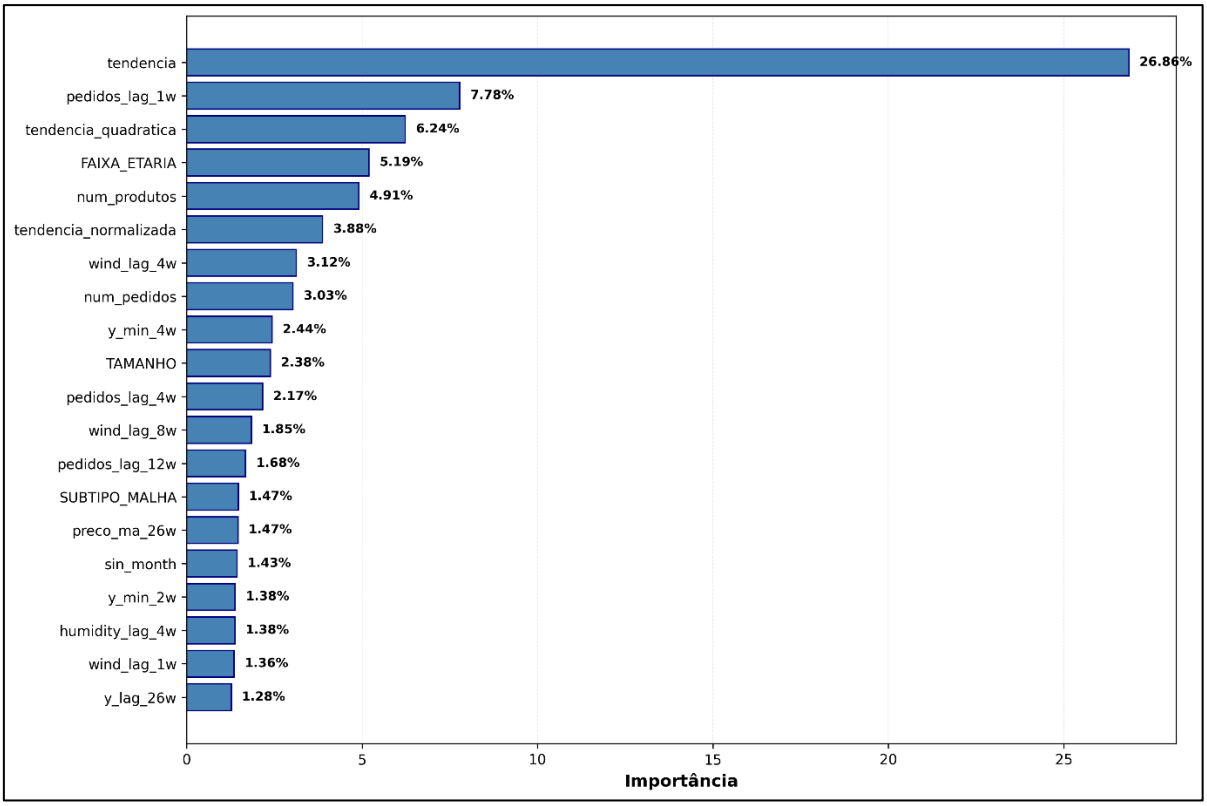
Na segunda estratégia (Ensemble Calibrado), o erro de volume reduziu para -3.0%, redução de 15.8 pontos percentuais em relação ao CatBoost Individual. O R^2 alcançou 0.0013, único valor positivo observado entre todas as estratégias. A contrapartida dessa abordagem é o aumento do MAE para 0.3199 unidades e do MAPE para -17.48%, refletindo maior erro nas previsões individuais. Cada previsão individual apresenta desvio maior da demanda real, mas a soma total das previsões aproxima-se melhor do volume real de vendas. O RMSE variou de 1.6599 para 1.6526 unidades, diferença de 0.0073 que não representa mudança relevante na distribuição de erros. O tempo de inferência permanece em aproximadamente 185ms.

Os resultados evidenciam que a escolha entre precisão pontual e acurácia agregada depende diretamente do objetivo do negócio. Para aplicações que requerem previsões individuais precisas por produto e semana, o CatBoost Individual demonstra superioridade. Por outro lado, para planejamento agregado de produção, onde o volume total importa mais que previsões individuais, o Ensemble Calibrado oferece um erro de volume menor.

4.4 IMPORTÂNCIA DAS CARACTERÍSTICAS PREDITIVAS

A análise da importância das características foi realizada utilizando o modelo CatBoost, selecionado por apresentar o menor MAE entre todos os modelos avaliados. A Figura 8 apresenta as vinte principais características preditivas identificadas pelo modelo CatBoost, ordenadas por seu ganho de informação médio durante a construção das árvores de decisão.

Figura 8 – Vinte principais características preditivas do modelo CatBoost



Fonte: elaborado pelo autor.

