UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DANILO TURKIEVICZ DOS SANTOS

Proposta de arquitetura de AutoML para aprendizado de múltiplos estimadores de séries temporais

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[ ] Dissertação [X] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Danilo Turkievicz dos Santos

3. Título do trabalho

Proposta de arquitetura de AutoML para aprendizado de múltiplos estimadores de séries temporais

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Anderson Da Silva Soares**, **Professor do Magistério Superior**, em 19/11/2025, às 22:11, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Danilo Turkievcz Dos Santos**, **Discente**, em 21/11/2025, às 11:36, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5800140** e o código CRC **52CDE479**.

Referência: Processo nº 23070.051435/2025-51 SEI nº 5800140

DANILO TURKIEVICZ DOS SANTOS

Proposta de arquitetura de AutoML para aprendizado de múltiplos estimadores de séries temporais

Tese apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito para obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação.

Linha de pesquisa: Sistemas Inteligentes e Aplicações.

Orientador: Prof. Anderson da Silva Soares

Goiânia
2025

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Santos, Danilo Turkievicz dos
Proposta de arquitetura de AutoML para aprendizado de múltiplos estimadores de séries temporais [manuscrito] / Danilo Turkievicz dos Santos. - 2025.
119 f.: il.

Orientador: Prof. Dr. Anderson da Silva Soares.
Tese (Doutorado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2025.
Bibliografia. Apêndice.

1. séries temporais. 2. automl. 3. clusterização. 4. agrupamento. I. da Silva Soares, Anderson, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

ATA DE DEFESA DE TESE

Ata nº **27** da sessão de Defesa de Tese de **Danilo Turkievicz dos Santos**, que confere o título de Doutor em Ciência da Computação, na área de concentração em Ciência da Computação.

Aos vinte dias do mês de outubro de dois mil e vinte e cinco, a partir das oito horas, via sistema de webconferência, realizou-se a sessão pública de Defesa de Tese intitulada **"Proposta de arquitetura de AutoML para aprendizado de múltiplos estimadores de séries temporais"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor Anderson da Silva Soares (INF/UFG) com a participação dos demais membros da Banca Examinadora: Professor Doutor Arlindo Rodrigues Galvão Filho (INF/UFG), membro titular interno; Professor Doutor Rafael Teixeira Sousa (UFMT), membro titular externo; Professor Doutor Rodrigo Zempulski Fanucchi (Copel Distribuição S.A.), membro titular externo e Professor Doutor Vinicius Sebba Patto (INF/UFG), membro titular externo. A realização da banca ocorreu por meio de videoconferência. Durante a arguição os membros da banca não fizeram sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Tese, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor Anderson da Silva Soares, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos vinte dias do mês de outubro de dois mil e vinte e cinco.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Anderson Da Silva Soares**, **Professor do Magistério Superior**, em 20/10/2025, às 10:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **RODRIGO ZEMPULSKI FANUCCHI**, **Usuário Externo**, em 20/10/2025, às 10:47, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Rafael Teixeira Sousa**, **Usuário Externo**, em 20/10/2025, às 10:48, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Vinicius Sebba Patto**, **Professor do Magistério Superior**, em 20/10/2025, às 10:48, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Danilo Turkievcz Dos Santos**, **Discente**, em 20/10/2025, às 11:23, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Arlindo Rodrigues Galvao Filho**, **Professor do Magistério Superior**, em 21/10/2025, às 13:40, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5730129** e o código CRC **D99E8D30**.

Referência: Processo nº 23070.051435/2025-51 SEI nº 5730129

Agradecimentos

A conclusão desta tese representa o fim de uma jornada de doutorado que, além dos desafios intelectuais inerentes, foi uma prova de resistência pessoal. Conduzir uma pesquisa desta magnitude em grande parte sob as circunstâncias extraordinárias e adversas impostas pela pandemia global adicionou uma camada de dificuldade que tornou o apoio de cada pessoa e instituição ainda mais vital. Este trabalho não teria sido possível sem eles.

Expresso minha mais profunda gratidão ao meu orientador, Prof. Dr. Anderson Soares da Silva. Sua orientação, rigor técnico, paciência e disponibilidade foram fundamentais. Agradeço pela confiança depositada desde o início, pelos ensinamentos que transcenderam a pesquisa e pela capacidade de guiar este trabalho por meio de suas fases mais complexas.

Agradeço aos membros da banca examinadora, Prof. Dr. Arlindo Rodrigues Galvão Filho, Prof. Dr. Vinicius Sebba Patto, Prof. Dr. Rafael Teixeira Sousa e Dr. Rodrigo Zempulski Fanucchi, por dedicarem seu tempo e conhecimento à leitura e avaliação desta tese. As vossas arguições e valiosas sugestões foram cruciais para o amadurecimento e aprimoramento da qualidade final desta pesquisa.

Estendo meus agradecimentos à Universidade Federal de Goiás (UFG) e ao seu corpo docente, pelo ambiente acadêmico que fomentou esta pesquisa. Em especial, ao Centro de Excelência em Inteligência Artificial (CEIA), que forneceu não apenas a infraestrutura, mas o contexto prático e os dados que formaram a base empírica deste estudo.

À minha família, em especial aos meus pais e sogros, meu muito obrigado. Vosso apoio incondicional, vossa torcida e vossos sacrifícios silenciosos ao longo de todos estes anos foram o alicerce que me permitiu perseguir este objetivo.

Por fim, e de forma mais profunda, dedico esta conquista à minha esposa, Regine Baptista Venturi. Esta tese é o resultado direto do seu amor, paciência infinita e compreensão nas inúmeras horas de ausência. Você foi meu suporte nos momentos de dúvida e minha maior incentivadora. Esta vitória não é apenas minha, é nossa.

Resumo

Santos, Danilo. **Proposta de arquitetura de AutoML para aprendizado de múltiplos estimadores de séries temporais**. Goiânia, 2025. 119p. Tese de Doutorado. Instituto de Informática, Universidade Federal de Goiás.

