INSTITUTO FEDERAL DO ESPÍRITO SANTO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

FERNANDA MAYUMI FUKAI

REDES NEURAIS DO TIPO *LONG SHORT-TERM MEMORY* APLICADAS EM PREVISÕES DE DEMANDA NO MERCADO VAREJISTA

SERRA
2024

FERNANDA MAYUMI FUKAI

REDES NEURAIS DO TIPO *LONG SHORT-TERM MEMORY* APLICADAS EM PREVISÕES DE DEMANDA NO MERCADO VAREJISTA

Dissertação apresentada ao Programa de Pós-graduação em Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo, como requisito parcial para a obtenção do título de Mestre em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Daniel Cruz Cavalieri

Coorientador: Prof. Dr. Fidelis Zanetti de Castro

SERRA

2024

Dados Internacionais de Catalogação na Publicação (CIP)

F961r Fukai, Fernanda Mayumi
2024 Redes neurais do tipo Long Short-Term Memory aplicadas em previsões de demanda no mercado varejista / Fernanda Mayumi Fukai. - 2024.
91 f.; il.; 30 cm

Orientador: Prof. Dr. Daniel Cruz Cavalieri.
Coorientador: Prof. Dr. Fidelis Zanetti de Castro.

Dissertação (mestrado) - Instituto Federal do Espírito Santo, Programa de Pós-graduação em Engenharia de Controle e Automação, 2024 .

1. Automação. 2. Redes neurais. 3. Varejo - vendas. 4. Machine Learning. 5. Long short-term memory (LSTM) . I. Cavalieri, Daniel Cruz. II. Instituto Federal do Espírito Santo. III. Título.

CDD 629.895

Bibliotecário: Valmir Oliveira de Aguiar - CRB6/ES 566

MINISTÉRIO DA EDUCAÇÃO

INSTITUTO FEDERAL DO ESPÍRITO SANTO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

FERNANDA MAYUMI FUKAI

"REDES NEURAIS DO TIPO LONG SHORT-TERM MEMORY APLICADAS EM PREVISÕES DE DEMANDA NO MERCADO VAREJISTA"

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Controle e Automação do Instituto Federal do Espírito Santo, como requisito parcial para obtenção de título de Mestre em Engenharia de Controle e Automação.

Aprovado em 07 de outubro de 2024

COMISSÃO EXAMINADORA

Documento assinado digitalmente
gov.br
DANIEL CRUZ CAVALIERI
Data: 05/12/2024 17:39:56-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Daniel Cruz Cavalieri
Instituto Federal do Espírito Santo
Orientador

Documento assinado digitalmente
gov.br
FIDELIS ZANETTI DE CASTRO
Data: 10/12/2024 13:48:42-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Fidelis Zanetti de Castro
Instituto Federal do Espírito Santo
Coorientador

Documento assinado digitalmente
gov.br
GABRIEL TOZATTO ZAGO
Data: 09/12/2024 09:56:38-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Gabriel Tozatto Zago
Instituto Federal do Espírito Santo
Membro Interno

Documento assinado digitalmente
gov.br
RODRIGO VAREJAO ANDREAO
Data: 09/12/2024 13:29:20-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Rodrigo Varejão Andreão
Instituto Federal do Espírito Santo
Membro Externo

Ao vovô Hideo, que, mesmo sem formação acadêmica, graduou-se engenheiro pela vida, e me ensinou sobre o imenso poder da educação e o valor transformador do conhecimento. Sua perseverança e respeito pelo aprendizado sempre me impulsionaram a buscar o melhor em mim.

AGRADECIMENTOS

Expresso minha mais profunda gratidão ao meu orientador, professor Dr. Daniel Cruz Cavalieri, cuja energia contagiante, entusiasmo pela docência e apoio incondicional foram pilares essenciais para a continuidade e a concretização deste trabalho.

Ao meu coorientador, Dr. Fidelis Zanetti de Castro, manifesto minha sincera gratidão pela amizade, pela constante disposição em ajudar e pelo incentivo indispensável, especialmente nos momentos mais desafiadores. Aos revisores e avaliadores, agradeço pelas críticas construtivas e sugestões valiosas, que contribuíram significativamente para o aprimoramento deste trabalho.

Ao meu amado esposo e companheiro de vida, Estevan, dedico uma menção especial. Sua fé inabalável em mim, seu apoio constante e sua presença em cada etapa desta jornada foram fontes inesgotáveis de força e inspiração. Este trabalho é, em muitos sentidos, uma prova do que construímos juntos – um reflexo de nossa parceria, amor e perseverança.

Finalmente, agradeço ao Instituto Federal do Espírito Santo e à Fundação de Amparo à Pesquisa e Inovação do Espírito Santo pelo suporte institucional e pelos recursos disponibilizados, que tornaram possível a realização deste trabalho.

"Boys, be ambitious!"

Hideo Fukai, 1981

RESUMO

A previsão de vendas é um componente crítico na indústria do varejo, essencial para decisões estratégicas, planejamento operacional e manutenção da estabilidade financeira, promovendo o crescimento dos negócios. Métodos estatísticos tradicionais têm sido amplamente utilizados para análise de séries temporais, mas frequentemente enfrentam dificuldades com relações complexas, não lineares e dinâmicas de alta dimensionalidade. Nesse contexto, redes Long Short-Term Memory (LSTM) oferecem vantagens significativas devido à sua capacidade de reter informações ao longo de longos períodos, tornando-as adequadas para lidar com interações dinâmicas e complexas nos dados. Esta dissertação avalia o desempenho de três redes neurais baseadas em variações da arquitetura LSTM: Vanilla LSTM, Conv1D-LSTM e Multiscale CNN-LSTM, aplicadas à previsão de vendas futuras. Para uma análise comparativa robusta, os modelos Average, XGBoost e LightGBM também são empregados como benchmarks, com desempenho avaliado por métricas como Erro Absoluto Médio (MAE), Raiz do Erro Quadrático Médio (RMSE) e Erro Absoluto Escalado Médio (MASE). A otimização de hiperparâmetros foi realizada para melhorar os resultados. Os resultados indicam que a arquitetura Conv1D-LSTM consistentemente supera os outros modelos LSTM em quase todas as métricas, alcançando o maior R^2 e os menores valores de MAE, RMSE e MASE após a otimização Bayesiana. No entanto, modelos como LightGBM e XGBoost demonstraram desempenho superior aos modelos LSTM, com custos computacionais significativamente menores, tornando-os alternativas eficientes para previsão de vendas, especialmente em cenários com recursos computacionais limitados. Embora os modelos baseados em LSTM, particularmente o Conv1D-LSTM, sejam altamente eficazes para lidar com a complexidade dos dados de séries temporais, modelos como LightGBM e XGBoost permanecem competitivos, oferecendo um equilíbrio valioso entre desempenho e eficiência computacional.

Palavras-chave: LSTM, Redes Neurais Convolucionais, Aprendizado Profundo, Previsão de Vendas, Varejo.

ABSTRACT

Sales forecasting is a critical component of the retail industry, essential for strategic decision-making, operational planning, and maintaining financial stability while promoting business growth. Traditional statistical methods have been widely used for time series analysis but often struggle with complex, nonlinear relationships and high-dimensional dynamics among variables. In this context, Long Short-Term Memory (LSTM) networks offer significant advantages due to their ability to retain information over long periods, making them particularly suited to handle the complex and dynamic interactions inherent in the data. This dissertation evaluates the performance of three neural networks based on variations of the LSTM architecture: Vanilla LSTM, Conv1D-LSTM, and Multiscale CNN-LSTM, applied to future sales forecasting. To provide a more robust comparative analysis, the Average, XGBoost, and LightGBM models are also employed as benchmarks. Model performance is assessed using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Scaled Error (MASE). Additionally, hyperpa- rameter tuning is conducted to optimize the models' results. The results indicate that the Conv1D-LSTM architecture consistently outperforms other LSTM models across nearly all evaluation metrics, achieving the highest R^2 value and the lowest MAE, RMSE, and MASE after Bayesian optimization of a hyperparameter set. However, models like LightGBM and XGBoost demonstrated superior performance compared to LSTM-type model's, while offering the advantage of significantly lower computational costs. This makes them highly efficient alternatives for sales forecasting, especially in scenarios where computational resources are limited. Overall, the study concludes that while LSTM-based models, particularly Conv1D-LSTM, are highly effective at navigating the complexities of time series data (in our context) and identifying intricate patterns over long periods, traditional models like LightGBM and XGBoost remain competitive, providing a valuable balance between performance and computational efficiency.

Keywords: LSTM, Convolutional Neural Networks, Deep Learning, Sales Forecasting, Retail.

LISTA DE ILUSTRAÇÕES

Figura 1 – Fatores influenciadores da demanda. 22
Figura 2 – Dimensões da previsão em nível de produto no varejo 24
Figura 3 – Arquitetura de uma CNN. 32
Figura 4 – Extração de *feature* usando convolução. 33
Figura 5 – Exemplos de funções de ativação. 34
Figura 6 – Exemplo de *Pooling* máximo. 35
Figura 7 – Exemplo de rede sem e com *dropout*. 35
Figura 8 – RNN simples de camada oculta única com uma única sequência de saída. 38
Figura 9 – Arquitetura de uma LSTM tradicional. 39
Figura 10 – A atualização da porta de esquecimento do estado da célula da LSTM. 40
Figura 11 – Operação da porta de entrada do estado da célula da LSTM. 41
Figura 12 – Operação da porta de saída do estado da célula da LSTM. 41
Figura 13 – Estrutura interna da ConvLSTM. 43
Figura 14 – Otimização em dois estágios para fluxo de trabalho de aprendizado de máquina. 45
Figura 15 – Função de Distribuição Cumulativa Empírica para *Sales* e *SalesPerCustomer*. 47
Figura 16 – Gráficos de dispersão ilustrando a relação entre clientes e distância da loja ao concorrente mais próximo, em relação a variável de alvo vendas. 47
Figura 17 – Distribuição das vendas nos dia da semana por tipo de loja. 48
Figura 18 – Relação entre variedade, tipo de loja e vendas. 48
Figura 19 – Análise da variáveis de promoção. 50
Figura 20 – Distribuição das vendas durante a ocorrência de feriados escolares e estaduais. 50
Figura 21 – Distribuição binomial para 20 interações, ilustrando as áreas de recusa, indecisão e aceitação. 55
Figura 22 – Vanilla LSTM. 61
Figura 23 – Arquitetura Multiscale CNN-LSTM. 63
Figura 24 – Arquitetura do modelo ConvLSTM. 64

Figura 25 – Comparação entre valores reais e previstos para o modelo Vanilla LSTM com *lookback* de 7 dias, antes e depois da otimização de hiperparâmetros. 78
Figura 26 – Comparação entre valores reais e previstos para o modelo Vanilla LSTM com *lookback* de 30 dias, antes e depois da otimização de hiperparâmetros. 78
Figura 27 – Comparação entre valores reais e previstos para o modelo Conv1D-LSTM com *lookback* de 7 dias, antes e depois da otimização de hiperparâmetros. 79
Figura 28 – Comparação entre valores reais e previstos para o modelo Conv1D-LSTM com *lookback* de 30 dias, antes e depois da otimização de hiperparâmetros. 79
Figura 29 – Comparação entre valores reais e previstos para o modelo Multiscale CNN-LSTM com *lookback* de 7 dias, antes e depois da otimização de hiperparâmetros. 80
Figura 30 – Comparação entre valores reais e previstos para o modelo Multiscale CNN-LSTM com *lookback* de 30 dias, antes e depois da otimização de hiperparâmetros. 81
Figura 31 – Gráfico de perda para o modelo Vanilla LSTM com *lookback* de 7 dias. 82
Figura 32 – Gráfico de perda para o modelo Vanilla LSTM com *lookback* de 30 dias. 82
Figura 33 – Gráfico de perda para o modelo Multiscale CNN-LSTM com *lookback* de 7 dias. 83
Figura 34 – Gráfico de perda para o modelo Multiscale CNN-LSTM com *lookback* de 30 dias. 83
Figura 35 – Gráfico de perda para o modelo Conv1D-LSTM com *lookback* de 7 dias. 84
Figura 36 – Gráfico de perda para o modelo Conv1D-LSTM com *lookback* de 30 dias. 84

LISTA DE TABELAS

Tabela 1 – Variáveis do conjunto de dados Rossmann. 46
Tabela 2 – Características da Amostragem 55
Tabela 3 – Otimização Baynesiana para o modelo XGBoost 59
Tabela 4 – Otimização Baynesiana para o modelo LightGBM. 61
Tabela 5 – Otimização Bayesiana para o modelo Vanilla LSTM. 62
Tabela 6 – Otimização Baynesiana para o modelo CONV1D-LSTM. 63
Tabela 7 – Otimização Baynesiana para Multiscale CNN-LSTM 65
Tabela 8 – Desempenho dos modelos no conjunto de Treinamento, Teste e Validação com e sem Seleção de Atributos). 71
Tabela 9 – Resultados de validação dos Modelos de LSTM para diferentes Épocas. 72
Tabela 10 – Resultados de Validação e Teste para Diferentes Configurações de Lookback (7 e 30 dias) 74
Tabela 11 – Resultados de teste e tempo de treinamento (T) para diferentes modelos com lookback 7 e 30 para modelos com hiperparâmetros ajustados e não ajustados 76

SUMÁRIO

1 INTRODUÇÃO 13
1.1 MOTIVAÇÃO 15
1.2 PROPOSTA DE PESQUISA 16
1.3 OBJETIVO GERAL 17
1.3.1 Objetivos Específicos 17
1.4 ORGANIZAÇÃO DA DISSERTAÇÃO 18
2 FUNDAMENTAÇÃO TEÓRICA 19
2.1 PREVISÃO DE DEMANDA NO VAREJO 19
2.1.1 O setor varejista 19
2.1.2 Fatores Influenciadores 20
2.1.3 Granularidade 23
2.2 SÍNTESE DA REVISÃO BIBLIOGRÁFICA 23
2.3 SISTEMAS DE APRENDIZADO SUPERVISIONADO 27
2.3.1 Modelos de Gradient Boosting 27
2.3.1.1 LightGBM 28
2.3.1.2 XGBoost 30
2.3.2 Redes Neurais Convolucionais (CNNs) 32
2.3.3 Redes Neurais Recorrentes (RNNs) 37
2.3.4 Redes do Tipo *Long Short-Term Memory* (LSTM) 39
2.3.5 ConvLSTM 42
3 METODOLOGIA 45
3.1 CONJUNTO DE DADOS 45
3.2 ANÁLISE EXPLORATÓRIA DE DADOS 46
3.3 PRÉ-PROCESSAMENTO DOS DADOS 51
3.3.1 Dados Ausentes 51
3.3.2 Enriquecimento de Dados 52
3.3.3 Codificação e Normalização 52
3.3.4 Transformações Cíclicas e Logarítmicas 52
3.3.5 Filtragem e Ajustes 53
3.4 ALGORITMO *BORUTA* PARA SELEÇÃO DE ATRIBUTOS 53
3.5 CONFIGURAÇÕES DOS AMBIENTES DE TESTES 55

3.5.1 Divisão em Treino e Teste 56
3.5.2 Bibliotecas e Ferramentas 56
3.6 CONFIGURAÇÕES 58
3.6.1 Modelo Average 58
3.6.2 XGBoost 58
3.6.3 LightGBM 60
3.6.4 Vanilla LSTM 61
3.6.5 Multiscale CNN-LSTM 62
3.6.6 ConvLSTM 64
3.7 OTIMIZAÇÃO DE HIPERPARÂMETROS 65
3.8 MÉTRICAS DE AVALIAÇÃO DE DESEMPENHO 66
4 RESULTADOS E DISCUSSÕES 69
4.1 SELEÇÃO DE ATRIBUTOS 70
4.2 AVALIAÇÃO DO NÚMERO DE ÉPOCAS PARA OS ALGORITMOS DE APRENDIZADO PROFUNDO 72
4.3 RESULTADOS PARA DIFERENTES CONFIGURAÇÕES DE LOOK-BACK E LAGS 73
4.4 AJUSTE DOS HIPERPARÂMETROS 75
4.5 ANÁLISE COMPARATIVA ENTRE VALORES REAIS E PREVISTOS PARA DIFERENTES MODELOS DE LSTM 76
4.6 ANÁLISE COMPARATIVA DOS GRÁFICOS DE PERDA PARA DIFERENTES MODELOS LSTM 81
5 CONSIDERAÇÕES FINAIS 86
REFERÊNCIAS 88

1 INTRODUÇÃO

A previsão precisa da demanda de produtos é essencial para o sucesso e eficiência das operações no setor varejista. Ao antecipar a quantidade de itens que os clientes irão comprar, os varejistas podem ajustar seus estoques e recursos de forma adequada, evitando tanto a falta de produtos quanto o excesso de estoque, que podem prejudicar a lucratividade e a satisfação do cliente. Por exemplo, um supermercado pode utilizar dados históricos de vendas e analisar tendências sazonais para planejar o estoque necessário durante períodos de alta demanda, como nas festas de fim de ano. Além disso, a previsão de demanda é fundamental para o planejamento de estratégias de marketing e promoções, permitindo atrair mais clientes e maximizar as vendas. Assim, a previsão de demanda não só apoia a eficiência operacional, mas também fortalece a base para decisões estratégicas, ajudando os varejistas a atender às expectativas dos clientes de forma mais eficiente e rentável.

Os fatores que influenciam vendas e demanda podem ser divididos em três categorias: internos, parcialmente internos e externos (ARUNRAJ; AHRENS, 2015). Fatores internos são elementos controlados pelo próprio varejista, os quais incluem promoções e descontos. Já os fatores parcialmente internos abrangem aspectos como canibalização e complementaridade entre produtos. Os fatores externos, por sua vez, englobam elementos observáveis, como eventos esportivos, festivais e feriados, bem como eventos atípicos como crises sanitárias e greves, além de previsões sobre a economia nacional e condições climáticas (ARUNRAJ; AHRENS, 2015).

Em decorrência desses fatores, os dados de vendas por produto frequentemente exibem alta volatilidade, assimetria, múltiplos ciclos sazonais, intermitência com períodos de ausência de vendas e esgotamento de estoques (FILDES; MA; KOLASSA, 2019). Entre os benefícios de uma previsão de demanda precisa, destacam-se o impacto positivo no gerenciamento de pessoas, que evita tanto o excesso quanto o deficit de mão de obra, e o planejamento eficaz das vendas. Com uma previsão acurada, os varejistas podem adaptar suas campanhas de marketing, ajustar preços e planejar lançamentos de novos produtos de maneira mais alinhada às tendências de mercado e às expectativas dos consumidores (MA; FILDES; HUANG, 2016).

Entretanto, um dos principais impulsionadores por trás da crescente demanda por técnicas de previsão reside no impacto que exercem sobre o planejamento de estoques. Estudos apontam que os custos de manutenção de estoque podem representar de 20% a 30% do valor total do inventário (INVESTOPEDIA, 2023). Diante disso, é crucial considerar que, embora os preços aplicados a grandes volumes de aquisição sejam geralmente mais baixos, o custo real do produto pode eventualmente superar o preço de venda de volumes menores, caso a demanda não seja suficiente e os custos de manutenção do estoque sejam levados em conta. Nesse contexto, além dos custos associados aos erros decorrentes de excesso de estoque, é igualmente relevante destacar as desvantagens do subestoque, uma vez que a falta de produtos pode resultar na insatisfação do cliente, danos à reputação da empresa varejista e redução dos seus lucros (BEUTEL; MINNER, 2012).

Com o advento do Aprendizado de Máquina (do inglês, *Machine Learning*), é possível notar uma transformação significativa na abordagem de previsão de demanda. Modelos de aprendizado de máquina passaram a oferecer a vantagem de manipular grandes volumes de dados e identificar padrões complexos que podem não ser capturados por abordagens tradicionais de previsão. Esses modelos têm sido utilizados para prever vendas no comércio eletrônico, varejo de moda e edifícios comerciais, demonstrando sua versatilidade em diferentes indústrias (FENG; NIU; SONG, 2022; GIRI; CHEN, 2022).

Por sua vez, a integração do aprendizado de máquina com a inteligência empresarial tem capacitado organizações a prever efetivamente demandas futuras e tomar decisões informadas (KHAN *et al.*, 2020). Nesse sentido, o uso de algoritmos de aprendizado de máquina, como *Random Forest*, *Extreme Gradient Boosting* e *Light GBM*, tem sido investigado para desenvolver modelos de previsão que possam se adaptar a condições dinâmicas e fornecer previsões precisas.

Já a aplicação de técnicas de aprendizado profundo, como *Convolutional Neural Networks* (CNNs) e *Long Short-Term Memories* (LSTMs), demonstrou resultados promissores na melhoria da precisão e desempenho dos modelos de previsão de demanda (GIRI; CHEN, 2022). Essas técnicas têm sido particularmente eficazes no

manuseio de dados de vendas de alta dimensão e na projeção destes em um espaço de alta dimensão para processamento, melhorando assim a capacidade de ajustar curvas de vendas mais próximas à realidade.

Embora algumas pesquisas venham sendo realizadas com o intuito de investigar a utilização de algoritmos de aprendizado profundo para a construção de modelos de previsão de demanda, poucas exploram a comparação de desempenho das variações de modelos baseados em LSTM. Diante dessa problemática, nesta dissertação será realizada uma comparação de desempenho entre modelos de redes neurais profundas que se baseiam em arquiteturas do tipo LSTM, utilizando como *benchmarks* os modelos XGBoost, LightGBM, Regressão Linear e Average.

- **Vanilla LSTM**: Representa a arquitetura mais simples de LSTM, processando sequências de dados em uma única direção e aprendendo padrões temporais ao longo das séries.
- **Multiscale CNN-LSTM**: Uma variante da LSTM que incorpora camadas convolucionais unidimensionais (*Conv1D*) para extrair características locais das séries temporais antes de processá-las com a LSTM. Essa abordagem captura padrões em diferentes níveis de granularidade.
- **Conv1D-LSTM**: Uma variante da LSTM que combina diretamente operações convolucionais unidimensionais (*Conv1D*) com a arquitetura LSTM, permitindo capturar características locais e temporais em séries temporais de forma eficiente.

1.1 MOTIVAÇÃO

A previsão precisa de demanda é um fator crucial para a eficiência e competitividade das operações no setor varejista. Em um mercado cada vez mais orientado por dados, as empresas que conseguem antecipar as necessidades dos consumidores têm uma vantagem significativa na otimização de estoques, planejamento de produção e alocação de recursos. Entretanto, a volatilidade das vendas, influenciada por uma combinação de fatores sazonais, econômicos e comportamentais, torna essa tarefa particularmente desafiadora.

Tradicionalmente, métodos estatísticos e técnicas de *Machine Learning*, como regressão linear e modelos baseados em árvores de decisão, têm sido amplamente utilizados para previsão de demanda. Embora eficazes em muitos cenários, esses métodos frequentemente apresentam limitações quando confrontados com padrões complexos e não lineares presentes em séries temporais de vendas. Com o avanço das tecnologias de *Deep Learning* (Aprendizado Profundo), novas oportunidades surgem para capturar essas complexidades de maneira mais eficiente e robusta.

Modelos baseados em *Long Short-Term Memory* (*LSTM*) têm mostrado grande potencial em tarefas de previsão de séries temporais, devido à sua capacidade de manter e processar informações ao longo de longas sequências. No entanto, apesar do crescente interesse acadêmico e das promissoras aplicações industriais, há ainda lacunas na literatura quanto à aplicação comparativa de diferentes variações de *LSTM* no contexto específico da previsão de demanda no varejo.

Diante desse cenário, a motivação desta dissertação reside na necessidade de explorar e avaliar o desempenho de modelos de *Deep Learning*, com ênfase em arquiteturas baseadas em *LSTM*, para previsão de vendas. A pesquisa busca não apenas aprofundar o entendimento teórico sobre essas técnicas, mas também fornecer conhecimento prático que possa orientar sua implementação em ambientes empresariais. Ao comparar diferentes variações de *LSTM* com métodos consagrados de *Machine Learning*, como *XGBoost* e *LightGBM*, este estudo visa identificar as circunstâncias em que os modelos de *Deep Learning* oferecem uma vantagem competitiva, contribuindo assim para o avanço do conhecimento na área e para a melhoria das práticas de previsão de demanda no varejo.

1.2 PROPOSTA DE PESQUISA

A proposta central desta pesquisa é aprimorar o entendimento e a aplicabilidade de modelos de aprendizado de máquina para o problema de séries temporais no contexto da previsão de demanda no varejo. Este estudo busca complementar o estado da arte na previsão de vendas e demanda, com ênfase no uso de modelos de *Deep Learning* baseados em *LSTM* e suas variantes, e compará-los com *benchmarks XGBoost*,

LightGBM, Regressão Linear e *Average*. Inicialmente, será realizada uma revisão bibliográfica para examinar os principais modelos de previsão baseados em *Deep Learning*, estabelecendo uma base teórica para a pesquisa, etapa fundamental para compreender o estado atual do conhecimento na área e identificar oportunidades de inovação.

Em seguida, a pesquisa se concentrará na implementação de três modelos diferentes de previsão de eventos temporais baseados na arquitetura *LSTM*: *Vanilla LSTM*, *Conv1DLSTM* e *Multiscale CNN-LSTM* com e sem ajuste de hiperparâmetros. Essa seleção permitirá uma comparação abrangente e detalhada entre diferentes abordagens de *LSTM*, avaliando suas capacidades e limitações no contexto de regressão em séries temporais. Por meio dessa análise, buscamos identificar qual modelo ou combinação de modelos oferece a melhor precisão e eficiência na previsão de vendas, e se os modelos *LSTM* são apropriados para a base de dados em estudo. Por fim, a pesquisa resulta em uma análise comparativa do desempenho de cada modelo, observando a eficácia das diferentes abordagens de *Deep Learning* no contexto da base da dados adotada.

1.3 OBJETIVO GERAL

O objetivo geral deste trabalho é realizar uma análise comparativa em termos de desempenho entre três variações de redes neurais com arquitetura *LSTM*, a saber: *Vanilla LSTM*, *Multiscale CNN+LSTM* e *ConvLSTM*, e os modelos clássicos *Average*, *XGBoost* e *LightGBM* - usando e não fine-tuning - na tarefa de previsão de vendas no mercado varejista.