A análise revela que características temporais dominam o modelo, representando cerca de 65% da importância total acumulada. A variável tendencia apresenta a maior importância individual (26.86%), evidenciando que o comportamento de crescimento ou redução da demanda ao longo do tempo constitui o principal preditor do modelo. Este resultado indica que padrões de trajetória histórica são mais informativos que flutuações pontuais de curto prazo. As defasagens de pedidos e vendas também se mostram relevantes, com pedidos_lag_1w contribuindo 7.78% e outras defasagens distribuídas entre as demais posições.

As características de produto somam 10.9% da importância total. A variável FAIXA_ETARIA com 5.19% aparece como a variável individual mais relevante, indicando que diferentes públicos apresentam padrões distintos de demanda. O TAMANHO contribui com 2.38%. As demais características de produto apresentam importância baixa, sugerindo que a segmentação por faixa etária e tamanho captura a maior parte da variabilidade explicada pelos atributos do produto. As variáveis climáticas defasadas acumulam 12.1% da importância, com destaque para wind_lag_4w (3.12%) e defasagens de umidade e temperatura em janelas de 4 a 8 semanas. A predominância de defasagens de médio prazo sugere que condições climáticas influenciam decisões de compra com antecedência, possivelmente relacionadas ao planejamento de coleções sazonais pelos clientes varejistas. Esse resultado justifica a inclusão de dados climáticos no modelo.

Em contraste, variáveis econômicas mensais apresentaram importância baixa, praticamente não contribuindo para o modelo. Das 130 características disponíveis, 82 apresentam importância maior que zero, enquanto 48 não contribuem para o modelo. Entre as variáveis com importância nula estão defasagens muito recentes (y_lag_1w, y_lag_2w) e todas as variáveis econômicas defasadas.

5 CONCLUSÕES

Este trabalho teve como objetivo disponibilizar um modelo de aprendizado de máquina para aprimorar a previsão de demanda na indústria têxtil. Para isso, foram implementados e comparados seis modelos CatBoost, XGBoost, ElasticNet, Polynomial Ridge, LSTM e BiLSTM e duas estratégias de ensemble, aplicados a dados reais de uma empresa têxtil do Vale do Itajaí. Os objetivos específicos foram atendidos. O primeiro avaliou o desempenho de seis modelos de aprendizado de máquina utilizando métricas MAE, MAPE, RMSE e R^2. O segundo foi analisado o impacto de variáveis externas como indicadores econômicos e sazonalidade na acurácia das previsões. Por fim, o terceiro objetivo foi atendido ao validar a eficácia dos modelos com dados reais de indústria têxtil do Vale do Itajaí, comparando os resultados obtidos com o processo de previsão atual da empresa.

O CatBoost apresentou o melhor desempenho em precisão pontual, com MAE de 0.2524 e MAPE de 9.27%, validado com dados do segundo semestre de 2024. A análise de importância revelou que características temporais representam 65% da importância total. Essas características temporais incluem a variável de tendência (que modela crescimento ou redução da demanda ao longo do tempo), defasagens de vendas e pedidos, e componentes sazonais (semana do ano, mês, trimestre). Em contraste, os indicadores econômicos mensais apresentaram importância nula devido à baixa variabilidade em horizontes semanais. Uma contribuição do trabalho foi a identificação de contrapartida entre precisão pontual e erro de volume total de vendas. Modelos de Gradient Boosting apresentam menor MAE individual (0.2524 unidades), mas subestimam o volume total em -19.8%, enquanto a estratégia de ensemble calibrado desenvolvida reduziu o erro de volume para -3.0%.

A validação prática do modelo demonstrou uma redução de erro em relação ao processo atual da empresa parceira. O Ensemble Calibrado subestimou o volume agregado em apenas -3.0%, enquanto o processo atual da empresa superestima em 12% a 15%, produzindo além da demanda real. Este resultado evidencia a viabilidade dos modelos desenvolvidos para o planejamento de produção e gestão do estoque, oferecendo melhorias sobre os métodos utilizados atualmente pela empresa. O trabalho diferencia-se dos trabalhos correlatos pela integração de 130 características derivadas, incluindo variáveis climáticas e indicadores macroeconômicos, e pela aplicação no contexto da indústria têxtil brasileira do Vale do Itajaí, Santa Catarina. A estratégia de Ensemble Calibrado representa uma contribuição metodológica ao demonstrar que a calibração de previsões pode contribuir para a melhora na previsão total, mas ao mesmo tempo, pode impactar negativamente a previsão individual.