A previsão de demanda no setor varejista é uma tarefa de alta complexidade, caracterizada pela vasta heterogeneidade de padrões em milhares de séries temporais. Abordagens tradicionais, como modelos únicos customizados, são custosas e pouco escaláveis, enquanto modelos fundacionais globais ainda apresentam desafios de aplicabilidade prática. Diante desse cenário, esta tese propõe e desenvolve uma metodologia de *Auto Machine Learning* (AutoML) que eleva a robustez e a eficiência computacional das predições, fundamentada na estratégia de agrupar para depois prever (*cluster-then-forecast*). O pilar desta metodologia é uma abordagem de clusterização inédita, que emprega *Self-Organizing Maps* (SOM) utilizando a métrica *LikelihoodDistance* para identificar séries com processos geradores subjacentes similares. A arquitetura foi validada em um cenário real e desafiador, com dados de vendas do varejo farmacêutico. Os resultados demonstraram que a abordagem por amostragem, derivada da clusterização, foi particularmente eficaz, conseguindo identificar os modelos de melhor desempenho a partir de pequenas amostras de séries. O resultado central da pesquisa é que a arquitetura proposta se mostrou competitiva nas lojas e métricas avaliadas, exibindo consistência. Adicionalmente, o sucesso da clusterização fornece indicativos de que a similaridade entre os modelos geradores das séries é um fator relevante para a seleção de uma técnica de previsão adequada. O trabalho contribui, assim, com um *framework* de AutoML que mitiga o problema da seleção de modelos em ambientes heterogêneos, oferecendo uma solução de previsão mais estável, escalável e computacionalmente viável para o setor varejista.

Palavras–chave

Previsão de Séries Temporais, Clusterização, AutoML

Abstract

Santos, Danilo. **Proposta de arquitetura de AutoML para aprendizado de múltiplos estimadores de séries temporais**. Goiânia, 2025. 119p. PhD. Thesis. Instituto de Informática, Universidade Federal de Goiás.

Demand forecasting in the retail sector is a highly complex task, characterized by the vast heterogeneity of patterns across thousands of time series. Traditional approaches, such as custom single models, are costly and poorly scalable, while global foundational models still face challenges in practical applicability. In this context, this thesis proposes and develops an AutoML methodology that enhances the robustness and computational efficiency of predictions, based on the cluster-then-forecast strategy. The cornerstone of this methodology is a novel clustering approach that employs SOM using the LikelihoodDistance metric to identify series with similar underlying generative processes. The architecture was validated in a real-world and challenging scenario, using sales data from the pharmaceutical retail sector. The results demonstrated that the sampling-based approach, derived from the clustering, was particularly effective, successfully identifying the best-performing models from small samples of series. The central finding of the research is that the proposed architecture not only proved to be competitive across all stores and evaluated metrics but also exhibited remarkable consistency and reliability. Furthermore, the success of the clustering provides strong evidence that the similarity between the series' generative models is a determining factor in selecting the most accurate forecasting technique. This work thus contributes an AutoML framework that mitigates the model selection problem in heterogeneous environments, offering a more stable, scalable, and computationally viable forecasting solution for the retail sector.

Keywords

Time Series, Clustering, AutoML

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

SOM *Self-Organizing Maps*
ML *Machine Learning*
ARIMA *AutoRegressive Integrated Moving Average*
ES *Exponential Smoothing*
AutoML *Auto Machine Learning*
ADI *Average Demand Interval*
CV *Coefficient of Variation*
GRU *Gated Recurrent Unit*
GRN *Gated Residual Networks*
SKU *Stock Keeping Unit*
SES *Single Exponential Smoothing*
SARIMA *Seasonal AutoRegressive Moving Average*
TFT *Temporal Fusion Transformer*
LSTM *Long Short-Term Memory*
TSB Teunter, Syntetos & Babai
SSE *Sum of Squared Errors*
PACF *Partial Autocorrelation Function*
KPSS Kwiatkowski-Phillips-Schmidt-Shin
ADF Augmented Dickey Fuller
CH Canova-Hansen
OCSB Osborn, Chui, Smith e Birchenhall
AICc *Akaike's Information Criterion with correction*
RNN *Recurrent Neural Network*
GBDT *Gradient boosting decision tree*
GOSS *Gradient-based One-Side Sampling*
EFB *Exclusive Feature Bundling*
HPO *Hyperparameter Optimization*
TSFRESH *Time Series Feature extraction based on scalable hypothesis tests*
SMAC *Sequential Model Algorithm Configuration*
TPE *tree-structured Parzen estimator*

RF *Random Forest*
CEIA Centro de Excelência em Inteligência Artificial
RMSE *Root Mean Square Error*
SMAPE *Symmetric Mean Absolute Percentage Error*
RMSSE *Root Mean Squared Scaled Error*
MASE *Mean Absolute Scaled Error*
MAPE *Mean Absolute Percentage Error*
DTW *Dynamic Time Warping*
MLOps *Machine Learning Operations*
BMU *Best Matching Unit*
SBD *Shape-Based Distance*
PFSA *Probabilistic Finite State Automata*
ACF *Autocorrelation Function*

CAPÍTULO 1

Introdução

Um conjunto de dados ordenados no tempo caracteriza uma série temporal [3]. As séries temporais podem ser classificadas em dois grandes grupos — contínuas e discretas — de acordo com a forma de observação desses dados. São denominadas contínuas quando a variável não é observada em tempos regulares. Distintivamente, as discretas são as que possuem dados amostrados em uma estampa de tempo definida [20].

A aplicação de séries temporais abrange diferentes especialidades, dentre elas, meteorologia, evolução de preço de ações, transmissão de dados e evolução demográfica [95]. Em certos casos, as projeções de uma série são perfeitamente previsíveis e, desta forma, nominadas determinísticas, tal qual a definição do Sistema Internacional de Unidades, que se fundamenta em ciclos de radiação emitida pelo Césio-133, para esta classificação. Em contrapartida, quando os valores passados são suficientes para prever valores com incerteza, a série é classificada como estocástica [29]. A previsibilidade é uma característica fundamental das séries temporais. O fato de as observações sucessivas serem, em sua maioria, dependentes do tempo torna possível prever um valor futuro fundamentado em medidas passadas.

O interesse por predição de valores das séries temporais abrange diversos segmentos, incluindo: saúde, financeiro, social, comercial, industrial, esportivo, meteorológico, etc. Devido à sua relevância e abrangência, este já é um assunto que vem sendo estudado há algumas décadas. No entanto, a busca contínua por melhores técnicas, motivada principalmente pelos resultados financeiros envolvidos, fornece subsídios permanentes para melhores predições [59].