1.3.1 Objetivos Específicos

- Realizar uma revisão bibliográfica sobre os principais modelos de previsão de eventos baseados *Deep Learning*;
- Realizar um pré-processamento no conjunto de dados da farmacêutica *Rossmann*;
- Realizar a seleção de atributos com o algoritmo *Boruta*;

- Gerar previsões de vendas considerando 7 e 30 dias passados, utilizando a base de dados da farmacêutica *Rossmann*;
- Realizar o ajuste dos hiperparâmetros dos modelos utilizando a otimização bayesiana;
- Realizar uma análise comparativa do desempenho de cada modelo em termos de eficiência e aplicabilidade com e sem o ajuste dos hiperparâmetros.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação se organiza da seguinte forma: o Capítulo 2 apresenta os conceitos que a dissertação está se apoiando e o referencial teórico. Em seguida, o Capítulo 3 descreve a metodologia aplicada. A análise dos resultados é, então, discutida no Capítulo 4. O trabalho é concluído no Capítulo 5, resumindo a pesquisa realizada.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 PREVISÃO DE DEMANDA NO VAREJO

2.1.1 O setor varejista

O setor varejista é uma das áreas mais importantes e dinâmicas da economia global, atuando como intermediário essencial entre fabricantes, produtores e consumidores. Além de facilitar o fluxo de bens e serviços, o varejo influencia diretamente a economia e as tendências de consumo, agregando valor à cadeia de suprimentos por meio da seleção de produtos, estratégias de marketing e serviços que melhoram a experiência de compra (ZENTES; MORSCHETT; SCHRAMM-KLEIN, 2012).

Em 2021, as 250 maiores empresas varejistas do mundo registraram vendas totais de US$ 5,6 trilhões, com gigantes como Walmart, Costco e Amazon.com liderando o setor (Deloitte Touche Tohmatsu Limited, 2023). O segmento de produtos de alta rotatividade (FMCG - *Fast-moving consumer goods*) domina o mercado, abrangendo supermercados, lojas de conveniência, drogarias, clubes de compras e lojas online. Essas cadeias de varejo operam múltiplas unidades sob uma única propriedade, centralizando seus processos decisórios (WEITZ; LEVY, 2011). Dado o alto custo fixo e as margens líquidas reduzidas, geralmente entre 2 e 3% das vendas, pequenos aumentos de receita podem ter um impacto significativo nos lucros, tornando o uso eficiente de dados e previsões precisas crucial para a rentabilidade (FISHER; RAMAN, 2018).

A gestão de estoques, especialmente devido à perecibilidade de muitos produtos, é uma função crítica no varejo (CHEN; PANG; PAN, 2014). Um dos maiores desafios é equilibrar a rotatividade dos produtos com a necessidade de minimizar os custos associados à falta de estoque (RANA, 2020). Previsões imprecisas podem causar a indisponibilidade de produtos, levando consumidores a recorrer a concorrentes e prejudicando a reputação da empresa, ou a gerar excesso de estoque (AGNEW; THORNES, 1995). Nesse contexto, o cenário de competição global intensa, o rápido desenvolvimento de novos produtos e a crescente diversidade de itens aumentam a complexidade das previsões de demanda (FISHER *et al.*, 1994). Portanto, a capacidade de realizar previsões acuradas é vital para otimizar decisões de gestão, influenciando

diretamente o sucesso e a sustentabilidade das operações varejistas.

2.1.2 Fatores Influenciadores

Uma previsão de demanda confiável é uma ferramenta fundamental para o sucesso do gerenciamento de empresas do varejo e planejamento. No entanto, a tarefa de estimar com precisão a demanda por produtos é complexa e desafiadora. Segundo Arunraj e Ahrens (2015), a dimensão direta da demanda em lojas de varejo também é dificultada pela dinâmica do varejo, onde os clientes só podem comprar os produtos disponíveis nas prateleiras. Como resultado, a demanda real é frequentemente estimada com base no volume de vendas, o que pode levar à subestimação da demanda em casos de ruptura de estoque, mascarando a demanda real do mercado. Para construir um modelo de previsão de demanda acurado, é crucial considerar os diversos fatores que podem influenciar o comportamento do consumidor e, consequentemente, a demanda por produtos. Entre eles, Arunraj e Ahrens (2015) destacam:

- **Apresentação do produto:** Aspecto visual, embalagem, qualidade, posição na loja, data de validade, tamanho do pacote, etc. É um consenso que a disposição atrativa de produtos na loja e a aderência a determinadas marcas tem relação com a quantidade de vendas.
- **Preço:** Promoções e variações no preço podem estimular o consumo e tornar a demanda mais volátil.
- **Eventos** A demanda pode variar de acordo com feriados e eventos, dependendo das tradições culturais e religiosas, localização geográfica, demografia e eventos anômalos. Lojas situadas próximas a atrações turísticas podem notar uma demanda mais flutuante durante temporadas de feriados e eventos, devido ao fluxo de turistas.
- **Padrões de demanda sazonais e tendências** Dia da semana, dia do mês, mês do ano, estação do ano, trimestre, podem estar relacionados a padrões de consumo.

- **Clima:** Temperatura, precipitação, umidade e vento. Eventos climáticos como frio extremo, neve e chuva podem influenciar as decisões de compra, levando os clientes a permanecerem em casa ou a frequentarem lojas mais próximas.
- **Canibalização e Complementaridade:** A demanda por um produto específico pode ser afetada pela introdução de novos produtos ou promoções de itens relacionados, seja na mesma loja ou entre lojas concorrentes.

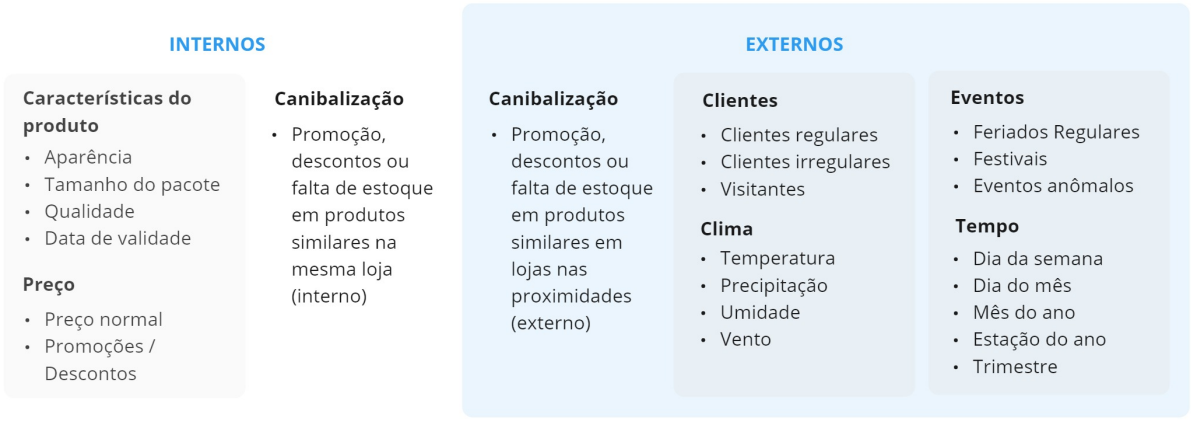
Em 2021, as 250 maiores empresas varejistas do mundo registraram vendas totais de US$ 5,6 trilhões, com gigantes como Walmart, Costco e Amazon.com liderando o setor (Deloitte Touche Tohmatsu Limited, 2023). O segmento de produtos de alta rotatividade (FMCG - *Fast-moving consumer goods*) domina o mercado, abrangendo supermercados, lojas de conveniência, drogarias, clubes de compras e lojas online. Essas cadeias de varejo operam múltiplas unidades sob uma única propriedade, centralizando seus processos decisórios (WEITZ; LEVY, 2011). Dado o alto custo fixo e as margens líquidas reduzidas, geralmente entre 2 e 3% do faturamento, pequenos aumentos de receita podem ter um impacto significativo nos lucros, tornando o uso eficiente de dados e previsões precisas crucial para a rentabilidade (FISHER; RAMAN, 2018).

Nesse contexto, a gestão de estoques, especialmente devido à perecibilidade de muitos produtos, também se apresenta como uma função crítica no varejo (CHEN; PANG; PAN, 2014). Um dos grandes desafios é equilibrar a rotatividade dos produtos com a necessidade de minimizar os custos associados à falta ou excesso de estoque (RANA, 2020). Previsões imprecisas podem causar a indisponibilidade de produtos, levando consumidores a recorrer a concorrentes e prejudicando a reputação da empresa, gerar custos extras de armazenamento ou perda de validade de produtos (AGNEW; THORNES, 1995). Em um cenário de competição global intensa, a constante introdução de novos produtos e a crescente diversidade de itens aumentam a complexidade da tarefa de previsão de demanda (FISHER *et al.*, 1994).

Na Figura 24, uma versão editada do trabalho Arunraj, Ahrens e Fernandes (2016), os fatores que influenciam a demanda são classificados em eventos, clima, preço,

canibalização, complementaridade, sazonalidade, características do produto e número de visitas de clientes. A versão editada inclui complementaridade e agrupa esses fatores em 2 categorias mencionadas pelos autores, a saber: internas e externas. As forças internas englobam o preço e as características do produto. No entanto, efeitos de substituição e canibalização são considerados apenas parcialmente internos, pois podem ocorrer dentro da loja ou entre lojas, como discutido anteriormente. As forças externas incluem eventos (feriados regulares, festivais e férias escolares), clima (temperatura, precipitação), sazonalidade e o número de visitas de clientes, sejam estes habituais, esporádicos ou visitantes especiais, e são essencialmente incontroláveis.

Figura 1 – Fatores influenciadores da demanda.



Fonte: Adaptada de Arunraj, Ahrens e Fernandes (2016).

Vários estudos têm destacado a importância de considerar múltiplos fatores que influenciam a demanda nas previsões de vendas. (DONSELAAR *et al.*, 2015) investigaram como promoções temporárias afetam as vendas de itens perecíveis, sugerindo que a adição de interações entre variáveis poderia melhorar a precisão dos modelos de regressão. No entanto, foi constatado que a incorporação de dados mais detalhados é benéfica apenas quando técnicas mais complexas e avançadas são utilizadas. Este trabalho inicialmente se concentrará em um grupo específico de fatores influenciadores da demanda, sendo eles restritos aos atributos contidos no conjunto de dados disponibilizado na plataforma Kaggle da empresa farmacêutica *Rossmann*.

2.1.3 Granularidade

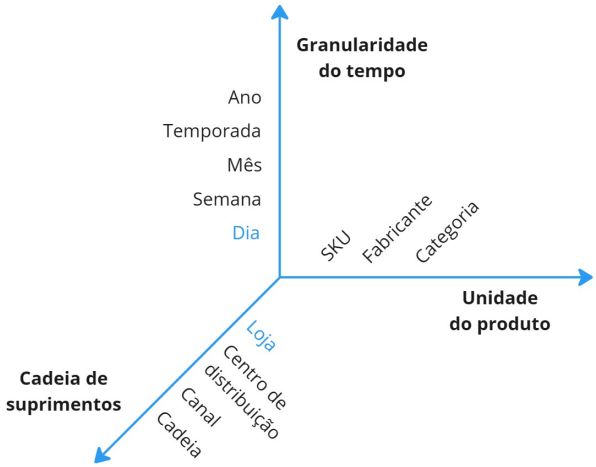
Neste segmento, será apresentado um panorama dos métodos de previsão de demanda existentes. Nesse contexto, é essencial reconhecer que a previsão de demanda é muitas vezes considerada um processo distinto da previsão de vendas devido às possíveis discrepâncias entre demanda e vendas, consequência da disponibilidade dos produtos. Desta forma, a venda não necessariamente reflete a demanda real, uma vez que é limitada a quantidade ofertada de produtos. Contudo, dada a sobreposição de métodos e o foco desta dissertação em dados de vendas quando os produtos estavam disponíveis, não diferenciamos essas duas abordagens aqui. Como definição geral, a demanda refere-se à quantidade de um produto ou serviço específico que um consumidor está disposto a comprar com um preço específico. De acordo com (LEWIS, 1997), a demanda pode ser classificada em demanda dependente ou independente, onde a demanda dependente é dependente de outros produtos ou serviços relacionados.

A previsão de demanda pode ser feita em diferentes níveis, e nesta tese nos concentramos na produção de previsão de demanda no nível da loja. A previsão de demanda é caracterizada por três dimensões, a granularidade do tempo, o nível na hierarquia do produto e a posição na cadeia de suprimentos do varejo. Geralmente, a granularidade do tempo aumenta à medida que passamos de decisões estratégicas até decisões operacionais. A hierarquia de produtos consiste em três diferentes níveis que são nível de SKU, nível de marca e nível de categoria, onde SKU (manutenção de estoque unidade operacional) é o menor nível necessário para decisões operacionais e táticas. Finalmente, o nível da cadeia de abastecimento refere-se ao nível da loja, do centro de distribuição ou da cadeia (FILDES; MA; KOLASSA, 2019). As três diferentes dimensões estão resumidas na Figura 2. Neste estudo, os níveis de previsão de demanda são dia e volume de vendas da loja.

2.2 SÍNTESE DA REVISÃO BIBLIOGRÁFICA

A previsão de demanda é uma área de estudo de extrema relevância, especialmente em contextos que envolvem decisões estratégicas em cadeias de suprimentos, planejamento de produção e alocação de recursos. Diversas abordagens têm sido propostas

Figura 2 – Dimensões da previsão em nível de produto no varejo



Fonte: Adaptada de Fildes, Ma e Kolassa (2019).

na literatura para lidar com o desafio de prever a demanda de diferentes produtos e serviços, utilizando técnicas que variam desde modelos estatísticos clássicos a métodos de aprendizado profundo.

Na revisão bibliográfica, é apresentada uma síntese das principais metodologias e práticas utilizadas na previsão de demanda, com foco na utilização de modelos de aprendizado profundo extraídos de estudos recentes ou relevantes para o tema. O conteúdo é organizado de acordo com o nome original do artigo, conjunto de dados, modelos utilizados, parâmetros de performance e o modelo que obteve melhor desempenho. A Tabela 2.2 resume as contribuições dos estudos analisados:

>p0.95 Abordagens de Aprendizado Profundo para problemas de previsão de séries temporais

---

(LOUREIRO; MIGUéIS; SILVA, 2018)

Título: *Exploring the use of deep neural networks for sales forecasting in fashion retail*

Dados: Dados de venda do varejo de modas

Modelos: Árvore de Decisão, Regressão por Vetores de Suporte, Redes Neurais, Regressão Lasso, Redes Neurais Profundas

Melhor Performance: *Regressão por Vetores de Suporte - R2: 0.923, RMSE: 618.5, MAPE: 0.069, MAE: 253.9, MSE: 39385*

(BANDARA *et al.*, 2019)

Título: *Sales Demand Forecast in E-commerce Using a Long Short-Term Memory Neural Network Methodology*

Dados: Conjunto de dados de vendas próprio

Modelos: *Prophet, SARIMA e LSTM*

Melhor Performance: *LSTM - MAPE: 0.793*

(HU *et al.*, 2019)

Título: *A hybrid deep learning framework with CNN and Bi-directional LSTM for store item demand forecasting*

Dados: Dados de vendas (Store Item Demand - Kaggle)

Modelos: *Gradiente Descendente Stochastico, KNN, Random Forest, XGBoost, Regressão Linear, CNN-LSTM, CNN-BiLSTM*

Melhor Performance: *CNN-BiLSTM*

(RGUIGA; MOUTTAKI; BENHRA, 2020)

Título: *Contribution to sales forecasting based on recurrent neural network in the context of a Moroccan company*

Dados: Não especificado

Modelos: *LSTM, ANN, RNN*

Melhor Performance: *RNN (LSTM)*

(CHEN *et al.*, 2021)

Título: *Sales Forecasting Using Deep Neural Network And SHAP techniques*

Dados: Conjunto de dados de vendas (Walmart - Kaggle)

Modelos: *Deep Neural Network, Support Vector Regression, Prophet, GRU, LSTM, ARIMA*

Melhor Performance: *Prophet + GRU*

(ALMENTERO; LI; BESSE, 2021)

Título: *Forecasting pharmacy purchases orders*

Dados: Conjunto de dados de farmacêutica

Modelos: *SARIMA, Prophet, Regressão Linear, Random Forest, LSTM*

Melhor Performance: *SARIMA*

(DAI; HUANG, 2021)

Título: *A Sales Prediction Method Based on LSTM with Hyper-Parameter Search*

Dados: Rossmann

Modelos: *LSTM, LSTM - Hyper-parameter Search*

Melhor Performance: *LSTM - Hyper-parameter Search*

(JADHAV; KAUNCHI; DANDAWATE, 2021)

Título: *Future Sales Prediction For Indian Products Using Convolutional Neural Network-Long Short Term Memory*

Dados: Conjunto de dados da loja Bigmart

Modelos: *CNN + LSTM*

Melhor Performance: *CNN + LSTM*

(NITHIN *et al.*, 2022)

Título: *Retail Demand Forecasting using CNN-LSTM Model*

Dados: Conjunto de dados próprio

Modelos: *Multilayer Perceptron, Long Short-Term Memory (LSTM), CNN, CNN+LSTM*

Melhor Performance: *CNN+LSTM*

(ECEVIT *et al.*, 2023)

Título: *Short-Term Sales Forecasting Using LSTM and Prophet Based Models in E-Commerce*

Dados: Dados de vendas de e-commerce

Modelos: *SARIMA, Prophet, LSTM*

Melhor Performance: *LSTM - R2: 0.9113, RMSE: 0.0763, MAPE 0.1623*

A Tabela 2.2 sintetiza diversos estudos que exploram a aplicação de técnicas de aprendizado profundo para a previsão de séries temporais em diferentes contextos. Entre os citados, destacam-se os estudos como o de (ECEVIT *et al.*, 2023) e (BANDARA *et al.*, 2019) que demonstraram que o modelo LSTM superou métodos tradicionais como SARIMA e Prophet na previsão de vendas em ambientes de e-commerce, evidenciando a eficácia das *RNNs* na captura de padrões temporais complexos.

Já a combinação de LSTM com outras técnicas, como otimização por enxame de partículas (PSO) e CNNs, mostrou-se promissora. Por exemplo, He, Wu e Si (2022)

utilizou LSTM com PSO, alcançando resultados superiores em comparação com LSTM puro, enquanto Hu *et al.* (2019) demonstrou que um modelo híbrido CNN-BiLSTM ofereceu melhor desempenho na previsão de demanda de itens em lojas.

O estudo de Nithin *et al.* (2022) destacou a eficácia do modelo CNN-LSTM na previsão da demanda no varejo, enquanto Chen *et al.* (2021) combinou Prophet e GRU, obtendo melhor performance na previsão de vendas de dados do Walmart. Por outro lado, Jadhav, kaunchi e Dandawate (2021) demonstraram que a abordagem CNN-LSTM foi particularmente eficaz na previsão de vendas para produtos indianos no Bigmart conjunto de dados.

Já o estudo de Kramar e Alchakov (2023) mostrou que o modelo XGBoost superou outros métodos, incluindo LSTM e SARIMA, na previsão de dados sazonais de tratamento de águas residuais, destacando sua capacidade de lidar com dados complexos e multivariados. Além disso, os modelos LightGBM e XGBoost têm se mostrado especialmente eficazes em cenários onde a rapidez e a precisão são cruciais, como apontado por Wu (2024), que indicou a superioridade desses modelos sobre LSTM na previsão de valores de ações no conjunto de dados S&P 500.

Em resumo, a síntese dos estudos indica que as redes LSTM, especialmente quando integradas a outras técnicas de aprendizado profundo ou métodos de otimização, oferecem vantagens significativas sobre os modelos estatísticos mais clássicos, como ARIMA, SARIMA e SVMs, na previsão de séries temporais. Da mesma forma, modelos como LightGBM e XGBoost destacam-se pela sua robustez e eficiência, sugerindo que a combinação de diferentes abordagens poderá ainda melhorar a precisão e a robustez das previsões em contextos variados.

2.3 SISTEMAS DE APRENDIZADO SUPERVISIONADO

2.3.1 Modelos de Gradient Boosting

Gradient Boosting Machines (GBM) é um algoritmo de aprendizado de máquina fundamentado no conceito de modelos aditivos estatísticos e na técnica de descida de gradiente. O GBM opera ao construir um modelo aditivo em etapas, aplicando a des-

cida de gradiente no espaço das funções, conforme proposto por (FRIEDMAN, 2001). Estes algoritmos são amplamente adotados em aprendizado de máquina devido à sua eficiência, precisão e capacidade de interpretação.

2.3.1.1 LightGBM

Com o rápido aumento no tamanho e diversidade dos dados nos últimos anos, a importância dada às otimizações de algoritmos está crescendo. Por essa razão, como uma alternativa ao algoritmo *Gradient Boosting*, foram desenvolvidos algoritmos como *XGBoost*, *LightGBM* e *Catboost*, que podem ser considerados versões do *Gradient Boosting*. Com esses algoritmos, busca-se alcançar um treinamento mais rápido e maior precisão.

O *LightGBM* é um algoritmo de *boosting* desenvolvido em 2017 como parte do projeto Microsoft DMTK (*Distributed Machine Learning Toolkit*). Em comparação com outros algoritmos de *boosting*, ele possui vantagens como alta velocidade de processamento, capacidade de lidar com grandes volumes de dados, menor uso de recursos (RAM), alta taxa de previsão, suporte para aprendizado paralelo e suporte para aprendizado em GPU. De acordo com o artigo Ke *et al.* (2017), no qual o modelo é apresentado, o *LightGBM* é 20 vezes mais rápido do que outros modelos.

Existem duas estratégias que podem ser usadas no aprendizado de árvores de decisão: a estratégia orientada por nível ou por profundidade (*level-wise or depth-wise*) e a estratégia orientada por folha (*leaf-wise*). Na estratégia orientada por nível, o equilíbrio da árvore é mantido à medida que a árvore cresce. Na estratégia focada na folha, por outro lado, o processo de divisão continua a partir das folhas, o que reduz a perda. Graças a essa característica, o *LightGBM* se diferencia de outros algoritmos de *boosting*. O modelo apresenta menor taxa de erro e aprende mais rapidamente com a estratégia orientada por folha. No entanto, uma estratégia de crescimento orientada por folha pode fazer com que o modelo seja propenso ao superaprendizado em casos onde o número de dados é baixo. Como resultado, a metodologia é mais adequada para uso com grandes quantidades de dados. Além disso, parâmetros como a profundidade da árvore e o número de folhas podem ser otimizados para evitar o superaprendizado.

O *LightGBM* também utiliza duas técnicas diferentes de outros algoritmos. Essas incluem *Gradient-based One-Side Sampling* (GOSS) e *Exclusive Feature Bundling* (EFB), que abordam a questão da quantidade de amostras de dados e variáveis (KE *et al.*, 2017). O GOSS visa reduzir o número de dados enquanto mantém a precisão das árvores de decisão. O *Gradient Boosting* tradicional analisa todas as amostras de dados para calcular o ganho de informação para cada característica, mas o GOSS utiliza apenas os dados importantes. Assim, o número de dados é reduzido sem afetar muito a distribuição dos dados.

Já o EFB visa reduzir o número de variáveis sem prejudicar a taxa de precisão e, consequentemente, aumentar a eficiência do treinamento do modelo. O EFB possui duas etapas de processamento: criar pacotes e mesclar variáveis no mesmo pacote. Com o EFB, características esparsas são combinadas para criar características mais densas. Isso leva à redução da complexidade e a um processo de treinamento mais rápido, com menor consumo de memória. Em resumo, o GOSS reduz o tamanho dos dados para calcular o ganho de informação, omitindo dados que podem ser considerados menos importantes, enquanto o EFB agrega variáveis para reduzir a dimensionalidade. Com essas duas funções, o *LightGBM* aumenta a eficiência do processo de treinamento.

Um dos principais pontos fortes do LightGBM é sua eficiência computacional. Ele é capaz de treinar modelos rapidamente, mesmo em grandes conjuntos de dados, graças a sua implementação otimizada (KE *et al.*, 2017). As técnicas GOSS e EFB são utilizadas para acelerar o treinamento e reduzir a complexidade dos dados, o que é especialmente útil em séries temporais de grande escala. Além disso, o LightGBM lida de forma eficaz com dados categóricos, que são comuns em séries temporais, incluindo variáveis como tipo de dia (dia útil ou fim de semana), estações do ano e feriados. Essa capacidade facilita o processo de engenharia de atributos, permitindo uma conversão direta de categorias em números.

O alto desempenho em previsões é outro ponto positivo do LightGBM. Como outros métodos de *boosting*, ele consegue capturar padrões complexos nos dados, combinando a robustez dos métodos baseados em árvores de decisão com a precisão dos métodos

de *boosting*. Isso resulta em previsões precisas, mesmo em cenários desafiadores. A versatilidade do LightGBM no uso de engenharia de atributos permite a criação de uma variedade de features derivadas, como atributos lag, médias móveis e diferenças entre períodos, que ajudam a capturar dinâmicas temporais complexas. Além disso, o LightGBM inclui mecanismos para controle de overfitting, como a profundidade máxima das árvores e técnicas de regularização, essenciais para garantir que o modelo não se ajuste excessivamente aos dados de treinamento e possa generalizar bem para novos dados.

No entanto, o uso do LightGBM em séries temporais apresenta algumas limitações.De maneira semelhante ao XGBoost, o LightGBM não reconhece intrinsecamente a ordem temporal dos dados, sendo necessário realizar também engenharia de atributos manual para criar variáveis que capturem as dependências temporais. Isso significa que a eficácia do modelo depende fortemente da qualidade dos atributos temporais criados, o que pode demandar tempo e expertise. O LightGBM também pode estar sujeito ao *overfitting*, especialmente em séries temporais com poucos dados ou alta variabilidade. Embora ofereça técnicas de regularização, o risco de *overfitting* persiste, e o ajuste cuidadoso dos hiperparâmetros é necessário para mitigar esse risco.

Por fim, a complexidade no ajuste de hiperparâmetros pode ser um desafio. O desempenho do LightGBM varia significativamente com base na combinação de parâmetros escolhidos, como taxa de aprendizado, número de árvores e profundidade máxima. Encontrar a configuração ideal pode ser um processo complexo e demorado, exigindo técnicas de validação cruzada e otimização.