Para aplicações práticas na indústria têxtil, recomenda-se a seleção do modelo conforme o objetivo de negócio. Para sistemas de reposição automática de estoque ou previsão por produto individual, o CatBoost Individual apresenta o melhor desempenho com MAE de 0.2524 unidades. Para planejamento agregado de produção, gestão de capacidade fabril ou previsão de receita, recomenda-se o Ensemble Calibrado que reduz o erro de volume total para -3.0%. Para casos que requerem equilíbrio entre ambas as métricas, o Ensemble Ponderado Simples apresenta-se como alternativa intermediária. Quanto às variáveis preditivas, recomenda-se priorizar características temporais (variável de tendência, defasagens de vendas e pedidos, componentes sazonais), características de produto (faixa etária, tamanho), e variáveis climáticas defasadas (vento, umidade e temperatura).

Quanto ao objetivo de subsidiar a otimização do mix de produtos, este aspecto não foi diretamente abordado. O foco foi direcionado à previsão de demanda, que constitui etapa anterior e necessária para posterior otimização de mix. A implementação de módulo de otimização requer modelagem adicional de margens de contribuição e restrições de capacidade, constituindo escopo para trabalhos futuros. Da mesma forma, a quantificação financeira da redução de estoques não foi mensurada, embora os resultados obtidos sugiram potencial de aplicação prática. Além disso, as limitações do trabalho incluem horizonte de 12 semanas que pode ser insuficiente para planejamento de coleções sazonais, impossibilidade de capturar eventos imprevisíveis como promoções não planejadas, e MAPE elevado para produtos com demanda muito baixa. O coeficiente R^2 negativo ou próximo de zero indica que a variabilidade da demanda em horizonte de 12 semanas é predominantemente estocástica, indicando alta variabilidade da demanda.

Como trabalhos futuros, sugere-se a implementação de comparação direta com métodos tradicionais como ARIMA e Holt-Winters para validação adicional dos ganhos obtidos, incorporação de dados de redes sociais para capturar tendências emergentes, quantificação do impacto financeiro através de implementação piloto, desenvolvimento de módulo de otimização de mix de produtos, extensão do horizonte de previsão para 26 ou 52 semanas, e aplicação de técnicas de redução de dimensionalidade como Análise de Componentes Principais (PCA) para simplificar o conjunto de 130 características e potencialmente reduzir a complexidade do modelo.

Em conclusão, este trabalho demonstra que modelos de aprendizado de máquina são ferramentas viáveis e eficazes para previsão de demanda na indústria têxtil. A implementação deste sistema pode contribuir para a competitividade da indústria têxtil do Vale do Itajaí através de melhor planejamento de produção e gestão de estoques.

REFERÊNCIAS

ABBASIMEHR, H.; SHABANI, M.; YOUSEFI, M. **An optimized model using LSTM network for demand forecasting**. Computers & Industrial Engineering, v. 143, p. 106435, 2020. DOI: 10.1016/j.cie.2020.106435.

ALBELADI, S. R.; ZAFAR, B.; MUEEN, A. **Time series forecasting using LSTM and ARIMA:** A comparative study on univariate time series data. International Journal of Advanced Computer Science and Applications, v. 14, n. 5, p. 671-680, 2023. DOI: 10.14569/IJACSA.2023.0140570.

BEDDAR-WIESING, S. et al. **Comprehensive Comparative Study of Multi-Label Classification Methods**. Entropy, v. 25, n. 12, p. 1641, 2023. DOI: 10.3390/e25121641.

CANDEIAS et al. **Previsão de demanda:** simulação em uma empresa do segmento de artigos para dança, fitness, natação e sportswear. Revista Produção Online, Florianópolis, v. 20, n. 1, p. 119-148, 2020. DOI: 10.14488/1676-1901.v20i1.3343. Disponível em: https://www.producaoonline.org.br/rpo/article/download/3343/1886. Acesso em: 20 nov. 2025.

CARVALHO, A. C. P. D. L. F. et al. **Inteligência Artificial:** uma abordagem de aprendizado de máquina. 2. ed. Rio de Janeiro: LTC, 2021.

CARVALHO, L.; VIDAL, C. **A indústria têxtil no Brasil:** uma análise dos últimos 10 anos. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 43., 2023, Fortaleza. Anais [...]. Fortaleza: ENEGEP, 2023.

CHAI, T.; DRAXLER, R. R. **Root mean square error (RMSE) or mean absolute error (MAE)?** – Arguments against avoiding RMSE in the literature. Geoscientific Model Development, v. 7, n. 3, p. 1247-1250, 2014. DOI: 10.5194/gmd-7-1247-2014.

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, San Francisco. Proceedings... New York: ACM, 2016. p. 785-794. DOI: http://dx.doi.org/10.1145/2939672.2939785.

DODGE, Y. **The Concise Encyclopedia of Statistics**. New York: Springer, 2008.

DUARTE, Filipe C. D. L. et al. **Previsão da arrecadação do ICMS**: uso do modelo Holt-Winters Aditivo na Paraíba. In: CONGRESSO USP DE CONTROLADORIA E CONTABILIDADE, 14., 2014, São Paulo. Anais [...]. São Paulo: FIPECAFI, 2014. Disponível em: https://congressousp.fipecafi.org/anais/artigos142014/507.pdf. Acesso em: 24 abr. 2025.