As empresas varejistas são um grande mercado e fonte de incentivos para os acadêmicos desta área [38]. Com a competitividade cada vez mais acirrada e melhores bases de dados, há inúmeros trabalhos abordando técnicas e resultados para este ramo em particular. A evolução dos modelos de previsão responde continuamente às limitações de seus predecessores. Modelos clássicos como o ARIMA, formalizado por [16], e o Holt-Winters, concebido para previsões de negócios [97], estabeleceram a base para lidar com tendências e sazonalidade, porém, eram restritos por pressupostos de linearidade. A ascensão do *Machine Learning* (ML) superou essa rigidez, com o *Random Forest* (RF) em [17],

oferecendo robustez e a capacidade de avaliar a importância das variáveis, embora sem poder de extrapolação de tendências. Um importante marco de evolução veio com os modelos de *Gradient Boosting*, como o XGBoost [21] e, principalmente, o LightGBM [52], projetados para escalabilidade e velocidade em grandes volumes de dados. A importância desta classe de modelos foi definitivamente comprovada pela Competição M5, considerada o *benchmark* mundial para previsão de varejo em alta granularidade. A M5 demonstrou que, para esta classe de dados — notavelmente incluindo séries intermitentes nos níveis mais baixos — os modelos de *Gradient Boosting* (como o LightGBM) superaram consistentemente as arquiteturas de *Deep Learning*, estabelecendo um novo estado da arte prático [67]. A análise crítica da Competição M5, detalhada por [67], consolidou a superioridade dessas abordagens, sendo que o sucesso foi atribuído não apenas ao algoritmo, mas a uma massiva engenharia de atributos e ao uso de modelos globais de *Cross-Learning*. Contudo, o estudo também revela uma limitação crítica: apenas 7,5% das equipes superaram um *benchmark* estatístico simples, indicando que a aplicação eficaz de ML é um desafio considerável e que o sucesso prático se tornou mais dependente de habilidades de experimentação e mineração de dados do que de conhecimento teórico aprofundado.

A vanguarda da pesquisa, por sua vez, avança em direção a arquiteturas de *Deep Learning*, as quais são projetadas para aprender dependências temporais de forma automática. As redes LSTM, propostas por [41], para resolver o problema do desaparecimento do gradiente e sua variante mais eficiente, a *Gated Recurrent Unit* (GRU) [24], foram pioneiras na captura de padrões de longo prazo. Inovações mais recentes desafiaram os paradigmas existentes: o N-BEATS [74] demonstrou que uma arquitetura de *Deep Learning* "pura", sem componentes específicos para séries temporais, poderia atingir o estado da arte, enquanto o TFT [60] foi projetado especificamente para o varejo, buscando unir alto desempenho com interpretabilidade. Apesar do potencial teórico, a análise de [67] evidencia que os modelos de *Deep Learning* não dominaram a prática do varejo na Competição M5, e foram superados por modelos de *Gradient Boosting* bem ajustados. O alto custo computacional e, fundamentalmente, a falta de interpretabilidade — o problema da "caixa-preta" — permanecem como barreiras críticas para sua adoção em um ambiente de negócios que exige embasamento para decisões de alto risco, mantendo o *Deep Learning* como uma vanguarda de pesquisa, enquanto o *Machine Learning* pragmático define o estado da arte na aplicação.

Uma vertente recente na busca por modelos generalistas mais poderosos é o desenvolvimento de *Foundation Models* para séries temporais, pré-treinadas em vastos conjuntos de dados, com a premissa de que poderiam transferir o conhecimento aprendido para novas tarefas com pouca ou nenhuma adaptação (*Zero-Shot Learning*) [73]. No entanto, a eficácia desses modelos ainda é limitada, pois seu desempenho tende a se degradar

em séries com ruído, longos períodos de sazonalidade ou formas complexas, características comuns nos dados de varejo [37]. Ademais, evidências demonstram que o conhecimento transferido de outros domínios não se traduz em ganhos de desempenho, com modelos mais simples frequentemente superando-os [88]. Embora demonstrem potencial, sua aplicação prática no varejo enfrenta barreiras como o alto custo computacional, a natureza de "caixa-preta" que dificulta a interpretabilidade e uma rigidez arquitetônica que limita a incorporação de variáveis causais cruciais, como promoções e eventos locais [85].

A multiplicidade de séries no varejo introduz uma complexidade singular, com a demanda exibindo uma alta diversidade de padrões que vão de regulares a erráticos, intermitentes ou esporádicos [87], o que atualmente é um obstáculo para modelos generalistas. A questão da "ultra-alta dimensionalidade" inerente aos dados de varejo, foi investigada por [64].Neste estudo abordam-se as interações promocionais intra e intercategorias para um único *Stock Keeping Unit* (SKU) e como o espaço de variáveis candidatas pode atingir a ordem de dezenas de milhares, tornando o custo de cada ciclo de treinamento proibitivo.

O equilíbrio entre acurácia preditiva e sustentabilidade computacional é um dilema para o retreinamento de modelos. O imperativo econômico para tal acurácia é inequívoco: a distorção de inventário custou ao varejo global $1.77 trilhão em 2023 [48], com melhorias de 10–20% na acurácia podendo se traduzir em aumentos de receita de 2–3% [68]. Neste contexto, a sabedoria convencional dita que o retreinamento frequente de modelos globais é essencial para mitigar o desvio do modelo (*model drift* ou *concept drift*). Apesar do trabalho de [98] desafiar esta premissa ao revelar que, para o conjunto de dados do M5 (diário), houve baixa degradação da acurácia com retreinos menos periódicos e redução no custo computacional de até 90%, para o conjunto de dados VN1 [1], *benchmark* de 2024 que se notabilizou por utilizar dados de varejo semanais com elevada intermitência, os resultados diferem bastante. Nos casos de dados semanais, como os do VN1, a penalidade por retreinos menos frequentes é significativamente mais pronunciada. Além disso, a análise de [98] foi exclusiva para modelos globais.

Além do dilema do retreinamento, a competição VN1 forneceu um panorama crucial sobre o estado da arte das arquiteturas de modelo. O *dataset* VN1, focado em dados semanais de alta intermitência, expôs os limites de abordagens excessivamente generalistas. Notavelmente, os resultados da competição [27] não consagraram uma única arquitetura. O pódio foi dividido entre um *Foundation Model* pré-treinado (Moirai) [99] e um *ensemble* de modelos especializados, que utilizou profundo conhecimento de domínio e modelagem detalhada [27]. Esta dualidade de vencedores reforça uma premissa central desta tese: a especialização e a seleção de modelos (como proposto pela arquitetura AutoML) permanecem uma estratégia competitiva e de ponta, mesmo na era dos modelos fundacionais.