2.3.1.2 XGBoost

XGBoost (*eXtreme Gradient Boosting*) é uma versão de alto desempenho do algoritmo *Gradient Boosting*, otimizada com várias modificações. Ele foi apresentado no trabalho desenvolvido por Chen e Guestrin (2016a). As características mais importantes do algoritmo são que ele pode alcançar alto poder preditivo, prevenir o superaprendizado, gerenciar dados ausentes e fazer isso rapidamente. Segundo Chen e Guestrin (2016a), o XGBoost é dez vezes mais rápido do que outros algoritmos populares.

O primeiro passo no XGBoost é prever a pontuação inicial. Esta estimativa pode ser qualquer número, pois o resultado correto será alcançado convergindo com as ações que serão tomadas nas etapas seguintes. Este número é 0,5 por padrão. A avaliação dessas estimativas é examinada com os resíduos do modelo. Esses erros são encontrados subtraindo-se o valor estimado do valor observado. O próximo passo é construir uma árvore de decisão que preveja os erros. O objetivo aqui é aprender com os erros e chegar mais perto da previsão correta.

A pontuação de similaridade é calculada para cada ramo da árvore criada. A pontuação de similaridade indica quão bem os dados estão agrupados nos ramos. Após calcular as pontuações de similaridade, a próxima pergunta é se uma previsão melhor pode ser feita. Árvores de todas as possibilidades possíveis são construídas para responder a essa pergunta. As pontuações de similaridade são calculadas para todas elas. O ganho será calculado para decidir qual árvore é melhor.

Enquanto os ramos são avaliados pela pontuação de similaridade, a árvore inteira é avaliada pelo ganho. A primeira árvore é concluída com as operações mencionadas acima (estabelecendo árvores que preveem erros, calculando as pontuações de similaridade e ganho para as árvores, podando e obtendo saídas de modelo). Após a conclusão da primeira árvore, com a primeira previsão, taxa de aprendizado e primeira árvore, a segunda previsão é feita. Dessa forma, as árvores são calculadas e as previsões continuam sendo corrigidas. Esses processos continuarão até que os erros sejam muito baixos ou até que o número especificado de árvores seja alcançado. Por padrão, os ramos são divididos seis vezes.

O uso do XGBoost para séries temporais apresenta alguns desafios. Um dos principais pontos negativos é a necessidade de um esforço considerável na *engenharia de atributos*. O XGBoost não lida nativamente com a estrutura temporal dos dados, o que significa que o sucesso do modelo depende fortemente da criação de atributos apropriados, como lags e diferenças temporais. Este processo pode ser manual e demorado, exigindo conhecimento especializado.

Além disso, o XGBoost não captura diretamente as relações temporais sequenciais, ao contrário de modelos específicos para séries temporais, como ARIMA ou redes

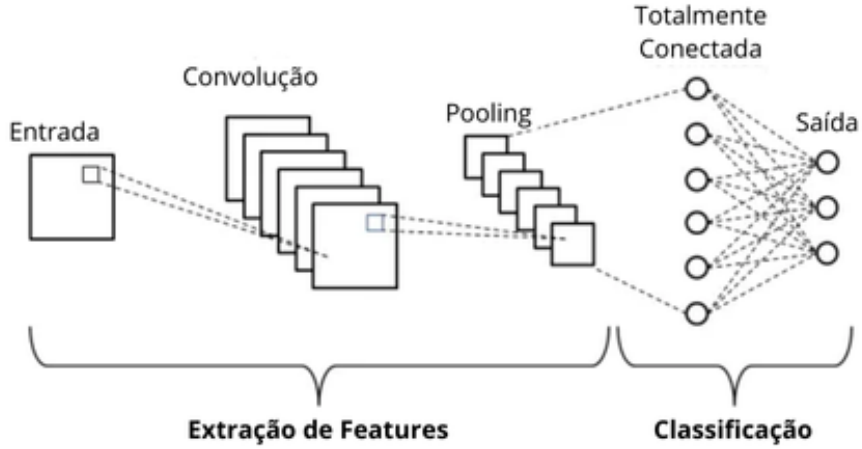
neurais recorrentes (RNNs) (ZHANG *et al.*, 2021). Isso pode limitar a capacidade do XGBoost de capturar certas dinâmicas intrínsecas dos dados de séries temporais, o que pode ser uma desvantagem em aplicações onde essas relações temporais são fundamentais.

Outro desafio é o potencial de overfitting, apesar das opções de regularização do XGBoost (CHEN; GUESTRIN, 2016b). Esse risco é particularmente alto quando o conjunto de dados de treinamento é pequeno ou possui alta variabilidade. Portanto, é necessário um ajuste cuidadoso dos hiperparâmetros para minimizar esse risco. Além disso, embora o XGBoost seja eficaz em lidar com séries temporais univariadas, sua aplicação em problemas multivariados é mais complexa. Lidar simultaneamente com múltiplas fontes de dados temporais requer abordagens específicas, o que pode adicionar complexidade ao processo de modelagem.

2.3.2 Redes Neurais Convolucionais (CNNs)

As Redes Neurais Convolucionais (CNNs), derivadas do estudo do córtex visual em felinos por Hubel e Wiesel (1962), ilustram que muitos neurônios neste córtex respondem exclusivamente a estímulos visuais em áreas restritas do campo visual. Nesse contexto, grupos de células são sensíveis a sub-regiões específicas da imagem de entrada. Dessa forma, a imagem não é processada como um bloco único, mas sim como uma composição de características menores (PATTERSON; GIBSON, 2017).

Figura 3 – Arquitetura de uma CNN.



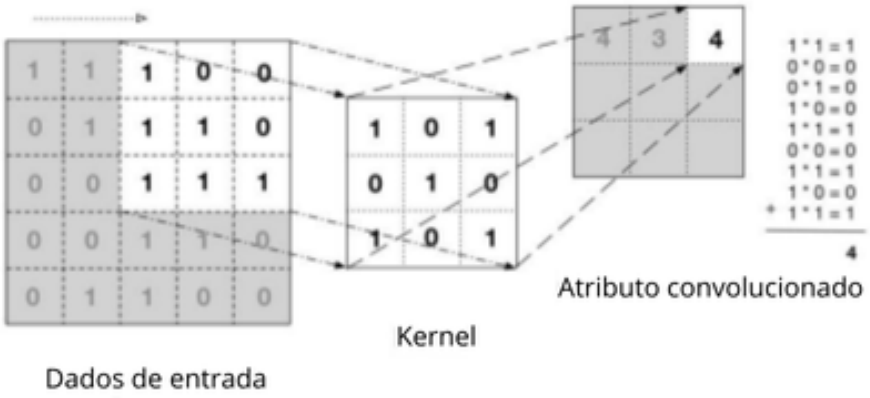
Fonte: Adaptada de Vargas, Carvalho e Vasconcelos (2016)..

A arquitetura CNN, uma variante da Multi-Layer Neural Network (MLNN), substitui camadas totalmente conectadas por uma alternância entre camadas de convolução e de *pooling*, utilizando filtros (kernels) para extrair características locais e uma camada totalmente conectada para classificação (VARGAS; CARVALHO; VASCONCELOS, 2016). A Figura 3 mostra uma visão de uma organização da CNN. Além da camada de entrada, as camadas intermediárias alcançam a extração de características, enquanto a parte final totalmente conectada realiza a classificação.

A principal diferença das redes neurais convolucionais em relação às redes neurais *feed-forward* está na ausência de camadas totalmente conectadas nas partes inicial e central da arquitetura. Camadas totalmente conectadas são utilizadas somente na parte final para produzir a saída, que é a distribuição de probabilidade de classificação. Do ponto de vista computacional, isso resulta em modelos que necessitam de um menor número de pesos, mesmo para uma grande quantidade de camadas, tornando-os mais gerenciáveis do ponto de vista da ocupação de memória. O processo de treinamento é realizado com a *retro-propagation*, assim como nas redes neurais *feed-forward*.

A seguir alguns elementos importantes da arquitetura das CNNs:

Figura 4 – Extração de *feature* usando convolução.



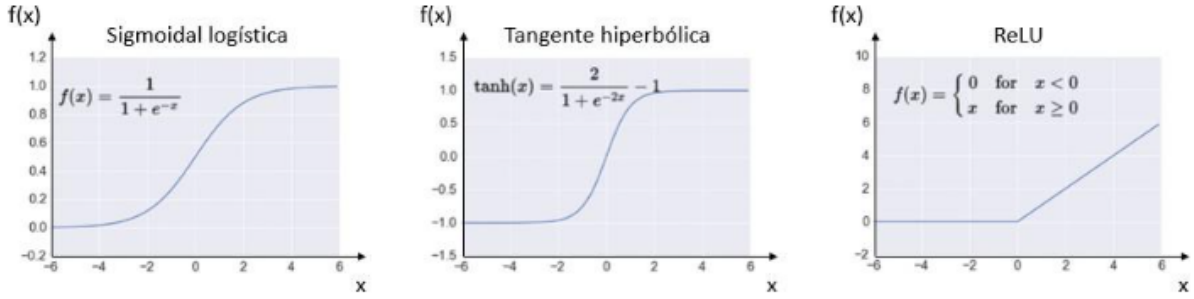
Fonte: Adaptada de Vargas, Carvalho e Vasconcelos (2016).

- **Camada Convolucional**: As camadas de convolução empregam filtros não lineares para explorar os dados de entrada (ou os provenientes da camada imediatamente anterior) de forma sequencial. Durante este processo, para cada posição sobre a entrada, realiza-se a multiplicação entre os elementos do filtro,

ou *kernel*, e os elementos correspondentes na entrada, com a soma dos produtos resultantes fornecendo o valor de saída para aquela posição específica. Este conjunto de valores de saída forma o que é conhecido como mapa de características.

Como mostrado na Figura 4, a operação de convolução é aplicada multiplicando-se um *kernel* por uma área na imagem de entrada (cada valor é multiplicado pelo peso correspondente, em seguida, o resultado é a soma de todas as multiplicações). O resultado é armazenado na matriz de saída chamada mapa de características ou mapa de ativação. Existem múltiplos filtros desse tipo. Uma vez que a entrada foi completamente processada, a rede usa o próximo filtro. É importante notar que a profundidade do mapa de ativação não está relacionada à profundidade da entrada ou do *kernel*, mas sim, é igual ao número de *kernels* aplicados à imagem de entrada.

Figura 5 – Exemplos de funções de ativação.

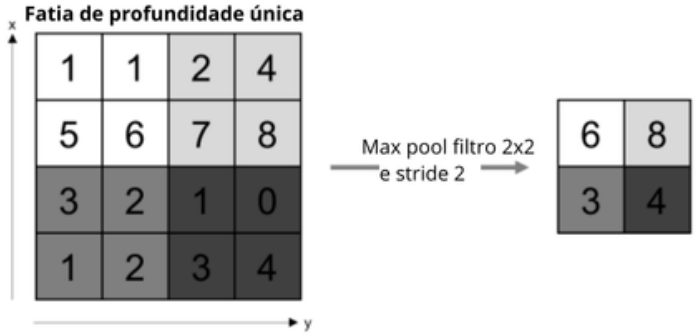


Fonte: Adaptada de Vargas, Carvalho e Vasconcelos (2016).

- **Função de Ativação**: Após a realização da convolução sobre a matriz de entrada e a inclusão de um termo de viés, os valores resultantes são fornecidos a uma função de ativação. Essa função é crucial para introduzir não-linearidade ao modelo, permitindo que a rede neural seja capaz de capturar e aprender uma ampla variedade de padrões e comportamentos nos dados. A função *ReLu* é frequentemente destacada como a mais adequada para uso em Redes Neurais Artificiais devido à sua eficácia em vários contextos. No entanto, outras funções de ativação, como a tangente hiperbólica e a sigmoide — exemplificadas na Figura 5, também são aplicáveis e podem ser empregadas em diversos tipos de

arquiteturas de rede, incluindo tanto as Redes Neurais Multicamadas (MLNN) quanto as Redes Neurais Convolucionais (CNN).

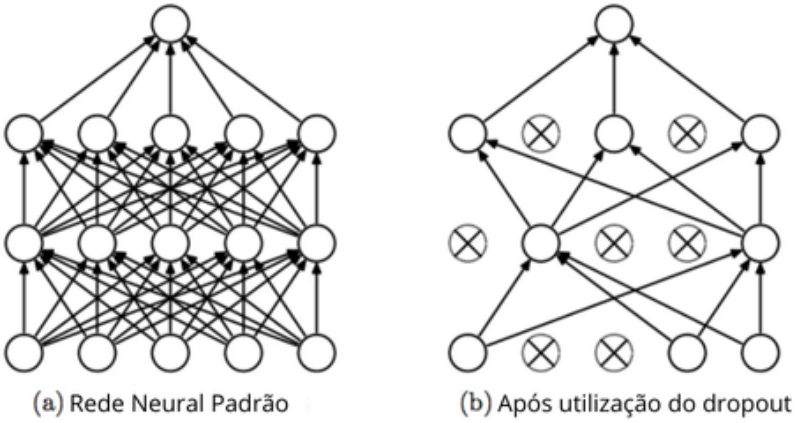
Figura 6 – Exemplo de *Pooling* máximo.



Fonte: Adaptada de Patterson e Gibson (2017).

- **Camada de *Pooling***: As camadas de *pooling* se fundem a características semanticamente similares encontradas no mapa de ativação anterior e ajudam a controlar o *overfitting* (PATTERSON; GIBSON, 2017). A camada mais comum é a de *max-pooling*, que extrai o valor máximo, como mostra a Figura 6.

Figura 7 – Exemplo de rede sem e com *dropout*.



Fonte: Extraída de Srivastava *et al.* (2014).

- **Dropout**: O *dropout* é uma técnica de regularização que procura mitigar o sobreajuste, desativando aleatoriamente neurônios durante o treinamento para promover a aprendizagem de características mais robustas, como exemplificado na Figura 7.

Uma camada totalmente conectada ocupa a maioria dos parâmetros e, portanto, os neurônios desenvolvem co-dependência entre si durante o treinamento, o que reduz o potencial individual de cada neurônio, levando ao ajuste excessivo dos dados de treinamento. A principal vantagem deste método é que ele impede que todos os neurônios em uma camada otimizem sincronicamente seus pesos (SRIVASTAVA *et al.*, 2014).

- **Camada totalmente conectada**: Por fim, a camada totalmente conectada pega a saída da convolução *pooling*, a transforma em uma matriz unidimensional e prediz o melhor rótulo para descrever a imagem. Como em uma rede neural *feed-forward* normal, as entradas para a camada totalmente conectada são multiplicadas pelos pesos e somadas. Então, uma função de ativação é usada para produzir a saída. Os resultados são propagados para a próxima camada totalmente conectada. A última possui um neurônio para cada rótulo de classe e produz a distribuição de probabilidade.

Algumas redes neurais convolucionais, por exemplo, as CNNs unidimensionais (Conv1D) são usadas para aplicações de sinais e séries temporais (FAWAZ *et al.*, 2019), as CNNs bidimensionais (Conv2D) para imagens (nível de pixel) e matrizes (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), e as tridimensionais (Conv3D) para dados volumétricos, como bancos de dados de imagens médicas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). A Conv1D é conhecida por sua menor complexidade computacional devido à sua arquitetura mais simples e compacta (KHAN *et al.*, 2020).

Estudos têm demonstrado a eficácia da Conv1D em várias aplicações, exibindo bom desempenho em tarefas como previsão de estoque e previsão de chuvas (KHAN *et al.*, 2020). A principal vantagem desta técnica sobre as Redes Neurais artificiais anteriores é que a Conv1D extrai características de um sinal considerando informações locais em vez de todo o sinal em cada camada de rede. Isso resulta em um treinamento mais rápido da rede com um número menor de parâmetros treináveis, o que causa menos custo computacional e potência.

Em um modelo Conv1D típico, existem camadas convolucionais unidimensionais que

aplicam filtros aos dados de entrada, extraindo características relevantes por meio de convoluções ao longo da dimensão temporal (BALLESTA *et al.*, 2023). Essas camadas convolucionais são frequentemente seguidas por funções de ativação como ReLU para introduzir a não-linearidade no modelo. Além disso, as camadas de *pooling* unidimensionais são comumente usadas, reduzindo a complexidade computacional enquanto retêm informações essenciais (BALLESTA *et al.*, 2023).

Além disso, para evitar o *overfitting*, camadas de *dropout* são frequentemente incorporadas em arquiteturas Conv1D, que desativam aleatoriamente uma fração de neurônios durante o treinamento para aumentar a generalização . As camadas finais de um modelo Conv1D normalmente incluem uma camada plana para transformar a saída multidimensional em um vetor unidimensional e uma camada softmax para tarefas de classificação, fornecendo as previsões do modelo (BALLESTA *et al.*, 2023).

Em aplicações mais complexas, as arquiteturas Conv1D podem ser aumentadas com componentes adicionais, como redes neurais recorrentes (RNNs) ou unidades de memória de curto prazo (LSTM) para capturar dependências temporais nos dados (ORDóñEZ; ROGGEN, 2016). Essas modificações permitem que os modelos Conv1D manipulem dados sequenciais de forma eficaz, tornando-os adequados para tarefas como reconhecimento de atividades e análise de séries temporais (ORDóñEZ; ROGGEN, 2016).

2.3.3 Redes Neurais Recorrentes (RNNs)

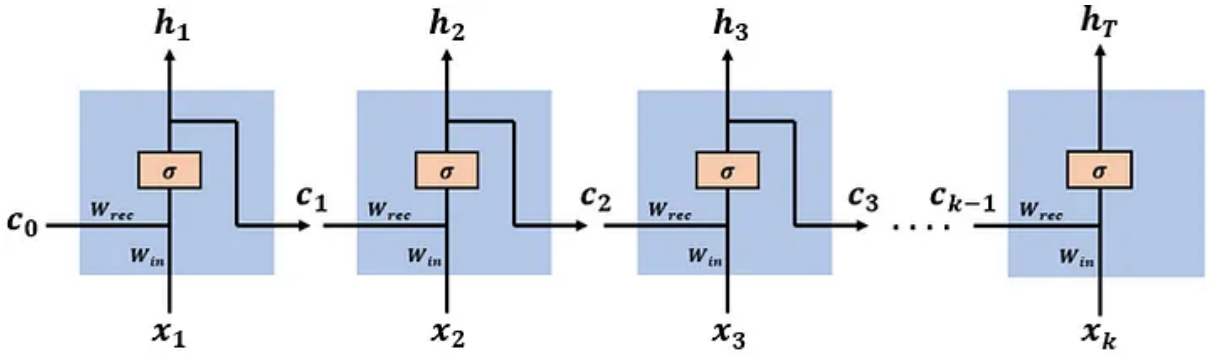
As Redes Neurais Recorrentes (RNNs) são uma classe de redes neurais projetadas para processar sequências de dados, capturando dinâmicas temporais e dependências de longo prazo. Caracterizam-se por suas conexões de retroalimentação, permitindo a um neurônio enviar sua saída de volta a si mesmo, criando uma forma de "memória"que retém informações de cálculos anteriores.

Uma RNN é um tipo especial de RNA adaptada para funcionar com dados de séries temporais ou dados que envolvem sequências. Ela é treinada para processar e converter uma entrada de dados sequencial em uma saída de dados sequencial específica. Dados sequenciais são dados como palavras, frases, dados de séries temporais onde

os componentes sequenciais se inter-relacionam com base em regras complexas de semântica e sintaxe.

Essa capacidade é ilustrada na Figura 8, mostrando um neurônio recorrente e sua representação ao longo do tempo.

Figura 8 – RNN simples de camada oculta única com uma única sequência de saída.



Fonte: Extraída de Houdt, Mosquera e Nápoles (2020).

A rede possui uma sequência de vetores de entrada $[x(1), x(2), \ldots, x(k)]$, no passo de tempo t a rede possui um vetor de entrada $x(t)$. As informações passadas e o conhecimento aprendido são codificados nos vetores de estado da rede $[c(1), c(2), \ldots, c(k-1)]$, no passo de tempo t a rede tem um vetor de estado de entrada $c(t-1)$. O vetor de entrada $x(t)$ e o vetor de estado $c(t-1)$ são concatenados para compor o vetor de entrada completo na etapa de tempo t, $[c(t-1), x(t)]$. A função sigmóide (σ) é usada como função de ativação na camada oculta.

A rede possui duas matrizes de peso: W_rec e W_in conectando $c(t-1)$ e $x(t)$, ou seja, $[c(t-1), x(t)]$, à camada oculta. Para simplificar, foi deixado de fora os vetores de polarização (b). A rede gera um único vetor no último intervalo de tempo h_T. Depois que o RNN gera o vetor de predição $h(k)$, é possível calcular o erro de predição $E(k)$ e utiliza-se o algoritmo *Back Propagation Through Time* (WERBOS, 1990) para calcular o gradiente, possibilitando a atualização dos pesos da rede ($W = [W_rec, W_in]$).

Porém, um problema comum neste tipo de algoritmo é o problema do desaparecimento do gradiente (do inglês, *Vanishing Gradient Problem*. Essencialmente, cada neurônio que participou do cálculo da saída, associada a uma função de custo, deveria ter seu peso atualizado para minimizar esse erro. E o que acontece com as RNNs é

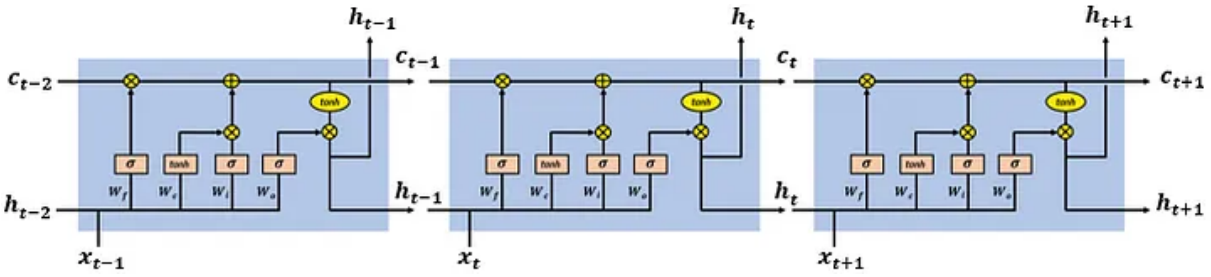
que não foram apenas os neurônios diretamente abaixo dessa camada de saída que contribuíram, mas todos os neurônios de muito tempo atrás. Então, é necessário propagar o erro de volta no tempo até esses neurônios. Caso os valores de W_rec sejam muito pequenos, mais difícil será para a rede atualizar os pesos e mais tempo levará para chegar ao resultado final.

Assim, embora as RNNs sejam eficazes para aprender dependências em curtos períodos temporais, elas enfrentam dificuldades com dependências de longo prazo, situação para a qual as *Long Short-Term Memory networks* (LSTMs) são recomendadas, como será discutido na próxima seção.

2.3.4 Redes do Tipo *Long Short-Term Memory* (LSTM)

O modelo LSTM representa um avanço significativo no âmbito das redes neurais recorrentes, projetado especificamente para solucionar os problemas de desaparecimento ou explosão de gradientes, que comumente surgem durante o aprendizado de dependências de longo prazo, inclusive em casos de longos intervalos temporais (HOCHREITER; SCHMIDHUBER, 1996).

Figura 9 – Arquitetura de uma LSTM tradicional.



Fonte: Extraída de Hochreiter e Schmidhuber (1997).

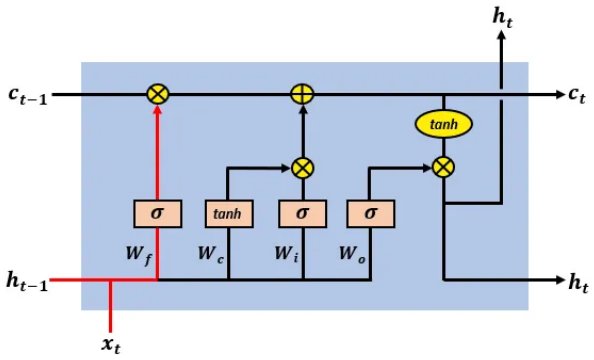
Essencialmente, as células do LSTM funcionam como redes recorrentes autônomas, dotadas de uma arquitetura inovadora que integra mecanismos adicionais como o portão de entrada e o portão de saída. Estas conexões autorecorrentes indicam um *feedback* com um intervalo de um passo temporal. Um bloco (ou célula) LSTM padrão é composto por um portão de entrada (*input gate*), um portão de saída (*output gate*) e um portão de esquecimento (*forget gate*). Este último, introduzido posteriormente por (GERS; SCHMIDHUBER; CUMMINS, 2000), permite que a rede reinicie seu estado,

facilitando a gestão da informação armazenada por intervalos temporais arbitrários através dos três portões, que regulam o fluxo de informações da célula.

A arquitetura de uma LSTM tradicional, ilustrada na Figura 9, é composta por um conjunto de blocos de memória, cada um contendo um estado de célula (c_t) e um estado oculto (h_t), além de três portões: o portão de entrada (i_t), o portão de esquecimento (f_t) e o portão de saída (o_t). Esses portões controlam o fluxo de informações dentro do bloco, permitindo que a LSTM armazene, atualize e recupere informações ao longo do tempo de forma seletiva. Uma rede LSTM possui um vetor de entrada $[h(t-1), x(t)]$ no intervalo de tempo t. O estado da célula da rede é denotado por $c(t)$. Os vetores de saída que passam pela rede entre intervalos de tempo consecutivos t, $t+1$ são denotados por $h(t)$.

Além disso, uma rede LSTM possui três portas que atualizam e controlam os estados da célula; estas são a porta de esquecimento (*forget gate*), a porta de entrada (*input gate*) e a porta de saída (*output gate*). As portas usam tangente hiperbólica (*tanh*) e funções de ativação sigmóide (σ). Além disso, a rede LSTM possui as matrizes de pesos W_f, W_c, W_i e W_o associadas, respectivamente, à porta de esquecimento, estados, entrada e saída. O funcionamento de um bloco de memória LSTM pode ser descrito pelas seguintes etapas:

Figura 10 – A atualização da porta de esquecimento do estado da célula da LSTM.