IEMI – INTELIGÊNCIA DE MERCADO. **Brasil Têxtil 2024:** Relatório Setorial da Indústria Têxtil Brasileira. 24. ed. São Paulo: IEMI, 2024. Apoio: ABIT e SENAI CETIQT. Disponível em: https://iemi.com.br/brasil-textil-2024.

FACISC – FEDERAÇÃO DAS ASSOCIAÇÕES EMPRESARIAIS DE SANTA CATARINA. **Vale é a segunda região do país que mais contratou na indústria, entre janeiro e agosto**. Centro de Inteligência e Estratégia da FACISC. Blumenau, out. 2024. Disponível em: https://www.facisc.org.br/noticias/vale-e-a-segunda-regiao-do-pais-que-mais-contratou-na-industria-entre-janeiro-e-agosto/. Acesso em: 20 nov. 2025.

FEBRATEX GROUP. **Gerenciamento de estoque no setor têxtil**. [S.l.]: FCEM, 27 jun. 2019. Disponível em: https://fcem.com.br/noticias/gerenciamento-de-estoque-no-setor-texti/. Acesso em: 17 abr. 2025.

FRIDGEIRSSON, H. E. et al. **An evaluation of feature selection methods for environmental data**. Ecological Informatics, v. 79, 2024. DOI: 10.1016/j.ecoinf.2023.102430.

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**: Concepts, Tools, and Techniques to Build Intelligent Systems. 2. ed. Sebastopol: O'Reilly Media, 2019.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning:** Data Mining, Inference, and Prediction. 2. ed. New York: Springer, 2009.

HENZEL, Joanna et al. **Demand Forecasting in the Fashion Business — An Example of Customized Nearest Neighbour and Linear Mixed Model Approaches**. In: FEDCSIS 2022 – Federated Conference on Computer Science and Intelligence Systems, Sofia, 2022. p. 61–65. DOI: https://annals-csis.org/Volume_30/drp/256.html. Acesso em: 16 abr. 2025.

HYNDMAN, R. J.; ATHANASOPOULOS, George. **Forecasting:** Principles and Practice. 2. ed. Melbourne: OTexts, 2018. Disponível em: https://otexts.com/fpp2/. Acesso em: 26 jun. 2025.

KELLEHER, J. D.; NAMEE, B. M.; D'ARCY, A. **Fundamentals of machine learning for predictive data analytics:** algorithms, worked examples, and case studies. Cambridge: The MIT Press, 2015.

LEE, J. et al. **Towards better diagnosis prediction using bidirectional recurrent neural networks**. In: IEEE INTERNATIONAL CONFERENCE ON HEALTHCARE INFORMATICS (ICHI), 10., 2022, Rochester. Proceedings [...]. Rochester: IEEE, 2022. p. 511–512. Disponível em: https://ieeexplore.ieee.org/document/9838724. Acesso em: 20 nov. 2025.

LORENTE-LEYVA, Leandro L. et al. **A Comparison of Machine Learning and Classical Demand Forecasting Methods:** A Case Study of Ecuadorian Textile Industry. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, OPTIMIZATION, AND DATA SCIENCE, 2020, València. Lecture Notes in Computer Science, v. 12566, p. 131–142. Springer. Disponível em: https://link.springer.com/chapter/10.1007/978-3-030-64580-9_11. Acesso em: 24 mar. 2025.

LV, Jieni et al. **Clothing Sales Forecast Considering Weather Information:** An Empirical Study in Brick-and-Mortar Stores by Machine-Learning. Journal of Textile Science and Technology, v. 9, p. 1–19, 2023. Disponível em: https://www.scirp.org/journal/paperinformation?paperid=122879. Acesso em: 24 mar. 2025.

MILNITZ, D. et al. **Mapeamento da indústria têxtil e de confecções em Santa Catar**ina. Revista de Ciências Empresariais da UNIPAR, v. 24, n. 1, 2023. Disponível em: https://www.apec.org.br/rce/index.php/rce/article/view/142.

PINHEIRO, J. M. H. **Um estudo sobre algoritmos de boosting e a otimização de hiperparâmetros utilizando Optuna**. 2023. 71 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Mecatrônica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023. Disponível em: https://bdta.abcd.usp.br/item/003122385. Acesso em: 20 nov. 2025.

PROKHORENKOVA, L.; GUSEV, G.; VOROBEV, A.; DOROGUSH, A. V.; GULIN, A. **CatBoost:** unbiased boosting with categorical features. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 31, 2018, Montréal. Proceedings... Montréal: NIPS, 2018. p. 6638-6648.