Soma-se à dimensionalidade, aspectos como a diversidade de mercados e cultu-

ras, que trazem a necessidade da adaptação frequente e até o desenvolvimento de novos modelos de predições, como os emergentes modelos fundacionais [58]. Por exemplo, é esperado que um modelo de previsão de consumo de carne bovina no Brasil não apresente desempenho semelhante quando aplicado à Índia. Por isso, cada vez mais, buscam-se modelos capazes de serem adaptáveis a diferentes cenários e que sejam autoparametrizáveis,reduzindo a necessidade de especialistas nos ajustes manuais dos parâmetros.

A implementação de modelos de aprendizado de máquina em ambientes produtivos, embora promissora, acarreta custos de manutenção que superam o desenvolvimento inicial. Tal fenômeno é adequadamente descrito pelo conceito de "dívida técnica", um passivo implícito gerado pela escolha de soluções rápidas em detrimento de abordagens mais robustas [82, 81]. Em sistemas de ML, essa dívida é agravada por fatores como o entrelaçamento de componentes e, de forma crítica para séries temporais, o inevitável *concept drift*, no qual o desempenho se degrada conforme as propriedades estatísticas dos dados se alteram [100, 62]. Essencialmente, o *concept drift* viola o pressuposto fundamental de que a relação entre as variáveis de entrada (X) e a variável alvo (Y), ou seja, a distribuição de probabilidade condicional $P(Y|X)$ é estacionária. No domínio do varejo, este desvio é particularmente agudo, podendo ser abrupto (início de uma pandemia, novas promoções de concorrentes) ou gradual (mudanças sazonais ou alteração na preferência do consumidor). O modelo, treinado em padrões históricos, torna-se obsoleto porque os padrões que ele aprendeu não mais representam a realidade do mercado. A mitigação deste desvio exige um ciclo contínuo de monitoramento e retreinamento, atividades que, embora essenciais, incorrem em custos operacionais recorrentes e significativos [6]. Com a popularidade do uso desses modelos e a facilidade da implantação, muitas empresas, incluindo as pequenas, têm investido na terceirização dessas predições ou no uso de computação em nuvem, elevando o custo das suas atividades [2]. Esses custos estão aliados principalmente ao tempo de uso e recursos empregados, por isso, é necessário balancear a complexidade dos modelos com os seus resultados obtidos.

Apesar da relevância prática desses desafios, a investigação acadêmica sobre o tema permanece incipiente. A própria disciplina de Operações de Aprendizado de Máquina *Machine Learning Operations* (MLOps), designada para gerenciar tais custos, é descrita na literatura como "em sua infância" e "fragmentada entre disciplinas" [55]. Há um foco desproporcional da academia na otimização de métricas de acurácia, tratando a viabilidade econômica como uma consideração posterior. Esta abordagem, criticada por [93] como um "processo sequencial" falho, ignora que múltiplos modelos com desempenho preditivo similar podem acarretar custos operacionais drasticamente distintos. A consequência direta é a carência de arcabouços para a análise do Custo Total de Propriedade (TCO), que, na prática, pode ser ordens de magnitude superiores aos custos de treinamento [83, 25]. Destarte, a desconexão entre a busca incessante por precisão e a

sustentabilidade econômica dos modelos evidencia uma negligência que justifica a investigação de métodos que integrem o custo como um objetivo de otimização.

Neste contexto, emerge a automação do aprendizado de máquina, ou AutoML, como uma solução estratégica e pragmática para os desafios de previsão em larga escala no varejo [96]. O AutoML visa automatizar o ciclo de vida da modelagem preditiva, desde a engenharia de atributos até a seleção de algoritmos e a otimização de seus hiperparâmetros [3]. A competição M5, um marco para o setor, forneceu um roteiro claro para a automação: as soluções vencedoras, embora manuais, validaram os princípios de um *pipeline* de AutoML sofisticado, convergindo universalmente para *ensembles* de modelos baseados em árvores (LightGBM) alimentados por uma rica e sistemática engenharia de características [67]. A ausência de *frameworks* de AutoML de prateleira no pódio não indica uma falha do conceito, mas sim uma oportunidade para o desenvolvimento de sistemas especializados que codifiquem essas melhores práticas comprovadas. A mais recente competição VN1 (2024) introduziu os modelos de fundação como uma nova forma de automação, com uma solução baseada no *fine-tuning* do modelo Moirai alcançando o primeiro lugar [99]. Contudo, é revelador que esta posição foi compartilhada com um *ensemble* customizado e orientado por especialistas, reforçando que a especialização de domínio e a modelagem detalhada permanecem no auge do desempenho [27].

A necessidade de sistemas AutoML especializados é impulsionada por uma confluência de fatores mercadológicos e tecnológicos. Primeiramente, o AutoML aborda a crescente demanda por capacidades analíticas frente à escassez de cientistas de dados e engenheiros de ML, democratizando o acesso a técnicas preditivas sofisticadas [96, 3]. Em segundo lugar, a computação em nuvem atua como um catalisador econômico, transformando o alto custo de capital (CapEx) em infraestrutura em um modelo de despesa operacional (OpEx) variável, tornando a construção de sistemas AutoML proprietários uma iniciativa estratégica viável [63]. Finalmente, a própria tecnologia AutoML amadureceu, com *frameworks* modernos que empregam técnicas robustas de *ensembling*, espelhando as estratégias vencedoras da M5 [84].

Adicionalmente, uma vertente de pesquisa recente alinha o objetivo da clusterização diretamente com o da previsão, tratando a seleção de modelos como um problema de otimização. Em vez de agrupar séries por similaridade de forma ou por características estatísticas, esta abordagem redefine a dissimilaridade como o erro de previsão incorrido por um determinado modelo. Metodologias, como a proposta por [101], formalizam esta concepção, gerando partições que são ótimas em termos de acurácia preditiva agregada e assegurando que as séries sejam agrupadas com base em sua previsibilidade compartilhada. Tal abordagem fundamenta a alocação estratégica de recursos computacionais, permitindo que modelos de alta complexidade sejam aplicados seletivamente aos clusters

que deles se beneficiam, enquanto grupos de séries com padrões mais simples podem ser tratados por modelos mais parcimoniosos, otimizando assim o balanço entre acurácia e custo computacional [98].