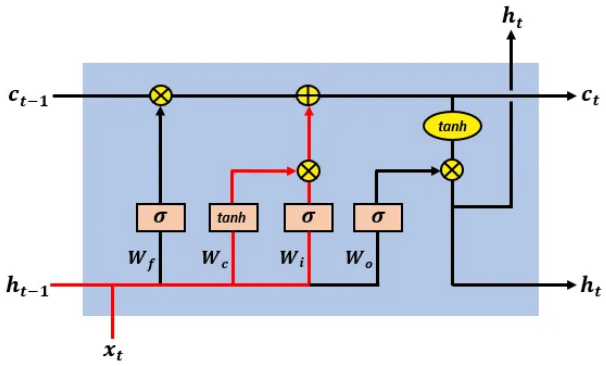
Fonte: Extraída de (HOCHREITER; SCHMIDHUBER, 1997).

1. **Porta de Esquecimento:** A porta de esquecimento controla quais informações no estado da célula devem ser esquecidas, dadas as novas informações que

entraram na rede. A Figura 10 e a equação 2.1 destacam o fluxo de operação dessa porta.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t]) \tag{2.1}$$

Figura 11 – Operação da porta de entrada do estado da célula da LSTM.

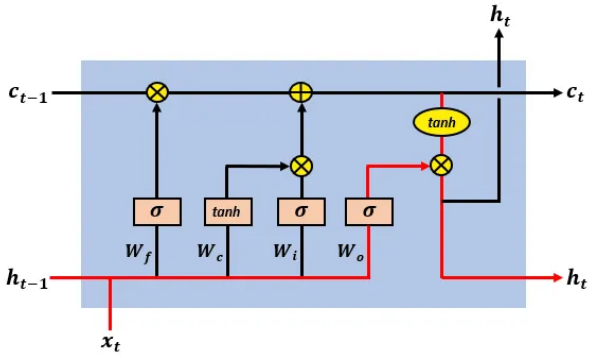


Fonte: Extraída de Hochreiter e Schmidhuber (1997).

2. **Porta de Entrada:** A porta de entrada controla quais novas informações serão codificadas no estado da célula (c), dadas as novas informações de entrada. A Figura 11 e a equação 2.2 ilustram essa operação.

$$i_t = tanh(W_c \cdot [h_{t-1}, x_t] \otimes \sigma(W_i \cdot [h_{t-1}, x_t]) \tag{2.2}$$

Figura 12 – Operação da porta de saída do estado da célula da LSTM.



Fonte: Extraída de Hochreiter e Schmidhuber (1997).

3. **Porta de Saída:** A porta de saída controla quais informações codificadas no estado da célula são enviadas para a rede como entrada no intervalo de tempo

seguinte, isso é feito através do vetor de saída $h(t)$, apresentado nas equações 2.3 e 2.4. A Figura 12 apresenta o fluxo de operação desta porta

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t]) \tag{2.3}$$

$$h_t = o_t \otimes tanh(c_t) \tag{2.4}$$

A introdução do portão de esquecimento é uma característica distintiva do LSTM, permitindo que a rede decida ativamente quais informações devem ser preservadas ou descartadas ao longo do tempo. Isso torna as LSTMs particularmente adequadas para aplicações em análise de séries temporais, processamento de linguagem natural, reconhecimento de fala e outras tarefas que envolvem dependências temporais complexas e longas.

Embora o LSTM padrão já apresente um desempenho notável, pesquisas continuam explorando maneiras de aprimorar essa arquitetura. Tais investigações revelam que, apesar da eficácia do modelo, ainda existem possibilidades de melhorias teóricas e práticas, demonstrando a robustez e a adaptabilidade do LSTM em uma ampla gama de aplicações.

2.3.5 ConvLSTM

A necessidade de compreender as dependências temporais em dados sequenciais, especialmente em imagens multitemporais, levou ao desenvolvimento da Rede Neural Convolucional de Longo Curto Prazo (ConvLSTM) por Shi *et al.* (2022). Introduzida como uma solução para capturar tanto as dependências temporais quanto as características espaço-espectrais dos dados, a ConvLSTM representa uma fusão inovadora entre a LSTM e as CNNs.

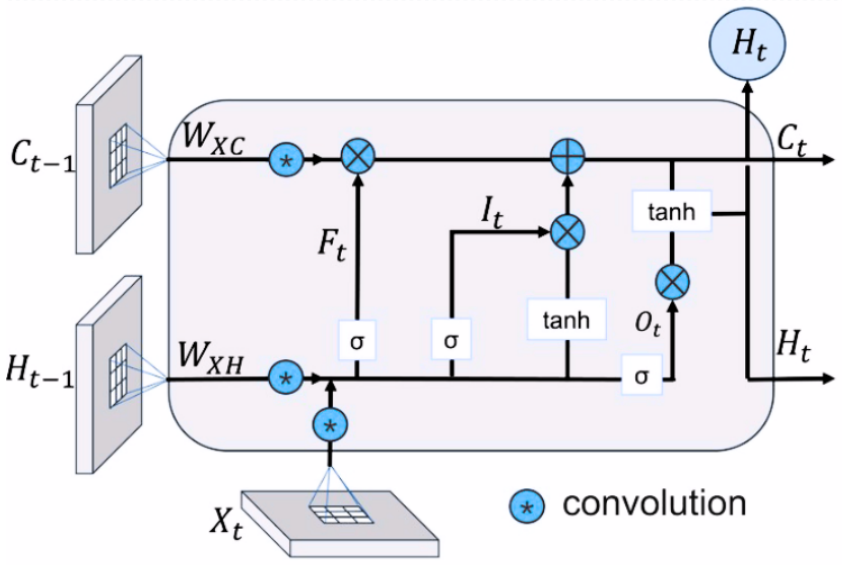
A ConvLSTM, apresentada e melhor detalhada em (SHI *et al.*, 2015), se distingue por incluir uma camada recorrente que opera de maneira similar à LSTM tradicional. A principal diferença reside no fato de que, na ConvLSTM, as operações padrão de

multiplicação de matrizes internas próprias da LSTM são trocadas por operações de convolução.

A ConvLSTM foi projetada para abordar limitações encontradas em CNNs tradicionais e LSTMs quando aplicadas separadamente. As CNNs são altamente eficazes na extração de características espaço-espectrais em imagens, aproveitando sua capacidade de aprender filtros que respondem a aspectos específicos das imagens, como bordas ou texturas. No entanto, as CNNs assumem que todas as entradas são independentes, uma premissa que não se mantém em tarefas onde a sequência temporal dos dados é crucial.

As operações fundamentais da ConvLSTM, ilustradas na Figura 13, podem ser descritas pelas equações 2.5, 2.6, 2.7, 2.8 e 2.9, onde $*$ denota a convolução e $\otimes$ denota o produto da matriz binária.

Figura 13 – Estrutura interna da ConvLSTM.



Fonte: Extraída de Shi *et al.* (2022)..

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \otimes C_{t-1} + b_i) \tag{2.5}$$

$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \otimes C_{t-1} + b_f) \tag{2.6}$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \tag{2.7}$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \otimes C_t + b_o) \tag{2.8}$$

$$H_t = o_t \otimes \tanh(C_t) \tag{2.9}$$

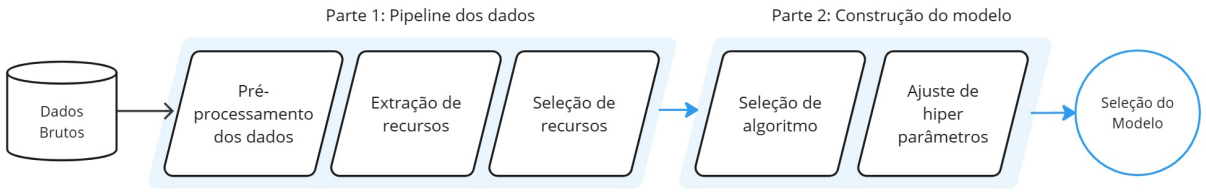
Essas equações demonstram como a ConvLSTM processa dados sequenciais, permitindo a extração de características espaço-espectrais enquanto leva em conta a dependência temporal. Isso a torna particularmente adequada para tarefas que requerem análise espaço-temporal detalhadas, como previsão meteorológica, análise de séries temporais multivariadas e detecção de mudanças com imagens hiperespectrais.

3 METODOLOGIA

Quemy (2019) descreve o processo padrão de aprendizado de máquina como sendo dividido em duas partes principais: a *pipeline* de dados e a construção do modelo. A *pipeline* de dados envolve a escolha e a aplicação de técnicas adequadas para preparar os dados de forma que possam ser efetivamente utilizados por um algoritmo de aprendizado de máquina; enquanto a construção do modelo se concentra na seleção do algoritmo de aprendizado de máquina apropriado e na configuração de seus parâmetros para alcançar um desempenho ótimo, conforme determinado pelas métricas de avaliação selecionadas.

Dentro da *pipeline* de dados, as fases comuns incluem o pré-processamento dos dados, a extração e a seleção de características. Já a construção do modelo engloba a escolha do algoritmo e o ajuste fino dos parâmetros. A Figura 14 ilustra o modelo de fluxo proposto:

Figura 14 – Otimização em dois estágios para fluxo de trabalho de aprendizado de máquina.



Fonte: Adaptada de Quemy (2019).

Neste capítulo, inicialmente serão apresentados o conjunto de dados da farmacêutica Rossmann, descrevendo suas principais características, e o pré-processamento efetuado na base de dados. Posteriormente, serão detalhados os dados que serão usados para o desenvolvimento do modelo de previsão de vendas.

3.1 CONJUNTO DE DADOS

O conjunto de dados fornecido pela *Retail Consult* e disponível em <https://www.kaggle.com/datasets/pratyushakar/rossmann-store-sales>, contém o registro de vendas de 1.115 estabelecimentos pertencentes a rede farmacêutica alemã *Rossmann*. Os dados históricos compreendem de 01 de janeiro de 2013 a 31 de julho de 2015 e são

compostos por três arquivos contendo: *train*, *test* e *store*. A composição e a descrição dos atributos dos dados relacionados ao treinamento e lojas são descritos na Tabela 1.

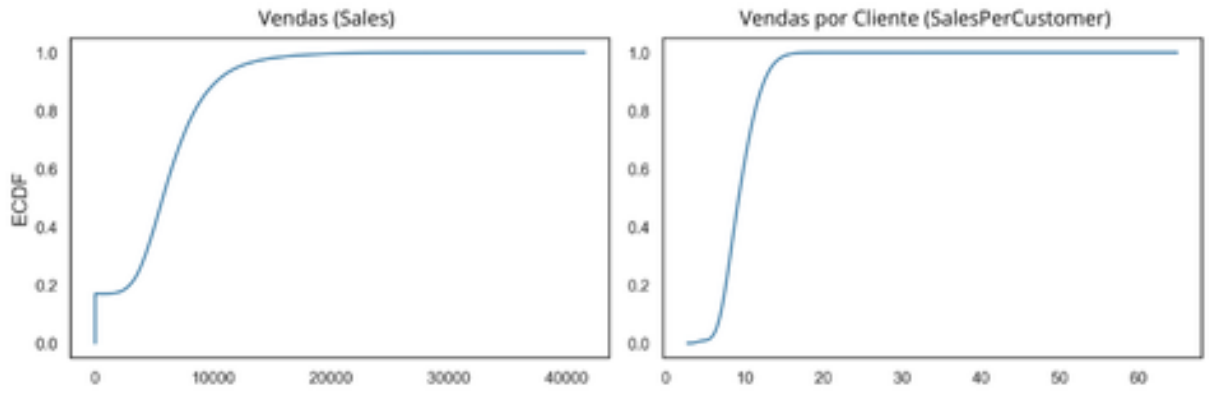
Tabela 1 – Variáveis do conjunto de dados Rossmann.

Variável	**Descrição**
Store	Um identificador único para cada loja
Store Type	Diferencia entre 4 diferentes modelos de loja: a, b, c, d
Assortment	Descreve o nível do sortimento: a = básico, b = extra, c = estendido
Competition Distance	Distância em metros até a loja concorrente mais próxima
Competition Open Since (Month ou Year)	Ano e mês aproximados em que o concorrente mais próximo foi inaugurado
Promo2	Promoção contínua e consecutiva para algumas lojas: 0 = a loja não participa, 1 = a loja participa
Promo2 Since (Week ou Year)	O ano e a semana em que a loja começou a participar da Promo2
Promo Interval	Os intervalos consecutivos em que a Promo2 é iniciada, nomeando os meses em que a promoção é renovada. Por exemplo, "Fev, Mai, Ago, Nov"indica que cada rodada começa em fevereiro, maio, agosto, novembro de qualquer ano para aquela loja
Day Of Week	Dia da semana
Date	Uma data formatada como AAAA-MM-DD
Sales	O faturamento de qualquer dia (variável alvo)
Customers	O número de clientes em um determinado dia
Open	Indicador se a loja estava aberta: 0 = fechada, 1 = aberta
Promo	Indica se a loja está realizando uma promoção naquele dia
State Holiday	Indica um feriado estadual: a = feriado público, b = feriado de Páscoa, c = Natal, 0 = nenhum
School Holiday	Indica um feriado escolar: 1 = feriado escolar, 0 = nenhum

3.2 ANÁLISE EXPLORATÓRIA DE DADOS

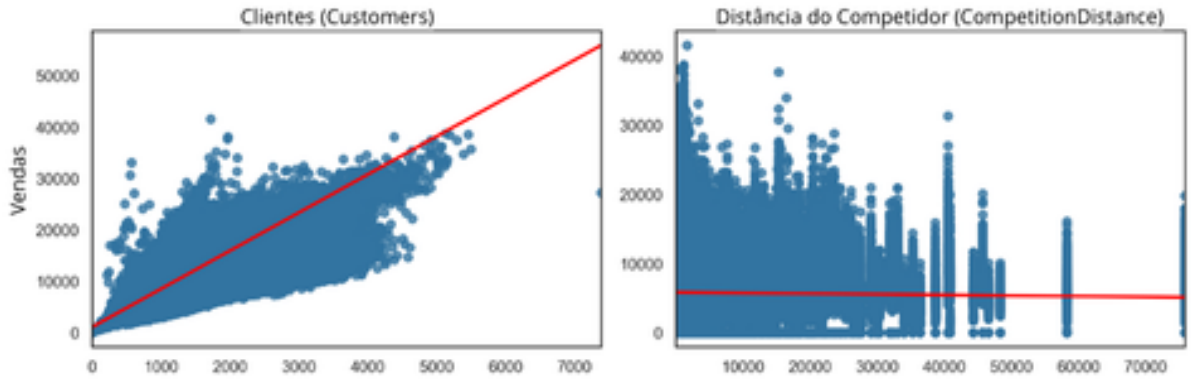
A fim de obter uma visão inicial dos dados, a variável *Sales* foi analisada utilizando a Função de Distribuição Cumulativa Empírica (*ECDF - Empirical Cumulative Distribution Function*), que mostra a distribuição dos dados de vendas em ordem crescente. Além disso, a variável *SalePerCustomer*, que representa a razão entre *Sales* e *Customers*, foi criada para examinar o padrão de vendas por cliente. Esta variável também foi explorada utilizando a ECDF. A Figura 15 apresenta os resultados dessas análises.

Figura 15 – Função de Distribuição Cumulativa Empírica para *Sales* e *SalesPerCustomer*.



Fonte: Elaborada pela autora (2024).

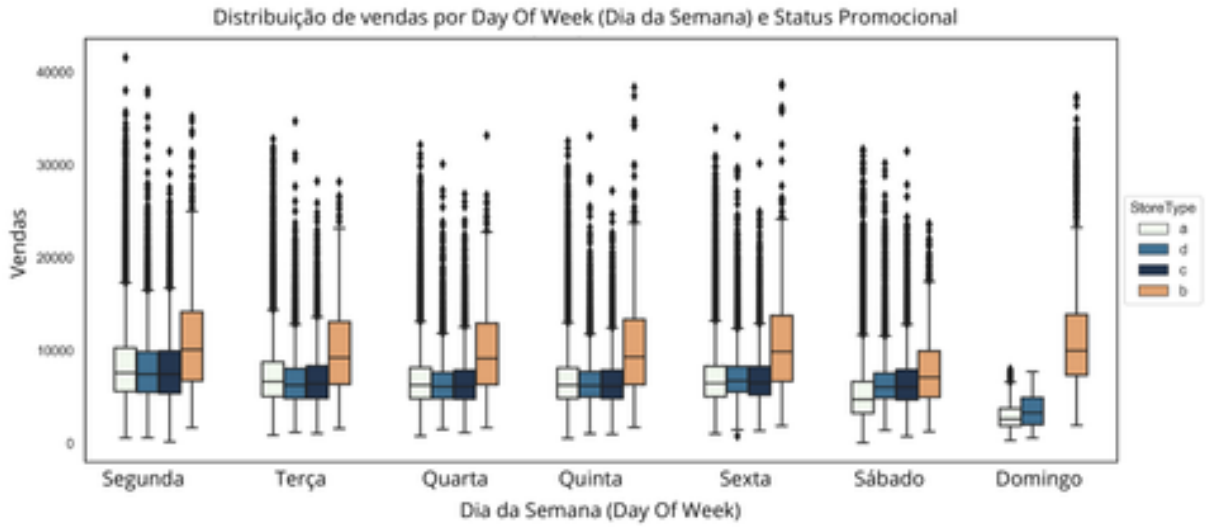
Figura 16 – Gráficos de dispersão ilustrando a relação entre clientes e distância da loja ao concorrente mais próximo, em relação a variável de alvo vendas.



Fonte: Elaborada pela autora (2024).

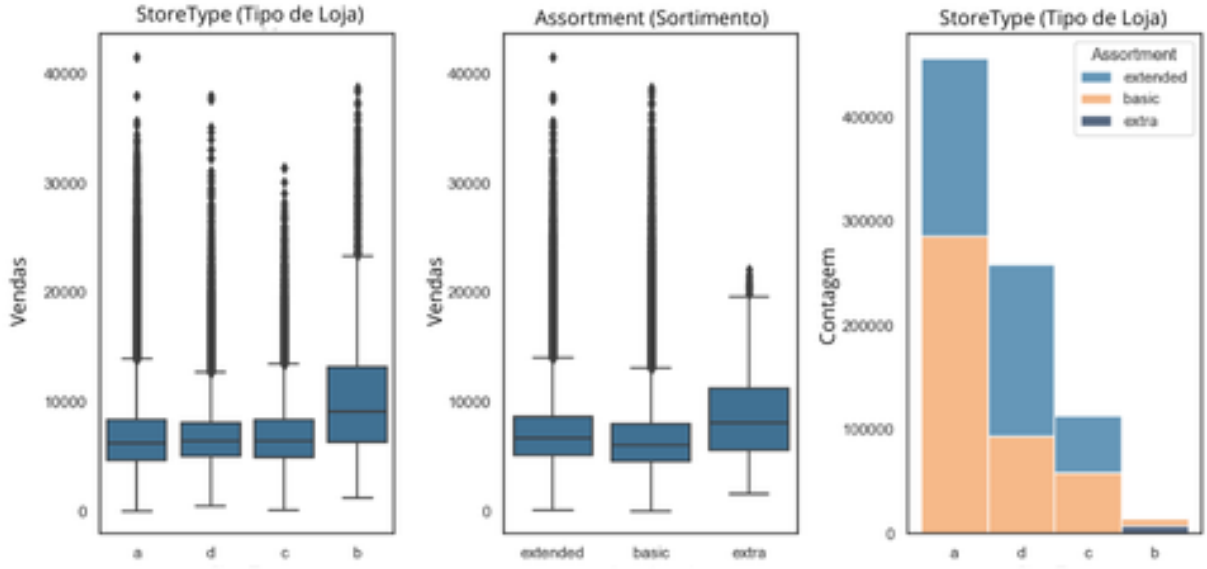
Analisando os gráficos da Figura 15, que mostram a distribuição das vendas no eixo Y e a quantidade de vendas no eixo X, é possível inferir que a maior parte da receita das lojas provém de um número relativamente pequeno de transações de alto valor, enquanto a maioria dos clientes realiza compras de menor valor, uma característica típica do varejo. Além disso, observa-se que aproximadamente 20% dos dados correspondem a casos com vendas por cliente iguais a zero, os quais serão tratados posteriormente. Já o gráfico da variável *SalesPerCustomer* revela que a maioria dos clientes gasta uma quantia relativamente pequena, com um número reduzido de clientes contribuindo significativamente para as vendas totais. Isso confirma o padrão observado no primeiro gráfico e reforça a importância das transações de alto valor para a receita total das lojas.

Figura 17 – Distribuição das vendas nos dia da semana por tipo de loja.



Fonte: Elaborada pela autora (2024).

Figura 18 – Relação entre variedade, tipo de loja e vendas.



Fonte: Elaborada pela autora (2024).

A Figura 17 apresenta a distribuição das vendas ao longo dos dias da semana, segmentada por tipo de loja. Observa-se que os valores médios de vendas diárias são maiores nas segundas-feiras (*monday*) e nas sextas-feiras (*friday*). No entanto, apenas 3% das lojas abrem aos domingos, o que justifica o baixo volume de vendas registrado nesse dia. Além disso, o gráfico revela que as lojas do tipo *c* não operam aos domingos,

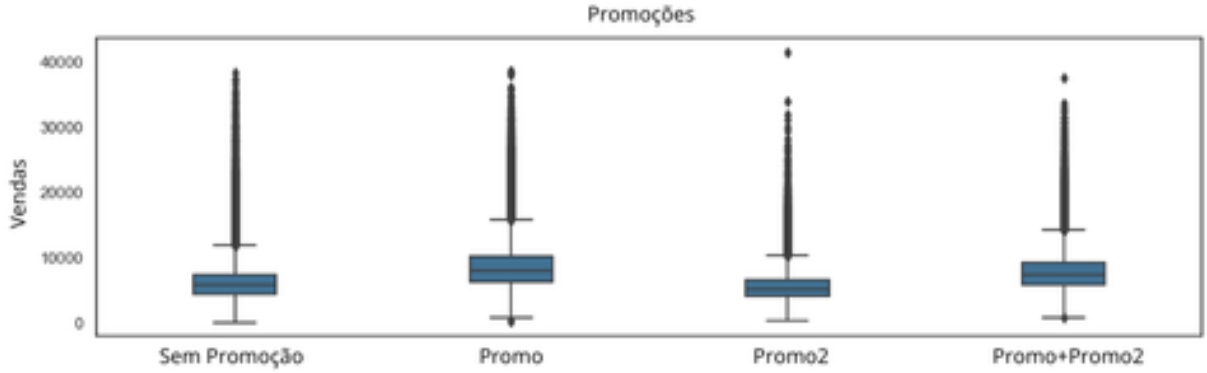
enquanto as lojas do tipo *a*, *b*, e *d* apresentam variações nas vendas ao longo da semana. Outro ponto importante é que as lojas do tipo *b* tendem a ter um desempenho de vendas mais elevado ao longo de toda a semana, destacando-se em relação aos outros tipos de loja, especialmente nas quintas-feiras e sábados. Já as lojas do tipo *a*, *c* e *d*, mostram uma distribuição mais uniforme de vendas, com um leve aumento observado nas sextas-feiras e sábados.

A Figura 18 apresenta a relação entre o tipo de loja (*StoreType*), o nível de sortimento (*Assortment*) e as vendas (*Sales*). No primeiro gráfico, observa-se que o *StoreType b* possui a maior mediana de vendas, sugerindo que, embora tenha um número menor de lojas em comparação ao *StoreType a*, essas lojas geram vendas de maior valor em média. O *StoreType a* exibe uma distribuição de vendas mais ampla, com muitos outliers, indicando uma variação significativa nas vendas entre as lojas desse tipo. O *StoreType d* também mostra uma mediana de vendas relativamente alta, sugerindo um bom desempenho em termos de vendas. O segundo gráfico, que analisa as vendas por nível de sortimento, destaca-se que as lojas com sortimento *extra* têm a maior mediana de vendas, indicando que esse sortimento está associado a transações de maior valor. O terceiro gráfico revela que o *StoreType b* é o único que oferece o sortimento *extra*, enquanto os outros tipos de loja (*a*, *c*, e *d*) não possuem esse sortimento, concentrando-se mais nos sortimentos *basic* e *extended*. Esses achados sugerem que tanto o tipo de loja quanto o nível de sortimento influenciam diretamente o valor das vendas, com o *StoreType b*, que oferece o sortimento *extra*, se destacando em termos de geração de receita por venda.

A Figura 19 ilustra como diferentes status promocionais afetam as vendas diárias. No gráfico, são apresentados os valores diários de vendas para os casos sem promoção (*No Promo*), com promoção (*Promo*), com a segunda promoção (*Promo2*) e a combinação de ambas as promoções (*Promo+Promo2*). Observa-se que os dias com *Promo* apresentam um volume médio de vendas diário maior do que os dias sem promoções, com valores superiores em todos os quartis. No entanto, a diferença na mediana de vendas diárias entre os dias com e sem *Promo2* é mínima, com os dias sem *Promo2* apresentando ligeiramente vendas diárias mais altas. Além disso, quando as duas promoções são combinadas (*Promo+Promo2*), as vendas médias diárias mostram um

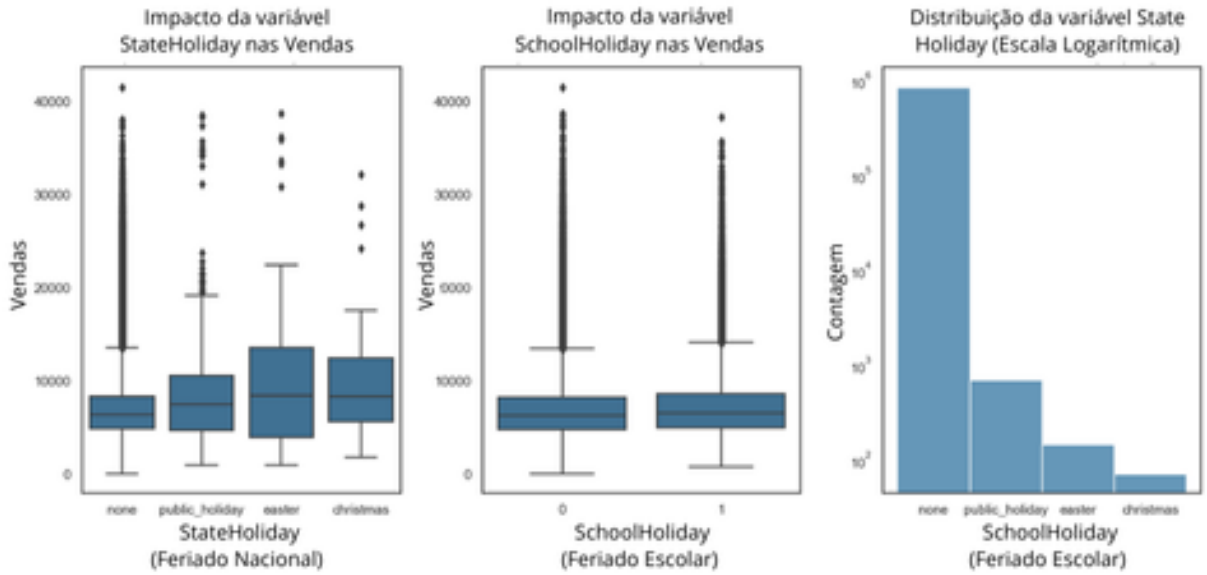
aumento, indicando que a combinação de promoções resulta em um maior ticket médio.