RIBEIRO, M. V. S. et al. **Previsão de demanda utilizando redes neurais artificiais:** um estudo de caso em uma distribuidora de bebidas. Revista de Gestão e Secretariado, v. 15, n. 2, p. 1466-1486, 2024.

RIBEIRO, Maria Vitória Sousa et al. **Aprendizado profundo baseado em CNN-LSTM para legendagem automática de imagens**. Revista Aracê, v. 6, n. 3, p. 6725-6749, 2024. Disponível em: https://periodicos.newsciencepubl.com/arace/article/download/1339/1902/5258.

SAADEDDIN, Zaina. **ARIMA para previsão de séries temporais:** um guia completo. [S.l.]: DataCamp, 2024. Disponível em: https://www.datacamp.com/pt/tutorial/arima. Acesso em: 24 abr. 2025.

SCHUSTER, M.; PALIWAL, K. K. **Bidirectional recurrent neural networks**. IEEE Transactions on Signal Processing, v. 45, n. 11, p. 2673–2681, 1997. Disponível em: https://ieeexplore.ieee.org/document/650093. Acesso em: 20 nov. 2025.

SURESH, R.; JARAPALA, S. R.; SUDEEP, V. LSTM **based stock price prediction using sentiment analysis**. International Journal of Computer Science and Mobile Computing, v. 11, n. 3, p. 56-62, 2022. DOI: 10.47760/ijcsmc.2022.v11i03.006.

YASIR, Muhammad et al. **Machine Learning–Assisted Efficient Demand Forecasting Using Endogenous and Exogenous Indicators for the Textile Industry**. International Journal of Logistics Research and Applications, v. 27, n. 12, p. 2867–2886, 2022. Disponível em: https://www.tandfonline.com/doi/full/10.1080/13675567.2022.2100334. Acesso em: 16 abr. 2025.

APÊNDICE A – CÓDIGO DOS MODELOS DE REGRESSÃO LINEAR

O Quadro 8 apresenta o código fonte da implementação do modelo PolynomialRidge, incluindo a configuração da sequência de processamento com normalização, transformação polinomial e regularização, além do espaço de hiperparâmetros explorado pelo Optuna.

Quadro 8 – Implementação do modelo PolynomialRidge

```
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.linear_model import Ridge
from sklearn.pipeline import Pipeline

# Espaço de hiperparâmetros específico
def objective(trial):
    degree = trial.suggest_int('degree', 1, 3)
    interaction_only = trial.suggest_categorical('interaction_only', [True, False])
    alpha = trial.suggest_float('alpha', 0.001, 100.0, log=True)

    # Pipeline: Scaler → Polynomial → Ridge
    pipeline = Pipeline([
        ('scaler', StandardScaler()),
        ('poly', PolynomialFeatures(
            degree=degree,
            interaction_only=interaction_only,
            include_bias=False
        )),
        ('ridge', Ridge(alpha=alpha, random_state=RANDOM_STATE))
    ])

    # Validação cruzada temporal (conforme Quadro 7)
    # ...

# Treinamento final com melhores parâmetros
final_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('poly', PolynomialFeatures(
        degree=study.best_params['degree'],
        interaction_only=study.best_params['interaction_only'],
        include_bias=False
    )),
    ('ridge', Ridge(
        alpha=study.best_params['alpha'],
        random_state=RANDOM_STATE
    ))
])
final_pipeline.fit(X_train, y_train)
```

Fonte: Elaborado pelo autor.

O Quadro 9 apresenta o código fonte da implementação do modelo ElasticNet, contendo a normalização dos dados, o espaço de hiperparâmetros e o treinamento final com os melhores parâmetros identificados.

Quadro 9 – Implementação do modelo ElasticNet

```
from sklearn.linear_model import ElasticNet
from sklearn.preprocessing import StandardScaler

# Normalização (crítica para ElasticNet)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Espaço de hiperparâmetros específico
def objective(trial):
    params = {
        'alpha': trial.suggest_float('alpha', 0.0001, 10.0, log=True),
        'l1_ratio': trial.suggest_float('l1_ratio', 0.0, 1.0),
        'max_iter': 10000,
        'random_state': RANDOM_STATE,
        'selection': trial.suggest_categorical('selection', ['cyclic', 'random'])
    }

    model = ElasticNet(**params)
    # Validação cruzada temporal (conforme Quadro 7)
    # ...

# Treinamento final
final_model = ElasticNet(
    max_iter=10000,
    random_state=RANDOM_STATE,
    **study.best_params
)
final_model.fit(X_train_scaled, y_train)
```

Fonte: Elaborado pelo autor.

APÊNDICE B – CÓDIGO DOS MODELOS DE GRADIENT BOOSTING

O Quadro 10 apresenta o código fonte da implementação do modelo XGBoost, contendo o espaço de hiperparâmetros, a configuração de interrupção antecipada e o treinamento final.