1.1 Proposta de trabalho

A revisão da literatura exposta neste capítulo revela um dilema de otimização central na previsão de demanda para o varejo moderno. Por um lado, a acurácia preditiva é economicamente imperativa, dada a complexidade de lidar com milhares de SKUs cujos dados são frequentemente dominados pela intermitência [87, 64] e os altos custos associados à distorção de inventário [48, 68]. Por outro lado, a necessidade de retreinamento frequente para mitigar o *concept drift* [100, 62], inerente a dados não estacionários, impõe um fardo significativo em termos de MLOps, gerando "dívida técnica" e custos operacionais recorrentes [82, 6].

Neste cenário, abordagens de modelo único mostram-se insuficientes, enquanto modelos fundacionais generalistas, embora promissores, ainda enfrentam barreiras de aplicabilidade prática no varejo de alta granularidade [37, 88]. A solução mais pragmática, corroborada pelos resultados das competições M5 e VN1 [67, 27], aponta para sistemas de AutoML especializados. Contudo, a aplicação direta de um *pipeline* de AutoML, que envolve o treinamento de múltiplos estimadores, a dezenas de milhares de séries temporais torna-se computacionalmente proibitiva. É precisamente este gargalo que a presente tese se propõe a resolver.

É explorada uma linha de pesquisa, fundamentada na estratégia *cluster-then-forecast* [101]. A justificativa para esta escolha reside na inadequação de métodos tradicionais de clusterização para o tipo de dado em questão. Abordagens *shape-based*, como *k-Shape* [76] ou *Dynamic Time Warping* (DTW), que agrupam séries por similaridade morfológica, obtêm bons resultados, mas pecam na interpretabilidade. Em contrapartida, esta tese investiga uma abordagem *model-based*, cuja premissa é agrupar séries que, embora visualmente distintas, compartilham o mesmo *processo estocástico gerador subjacente*, ou seja, a mesma "física" estatística [43]. A "dificuldade" introduzida por esta abordagem reside no custo computacional inicial necessário para inferir os modelos estocásticos (como Autômatos Finitos Probabilísticos ou *Probabilistic Finite State Automata* (PFSA)) e treinar o mapa de clusterização (como o SOM). Contudo, este custo inicial representa um investimento amortizável que habilita uma arquitetura de AutoML substancialmente mais eficiente em longo prazo.

Para abordar o problema de forma estruturada e testar a viabilidade desta proposta, foram formuladas duas hipóteses centrais. A **primeira hipótese (H1)** postula que a similaridade no processo estocástico gerador de séries intermitentes, quantificada por

uma métrica *model-based* como a `LikelihoodDistance` derivada de PFSA, constitui um preditor estatisticamente significativo. A **segunda hipótese (H2)** propõe que uma arquitetura de AutoML utilizando seleção de modelos por amostragem, guiada pela clusterização *model-based* (conforme H1), pode alcançar acurácia preditiva estatisticamente comparável ao *benchmark* (treino completo de todos os modelos), porém com um custo computacional (tempo de treino) ordens de magnitude inferior.

Assim, o **objetivo geral** desta tese é propor, implementar e validar uma nova arquitetura de AutoML para previsão de demanda em larga escala, propondo uma abordagem de clusterização SOM em conjunto com uma métrica *model-based*, trazendo interpretabilidade, balanço entre acurácia preditiva e sustentabilidade computacional.

Para alcançar este objetivo geral e validar as hipóteses formuladas, foram definidos os seguintes **objetivos específicos**. Primeiramente, é implementada a contribuição métrica central: a métrica de dissimilaridade `LikelihoodDistance`, baseada na inferência de PFSA, para ser utilizada como função de distância em um mapa auto-organizável (SOM). Em seguida, é testada a Hipótese 1, validando estatisticamente, por meio da Análise de Variância Multivariada Permutacional (PERMANOVA), se a clusterização resultante agrupa séries que compartilham o mesmo modelo preditivo ótimo. Posteriormente, é construída a arquitetura de AutoML (*cluster-then-forecast*), que empregará os *clusters* gerados para realizar uma seleção eficiente de modelos por amostragem percentual. Finalmente, é valida-se a Hipótese 2, quantificando o *trade-off* da arquitetura proposta em um conjunto de dados real de varejo farmacêutico (Capítulo 6). Esta validação envolverá uma comparação rigorosa com o *benchmark* (treino empírico de todos os modelos) em termos de acurácia preditiva (utilizando métricas como RMSE, SMAPE, RMSSE) e custo computacional (tempo de processamento).

1.2 Relação da Pesquisa com Projetos de Pesquisa, Desenvolvimento e Inovação

A articulação estratégica entre a pesquisa acadêmica e o ambiente corporativo configura-se como um paradigma robusto para a aceleração da inovação, atuando como uma síntese entre a investigação de fronteira e a aplicação pragmática orientada ao mercado. A premissa fundamental desta aliança reside na translação do capital intelectual e dos achados científicos, maturados no ambiente universitário, para a resolução de desafios tecnológicos complexos e a geração de valor tangível no ambiente de negócios [32]. Este processo de co-criação permite que a indústria acesse metodologias rigorosas e explore novos domínios do conhecimento, mitigando os riscos inerentes à pesquisa e desenvolvimento (P&D) intramuros. Em contrapartida, para a academia, a colaboração representa

um mecanismo de validação empírica de suas hipóteses e um canal direto para a transferência de tecnologia, cujo resultado não apenas fomenta a geração de patentes e licenciamentos, mas também retroalimenta o ciclo de pesquisa com problemas de relevância imediata, garantindo que a produção científica tenha um impacto mensurável e direto [77].

A presente tese encontra seu ponto de partida na trajetória de pesquisa do autor, marcada pela participação em dois projetos de pesquisa focados na predição de vendas para o varejo farmacêutico: um concluído na fase inicial deste doutorado e outro atualmente em andamento. Tais projetos serviram como laboratório para a validação das hipóteses que norteiam este trabalho, materializando a colaboração como um vetor de benefício mútuo. De um lado, a empresa parceira obteve ganhos de eficiência por meio da implementação de modelos preditivos e repasse de conhecimento; de outro, o grupo de pesquisa pôde desenvolver, refinar e validar suas metodologias em um cenário real, com dados de alta complexidade. Os resultados do primeiro projeto e o progresso da iniciativa vigente, ao gerarem inovação de aplicação direta, atestam de forma inequívoca que a intersecção entre a pesquisa acadêmica e os desafios do varejo farmacêutico é um tema atual, de extrema relevância e com impacto econômico localizado. Assim, essa imersão prática e contínua não apenas inspira, mas fundamenta a necessidade desta investigação, que busca sistematizar e aprofundar o conhecimento gerado a partir dessas interações.