Figura 19 – Análise da variáveis de promoção.



Fonte: Elaborada pela autora (2024).

Figura 20 – Distribuição das vendas durante a ocorrência de feriados escolares e estaduais.



Fonte: Elaborada pela autora (2024).

A relação entre as vendas e os consumidores durante a ocorrência de feriados escolares e estaduais pode ser ilustrada pela Figura 20. Durante os feriados estaduais, observa-se que os valores médios de vendas diárias são superiores aos dias normais, com destaque para a Páscoa (*StateHoliday=easter*), que registra o maior faturamento médio. Esse aumento pode ser atribuído ao fato de que feriados como a Páscoa incentivam o consumo, talvez devido a promoções especiais ou ao aumento na demanda por produtos específicos. No entanto, o impacto dos feriados escolares nas vendas parece ser marginal, conforme indicado pelo segundo gráfico, onde as vendas médias diárias

são semelhantes nos dias com e sem feriados escolares. Isso sugere que o público-alvo das lojas não é significativamente influenciado por esses feriados. O terceiro gráfico, que mostra a distribuição dos tipos de feriados em escala logarítmica, destaca que a maioria dos dias no conjunto de dados são dias normais, sem feriados. No entanto, mesmo com uma ocorrência menor, os feriados estaduais, especialmente a Páscoa, têm um impacto desproporcionalmente maior nas vendas, o que pode ser um indicativo de um padrão de compra apresentado pelos consumidores durante esses períodos.

3.3 PRÉ-PROCESSAMENTO DOS DADOS

Nesta seção, são detalhadas as etapas de pré-processamento aplicadas ao conjunto de dados bruto para preparar os dados para os modelos preditivos. As transformações realizadas asseguram que os dados estejam em um formato adequado, limpo e consistente, facilitando a aplicação dos algoritmos de aprendizado de máquina.

3.3.1 Dados Ausentes

A presença de dados ausentes em uma base de dados pode levar a análises e previsões imprecisas, já que informações essenciais podem estar faltando. Por isso, é crucial tratar os valores ausentes de maneira apropriada. No caso da variável *competition distance*, os valores faltantes foram substituídos por um valor elevado (100.000), que representa a ausência de concorrentes próximos. Essa abordagem permite que o modelo interprete corretamente os casos onde não há competição relevante, evitando assim interpretações errôneas.

Para outras colunas, como *competition open since month*, *competition open since year*, *promo2 since year* e *promo2 since week*, os valores ausentes foram preenchidos com base nas informações de data do próprio registro. Essa técnica assegura a consistência temporal dos dados, evitando que dados anômalos ou inconsistentes afetem a qualidade das análises.

3.3.2 Enriquecimento de Dados

O enriquecimento de dados envolve a criação de novas variáveis que podem fornecer *insights* adicionais. No contexto apresentado, variáveis derivadas como *competition since* e *promo2 since* foram calculadas para indicar o tempo desde a abertura da concorrência e a implementação da *Promo2*, respectivamente. Essas variáveis são importantes porque permitem medir o impacto temporal desses fatores sobre as vendas.

Além disso, características temporais como dia, mês, ano, semana do ano e se o dia é um dia útil foram extraídas da coluna *Date*. A extração dessas características é fundamental para capturar padrões sazonais e flutuações periódicas, que são cruciais para previsões precisas em séries temporais.

3.3.3 Codificação e Normalização

Muitas vezes, variáveis categóricas precisam ser convertidas em um formato que possa ser utilizado por algoritmos de aprendizado de máquina. Neste trabalho, as variáveis *year week* e *states* foram codificadas utilizando o *Label Encoding*, enquanto a variável *assortment* foi codificada ordinalmente. A codificação transforma dados categóricos em números que o modelo pode entender e processar.

A normalização, por outro lado, é crucial para garantir que todas as variáveis estejam na mesma escala. Isso é importante porque variáveis com escalas diferentes podem distorcer o desempenho do modelo, dando mais importância a variáveis com maior amplitude. A técnica *MinMaxScaler*foi aplicada para normalizar variáveis como *promo2 since year*, *competition open since year*, *year*, *assortment* e *competition distance*, assegurando que todas as variáveis estejam em uma escala comparável.

3.3.4 Transformações Cíclicas e Logarítmicas

Variáveis temporais geralmente têm natureza cíclica, como os meses do ano ou os dias da semana. Para capturar essa natureza cíclica, foram aplicadas transformações seno e cosseno às variáveis *month*, *day*, *week of year* e *day of week*. Essas transformações permitem que os modelos de aprendizado de máquina compreendam melhor

os padrões cíclicos e sazonais nos dados, melhorando a precisão das previsões.

A transformação logarítmica da variável *sales* ajuda a normalizar a distribuição dos dados, o que é especialmente útil para reduzir o impacto de *outliers*. *Outliers* podem distorcer análises e previsões, e a transformação logarítmica suaviza as diferenças extremas em vendas, tornando o modelo mais robusto.

3.3.5 Filtragem e Ajustes

A filtragem de colunas desnecessárias é um passo importante para melhorar a eficiência do modelo e reduzir o ruído nos dados. Remover colunas como *customers*, *promo interval*, *open*, *month map*, *promo2 since* e *competition since* ajuda a focar nas variáveis mais relevantes, tornando o modelo mais preciso e eficiente.

Ajustar os tipos de dados também é uma prática importante para otimizar o uso de memória e melhorar o desempenho computacional. Converter variáveis inteiras para o tipo int32 e variáveis em ponto flutuante para o tipo float32 pode reduzir significativamente o consumo de memória, permitindo que grandes volumes de dados sejam processados de maneira mais eficiente.

3.4 ALGORITMO *BORUTA* PARA SELEÇÃO DE ATRIBUTOS

A seleção de atributos é um passo crucial no pré-processamento de dados para modelagem preditiva. O objetivo da seleção de atributos é identificar e selecionar um subconjunto de atributos relevantes do conjunto de dados original que contribuem significativamente para a precisão do modelo. A redução da dimensionalidade do conjunto de dados pode não apenas melhorar o desempenho do modelo, mas também reduzir a complexidade computacional e aumentar a interpretabilidade do modelo.

Criado por Kursa e Rudnicki (2010) para aplicação em R, o *Boruta* é um algoritmo de seleção de atributos todo baseado em importâncias de atributos derivadas do *Random Forest*. A ideia principal do *Boruta* é criar um conjunto de atributos "sombra"duplicando todos os atributos reais no conjunto de dados, mas embaralhando os valores em cada coluna para remover qualquer associação com a resposta. Em seguida, um modelo

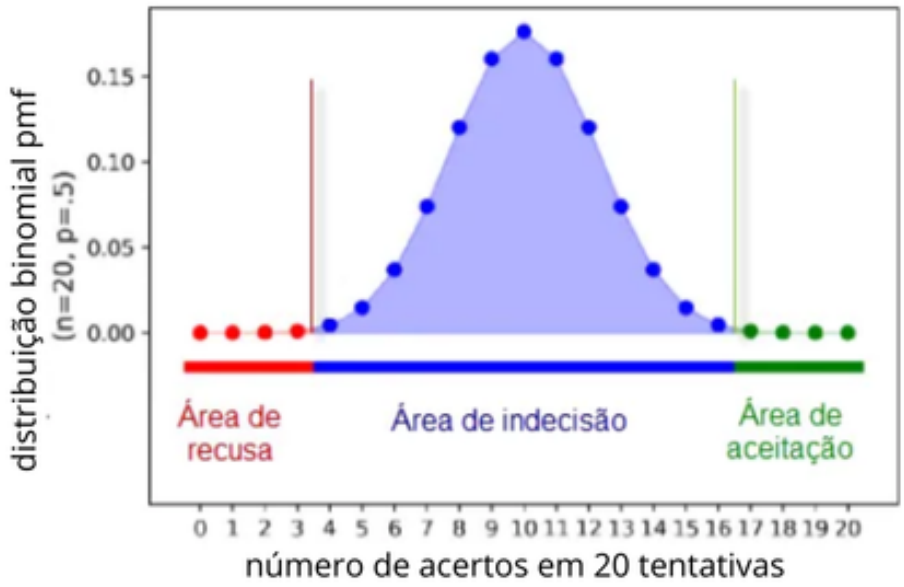
Random Forest é ajustado ao conjunto de dados expandido, incluindo os atributos reais e sombra, e a importância dos atributos é registrada. Para cada atributo real, o *Boruta* testa se sua importância é significativamente maior do que a máxima importância entre seus pares sombra. Aqueles atributos que têm uma importância significativamente maior são considerados relevantes para o modelo. O processo é repetido várias vezes para garantir a robustez da seleção de atributos (KURSA; RUDNICKI, 2010). A aplicação prática do algoritmo *Boruta* segue os seguintes passos:

1. **Duplicação** dos atributos do conjunto de dados para criar atributos sombra.
2. **Embaralhamento** dos valores em cada atributo sombra para remover qualquer correlação com o vetor alvo.
3. **Treinamento** de um modelo *Random Forest* no conjunto de dados expandido e cálculo da importância dos atributos.
4. **Comparação** da importância de cada atributo real com a importância máxima de seus respectivos atributos sombra.
5. **Marcação** dos atributos reais que se mostram consistentemente mais importantes do que os sombra como relevantes.
6. **Repetição** do processo até que a decisão sobre cada atributo seja tomada ou o número máximo de iterações seja alcançado.

Conforme mencionado, o *Boruta* avalia um recurso cuja utilidade é desconhecida, classificando-o como aceito ou recusado com base em três áreas: a área de recusa, onde os atributos são descartados por serem considerados ruído; a área de indecisão, onde o *Boruta* permanece incerto sobre a relevância da variável; e a área de aceitação, onde as variáveis são consideradas preditivas. Esse processo pode ser observado na Figura 21.

Neste trabalho, o algoritmo *Boruta* reduziu de 32 para 23 os atributos da base de dados. Os resultados em relação à performance e interpretabilidade serão discutidos no capítulo de resultados.

Figura 21 – Distribuição binomial para 20 interações, ilustrando as áreas de recusa, indecisão e aceitação.



Fonte: Extraída de Arcanjo (2021)..

3.5 CONFIGURAÇÕES DOS AMBIENTES DE TESTES

Para fins de estudo e aceleração do treinamento, foi separada uma amostra do conjunto de dados. O registro de todo o ano de 2014 foi definido como conjunto de treinamento, e os dados de 1º de janeiro de 2015 a 31 de julho de 2015 como conjunto de teste. Para atender às necessidades reais, previsões semanais e mensais foram feitas. As informações dos conjuntos de dados são detalhadas na Tabela 2, apresentando o conjunto de dados completo após o pré-processamento, características da amostragem retirada e da amostragem após a realização da seleção de atributos.

Tabela 2 – Características da Amostragem

Conjunto de dados	**Data**	**Meses**	**Atributos**	**Linhas**
Completo	01-01-2013 a 31-07-2015	31	32	1017209
Amostra	2014-01-31 a 2015-07-31	18	32	478820

3.5.1 Divisão em Treino e Teste

Se tratando de uma série temporal, a divisão do conjunto de dados em segmentos de treino, teste e validação, conforme descrito, garante a integridade cronológica. Este método evita o viés antecipado, onde informações futuras podem influenciar inadvertidamente o treinamento do modelo. Ao dividir os dados em 70% para treinamento, 20% para teste e os últimos 10% para validação, garantimos que o modelo possa ser treinado em dados anteriores, testado em dados posteriores e, finalmente, validado nos dados mais recentes, respeitando a ordem natural da série temporal. Técnicas de validação cruzada para séries temporais como a *TimeSeriesSplit* da biblioteca *scikit-learn* foram utilizadas nos testes iniciais, em detrimento do custo computacional, a primeira abordagem foi escolhida.

Além disso, o treinamento foi otimizado utilizando as seguintes estratégias:

- **Early Stopping**: Interrompe o treinamento automaticamente quando a melhoria na validação do modelo é mínima, evitando sobreajuste.
- **Model Checkpoint**: Armazena os melhores pesos do modelo com base no desempenho de validação durante o treinamento.

A combinação dessas técnicas e ferramentas permitiu a construção de modelos robustos, otimizados para a previsão de séries temporais, enquanto maximizava o uso eficiente dos recursos computacionais disponíveis.

3.5.2 Bibliotecas e Ferramentas

Os experimentos foram realizados em uma máquina virtual utilizando o Google Colab Pro, com duas configurações diferentes. A configuração utilizou uma CPU de alta capacidade com 51 GB de RAM, adequada para tarefas intensivas de processamento. Nesse contexto, as seguintes bibliotecas *Python* foram empregadas na análise dos dados e na construção dos modelos:

- **pandas**: Utilizada para a manipulação de dados por meio de *dataframes*, facilitando operações de limpeza, transformação e análise dos dados.
- **numpy**: Responsável por realizar operações matemáticas e manipulações de *arrays*, fundamentais para o processamento eficiente de dados numéricos.
- **matplotlib**: Utilizada para a visualização de dados por meio de gráficos, permitindo a análise exploratória dos dados e a avaliação dos resultados dos modelos.
- **scikit-learn**: Usada para a modelagem de aprendizado de máquina e para executar tarefas como a divisão dos dados, cálculo de métricas de erro e operações de pré-processamento.
- **tensorflow**: Utilizada para a construção, treinamento e avaliação dos modelos de redes neurais, incluindo LSTM e CNNs.
- **lightgbm**: Implementada para a execução de experimentos utilizando o algoritmo LightGBM, que é otimizado para tarefas de aprendizado em larga escala.
- **xgboost**: Implementada para a execução de experimentos utilizando o algoritmo XGBoost, que é otimizado para tarefas de aprendizado em larga escala.
- **optima**: Utilizada para a otimização de hiperparâmetros, ajustando os modelos de forma mais eficiente para alcançar a melhor performance possível.
- **mixed_precision**: Configurada para utilizar a política de precisão mista (*mixed_float16*), o que melhora o desempenho ao treinar modelos em GPUs.
- **random**: Utilizada para a definição de sementes aleatórias, garantindo a reprodutibilidade dos resultados obtidos.

Dessa forma, o ambiente de testes foi cuidadosamente configurado para maximizar a eficiência no processamento dos dados e no treinamento dos modelos, utilizando o que há de mais moderno em termos de infraestrutura e bibliotecas de aprendizado de máquina.

3.6 CONFIGURAÇÕES

3.6.1 Modelo Average

A avaliação de modelos preditivos exige a definição de *benchmarks* apropriados, os quais estabelecem uma referência mínima contra a qual o desempenho de modelos mais sofisticados pode ser comparado. O modelo de previsão baseado na média, muitas vezes descrito como um método *naive*, serve como um desses *benchmarks* devido à sua simplicidade e à sua capacidade de definir uma expectativa mínima de desempenho para qualquer conjunto de dados (HEWAMALAGE; ACKERMANN; BERGMEIR, 2022).

O modelo de média gera a previsão $\hat{y}_t$ ao assumir que todas as futuras observações serão iguais à média aritmética das observações passadas. Isso é matematicamente expresso pela equação 3.1, onde $\hat{y}_t$ representa o valor previsto, e y_i representa os valores observados no conjunto de treinamento, e n é o número total de observações no treinamento.

$$\hat{y}_t = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3.1}$$

Este modelo é útil como uma linha de base devido à sua implementação simples e ao custo computacional mínimo. Contudo, sua simplicidade também traz limitações, especialmente em séries temporais com padrões complexos, como tendências e sazonalidades, onde se espera que modelos mais avançados superem significativamente esse *benchmark*. O modelo é ajustado aos dados de treinamento e a média calculada é usada para prever os valores no conjunto de teste.

3.6.2 XGBoost

O XGBoost (Extreme Gradient Boosting) é amplamente reconhecido por sua eficácia em problemas de aprendizado supervisionado, incluindo séries temporais (ZHANG *et al.*, 2021). Entre seus principais pontos positivos, destaca-se a alta precisão e desempenho, o que o torna uma escolha popular em diversos *benchmarks* de aprendizado de

máquina (CHEN; GUESTRIN, 2016b). Essa precisão é particularmente valiosa em séries temporais, onde o XGBoost é capaz de capturar padrões complexos nos dados sequenciais, resultando em previsões precisas.

Outro benefício significativo do XGBoost é sua robustez em relação aos outliers. Devido à natureza dos métodos baseados em árvores de decisão utilizados pelo algoritmo, ele lida de maneira eficaz com valores atípicos, que são comuns em séries temporais. Além disso, o XGBoost oferece grande flexibilidade no uso de *feature engineering*, permitindo a inclusão de diversas características derivadas de variáveis temporais, como *lag features*, tendências e padrões sazonais. Essa flexibilidade ajuda a capturar padrões complexos e a melhorar a acurácia das previsões (ZHANG *et al.*, 2021). Outro ponto forte é o suporte à regularização, que auxilia na prevenção do *overfitting*, garantindo que o modelo generalize bem para novos dados. Por fim, o XGBoost é otimizado para velocidade e eficiência computacional, tornando-o capaz de lidar com grandes conjuntos de dados de séries temporais de forma rápida e eficaz.

Conforme mostrado na Tabela 3, a otimização dos hiperparâmetros do XGBoost por meio de *Bayesian Optimization* resultou em ajustes significativos em várias configurações-chave, como o número de folhas (num_leaves), a taxa de aprendizado ($learning_rate$) e a profundidade máxima (max_depth). Esses ajustes contribuíram para um modelo mais eficiente e preciso, evidenciando a importância de uma boa afinação dos hiperparâmetros para o desempenho do XGBoost.

Tabela 3 – Otimização Baynesiana para o modelo XGBoost

Hiperparâmetros	**Originais**	**Ajustados**	**Valores Testados**
num_leaves	10	9	5, 20
learning_rate	0.03	0.027	0.01, 0.3
n_estimators	300	264	100, 1000
max_depth	300	2	-1, 15
subsample	0.8	0.774	0.5, 1.0
colsample_bytree	0.85	0.901	0.5, 1.0
min_child_weight	8	2	0.5, 1.0
learning_rate	0.01	0.01	0.01, 0.3

3.6.3 LightGBM

O LightGBM é um algoritmo de *boosting* que tem ganhado destaque pela sua eficiência e alta performance, especialmente em tarefas que envolvem grandes volumes de dados e modelos complexos, como a previsão de séries temporais. Desenvolvido por Ke *et al.* (2017) para ser mais rápido e consumir menos memória do que outros algoritmos de *boosting*, como o XGBoost, o LightGBM utiliza técnicas avançadas como o *Gradient-based One-Side Sampling* (GOSS) e o *Exclusive Feature Bundling* (EFB) para acelerar o treinamento e melhorar a precisão. Em séries temporais, os hiperparâmetros principais, como o número de folhas (*num_leaves*), a profundidade máxima das árvores (*max_depth*), e a taxa de aprendizado (*learning_rate*), são ajustados para capturar eficazmente as dependências temporais e tendências, garantindo que o modelo possa prever com precisão as variações futuras com eficiência computacional.

Conforme mostrado na Tabela 4, a otimização dos hiperparâmetros do LightGBM por meio de *Bayesian Optimization* resultou em ajustes significativos. Entre os parâmetros ajustados, destaca-se o aumento do número de folhas (*num_leaves*), a redução da taxa de aprendizado (*learning_rate*), e a alteração na profundidade máxima (*max_depth*). Esses ajustes permitiram que o LightGBM capturasse melhor as complexidades das séries temporais, aumentando a precisão do modelo ao mesmo tempo em que manteve a eficiência computacional.

Neste trabalho, foram selecionadas três arquiteturas de redes neurais para modelar as vendas das lojas *Rossmann*: a Vanilla LSTM, representando a forma mais simples das arquiteturas LSTM; a Conv1D-LSTM, que utiliza convoluções unidimensionais para capturar padrões temporais locais antes de processá-los com LSTMs; e a Multi-Scale CNN-LSTM, projetada para identificar padrões em múltiplas escalas temporais. Essa seleção abrange desde abordagens tradicionais até técnicas mais avançadas, buscando capturar a dinâmica temporal das vendas com precisão e diferentes níveis de complexidade computacional.

Para a construção dos modelos de aprendizado profundo, utilizamos a biblioteca *TensorFlow*, juntamente com sua API de alto nível *Keras*. As arquiteturas utilizadas são descritas a seguir.

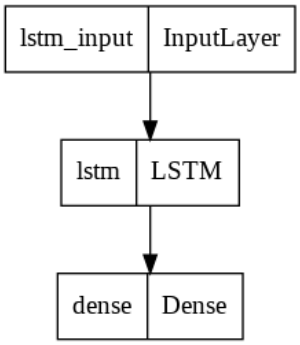
Tabela 4 – Otimização Baynesiana para o modelo LightGBM.

Hiperparâmetros	Originais	Ajustados	Valores Testados
num_leaves	50	195	31, 256
n_estimators	300	987	100, 1000
max_depth	-1	11	-1, 15
learning_rate	0.1	0.010	0.01, 0.3
subsample	1	0.759	0.5, 1.0
colsample_bytree	0.8	0.888	0.5, 1.0
reg_alpha	0.5	5.938	1e-8, 10.0
reg_lambda	0.5	1.110	1e-8, 10.0

3.6.4 Vanilla LSTM

O modelo Vanilla LSTM é a versão mais simples de uma rede LSTM, utilizada para capturar dependências temporais de longo prazo nos dados sequenciais. A arquitetura consiste em uma camada LSTM com 50 unidades, o que proporciona uma capacidade moderada para aprender padrões temporais complexos. Após a camada LSTM, há duas camadas densas com 100 unidades cada, usando a função de ativação ReLU, que adicionam transformações não lineares ao modelo, aumentando sua expressividade e capacidade de generalização. A camada de saída possui uma única unidade com ativação linear, responsável por gerar a previsão final, adequada para tarefas de regressão. A Figura 22 apresenta a estrutura do modelo.

Figura 22 – Vanilla LSTM.



Fonte: Elaborada pela autora (2024).

A otimização bayesiana para o modelo Vanilla LSTM foi conduzida ao longo de 7 tentativas, com um tempo total de 01:19:44. A perda de validação na sétima tentativa foi de 0.1449. A arquitetura ajustada incluiu um aumento no número de unidades da camada LSTM, o que melhorou a capacidade do modelo de capturar dependências

temporais nos dados. O ajuste dos hiperparâmetros, como o número de unidades densas e a taxa de aprendizado, contribuiu para um melhor desempenho geral (ver Tabela 5).

Tabela 5 – Otimização Bayesiana para o modelo Vanilla LSTM.

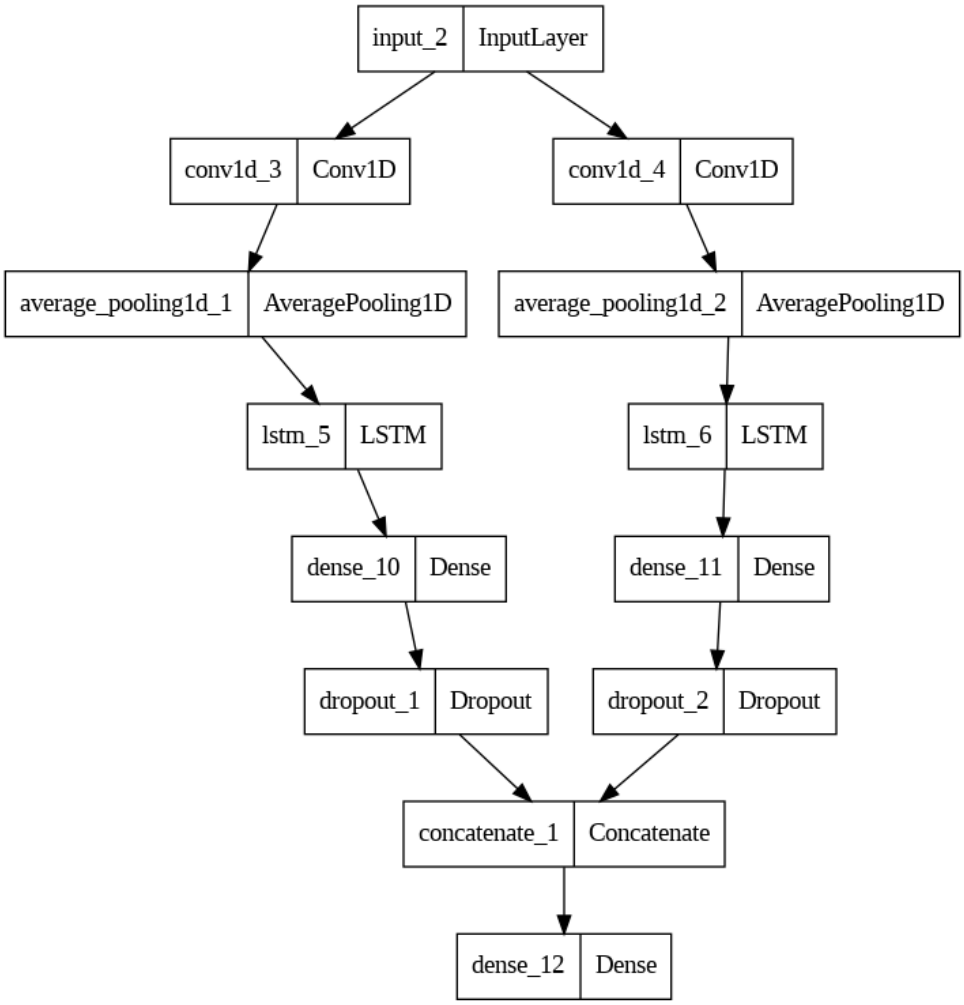
Hiperparâmetros	**Originais**	**Ajustados**	**Testados**
units	50	128	32, 256
dense_units	100	64	64, 256
dense_units_2	100	128	64, 256
learning_rate	0.001	0.001	0.01, 0.001

3.6.5 Multiscale CNN-LSTM

O modelo Multiscale CNN-LSTM é projetado para capturar diferentes escalas temporais simultaneamente, utilizando duas ramificações convolucionais paralelas. A primeira ramificação utiliza uma camada Conv1D com 60 filtros e kernel de tamanho 7, seguida por uma camada LSTM com 30 unidades para capturar dependências temporais em uma escala menor. A segunda ramificação, com uma configuração similar de 60 filtros, mas com um kernel maior de tamanho 30, foca em capturar padrões de longo prazo, também seguida por uma LSTM com 30 unidades. As saídas dessas duas LSTMs são então concatenadas e passam por uma camada densa final com uma unidade e ativação ReLU para gerar a previsão. Esta arquitetura permite ao modelo considerar padrões em múltiplas escalas temporais, garantindo uma compreensão abrangente dos dados. A Figura 23 apresenta o modelo implementado.