Quadro 10 – Implementação do modelo XGBoost

```
import xgboost as xgb
RANDOM_STATE = 42
# Espaço de hiperparâmetros específico
def objective(trial):
    params = {
        'objective': 'reg:squarederror',
        'eval_metric': 'mae',
        'booster': 'gbtree',
        'random_state': RANDOM_STATE,
        'n_estimators': trial.suggest_int('n_estimators', 100, 1000, step=50),
        'max_depth': trial.suggest_int('max_depth', 3, 10),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3, log=True),
        'subsample': trial.suggest_float('subsample', 0.6, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.6, 1.0),
        'min_child_weight': trial.suggest_int('min_child_weight', 1, 10),
        'gamma': trial.suggest_float('gamma', 0, 5),
        'reg_alpha': trial.suggest_float('reg_alpha', 0, 1),
        'reg_lambda': trial.suggest_float('reg_lambda', 0, 1),
        'verbosity': 0
    }

    model = xgb.XGBRegressor(**params)
    # Validação cruzada temporal (conforme Quadro 7)
    # ...

# Treinamento final com early stopping
final_params = {
    'objective': 'reg:squarederror',
    'eval_metric': 'mae',
    'booster': 'gbtree',
    'random_state': RANDOM_STATE,
    'verbosity': 0,
    'early_stopping_rounds': 50,
    **study.best_params
}

final_model = xgb.XGBRegressor(**final_params)

# Separar validação para early stopping
split_point = int(len(X_train) * 0.8)
final_model.fit(
    X_train_model, y_train_model,
    eval_set=[(X_val_model, y_val_model)],
    verbose=True
)
```

Fonte: Elaborado pelo autor.

O Quadro 11 apresenta o código fonte da implementação do modelo CatBoost, incluindo a identificação automática de variáveis categóricas, o uso da classe `Pool` e a configuração de interrupção antecipada.

Quadro 11 – Implementação do modelo CatBoost

```
from catboost import CatBoostRegressor, Pool
RANDOM_STATE = 42
# Identificar features categóricas
categorical_features = df_forecasting[feature_cols].select_dtypes(
    include=['object', 'category']
).columns.tolist()

# Espaço de hiperparâmetros específico
def objective(trial):
    params = {
        'iterations': trial.suggest_int('iterations', 100, 1000, step=50),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3, log=True),
        'depth': trial.suggest_int('depth', 4, 10),
        'l2_leaf_reg': trial.suggest_float('l2_leaf_reg', 1, 10),
        'min_data_in_leaf': trial.suggest_int('min_data_in_leaf', 1, 100),
        'random_strength': trial.suggest_float('random_strength', 0, 10),
        'bagging_temperature': trial.suggest_float('bagging_temperature', 0, 1),
        'border_count': trial.suggest_int('border_count', 32, 255),
        'random_state': RANDOM_STATE,
        'verbose': False,
        'loss_function': 'MAE',
        'eval_metric': 'MAE'
    }

    # Criar Pool com features categóricas
    train_pool = Pool(
            X_fold_train,
            y_fold_train,
            cat_features=categorical_features
        )
    val_pool = Pool(
            X_fold_val,
            y_fold_val,
            cat_features=categorical_features
        )

    model = CatBoostRegressor(**params)
    model.fit(train_pool, eval_set=val_pool, verbose=False)
    # ...

# Treinamento final
final_params = {
    'random_state': RANDOM_STATE,
    'verbose': 100,
    'loss_function': 'MAE',
    'eval_metric': 'MAE',
    'early_stopping_rounds': 50,
    **study.best_params
}

final_model = CatBoostRegressor(**final_params)

train_pool = Pool(
    X_train_model,
    y_train_model,
    cat_features=categorical_features
)
val_pool = Pool(
    X_val_model,
    y_val_model,
    cat_features=categorical_features
)

final_model.fit(train_pool, eval_set=val_pool)
```

Fonte: Elaborado pelo autor.

APÊNDICE C – CÓDIGO DOS MODELOS DE REDES NEURAIS RECORRENTES

O Quadro 12 apresenta o código fonte da implementação do modelo LSTM, contendo a preparação dos dados, a arquitetura condicional e a configuração das funções de controle do treinamento.