1.3 Organização da tese

Esta tese está estruturada em sete capítulos. No Capítulo 1 apresenta-se a introdução, contextualizando o desafio da previsão de demanda no varejo e o dilema entre acurácia e custo computacional, culminando nos objetivos e hipóteses desta pesquisa.

No Capítulo 2, estabelece-se o arcabouço técnico dos modelos preditivos, realizando uma revisão dos métodos clássicos, estatísticos e das arquiteturas de ML e *Deep Learning* que compõem o *pool* de estimadores.

No Capítulo 3, são aprofundadas as técnicas de agrupamento para séries temporais. São discutidos os paradigmas *shape-based* e *model-based*, fundamentando a escolha das métricas de dissimilaridade, como a `LikelihoodDistance`, e dos algoritmos, como o SOM, que são centrais para esta tese.

No Capítulo 4, formaliza-se o conceito de Aprendizado de Máquina Automatizado (AutoML), dissecando o problema CASH e os desafios específicos de sua aplicação em séries temporais, como o *concept drift* e o custo operacional em larga escala, justificando a necessidade da arquitetura proposta.

No Capítulo 5, detalha-se a arquitetura de AutoML (*cluster-then-forecast*) proposta. É descrito o *pipeline* completo, desde o pré-processamento e engenharia de *features*

até a implementação das estratégias de seleção de modelos por amostragem (janela reduzida e conjunto reduzido).

No Capítulo 6, apresenta-se a validação empírica da metodologia em um *dataset* real de varejo farmacêutico. Neste capítulo, quantifica-se o *trade-off* entre acurácia preditiva e custo computacional, validando estatisticamente (via PERMANOVA) a eficácia da abordagem de clusterização *model-based*.

Finalmente, no Capítulo 7 são sumarizadas as contribuições do trabalho, reiterada a validação das hipóteses e propostas as direções para trabalhos futuros.

CAPÍTULO 2

Séries Temporais

A série temporal pode descrever uma função, por exemplo, os dados plotados em um gráfico. Desse modo, alguns valores que se presumem fora do contexto, denominados *outliers* ou pontos de mudança, são facilmente identificados, de igual maneira a variação cambial logo após o início de uma crise financeira. Também são objetivos das séries: explicar a relação entre duas variáveis, fazer a predição de valores ou controle de um processo, como quando aplicado a um processo de ajuste de velocidade de um motor.

Se uma série temporal não dispõe, ao longo do tempo, de grandes irregularidades na média, variância e se podem ser fortemente amortizados os efeitos de sazonalidades, então, ela pode ser enquadrada como uma série temporal estacionária. Tal fato é relevante em razão da viabilidade da aplicação da maior parte das teorias de probabilidade e de predição já desenvolvidas; em contrapartida, tais séries não representam os processos reais em sua completude. O emprego de tais teorias torna-se factível com a conversão das séries em estacionárias, proporcionada por: aplicação de filtros para retirada de *outliers*, retirada do efeito da tendência e eliminação dos efeitos de sazonalidade.

Para o escopo deste trabalho foram testados modelos das três famílias de métodos de predição: clássicos, compostos pelo *Naive Drift* e uma versão sazonal, *Naive Seasonal*; estatísticos, nos quais estão incluídas: a suavização exponencial de Holt-Winters [42], ARIMA [16], sua versão sazonal *Seasonal AutoRegressive Moving Average* (SARIMA), TSB [91], além dos mais recentes PROPHET [90] e THETA [4]; e ML, nos quais fazem parte os LightGBM [52], DeepAR [79], LSTM [41] e TFT [60].

2.1 *Naive*

Na sua implementação básica, o modelo prevê que os valores futuros são iguais ao último valor da série, sendo sua formulação apresentada pela Equação (2-1). Por sua simplicidade, baixo custo computacional é amplamente utilizado como um modelo de referência, além de aplicações em séries econômicas e financeiras [47].

No *Naive Drift* tem-se uma adaptação que permite corrigir os valores pela média dos valores passados, captando assim alguma tendência, tendo sua formulação na Equação (2-2).

A versão sazonal permite que os valores futuros se repitam com base na sazonalidade m, descrita na Equação (2-3). Combinando *Naive Drift* e *Naive Seasonal* é possível obter as vantagens de tendência oferecida pelo primeiro modelo e de sazonalidade do segundo, sendo apresentado como *Naive Drift Seasonal*. A Figura 2.1 ilustra as diferentes predições para os modelos apresentados.

$$\hat{y}_{T+h|T} = y_T \tag{2-1}$$

$$\hat{y}_{T+h|T} = y_T + h\left(\frac{y_T - y_1}{T-1}\right) \tag{2-2}$$

$$\hat{y}_{T+h|T} = y_{T+h-m} \tag{2-3}$$

em que:

y_T = valor da série no tempo T
h = horizonte de predição
$\hat{y}_{T+h|T}$ = valor previsto
m = sazonalidade

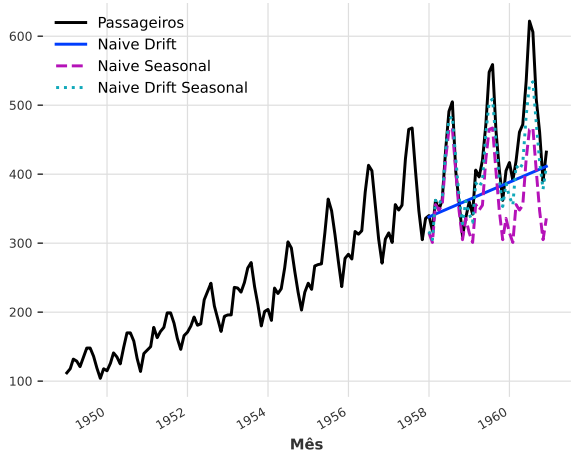


Figura 2.1: Predição utilizando diferentes modelos Naives

2.2 Suavização Exponencial

A suavização exponencial aplicada à predição de séries temporais tem sua origem na década de 50 no trabalho de Brown [18], com sua versão sazonal elaborada por Holt [42] e Winters [97].