A otimização bayesiana para o modelo ConvLSTM envolveu 7 tentativas, com um tempo total de 03:16:59, resultando em uma perda de validação de 0.1345 na sétima tentativa. A arquitetura ajustada incluiu modificações nos tamanhos dos filtros e kernels das camadas convolucionais, aumentando a capacidade do modelo de extrair características relevantes, especialmente em séries temporais com forte componente espacial. A taxa de dropout foi ligeiramente reduzida, mantendo o equilíbrio entre regularização e aprendizado (ver Tabela 6).

Figura 23 – Arquitetura Multiscale CNN-LSTM.



Fonte: Elaborada pela autora (2024).

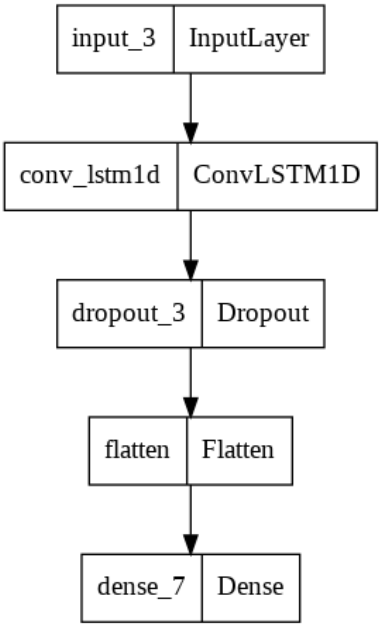
Tabela 6 – Otimização Baynesiana para o modelo CONV1D-LSTM.

Hiperparâmetros	**Originais**	**Ajustados**	**Testados**
conv1_filters	128	128	32, 256
conv1_kernel_size	3	5	2, 5
dropout_rate	0.3	0.1	0.1, 0.5
conv2_filters	64	64	32, 256
conv2_kernel_size	3	2	2, 5
lstm_units1	100	100	50, 200
lstm_units2	50	50	20, 100
dense_units	100	100	50, 200
learning_rate	0.001	0.001	0.01, 0.001

3.6.6 ConvLSTM

O modelo ConvLSTM combina convoluções unidimensionais com LSTMs para capturar padrões temporais e processá-los de forma hierárquica. A primeira camada Conv1D, com 128 filtros e kernel de tamanho 3, é projetada para extrair padrões temporais locais nas sequências, focando em detalhes específicos. A segunda camada Conv1D, com 64 filtros e kernel também de tamanho 3, continua essa extração, mas com menos filtros, o que ajuda a reduzir a complexidade do modelo. Seguem-se duas camadas LSTM, a primeira com 100 unidades, configurada para retornar a sequência, permitindo a captura de dependências temporais de longo prazo, e a segunda com 50 unidades, que condensa essas informações em um vetor único. Finalmente, as características extraídas passam por uma camada densa com 50 unidades e ativação ReLU antes de chegar à camada de saída, que possui uma única unidade com ativação linear para realizar a previsão final. A Figura 24 apresenta a estrutura do modelo implementado.

Figura 24 – Arquitetura do modelo ConvLSTM.



Fonte: Elaborada pela autora (2024).

A otimização bayesiana para o modelo LSTM Multiscale também foi realizada em 7 tentativas, com um tempo total de 02:32:4s, alcançando uma perda de validação de 0.1787 na sétima tentativa. A arquitetura ajustada beneficiou-se de ajustes nos tamanhos de kernel e nas unidades LSTM em diferentes escalas temporais. Isso

permitiu ao modelo capturar padrões em múltiplos horizontes temporais, resultando em uma melhor generalização nos dados de validação. A modificação nas camadas densas também contribuiu para uma melhora na performance (ver Tabela 7).

Tabela 7 – Otimização Baynesiana para Multiscale CNN-LSTM

Hiperparâmetros	**Originais**	**Ajustados**	**Testados**
conv1_filters	60	96	32, 128
conv1_kernel_size	7	5	5, 15
pool1_size	7	4	2, 10
lstm1_units	30	70	30, 100
dense1_units	30	50	30, 100
dropout_rate	0.2	0.2	0.1, 0.5
conv2_filters	60	96	32, 128
conv2_kernel_size	30	30	10, 30
pool2_size	7	8	2, 10
lstm2_units	30	90	30, 100
dense2_units	30	90	30, 100
learning_rate	0.001	0.001	0.01, 0.001

3.7 OTIMIZAÇÃO DE HIPERPARÂMETROS

O desempenho dos algoritmos de aprendizado de máquina, especialmente modelos preditivos baseados em aprendizado profundo, pode ser melhorado por meio da otimização de hiperparâmetros. A busca em grade (*grid search*) e a busca aleatória (*random search*) são os métodos mais comumente usados para encontrar a combinação ideal de hiperparâmetros para produzir modelos mais precisos. No entanto, a eficiência da busca em grade diminui com o aumento do número de hiperparâmetros, e sua complexidade de tempo computacional é alta devido ao aumento exponencial no número de avaliações necessárias com cada parâmetro adicional. Por outro lado, a busca aleatória utiliza uma combinação de parâmetros amostrados aleatoriamente com base em uma distribuição estatística, o que pode ser ineficaz na busca de pontos de hiperparâmetros ideais para alguns modelos complexos.

A otimização bayesiana é uma estratégia eficiente para resolver funções computacionalmente caras sem uma expressão de forma fechada. Ela constrói um modelo probabilístico da função objetivo para determinar os hiperparâmetros ideais de maneira informada, reduzindo o número de vezes que a função objetivo precisa ser execu-

tada ao escolher apenas o conjunto mais promissor de hiperparâmetros (SNOEK; LAROCHELLE; ADAMS, 2012). A abordagem consiste em dois componentes básicos:

- **Modelo substituto (probabilístico)**: A otimização bayesiana é guiada pelo teorema de Bayes e, a cada iteração, usa um modelo substituto para aproximar a função objetivo, que pode ser amostrado de maneira eficiente. Um processo gaussiano é o modelo substituto mais eficaz para selecionar o conjunto promissor de hiperparâmetros a serem avaliados na verdadeira função objetivo. O modelo substituto estima a função objetivo, que é usada para guiar futuras amostragens.
- **Função de aquisição**: Usa uma função de aquisição para determinar quais pontos no espaço de busca devem ser avaliados e para fornecer informações sobre o valor ótimo. O objetivo da função de aquisição é usar informações posteriores para encontrar o melhor ponto de amostra em cada iteração e propor um novo ponto de amostragem para identificar o conjunto mais promissor de hiperparâmetros a serem avaliados em seguida. A função de aquisição equilibra exploração e exploração. Exploração envolve focar no espaço de busca com maior probabilidade de melhorar a solução atual com base no modelo substituto atual, enquanto exploração é a estratégia de mover-se para regiões menos exploradas do espaço de busca.

De maneira geral, a otimização bayesiana fornece uma maneira mais eficiente e eficaz de ajustar os hiperparâmetros de um modelo LSTM do que os métodos tradicionais de busca em grade ou busca aleatória. Ela faz isso ao selecionar inteligentemente o próximo conjunto de hiperparâmetros a ser avaliado com base nos resultados das avaliações anteriores, levando a uma convergência mais rápida e melhores resultados (SNOEK; LAROCHELLE; ADAMS, 2012). Este método foi utilizado e descrito anteriormente nos modelos aplicados.

3.8 MÉTRICAS DE AVALIAÇÃO DE DESEMPENHO

Neste estudo, utilizamos quatro métricas amplamente reconhecidas para avaliar o desempenho dos modelos de regressão: *Erro Médio Absoluto* (MAE), *Raiz do Erro*

Médio Quadrático (RMSE), *Erro Médio Absoluto Escalonado* (MASE) e o coeficiente de determinação (R^2). Essas métricas fornecem diferentes perspectivas sobre o desempenho do modelo e são formalmente definidas nas equações 3.2, 3.3, 3.4 e 3.5, respectivamente.

Erro Médio Absoluto (MAE): O MAE mede a magnitude média dos erros nas previsões sem considerar a direção. Esta métrica é vantajosa devido à sua interpretabilidade simples, uma vez que representa a média dos erros absolutos, sendo útil para a comparação direta com os valores observados. Contudo, o MAE não penaliza grandes erros de forma mais intensa, tratando todos os desvios da mesma forma, o que pode ser uma desvantagem em contextos onde grandes erros são críticos.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.2}$$

Raiz do Erro Médio Quadrático (RMSE): O RMSE é sensível a grandes erros, pois utiliza o quadrado das diferenças. Ele é particularmente útil quando erros grandes precisam ser penalizados mais severamente, sendo uma métrica recomendada em casos onde o desempenho em previsões com grandes desvios é relevante. No entanto, essa sensibilidade pode ser desvantajosa em séries temporais com muitos outliers, já que pequenos desvios também podem resultar em um aumento substancial na métrica.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.3}$$

Erro Médio Absoluto Escalonado (MASE): O MASE normaliza o MAE dividindo-o pela média do erro de um modelo de referência, geralmente um modelo de previsão ingênuo que usa o valor anterior como previsão. Isso permite a comparação direta entre diferentes séries temporais, especialmente quando há variações nas unidades de medida ou padrões intermitentes. É uma métrica adequada para dados intermitentes, onde o desempenho de modelos tradicionais pode ser prejudicado devido à presença de zeros ou valores esparsos.

$$\text{MASE} = \frac{\text{MAE}}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|} \tag{3.4}$$

Coeficiente de Determinação (R^2): O R^2 mede a proporção da variabilidade total dos dados que é explicada pelo modelo. Varia entre 0 e 1, com valores próximos de 1 indicando que o modelo é capaz de explicar grande parte da variabilidade dos dados observados. Um valor de R^2 baixo pode sugerir que o modelo não está capturando bem os padrões subjacentes, enquanto valores próximos de 1 indicam um bom ajuste do modelo aos dados. Apesar de ser uma métrica amplamente utilizada, o R^2 pode ser influenciado pela complexidade do modelo, aumentando artificialmente conforme o número de parâmetros, o que requer a análise crítica de seu valor.

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} \tag{3.5}$$

onde,

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.6}$$

$$\text{TSS} = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{3.7}$$

Essas métricas oferecem diferentes perspectivas sobre a performance do modelo, sendo que o MAE é mais interpretável em termos absolutos, o RMSE é mais sensível a grandes erros, o MASE é robusto para séries intermitentes, e o R^2 avalia a proporção de variabilidade explicada pelo modelo. A escolha das métricas deve sempre considerar as características específicas dos dados e o objetivo do modelo preditivo.

4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados obtidos ao longo do desenvolvimento do trabalho, divididos em diferentes fases de experimentação. Inicialmente, utilizou-se uma amostra de 18 meses retirada do conjunto de dados para analisar o desempenho dos modelos em diferentes configurações e hiperparâmetros. Posteriormente, o conjunto de dados completo, após o devido pré-processamento, foi utilizado para o ajuste final de hiperparâmetros. As métricas analisadas incluem R^2, MAE (Erro Médio Absoluto), RMSE (Raiz do Erro Quadrático Médio), MASE (Erro Absoluto Escalado Médio) e o tempo de treinamento. Valores maiores de R^2 indicam uma melhor explicação da variabilidade dos dados, enquanto menores valores de MAE, RMSE e MASE refletem menor erro preditivo. O tempo de treinamento foi avaliado como uma medida de eficiência computacional.

A primeira etapa, referente à **Seleção de Atributos**, avaliou os modelos com e sem o método Boruta, utilizando a amostra de 18 meses para os conjuntos de treinamento, validação e teste. O objetivo foi verificar se a seleção de atributos resultaria em ganhos de desempenho.

Na subseção **Avaliação do Número de Épocas**, o impacto do número de épocas foi investigado em modelos como Vanilla LSTM, ConvLSTM e Multiscale CNN-LSTM. Essa análise preliminar foi feita também na amostra, mas sem a variável "store".

A subseção **Resultados para Configurações de Lookback e Lags** explorou como diferentes valores de *lookback* e *lags* afetaram o desempenho dos modelos de aprendizado profundo, com foco na otimização desses parâmetros.

Por fim, o **Resultados Finais com Ajustes de hiperparâmetros** foi realizado utilizando o conjunto de dados. Nesta etapa, os hiperparâmetros foram otimizados, resultando em uma análise final mais robusta e precisa.

Os resultados são discutidos detalhadamente nas próximas seções, destacando os efeitos da seleção de atributos, ajustes de hiperparâmetros e configuração de *lookback* e *lags*.

4.1 SELEÇÃO DE ATRIBUTOS

Nesta subseção, são apresentados os resultados obtidos pelos modelos de aprendizagem de máquina no conjunto de dados com 64 épocas e *lookback* de 7, utilizando uma amostra do conjunto de dados. A avaliação foi realizada para os conjuntos de treinamento, validação e teste, tanto com a aplicação do método de seleção de atributos Boruta quanto sem ele. A Tabela 8 apresenta os resultados dessas etapas.

A aplicação do método Boruta de seleção de atributos trouxe algumas melhorias específicas, mas seus impactos variaram conforme o modelo analisado. No caso do XGBoost, houve uma melhora marginal em todas as métricas de desempenho após a aplicação do Boruta, com um aumento no R^2 de 0.732 para 0.756 no conjunto de teste e uma redução no MAE de 0.161 para 0.152. Essas melhorias, embora pequenas, indicam que a seleção de atributos contribuiu para uma leve melhora no desempenho preditivo sem alterar significativamente o tempo de treinamento (mantido em 35 segundos).

O LightGBM apresentou uma redução significativa no tempo de treinamento ao utilizar o Boruta, de 92 para 47 segundos, além de melhorias em todas as métricas de erro, como o MAE (de 0.109 para 0.104) e o RMSE (de 0.154 para 0.148). Esses resultados sugerem que o Boruta foi eficaz em reduzir a complexidade do modelo sem comprometer a precisão, tornando-o mais eficiente.

Por outro lado, o Vanilla LSTM apresentou um comportamento diferente com a utilização do Boruta. Além da redução no tempo de treinamento em 29% (de 611 para 434 segundos), o desempenho preditivo melhorou somente na etapa de teste. Especificamente, no treino e validação o algoritmo sem a utilização do Boruta apresentou melhores resultados, porém, no teste o algoritmo apresentou uma melhora no R^2 subindo de 0.178 para 0.256, enquanto o MAE, RMSE e MASE diminuíram.

Os modelos mais complexos, como o ConvLSTM e o Multiscale CNN-LSTM, também se beneficiaram da aplicação do Boruta, especialmente em termos de tempo de treinamento. No caso do ConvLSTM, houve uma redução no tempo de 12% e uma melhora significativa nas métricas de erro no conjunto de teste. Para o Multiscale

Tabela 8 – Desempenho dos modelos no conjunto de Treinamento, Teste e Validação com e sem Seleção de Atributos).

Modelo	**Atributos**	**R^2**	**MAE**	**RMSE**	**MASE**	**T(S)**
Treinamento						
Average Model	-	0.000	0.326	0.419	0.827	3
Average Model	Boruta	0.000	0.326	0.419	0.827	4
XGBoost	-	0.826	0.133	0.174	0.336	**35**
XGBoost	Boruta	**0.839**	**0.128**	**0.168**	**0.324**	35
LightGBM	-	0.957	0.065	0.088	0.163	92
LightGBM	Boruta	**0.964**	**0.058**	**0.080**	**0.148**	**47**
Vanilla LSTM	-	**0.602**	**0.200**	**0.264**	**0.508**	611
Vanilla LSTM	Boruta	0.459	0.236	0.308	0.598	**434**
Conv1D-LSTM	-	**0.828**	**0.127**	**0.173**	**0.321**	2353
Conv1D-LSTM	Boruta	0.816	0.129	0.179	0.327	**2061**
Multiscale CNN-LSTM	-	0.716	0.165	0.223	0.419	1440
Multiscale CNN-LSTM	Boruta	**0.730**	**0.161**	**0.217**	**0.407**	**1180**
Validação						
Average Model	-	-0.001	0.331	0.423	0.838	-
Average Model	Boruta	-0.001	0.331	0.423	0.838	-
XGBoost	-	0.747	0.166	0.213	0.421	-
XGBoost	Boruta	**0.771**	**0.157**	**0.202**	**0.397**	-
LightGBM	-	0.859	0.121	0.159	0.306	-
LightGBM	Boruta	**0.870**	**0.115**	**0.153**	**0.291**	-
Vanilla LSTM	-	**0.388**	**0.251**	**0.331**	**0.635**	-
Vanilla LSTM	Boruta	0.375	0.259	0.334	0.656	-
Conv1D-LSTM	-	0.734	0.158	0.218	0.400	-
Conv1D-LSTM	Boruta	**0.758**	**0.149**	**0.208**	**0.378**	-
Multiscale CNN-LSTM	-	0.483	0.233	0.304	0.590	-
Multiscale CNN-LSTM	Boruta	**0.550**	**0.216**	**0.284**	**0.547**	-
Teste						
Average Model	-	-0.016	0.325	0.416	0.823	-
Average Model	Boruta	-0.016	0.325	0.416	0.823	-
XGBoost	-	0.732	0.161	0.214	0.408	-
XGBoost	Boruta	**0.756**	**0.152**	**0.204**	**0.386**	-
LightGBM	-	0.861	0.109	0.154	0.277	-
LightGBM	Boruta	**0.871**	**0.104**	**0.148**	**0.263**	-
Vanilla LSTM	-	0.178	0.284	0.374	0.721	-
Vanilla LSTM	Boruta	**0.256**	**0.275**	**0.356**	**0.697**	-
Conv1D-LSTM	-	0.643	0.176	0.247	0.446	-
Conv1D-LSTM	Boruta	**0.707**	**0.158**	**0.224**	**0.399**	-
Multiscale CNN-LSTM	-	0.323	0.261	0.340	0.661	-
Multiscale CNN-LSTM	Boruta	**0.427**	**0.238**	**0.312**	**0.602**	-

CNN-LSTM, a redução no tempo de treinamento foi de cerca de 18%, acompanhada de uma melhora modesta no R^2 e nas métricas de erro.

Em termos gerais, o Boruta mostrou-se útil em cenários onde a eficiência computacional é uma prioridade, particularmente para modelos mais complexos . A utilidade do Boruta parece estar relacionada à complexidade do modelo e à natureza do conjunto de dados, sendo mais eficaz em reduzir a dimensionalidade para modelos que lidam bem com menos atributos sem perder precisão significativa.

4.2 AVALIAÇÃO DO NÚMERO DE ÉPOCAS PARA OS ALGORITMOS DE APRENDIZADO PROFUNDO

Foi realizado um estudo preliminar para avaliar o impacto do número de épocas no desempenho dos modelos na previsão das vendas diárias gerais. A Tabela 9 apresenta os resultados de validação para diferentes configurações de épocas, comparando os modelos Vanilla LSTM, Multiscale CNN-LSTM e Conv1D-LSTM. Os experimentos foram conduzidos utilizando uma amostra do conjunto de dados, previamente filtrada pelo algoritmo Boruta. As análises consideraram uma janela de Lookback de 7 períodos e desconsideraram a variável *store* na análise.

Tabela 9 – Resultados de validação dos Modelos de LSTM para diferentes Épocas.

Modelo	**Épocas**	**R^2**	**MAE**	**RMSE**	**MASE**	**T(S)**
Vanilla LSTM	128	-0.740	0.409	0.544	1.037	**223**
Vanilla LSTM	64	**0.399**	**0.239**	**0.320**	**0.606**	1193
Vanilla LSTM	32	-2.061	0.617	0.722	1.564	5462
Vanilla LSTM	16	-0.757	0.440	0.547	1.115	1017
Conv1D-LSTM	128	0.592	0.190	0.562	0.295	21434
Conv1D-LSTM	64	**0.602**	**0.183**	**0.260**	**0.290**	**1342**
Conv1D-LSTM	32	0.549	0.203	0.555	0.277	1706
Conv1D-LSTM	16	0.228	0.280	0.323	0.363	1387
Multiscale CNN-LSTM	128	0.489	0.222	0.562	0.295	8208
Multiscale CNN-LSTM	64	**0.506**	**0.216**	0.547	**0.290**	8108
Multiscale CNN-LSTM	32	0.494	0.219	0.555	0.294	5462
Multiscale CNN-LSTM	16	-0.384	0.244	**0.323**	0.324	**1184**

Os resultados demonstram que o número de épocas influencia significativamente o desempenho dos modelos. Em geral, o aumento do número de épocas tende a melhorar as métricas de desempenho, como R^2, MAE, RMSE e MASE, até certo

ponto, após o qual podem ocorrer efeitos adversos, como o *overfitting*. No entanto, observa-se que o modelo Multiscale CNN-LSTM, com 64 épocas, obteve o melhor desempenho geral, enquanto o Conv1D-LSTM também se destacou com 64 épocas. Esses resultados indicam que, para os testes preliminares conduzidos, 64 épocas podem ser uma escolha adequada para o equilíbrio entre desempenho e tempo de treinamento.

4.3 RESULTADOS PARA DIFERENTES CONFIGURAÇÕES DE LOOKBACK E LAGS

No contexto de modelos de aprendizado de máquina aplicados a séries temporais, como LSTM (Long Short-Term Memory), dois parâmetros importantes são o lookback e o lag. O lookback refere-se ao número de observações anteriores que o modelo utiliza para fazer previsões. Por exemplo, um lookback de 7 ou 30 dias significa que o modelo considera os dados desses últimos dias para prever o próximo valor da série. A escolha desse parâmetro é crucial, pois afeta diretamente a capacidade do modelo de capturar padrões de curto ou longo prazo. Um lookback muito curto pode não identificar padrões sazonais, enquanto um lookback muito longo pode aumentar a complexidade do modelo e introduzir ruído, o que pode levar a problemas de overfitting.

O lag refere-se à introdução de atrasos adicionais nos dados, permitindo que o modelo capture padrões temporais que ocorrem em intervalos específicos. Ao utilizar lags de 1, 7 e 30 dias, o modelo considera não apenas as observações mais recentes, mas também as observações em intervalos maiores, ajudando a detectar padrões periódicos e sazonais. A análise apresentada na Tabela 10 compara três modelos – Vanilla LSTM, Conv1D-LSTM e Multiscale CNN-LSTM – para diferentes configurações de lookback (7 e 30 dias) e a utilização de lags (1, 7, 30). As métricas de avaliação incluem R^2, MAE, RMSE e MASE, com o tempo de treinamento registrado para cada configuração.

Os resultados indicam que o desempenho dos modelos varia conforme a configuração de *lookback* e a utilização de lags. De modo geral, os modelos tendem a ter um desempenho superior com um *lookback* de 7 dias, especialmente quando lags são utilizados. No entanto, para um *lookback* de 30 dias, o Conv1D-LSTM apresentou

Tabela 10 – Resultados de Validação e Teste para Diferentes Configurações de Look-back (7 e 30 dias)

Modelo	**LB**	**LG**	**R^2**	**MAE**	**RMSE**	**MASE**	**T(S)**
Treinamento							
Vanilla LSTM	7	-	**0.459**	**0.236**	**0.308**	**0.598**	**433**
	7	(1,7,30)	-0.001	0.326	0.419	0.827	1113
	30	-	-0.001	0.327	0.419	0.828	1329
	30	(1,7,30)	-0.001	0.326	0.419	0.827	937
Conv1D-LSTM	7	-	0.816	0.129	0.179	0.327	**2353**
	7	(1,7,30)	0.900	0.096	0.132	0.242	6009
	30	-	0.886	0.101	0.141	0.255	9357
	30	(1,7,30)	**0.913**	**0.089**	**0.124**	**0.225**	8205
Multiscale CNN-LSTM	7	-	**0.730**	**0.161**	**0.217**	**0.407**	1180
	7	(1,7,30)	0.711	0.168	0.225	0.426	2175
	30	-	0.698	0.171	0.230	0.434	1900
	30	(1,7,30)	-0.001	0.327	0.419	0.828	**504**
Teste							
Vanilla LSTM	7	-	**0.256**	**0.275**	**0.356**	**0.697**	-
	7	(1,7,30)	-0.023	0.326	0.418	0.826	-
	30	-	-0.010	0.324	0.415	0.820	-
	30	(1,7,30)	-0.023	0.326	0.418	0.826	-
Conv1D-LSTM	7	-	0.707	0.158	0.224	0.399	-
	7	(1,7,30)	0.808	0.126	0.181	0.318	-
	30	-	**0.825**	**0.120**	**0.173**	**0.305**	-
	30	(1,7,30)	0.820	**0.120**	0.175	**0.305**	-
Multiscale CNN-LSTM	7	-	**0.427**	**0.238**	**0.312**	**0.602**	-
	7	(1,7,30)	0.374	0.249	0.327	0.630	-
	30	-	0.401	0.242	0.320	0.613	-
	30	(1,7,30)	-0.009	0.323	0.415	0.820	-
Validação							
Vanilla LSTM	7	-	**0.375**	**0.259**	**0.334**	**0.656**	-
	7	(1,7,30)	-0.003	0.331	0.423	0.839	-
	30	-	-0.001	0.331	0.423	0.838	-
	30	(1,7,30)	-0.003	0.331	0.423	0.839	-
Conv1D-LSTM	7	-	0.758	0.149	0.208	0.378	-
	7	(1,7,30)	0.880	0.108	0.147	0.273	-
	30	-	0.833	0.130	0.172	0.329	-
	30	(1,7,30)	**0.893**	**0.101**	**0.139**	**0.256**	-
Multiscale CNN-LSTM	7	-	**0.550**	**0.216**	**0.284**	**0.547**	-
	7	(1,7,30)	0.532	0.221	0.289	0.560	-
	30	-	0.514	0.223	0.295	0.566	-
	30	(1,7,30)	0.000	0.331	0.423	0.838	-

melhorias significativas nas métricas de desempenho, embora o tempo de treinamento tenha aumentado consideravelmente. Isso sugere que a escolha da janela de *lookback* e o uso de *lags* são fatores importantes para otimizar o desempenho do modelo.