Quadro 12 – Implementação do modelo LSTM

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler

# Preparação de dados para LSTM
scaler_X = StandardScaler()
X_train_scaled = scaler_X.fit_transform(X_train)

scaler_y = StandardScaler()
y_train_scaled = scaler_y.fit_transform(y_train.values.reshape(-1, 1)).ravel()

# Reshape para formato LSTM (samples, timesteps=1, features)
X_train_lstm = X_train_scaled.reshape((X_train_scaled.shape[0], 1,
X_train_scaled.shape[1]))

# Espaço de hiperparâmetros específico
def objective(trial):
    lstm_units_1 = trial.suggest_int('lstm_units_1', 32, 256, step=32)
    dropout_rate = trial.suggest_float('dropout_rate', 0.1, 0.5)
    use_batch_norm = trial.suggest_categorical('use_batch_norm', [True, False])
    use_second_lstm = trial.suggest_categorical('use_second_lstm', [True, False])
    learning_rate = trial.suggest_float('learning_rate', 1e-4, 1e-2, log=True)
    batch_size = trial.suggest_categorical('batch_size', [32, 64, 128, 256])

    # Construir arquitetura LSTM
    model = Sequential()

    if use_second_lstm:
        model.add(LSTM(lstm_units_1, return_sequences=True,
                       input_shape=(X_train_lstm.shape[1], X_train_lstm.shape[2])))
    else:
        model.add(LSTM(lstm_units_1,
                       input_shape=(X_train_lstm.shape[1], X_train_lstm.shape[2])))

    if use_batch_norm:
        model.add(BatchNormalization())
    model.add(Dropout(dropout_rate))

    if use_second_lstm:
        model.add(LSTM(lstm_units_2))
        if use_batch_norm:
            model.add(BatchNormalization())
        model.add(Dropout(dropout_rate))

    model.add(Dense(1))

    # Compilar com Adam e clipnorm
    optimizer = Adam(learning_rate=learning_rate, clipnorm=1.0)
    model.compile(optimizer=optimizer, loss='mae', metrics=['mae'])

    # Callbacks para controle de treinamento
    early_stop = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5,
min_lr=1e-6)
```

Fonte: Elaborado pelo autor.

O Quadro 13 apresenta o código fonte da implementação do modelo BiLSTM, utilizando a camada Bidirectional do Keras e seguindo a mesma estrutura condicional do modelo LSTM.

Quadro 13 – Implementação do modelo BiLSTM (BRNN)

```
1  from tensorflow.keras.layers import LSTM, Bidirectional
2
3  # Espaço de hiperparâmetros (idêntico ao LSTM)
4  def objective(trial):
5      brnn_units_1 = trial.suggest_int('brnn_units_1', 32, 256, step=32)
6      dropout_rate = trial.suggest_float('dropout_rate', 0.1, 0.5)
7      use_batch_norm = trial.suggest_categorical('use_batch_norm', [True, False])
8      use_second_lstm = trial.suggest_categorical('use_second_lstm', [True, False])
9      learning_rate = trial.suggest_float('learning_rate', 1e-4, 1e-2, log=True)
10     batch_size = trial.suggest_categorical('batch_size', [32, 64, 128, 256])
11
12     # Diferença principal: wrapper Bidirectional
13     model = Sequential()
14     if use_second_lstm:
15         model.add(Bidirectional(LSTM(brnn_units_1, return_sequences=True,
16                                 input_shape=(X_train_brnn.shape[1],
17 X_train_brnn.shape[2]))))
18     else:
19         model.add(Bidirectional(LSTM(brnn_units_1,
20                                 input_shape=(X_train_brnn.shape[1],
21 X_train_brnn.shape[2]))))
22
23     if use_batch_norm:
24         model.add(BatchNormalization())
25     model.add(Dropout(dropout_rate))
26
27     if use_second_lstm:
28         model.add(Bidirectional(LSTM(brnn_units_2)))
29         if use_batch_norm:
30             model.add(BatchNormalization())
31         model.add(Dropout(dropout_rate))
```

Fonte: Elaborado pelo autor.

APÊNDICE D – CÓDIGO DA ESTRATÉGIA DE ENSEMBLE

O Quadro 14 apresenta o código fonte das funções de ensemble desenvolvidas, incluindo a média ponderada simples e a estratégia com calibração de previsões.

Quadro 14 – Implementação da estratégia de Ensemble

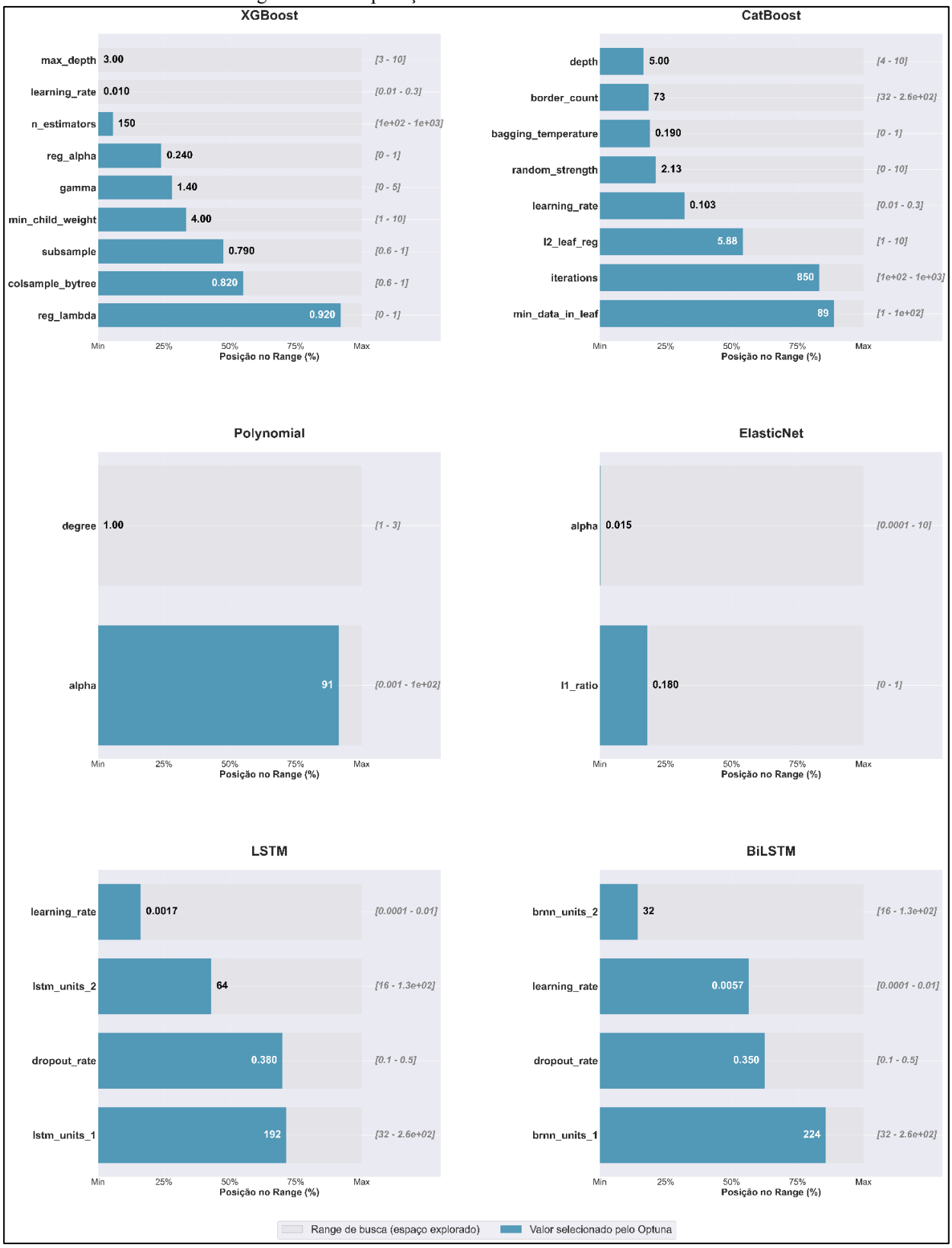
```
import numpy as np
from sklearn.metrics import mean_absolute_error