O princípio da suavização exponencial reside no fato de que os fatos recentes possuem maior relevância em relação aos valores passados, sendo definida pela Equação (2-4) [47]. Dessa forma, a cada valor passado é aplicado um peso α exponencialmente proporcional à distância com o valor a ser previsto, suavizando exponencialmente as previsões.

$$\hat{y}_{T+1|T} = \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T l_0 \tag{2-4}$$

Na versão de Holt-Winters três aspectos são levados em consideração: média (l_t), tendência (b_t) e sazonalidade (s_t) cada um com seu respectivo parâmetro de suavização, α, β, γ respectivamente. Há duas formas de abordagem: aditiva e multiplicativa, no entanto, apresenta-se somente a versão aditiva, definida pela Equação (2-5) [47] e ilustrada pela Figura 2.2.

$$\begin{aligned} \hat{y}_{t+h|t} &= l_t + hb_t + s_{t+h-m} \\ l_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \end{aligned} \tag{2-5}$$

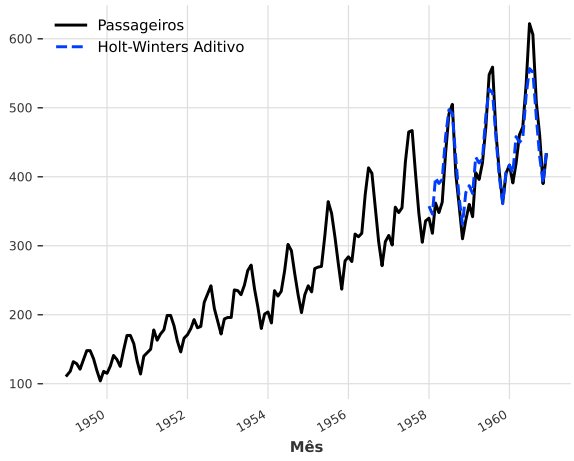


Figura 2.2: Predição utilizando suavização exponencial aditiva de Holt-Winters

Para a solução das Equações (2-4) e (2-5) é necessário um processo de otimização para obtenção dos valores dos parâmetros de suavização, sendo a minimização da soma dos erros ao quadrado ou *Sum of Squared Errors* (SSE) a otimização usualmente empregada.

2.3 *AutoRegressive Integrated Moving Average* (ARIMA)

O modelo *AutoRegressive Integrated Moving Average* (ARIMA) (*AutoRegressive Integrated Moving Average*), formalizado por Box e Jenkins [16], representa uma classe de modelos estatísticos robusta e amplamente utilizada para análise e previsão de séries temporais. Sua principal vantagem reside na capacidade de modelar dados não estacionários, que são prevalentes em cenários do mundo real, por meio da combinação sistemática de três componentes fundamentais: Auto-Regressão (AR), Integração (I) e Média Móvel (MA).

A componente **Auto-Regressiva (AR)** de ordem p, denotada AR (p), expressa o valor atual da série (y_t) como uma função linear de seus p valores passados ($y_{t-1}, \ldots, y_{t-p}$), adicionado a um termo de erro estocástico (ϵ_t). Matematicamente, desconsiderando uma constante, é definida pela Equação (2-6).

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \tag{2-6}$$

Em que $\phi_1, \ldots, \phi_p$ são os parâmetros do modelo que quantificam a influência dos *lags* passados, e ϵ_t é um termo de ruído branco (tipicamente assumido como sendo normalmente distribuído com média zero e variância constante σ_ϵ^2). Este componente captura a inércia ou a memória de curto prazo da série temporal.

A componente **Integrada (I)** de ordem d, denotada I (d), é o mecanismo pelo qual o modelo ARIMA lida com a não estacionariedade, particularmente a presença de tendências ou mudanças de nível. A integração consiste na aplicação de d operações de diferenciação à série original. A diferenciação de primeira ordem é definida como $y'_t = y_t - y_{t-1}$. Se uma diferenciação não for suficiente para tornar a série estacionária (isto é, com média e variância constantes ao longo do tempo), aplica-se a diferenciação de segunda ordem $y''_t = y'_t - y'_{t-1} = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$, e assim por diante até a ordem d. O objetivo é transformar a série não estacionária em uma série estacionária, sobre a qual os componentes AR e MA podem ser modelados de forma eficaz.

A componente de **Média Móvel (MA)** de ordem q, denotada MA (q), modela o valor atual da série (y_t) não em função dos valores passados da série em si, mas como uma combinação linear dos q erros de previsão passados ($\epsilon_{t-1}, \ldots, \epsilon_{t-q}$) mais o erro de previsão atual (ϵ_t). A forma matemática, desconsiderando uma constante, é definida pela Equação (2-7).

$$y_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} \tag{2-7}$$

Em que $\theta_1, \ldots, \theta_q$ são os parâmetros do modelo que medem o impacto dos choques passados, e $\epsilon_t, \ldots, \epsilon_{t-q}$ são os termos de erro (ruído branco). Este componente é

útil para capturar dependências de curto prazo que podem surgir de eventos aleatórios ou choques não observados.

Um modelo *AutoRegressive Integrated Moving Average* (ARIMA) (p, d, q) completo combina estas três partes. É comum representá-lo de forma compacta utilizando o operador de *backshift* (ou *lag*) B, em que $By_t = y_{t-1}$, e o operador de diferença $\nabla = (1 - B)$, em que $\nabla y_t = y_t - y_{t-1}$ e $\nabla^d = (1 - B)^d$. A equação geral para o modelo ARIMA (p, d, q) é:

$$\phi_p(B)(1 - B)^d y_t = \theta_q(B)\epsilon_t \tag{2-8}$$

em que $\phi_p(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$ é o polinômio auto-regressivo de ordem p, e $\theta_q(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$ é o polinômio de média móvel de ordem q. A série y_t é diferenciada d vezes antes de ser modelada pelos componentes AR (p) e MA (q). Diversos modelos mais simples são casos especiais do ARIMA, conforme ilustrado na Tabela 2.1, adaptada de [47].

Tabela 2.1: Casos especiais do ARIMA [47]

Ruído branco	ARIMA (0,0,0)
Naive	ARIMA (0,1,0) sem constante
Naive Drift	ARIMA (0,1,0) com constante
Auto regressão (AR)	ARIMA (p,0,0)
Média Móvel (MA)	ARIMA (0,0,q)
ARMA	ARIMA (p,0,q)

Para séries que exibem padrões sazonais claros, o modelo ARIMA é estendido para o *Seasonal AutoRegressive Moving Average* (SARIMA) (*Seasonal ARIMA*). Este modelo adiciona componentes sazonais, denotados por $(P, D, Q)_m$, em que m é o período sazonal (por exemplo, $m = 12$ para dados mensais com sazonalidade anual). A lógica é análoga à do ARIMA não sazonal, mas aplicada aos *lags* sazonais. O modelo SARIMA $(p, d, q)(P, D, Q)_m$ é expresso de forma compacta como:

$$\Phi_P(B^m)\phi_p(B)(1 - B^m)^D(1 - B)^d y_t = \Theta_Q(B^m)\theta_q(B)\epsilon_t \tag{2-9}$$

em que $\Phi_P(B^m) = 1 - \Phi_1 B^m - \cdots - \Phi_P B^{Pm}$ é o polinômio AR sazonal, $\Theta_Q(B^m) = 1 + \Theta_1 B^m + \cdots - \Theta_Q B^{Qm}$ é o polinômio MA sazonal, e $(1 - B^m)^D$ representa a diferenciação sazonal de ordem D. Este modelo é capaz de capturar simultaneamente as dependências de curto prazo (via p, d, q) e as dependências sazonais (via P, D, Q, m).