4.4 AJUSTE DOS HIPERPARÂMETROS

A Tabela 11 apresenta uma visão geral do desempenho dos modelos testados neste estudo sem e com a utilização da otimização bayesiana. As métricas analisadas incluem novamente o R^2, MAE, RMSE, MASE e o tempo de treinamento (T) em segundos.

O modelo médio (Average Model) serviu como uma linha de base, apresentando valores baixos de R^2 e métricas de erro elevadas, o que era esperado. Os modelos XGBoost e LightGBM, mesmo antes da otimização, superaram significativamente o modelo médio, com melhorias notáveis em todas as métricas.

A aplicação da Otimização Bayesiana nos modelos XGBoost e LightGBM resultou em uma melhoria considerável no desempenho. Por exemplo, o modelo LightGBM otimizado obteve um R^2 de 0.864, uma redução significativa no MAE para 0.108, e um RMSE de 0.152. Embora o tempo de treinamento tenha aumentado, as melhorias na precisão justificam o custo computacional adicional.

Já as arquiteturas LSTM mostraram melhorias substanciais após a aplicação da Otimização Bayesiana. O modelo Vanilla LSTM otimizado, por exemplo, teve um aumento no R^2 de 0.375 para 0.658, com reduções correspondentes nas métricas de erro. No entanto, isso veio ao custo de um aumento significativo no tempo de treinamento, de 433 segundos para 1608 segundos.

Os modelos CNN-LSTM Multiescala e Conv1D-LSTM também se beneficiaram da Otimização Bayesiana. O Conv1D-LSTM otimizado, em particular, apresentou o melhor desempenho geral, com um R^2 de 0.907, o menor MAE de 0.091, e o menor RMSE de 0.127 entre todos os modelos. O aumento no tempo de treinamento para 5552 segundos reflete a maior complexidade computacional, mas é acompanhado por uma melhoria significativa na precisão preditiva.

Em suma, a otimização Bayesiana provou ser uma abordagem eficaz para melhorar o desempenho dos modelos em termos de métricas de erro e poder preditivo, com cada modelo ajustado superando suas versões não otimizadas. A análise detalhada demonstra a eficácia desta técnica em várias arquiteturas de modelos, desde modelos baseados em árvores de decisão até redes neurais profundas.

Tabela 11 – Resultados de teste e tempo de treinamento (T) para diferentes modelos com lookback 7 e 30 para modelos com hiperparâmetros ajustados e não ajustados

Modelo	**LB**	**R^2**	**MAE**	**RMSE**	**MASE**	**T(S)**
Average	-	-0.028	0.348	0.450	0.878	-
XGBoost	-	**0.806**	**0.136**	**0.195**	**0.344**	**63**
XGBoost Ajustado	-	0.680	0.184	0.251	0.466	68
LightGBM	-	0.755	0.157	0.220	0.396	**31**
LightGBM Ajustado	-	**0.815**	**0.132**	**0.191**	**0.333**	116
Vanilla LSTM	7	0.353	0.268	0.357	0.678	**1487**
Vanilla LSTM Ajustado	7	**0.511**	**0.228**	**0.310**	**0.577**	3488
Vanilla LSTM	30	-0.063	0.350	0.457	0.885	**1928**
Vanilla LSTM Ajustado	30	**0.154**	**0.314**	**0.408**	**0.794**	3852
Conv1D-LSTM	7	0.714	0.161	0.237	0.407	5735
Conv1D-LSTM Ajustado	7	**0.762**	**0.145**	**0.216**	**0.368**	**3764**
Conv1D-LSTM	30	0.762	0.144	0.217	0.365	21383
Conv1D-LSTM Ajustado	30	**0.794**	**0.130**	**0.201**	**0.329**	**20223**
Multiscale CNN-LSTM	7	0.311	0.279	0.368	0.705	**987**
Multiscale CNN-LSTM Ajustado	7	**0.411**	**0.254**	**0.340**	**0.642**	2304
Multiscale CNN-LSTM	30	0.312	0.279	0.368	0.705	**4442**
Multiscale CNN-LSTM Ajustado	30	**0.462**	**0.243**	**0.325**	**0.614**	4960

4.5 ANÁLISE COMPARATIVA ENTRE VALORES REAIS E PREVISTOS PARA DIFERENTES MODELOS DE LSTM

Nesta seção, analisamos a performance dos modelos Vanilla LSTM, Conv1D-LSTM e Multiscale CNN-LSTM através de gráficos que comparam os valores reais com os valores previstos (*Actual vs Predicted*). Cada par de gráficos representa a comparação entre a versão original e a versão ajustada do modelo, após a otimização dos hiperparâmetros. Além disso, utilizamos *lookbacks* de 7 e 30 dias para avaliar o impacto do tamanho da janela temporal nas previsões dos modelos.

A Figura 25 mostra a relação entre os valores reais e os valores previstos pelo modelo Vanilla LSTM com *lookback* de 7 dias. No gráfico da esquerda, observamos o

desempenho do modelo não ajustado. Nota-se uma dispersão significativa dos pontos, indicando que as previsões feitas pelo modelo estão distantes dos valores reais, com muitos pontos afastados da linha ideal de 45 graus. Isso demonstra que o modelo original tem dificuldades em capturar a dinâmica dos dados com precisão.

No gráfico da direita, que representa o modelo ajustado após a otimização dos hiperparâmetros, a dispersão dos pontos em torno da linha ideal diminui consideravelmente. A maior concentração de pontos ao longo da linha de 45 graus sugere que o modelo ajustado é capaz de prever os valores de forma mais precisa, indicando uma melhoria significativa na performance após o ajuste de hiperparâmetros.

A Figura 26 apresenta o desempenho do modelo Vanilla LSTM para um *lookback* de 30 dias. No gráfico à esquerda, que corresponde ao modelo não ajustado, vemos uma dispersão ainda maior dos pontos, o que indica uma acurácia preditiva inferior. Essa dispersão sugere que o modelo original tem dificuldades em capturar padrões de longo prazo nos dados, resultando em previsões que se afastam consideravelmente dos valores reais.

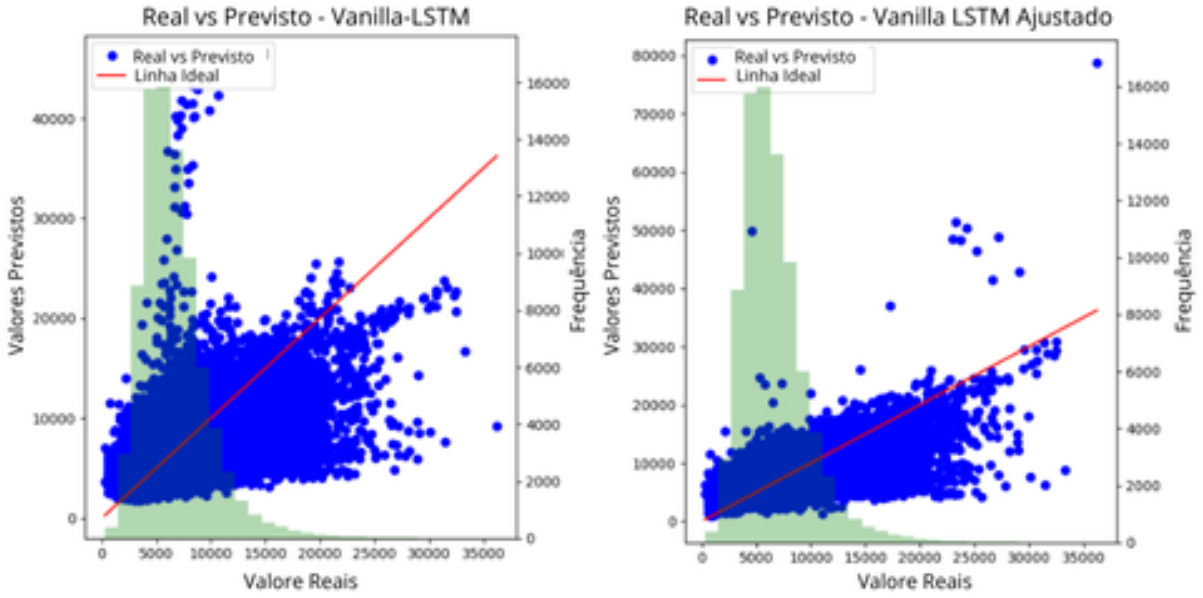
Por outro lado, o gráfico da direita, referente ao modelo ajustado, apresenta uma distribuição mais concentrada ao redor da linha ideal. A otimização dos hiperparâmetros claramente melhorou a capacidade preditiva do modelo para um *lookback* maior, reduzindo o erro nas previsões e tornando o modelo mais robusto.

Na Figura 27, vemos o desempenho do modelo Conv1D-LSTM para um *lookback* de 7 dias. O gráfico da esquerda, que apresenta o modelo não ajustado, exibe uma considerável dispersão dos pontos em relação à linha de 45 graus, indicando que as previsões feitas pelo modelo ainda são imprecisas.

Após a otimização dos hiperparâmetros, no gráfico à direita, a dispersão é reduzida, com uma maior concentração de pontos ao longo da linha ideal. Isso mostra que a otimização teve um impacto positivo, aumentando a precisão do modelo e melhorando sua capacidade de previsão para janelas temporais mais curtas.

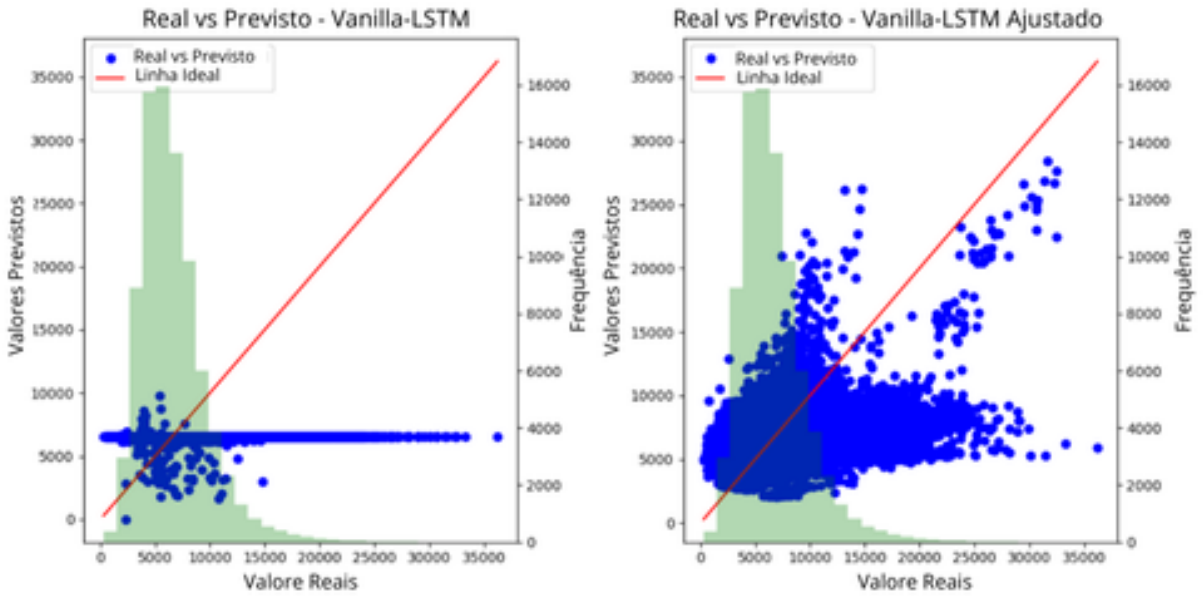
A Figura 28 apresenta os resultados para o Conv1D-LSTM com um *lookback* de 30

Figura 25 – Comparação entre valores reais e previstos para o modelo Vanilla LSTM com *lookback* de 7 dias, antes e depois da otimização de hiperparâmetros.



Fonte: Elaborada pela autora (2024).

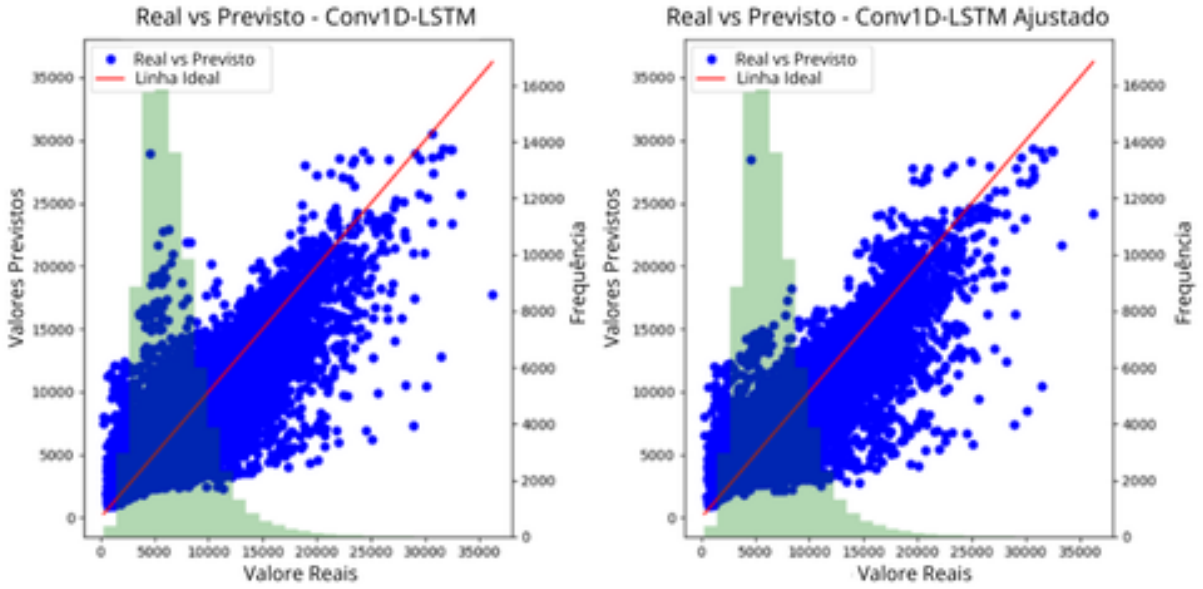
Figura 26 – Comparação entre valores reais e previstos para o modelo Vanilla LSTM com *lookback* de 30 dias, antes e depois da otimização de hiperparâmetros.



Fonte: Elaborada pela autora (2024)..

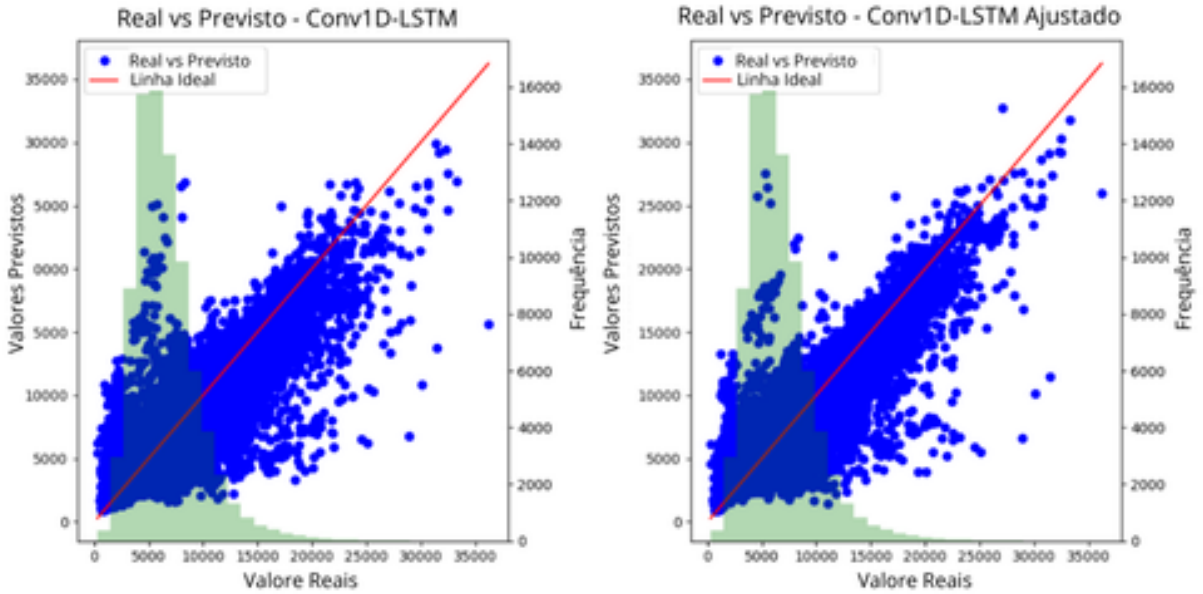
dias. No gráfico da esquerda, referente ao modelo não ajustado, há uma dispersão mais acentuada em comparação com o *lookback* de 7 dias, o que sugere dificuldades do modelo em realizar previsões precisas quando são utilizadas janelas temporais

Figura 27 – Comparação entre valores reais e previstos para o modelo Conv1D-LSTM com *lookback* de 7 dias, antes e depois da otimização de hiperparâmetros.



Fonte: Elaborada pela autora (2024).

Figura 28 – Comparação entre valores reais e previstos para o modelo Conv1D-LSTM com *lookback* de 30 dias, antes e depois da otimização de hiperparâmetros.



Fonte: Elaborada pela autora (2024).

maiores.

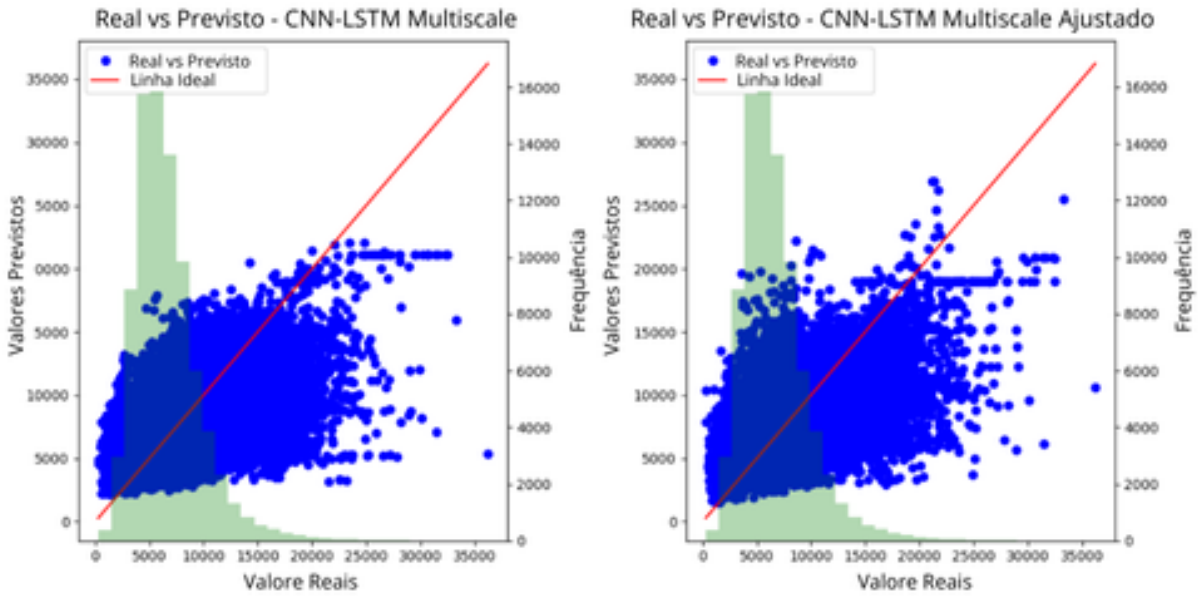
O gráfico da direita mostra o desempenho do modelo após a otimização. A dispersão dos pontos em torno da linha ideal diminui, sugerindo que o ajuste dos hiperparâmetros

melhora a capacidade do modelo de lidar com períodos mais longos.

A Figura 29 mostra o desempenho do modelo Multiscale CNN-LSTM com *lookback* de 7 dias. O gráfico da esquerda, que representa o modelo não ajustado, exibe uma dispersão considerável, sugerindo que o modelo tem dificuldades em realizar previsões precisas com janelas temporais curtas.

Após a otimização dos hiperparâmetros, no gráfico à direita, é apresentada maior concentração de pontos próximos à linha ideal de 45 graus.

Figura 29 – Comparação entre valores reais e previstos para o modelo Multiscale CNN-LSTM com *lookback* de 7 dias, antes e depois da otimização de hiperparâmetros.

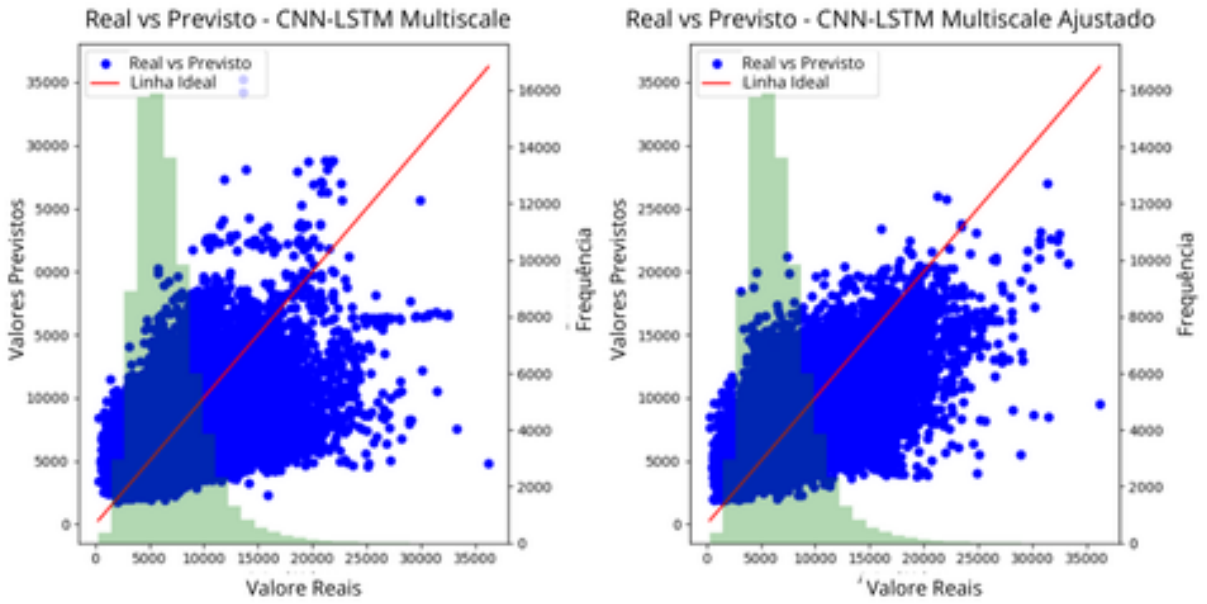


Fonte: Elaborada pela autora (2024).

Por fim, a Figura 30 apresenta o desempenho do modelo Multiscale CNN-LSTM para um *lookback* de 30 dias. No gráfico da esquerda, referente ao modelo não ajustado, observa-se uma dispersão ainda maior, indicando dificuldades em realizar previsões de longo prazo.

O gráfico da direita, que mostra o desempenho do modelo ajustado, apresenta uma melhora considerável na previsão, com uma concentração mais densa de pontos próximos à linha de 45 graus. Isso sugere que, apesar da maior janela temporal, o ajuste de hiperparâmetros do modelo foi eficaz em reduzir o erro nas previsões.

Figura 30 – Comparação entre valores reais e previstos para o modelo Multiscale CNN-LSTM com *lookback* de 30 dias, antes e depois da otimização de hiperparâmetros.



Fonte: Elaborada pela autora (2024).

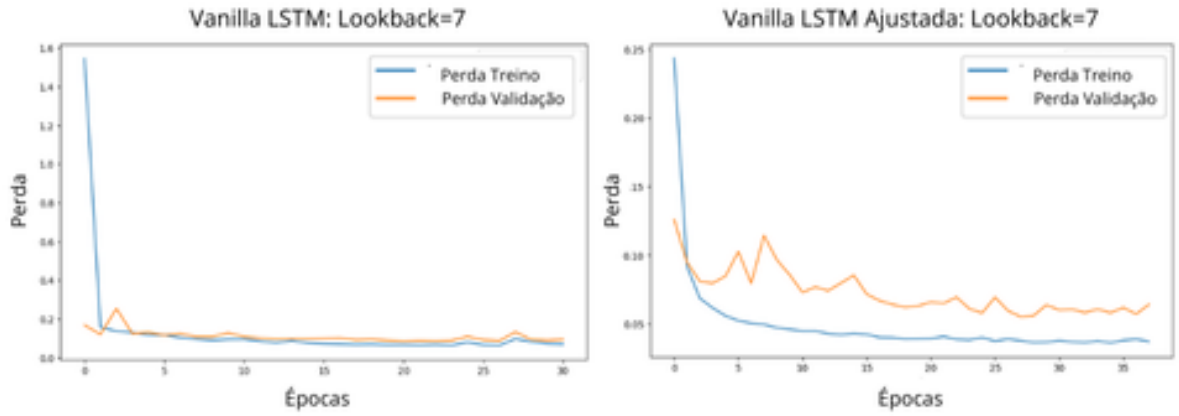
Os gráficos de comparação entre valores reais e previstos para os diferentes modelos de LSTM, com *lookbacks* de 7 e 30 dias, mostram que a otimização de hiperparâmetros melhora o desempenho preditivo dos modelos. A redução da dispersão dos pontos em torno da linha ideal de 45 graus nos modelos ajustados indica que eles são capazes de capturar melhor as dinâmicas dos dados, resultando em previsões mais precisas.