def ensemble_weighted_average(predictions_dict, weights):
    ensemble_pred = np.zeros_like(next(iter(predictions_dict.values())))

    for model, pred in predictions_dict.items():
        ensemble_pred += weights[model] * pred

    return ensemble_pred

def ensemble_calibrated(predictions_dict, calibration_factors):
    calibrated_preds = {}

    for model, pred in predictions_dict.items():
        if model in calibration_factors:
            calibrated_preds[model] = pred * calibration_factors[model]
        else:
            calibrated_preds[model] = pred

    return calibrated_preds

# Configuração do ensemble
models_config = {
    'catboost_20251002_193349': {
        'name': 'CatBoost',
        'weight': 0.50,
        'calibration': 1.246  # 26401/21185
    },
    '20251002_155938': {
        'name': 'XGBoost',
        'weight': 0.30,
        'calibration': 1.0
    },
    'lasso_ridge_20251007_222123': {
        'name': 'ElasticNet',
        'weight': 0.20,
        'calibration': 1.0
    }
}

# ESTRATÉGIA 1: Ensemble simples (média ponderada)
weights = {key: config['weight'] for key, config in models_config.items()}
ensemble_simple = ensemble_weighted_average(predictions, weights)

# ESTRATÉGIA 2: Ensemble calibrado (calibração + média ponderada)
calibration = {key: config['calibration'] for key, config in models_config.items()}
predictions_calibrated = ensemble_calibrated(predictions, calibration)
ensemble_calibrated_pred = ensemble_weighted_average(predictions_calibrated,
weights)
metrics_calibrated = calculate_metrics(y_true, ensemble_calibrated_pred)
```

Fonte: Elaborado pelo autor.

APÊNDICE E – COMPARAÇÃO DE MÉTRICAS OPTUNA POR MODELO

A Figura 6 apresenta a visualização dos espaços de busca de hiperparâmetros explorados pelo `Optuna`, apresentando os intervalos testados e os valores ótimos selecionados para cada modelo.

Figura 6 – Comparação de métricas dos modelos individuais



Fonte: elaborado pelo autor.