4.6 ANÁLISE COMPARATIVA DOS GRÁFICOS DE PERDA PARA DIFERENTES MODELOS LSTM

Nesta seção, apresentamos uma análise comparativa dos gráficos de perda (*loss*) gerados para diferentes arquiteturas de redes neurais Vanilla LSTM, Multiscale CNN-LSTM e Conv1D-LSTM, tanto antes quanto após a otimização dos hiperparâmetros. Avaliamos o desempenho dos modelos utilizando *lookbacks* de 7 e 30 dias.

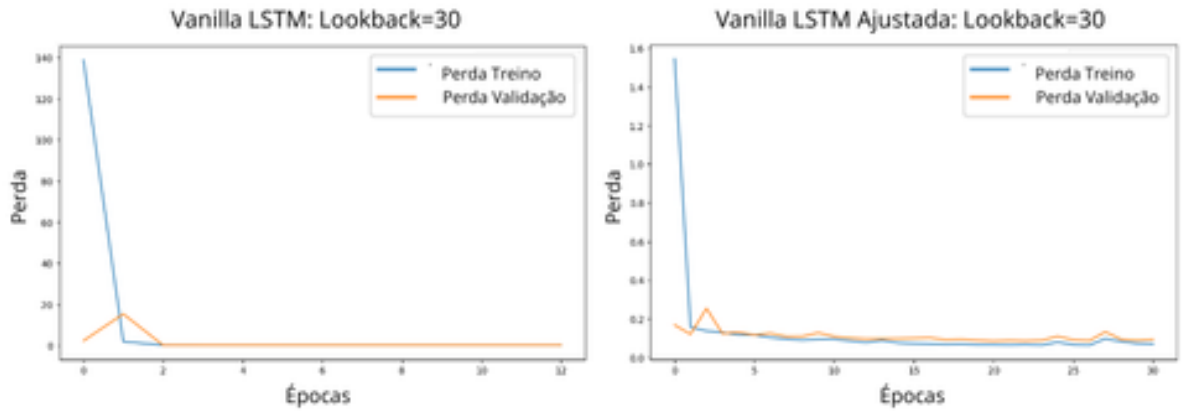
A análise dos gráficos de perda do Vanilla LSTM, mostrada nas Figuras 31 e 32, revela um comportamento distinto entre os modelos originais e ajustados. Para o *lookback* de 7 dias, observa-se que o modelo ajustado apresenta uma convergência mais rápida, com a perda de treino diminuindo substancialmente nas primeiras épocas, enquanto a perda

Figura 31 – Gráfico de perda para o modelo Vanilla LSTM com *lookback* de 7 dias.



Fonte: Elaborada pela autora (2024).

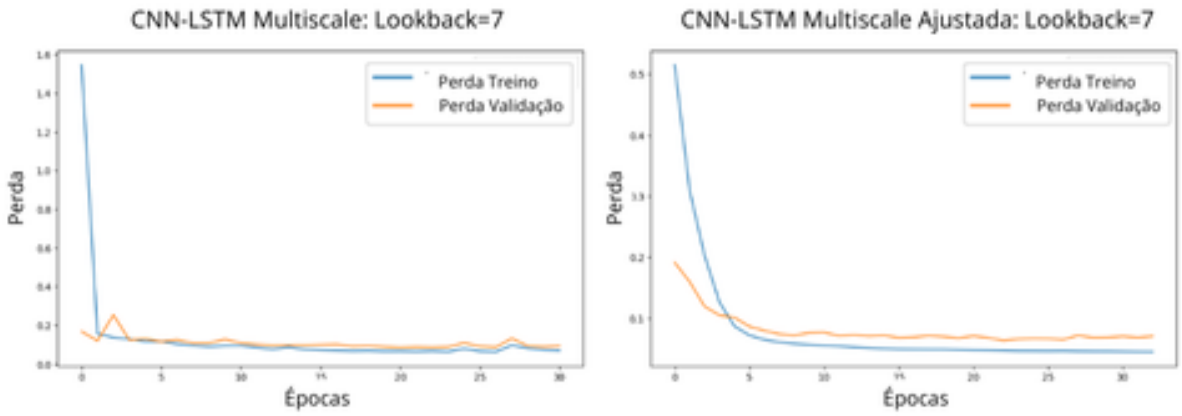
Figura 32 – Gráfico de perda para o modelo Vanilla LSTM com *lookback* de 30 dias.



Fonte: Elaborada pela autora (2024).

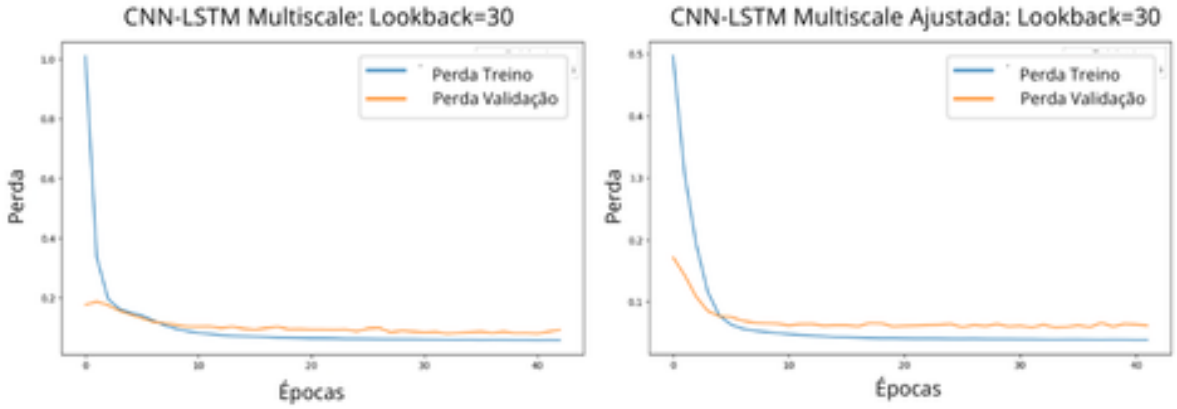
de validação estabiliza de maneira mais consistente em comparação ao modelo não ajustado. Esse comportamento indica uma maior capacidade de generalização após a otimização dos hiperparâmetros, conforme evidenciado na Figura 31. Para o *lookback* de 30 dias (Figura 32), o modelo ajustado também apresenta uma melhora expressiva, com uma redução significativa na diferença entre a perda de treino e validação. A menor discrepância entre essas curvas sugere que o modelo ajustado apresenta menor propensão ao *overfitting*. Os gráficos de perda para o modelo Multiscale CNN-LSTM, apresentados nas Figuras 33 e 34, mostram um comportamento semelhante ao do modelo Vanilla LSTM. O modelo ajustado, após a otimização dos hiperparâmetros, apresenta uma curva de perda mais estável, com menor variabilidade tanto na perda de treino quanto na de validação. Para o *lookback* de 7 dias (Figura 33), observa-se uma

Figura 33 – Gráfico de perda para o modelo Multiscale CNN-LSTM com *lookback* de 7 dias.



Fonte: Elaborada pela autora (2024).

Figura 34 – Gráfico de perda para o modelo Multiscale CNN-LSTM com *lookback* de 30 dias.

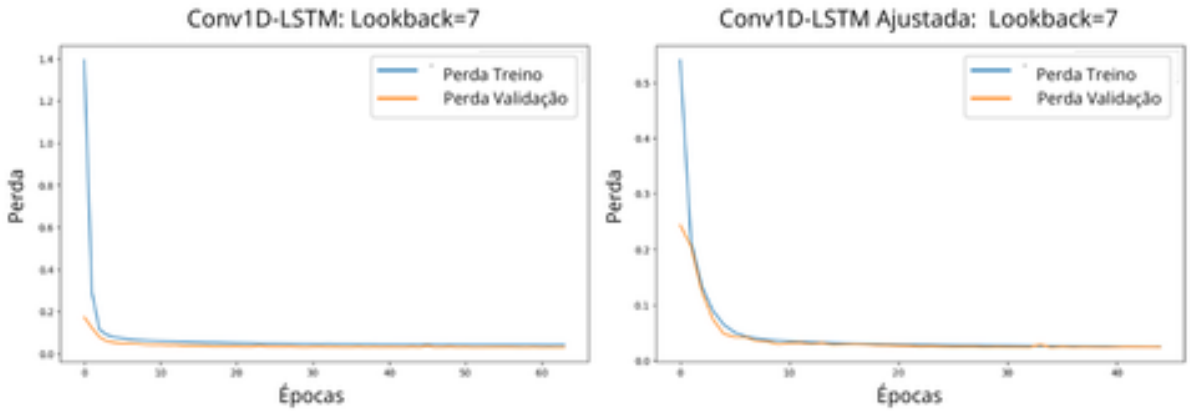


Fonte: Elaborada pela autora (2024).

redução significativa na perda de validação em comparação ao modelo não ajustado, evidenciando a eficácia da otimização de hiperparâmetros.

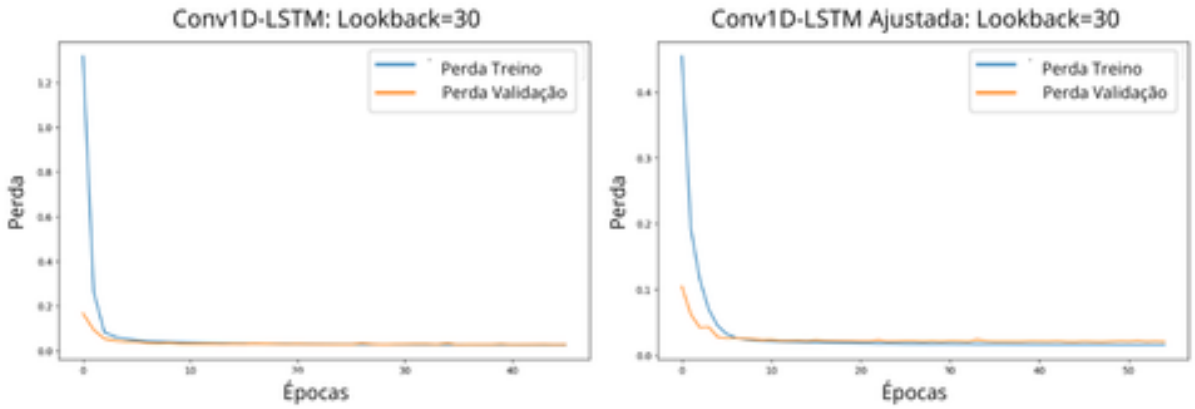
No *lookback* de 30 dias (Figura 34), o modelo ajustado continua a demonstrar melhor performance, com uma discrepância menor entre as perdas de treino e validação, sugerindo uma capacidade preditiva mais robusta e generalizada. Os gráficos de perda do modelo Conv1D-LSTM, apresentados nas Figuras 35 e 36, mostram que o modelo ajustado, após a otimização dos hiperparâmetros, apresenta uma curva de perda mais estável e com menor variação tanto para o *lookback* de 7 dias quanto para o de 30

Figura 35 – Gráfico de perda para o modelo Conv1D-LSTM com *lookback* de 7 dias.



Fonte: Elaborada pela autora (2024).

Figura 36 – Gráfico de perda para o modelo Conv1D-LSTM com *lookback* de 30 dias.



Fonte: Elaborada pela autora (2024).

dias. Para o *lookback* de 7 dias (Figura 35), o modelo ajustado apresenta uma redução significativa na perda de validação, com uma rápida convergência da curva de treino, indicando uma melhoria considerável após a otimização.

No *lookback* de 30 dias, o comportamento do modelo segue uma tendência semelhante, com uma redução significativa nas perdas de treino e validação, sugerindo que a otimização dos hiperparâmetros foi eficaz em melhorar a generalização do modelo e em reduzir o *overfitting*. Os gráficos de perda apresentados evidenciam que a otimização de hiperparâmetros resultou em uma melhora significativa no desempenho dos modelos LSTM, Multiscale CNN-LSTM e Conv1D-LSTM. Esta melhora é refletida pela menor discrepância entre as curvas de perdas de treino e validação, além de

uma convergência mais rápida observada nos modelos ajustados. A otimização de hiperparâmetros demonstrou ser uma técnica eficaz para reduzir o *overfitting* e melhorar a capacidade de generalização dos modelos preditivos.

5 CONSIDERAÇÕES FINAIS

Os resultados demonstraram que o modelo Conv1D-LSTM apresentou o melhor desempenho entre os modelos de LSTM utilizados, alcançando o maior valor de R^2 e os menores valores de MAE, RMSE e MASE no conjunto de teste. No entanto, é importante destacar que os modelos LightGBM e XGBoost superaram consistentemente os modelos de LSTM, também apresentando desempenho competitivo, com a vantagem de um custo computacional significativamente menor, tornando-os alternativas eficientes em cenários onde a capacidade computacional é limitada.

Os achados deste estudo destacam também a importância de selecionar e ajustar corretamente os hiperparâmetros dos modelos, o que foi evidenciado pela melhoria substancial nas métricas de desempenho após a aplicação da Otimização Bayesiana. No entanto, a abordagem de ajuste de hiperparâmetros não é única. Técnicas alternativas, como Algoritmos Genéticos, Otimização de Enxame de Partículas (PSO) e Pesquisa Aleatória Estratificada, poderiam ser exploradas para verificar se oferecem resultados ainda mais satisfatórios ou se aceleram o processo de otimização.

Além disso, o sucesso da Conv1D-LSTM em capturar padrões temporais complexos levanta a possibilidade de combinar o poder preditivo das redes LSTM com a eficiência computacional dos modelos baseados em árvores de decisão. Uma arquitetura híbrida, que combine LightGBM com LSTM, pode ser uma direção promissora. Tal arquitetura poderia aproveitar a capacidade do LightGBM de manejar grandes volumes de dados e identificar interações complexas entre *features*, enquanto a LSTM lidaria com a modelagem das dependências temporais. Por fim, cabe enfatizar que os objetivos propostos estabelecidos para este trabalho foram atingidos. Com efeito, inicialmente, foi realizada uma revisão bibliográfica abrangente sobre os principais modelos de previsão de eventos baseados em Deep Learning, proporcionando uma base teórica sólida para o estudo.

O pré-processamento conjunto de dados da farmacêutica Rossmann foi realizado a fim de garantir a qualidade e a integridade das informações utilizadas nas análises subsequentes. Em seguida, a seleção de atributos foi realizada utilizando o algoritmo Boruta, o que possibilitou identificar as variáveis mais relevantes para a previsão de

vendas. Além disso, os hiperparâmetros dos modelos foram ajustados de maneira eficiente por meio de Otimização Bayesiana, otimizando o desempenho dos modelos selecionados. Por fim, uma análise comparativa foi conduzida, permitindo avaliar o desempenho de cada modelo em termos de eficiência e aplicabilidade.

Os resultados obtidos demonstram que os métodos aplicados foram eficazes e proporcionaram conhecimentos importantes para a previsão de vendas. A pesquisa contribui não apenas para o entendimento dos modelos de aprendizado profundo, mas também para a aplicação prática no setor varejista, destacando a importância da análise de dados e da modelagem preditiva.

Para trabalhos futuros, são propostas as seguintes direções:

- **Exploração de Outras Técnicas de Ajuste de Hiperparâmetros:** Implementar e comparar outras técnicas de ajuste de hiperparâmetros, como Algoritmos Genéticos, Otimização de Enxame de Partículas (PSO) e Pesquisa Aleatória Estratificada, a fim de avaliar se essas técnicas podem superar a Otimização Bayesiana em termos de eficiência e resultados.
- **Desenvolvimento de Arquiteturas Híbridas:** Investigar a implementação de uma arquitetura híbrida que combine a robustez dos modelos baseados em árvores de decisão, como LightGBM, com a capacidade das redes LSTM de modelar sequências temporais. Tal arquitetura poderia potencialmente melhorar a precisão preditiva ao combinar o melhor de ambos os mundos.
- **Análise de Robustez com Diferentes Conjuntos de Dados:** Testar as arquiteturas propostas em diferentes conjuntos de dados de séries temporais para avaliar a robustez e capacidade de generalização dos modelos em diferentes contextos e setores.

REFERÊNCIAS

AGNEW, M. D.; THORNES, J. E. The weather sensitivity of the uk food retail and distribution industry. **Meteorological Applications**, Cambridge University Press, v. 2, n. 2, p. 137–147, 1995.

ALMENTERO, B.; LI, J.; BESSE, C. Forecasting pharmacy purchases orders. In: **2021 IEEE 24th International Conference on Information Fusion (FUSION)**. [S.l.: s.n.], 2021.

ARCANJO, J. **Como selecionar as melhores Variáveis para o seu modelo com Boruta**. 2021. Disponível em: <https://medium.com/data-hackers/como-selecionar-melhores-vari%C3%A1veis-para-o-seu-modelo-com-boruta-ef7cbfb3fc35>.

ARUNRAJ, N.; AHRENS, D.; FERNANDES, M. Application of sarimax model to forecast daily sales in food retail industry. **International Journal of Operations Research and Information Systems**, v. 7, p. 1–21, 04 2016.

ARUNRAJ, N. S.; AHRENS, D. A hybrid seasonal autoregressive integrated moving average and quantile regression for daily food sales forecasting. **International Journal of Production Economics**, Elsevier, v. 170, p. 321–335, 2015.

BALLESTA, P. *et al.* Spectral-based classification of genetically differentiated groups in spring wheat grown under contrasting environments. **Plants**, v. 12, p. 440, 2023.

BANDARA, K. *et al.* Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In: ____. [S.l.: s.n.], 2019. p. 462–474. ISBN 978-3-030-36717-6.

BEUTEL, A. L.; MINNER, S. Safety stock planning under causal demand forecasting. **International Journal of Production Economics**, Elsevier, v. 140, n. 2, p. 637–645, 2012.

CHEN, J. *et al.* Sales forecasting using deep neural network and shap techniques. p. 135–138, 2021.

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**. ACM, 2016. (KDD '16). Disponível em: <http://dx.doi.org/10.1145/2939672.2939785>.

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**. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 785–794. ISBN 9781450342322. Disponível em: <https://doi.org/10.1145/2939672.2939785>.

CHEN, X.; PANG, Z.; PAN, L. Coordinating inventory control and pricing strategies for perishable products. **Operations Research**, v. 62, n. 2, 2014.

DAI, Y.; HUANG, J. A sales prediction method based on lstm with hyper-parameter search. **Journal of Physics: Conference Series**, v. 1756, p. 012015, 02 2021.

Deloitte Touche Tohmatsu Limited. **Global Powers of Retailing 2023**. 2023. <https://www.deloitte.com/global/en/Industries/consumer/analysis/global-powers-of-retailing.html>. Accessed: 23-02-23.

DONSELAAR, K. van *et al.* Analysis and forecasting of demand during promotions for perishable items. **International Journal of Production Economics**, v. 172, 11 2015.

ECEVIT, A. *et al.* Short-term sales forecasting using lstm and prophet based models in e-commerce. **ACTA INFOLOGICA**, v. 7, n. 1, p. 59–70, 2023. RESEARCH ARTICLE, E-Ticarette LSTM ve Prophet Esaslı Modeller Kullanarak Kısa Dönemli Satış Tahmini. Disponível em: <https://dergipark.org.tr/acin>.

FAWAZ, H. I. *et al.* Deep learning for time series classification: a review. **Data Mining and Knowledge Discovery**, Springer Science and Business Media LLC, v. 33, n. 4, p. 917–963, mar. 2019. ISSN 1573-756X. Disponível em: <http://dx.doi.org/10.1007/s10618-019-00619-1>.

FENG, T.; NIU, C.; SONG, Y. Short term e-commerce sales forecast method based on machine learning models. In: **Proceedings of the 2022 6th International Seminar on Education, Management and Social Sciences (ISEMSS 2022)**. [S.l.: s.n.], 2022. p. 1020–1030.

FILDES, R.; MA, S.; KOLASSA, S. Retail forecasting: Research and practice. **International Journal of Forecasting**, 2019.

FISHER, M.; RAMAN, A. Using Data and Big Data in Retailing. **Production and Operations Management**, v. 27, n. 9, p. 1665–1669, September 2018. Disponível em: <https://ideas.repec.org/a/bla/popmgt/v27y2018i9p1665-1669.html>.

FISHER, M. L. *et al.* Making supply meet demand in an uncertain world. **Harvard Business Review**, v. 72, n. 3, p. 83–94, 1994.

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. **The Annals of Statistics**, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189 – 1232, 2001. Disponível em: <https://doi.org/10.1214/aos/1013203451>.

GERS, F.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: Continual prediction with lstm. **Neural computation**, v. 12, p. 2451–71, 10 2000.

GIRI, C.; CHEN, Y. Deep learning for demand forecasting in the fashion and apparel retail industry. **Forecasting**, v. 4, n. 2, p. 565–581, 2022.

HE, Q.-Q.; WU, C.; SI, Y.-W. Lstm with particle swam optimization for sales forecasting. **Electronic Commerce Research and Applications**, v. 51, p. 101118, 2022. ISSN 1567-4223. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1567422322000023>.

HEWAMALAGE, H.; ACKERMANN, K.; BERGMEIR, C. Forecast evaluation for data scientists: common pitfalls and best practices. **Data Mining and Knowledge Discovery**, v. 37, 12 2022.

HOCHREITER, S.; SCHMIDHUBER, J. Lstm can solve hard long time lag problems. In: . [S.l.: s.n.], 1996. p. 473–479.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, v. 9, p. 1735–80, 12 1997.

HOUDT, G. V.; MOSQUERA, C.; NáPOLES, G. A review on the long short-term memory model. **Artificial Intelligence Review**, v. 53, 12 2020.

HU, P. *et al.* A hybrid model based on cnn and bi-lstm for urban water demand prediction. p. 1088–1094, 06 2019.

HUBEL, D.; WIESEL, T. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. **Journal of Physiology**, v. 160, p. 106–154, 1962.

INVESTOPEDIA. **Carrying Cost of Inventory**. 2023. Available at: <https://www.investopedia.com/terms/c/carryingcostofinventory.asp>, Accessed on: July 24, 2023.

JADHAV, T.; KAUNCHI, p.; DANDAWATE, Y. Future sales prediction for indian products using convolutional neural network-long short term memory. 11 2021.

KE, G. *et al.* Lightgbm: A highly efficient gradient boosting decision tree. In: GUYON, I. *et al.* (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2017. v. 30. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf>.

KHAN, M. A. *et al.* Effective demand forecasting model using business intelligence empowered with machine learning. **IEEE Access**, v. 8, p. 116013–116023, 2020.

KRAMAR, V.; ALCHAKOV, V. Time-series forecasting of seasonal data using machine learning methods. **Algorithms**, v. 16, n. 5, 2023. ISSN 1999-4893. Disponível em: <https://www.mdpi.com/1999-4893/16/5/248>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. Imagenet classification with deep convolutional neural networks. **Neural Information Processing Systems**, v. 25, 01 2012.

KURSA, M. B.; RUDNICKI, W. R. Feature selection with the boruta package. **Journal of Statistical Software**, v. 36, p. 1–13, 2010. Disponível em: <https://api.semanticscholar.org/CorpusID:16510361>.

LEWIS, C. **Demand Forecasting and Inventory Control**. 1. ed. Routledge, 1997. Disponível em: <https://doi.org/10.4324/9781856179898>.

LOUREIRO, A.; MIGUéIS, V.; SILVA, L. da. Exploring the use of deep neural networks for sales forecasting in fashion retail. **Decision Support Systems**, v. 114, p. 81–93, 2018.

MA, S.; FILDES, R.; HUANG, T. Demand forecasting with high dimensional data: The case of sku retail sales forecasting with intra-and inter-category promotional information. **European Journal of Operational Research**, v. 249, n. 1, p. 245–257, 2016.

NITHIN, S. S. J. *et al.* Retail demand forecasting using cnn-lstm model. 2022.

ORDóñEZ, F. J.; ROGGEN, D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. **Sensors**, v. 16, p. 115, 2016.

PATTERSON, J.; GIBSON, A. **Deep Learning: A Practitioners Approach**. [S.l.]: O'Reilly, 2017.

QUEMY, A. Two-stage optimization for machine learning workflow. **Information Systems**, v. 92, p. 101483, 12 2019.

RANA, S. M. S. Supply chain drivers and retail supply chain responsiveness: strategy as moderator. **International Journal of Management Practice**, v. 13, n. 1, 2020.

RGUIGA, G.; MOUTTAKI, N.; BENHRA, J. Contribution to sales forecasting based on recurrent neural network in the context of a moroccan company. 2020.

SHI, C. *et al.* Learning multiscale temporal–spatial–spectral features via a multipath convolutional lstm neural network for change detection with hyperspectral images. **IEEE Transactions on Geoscience and Remote Sensing**, 05 2022.

SHI, X. *et al.* Convolutional lstm network: a machine learning approach for precipitation nowcasting. In: **Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1**. Cambridge, MA, USA: MIT Press, 2015. (NIPS'15), p. 802–810.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. In: PEREIRA, F. *et al.* (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf>.

SRIVASTAVA, N. *et al.* Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 06 2014.

VARGAS, A. C. G.; CARVALHO, A. M. P.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: **Proceedings of the XXIX Conference on Graphics, Patterns and Images**. [S.l.: s.n.], 2016. v. 1, p. 1–4.

WEITZ, B.; LEVY, M. **Retailing Management**. McGraw-Hill Education, 2011. ISBN 9780073530024. Disponível em: <https://books.google.com.br/books?id=fMc7PgAACAAJ>.

WERBOS, P. Backpropagation through time: what it does and how to do it. **Proceedings of the IEEE**, v. 78, n. 10, p. 1550–1560, 1990.

WU, N. Analysis of arima, lightgbm, xgboost, and lstm models for stock prediction. **Applied and Computational Engineering**, v. 54, p. 76–81, 03 2024.

ZENTES, J.; MORSCHETT, D.; SCHRAMM-KLEIN, H. **Strategic Retail Management**. [S.l.]: Gabler Verlag, 2012.

ZHANG, L. *et al.* Time series forecast of sales volume based on xgboost. **Journal of Physics: Conference Series**, v. 1873, p. 012067, 04 2021.