A principal dificuldade prática na aplicação do SARIMA reside na identificação correta das ordens p, d, q, P, D, Q. Embora a análise das funções de autocorrelação (*Autocorrelation Function* (ACF)) e autocorrelação parcial (*Partial Autocorrelation Function* (PACF)) possa oferecer indicações iniciais, métodos mais objetivos são frequentemente empregados. Testes estatísticos de hipótese, como o Kwiatkowski-Phillips-

Schmidt-Shin (KPSS) [57] e o Augmented Dickey Fuller (ADF) [89], auxiliam na determinação da ordem de diferenciação não sazonal d, avaliando a estacionariedade da série. Analogamente, testes como o Osborn, Chui, Smith e Birchenhall (OCSB) [75] e o Canova-Hansen (CH) [19] são usados para determinar a necessidade e a ordem da diferenciação sazonal D. Para as ordens restantes (p, q, P, Q), o algoritmo AutoARIMA, proposto por [46], automatiza o processo. Ele realiza uma busca heurística no espaço de parâmetros, tipicamente minimizando um critério de informação como o *Akaike's Information Criterion with correction* (AICc) (*Akaike Information Criterion corrected*), para encontrar a combinação que melhor se ajusta aos dados sem excesso de complexidade.

Na Figura 2.3 são apresentados os resultados alcançados por meio deste modelo automatizado. Apesar de ser um modelo desenvolvido há várias décadas, o ARIMA e suas variantes sazonais continuam sendo ferramentas relevantes e competitivas em diversos cenários de previsão, frequentemente servindo como um *benchmark* robusto contra o qual modelos mais modernos são comparados, conforme evidenciado em estudos recentes como [53], [11], [72] e [28].

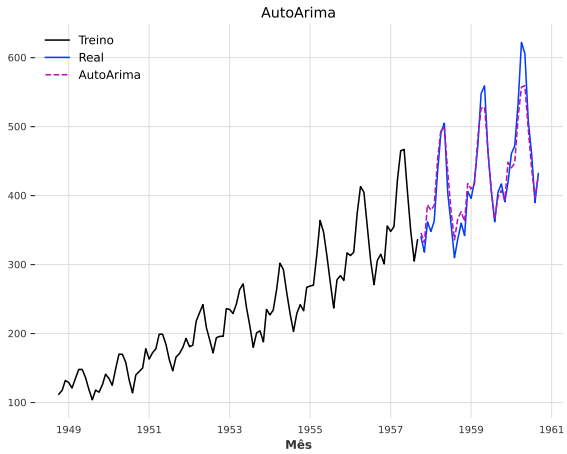


Figura 2.3: Predição utilizando AutoARIMA

2.4 Teunter, Syntetos & Babai (TSB)

Visando principalmente a predição para demandas intermitentes, o TSB [91] é uma adaptação do modelo original Croston [26], que é composto por 3 etapas: avaliação da demanda média quando existe demanda; avaliação do tempo entre a ocorrência de duas demandas; e previsão da demanda. Sendo formulado pela Equação (2-10). Dessa forma, o resultado é fruto de duas previsões: a primeira, que está relacionada à probabilidade de ter uma demanda naquele período, e outra relacionada ao valor propriamente dito dessa demanda neste determinado período.

$$\text{se } q_i > 0 \quad \begin{cases} \hat{q}_{i+1|i} = \alpha q_i + (1-\alpha)\hat{q}_{i|i-1} \\ \hat{a}_{i+1|i} = \alpha a_i + (1-\alpha)\hat{a}_{i|i-1} \\ \hat{y}_{T+h|T} = \dfrac{\hat{q}_{i+1|i}}{\hat{a}_{i+1|i}} \end{cases}$$
$$\text{se } q_i = 0 \quad \begin{cases} \hat{q}_{i+1|i} = \hat{q}_i \\ \hat{a}_{i+1|i} = \hat{a}_i \\ \hat{y}_{T+h|T} = \hat{y}_T \end{cases} \tag{2-10}$$

em que:

q_i = i-nésima demanda não nula

a_i = i-nésimo intervalo entre q_{i-1} e q_i

A principal limitação ocorre quando não há demanda ($q_i = 0$) e, portanto, os valores não são atualizados. Mesmo existindo diversos períodos nulos, este valor não é atualizado. Para mitigar esta restrição a implementação de TSB atualiza o modelo segundo um novo parâmetro de suavização β, mesmo quando não há demanda. A nova formulação é, então, descrita pela Equação (2-11) e ilustrada pela Figura 2.4.

$$\text{se } q_i > 0 \quad \begin{cases} \hat{q}_{i+1|i} = \alpha q_i + (1-\alpha)\hat{q}_{i|i-1} \\ \hat{a}_{i+1|i} = \beta + (1-\beta)\hat{a}_{i|i-1} \\ \hat{y}_{t+h|t} = (\hat{q}_{i+1|i})\ (\hat{a}_{i+1|i}) \end{cases}$$
$$\text{se } q_i = 0 \quad \begin{cases} \hat{q}_{i+1|i} = \hat{q}_i \\ \hat{a}_{i+1|i} = (1-\beta)\hat{a}_i \\ \hat{y}_{T+h|T} = (\hat{q}_{i+1|i})\ (\hat{a}_{i+1|i}) \end{cases} \tag{2-11}$$

2.5 PROPHET

Desenvolvido pela equipe de *Core Data Science* do Facebook e apresentado por Taylor e Letham [90], o Prophet é um procedimento de previsão para séries temporais baseado em um modelo aditivo em que tendências não lineares são ajustadas a sazonalidades anual, semanal e diária, além de efeitos de feriados. Ele foi projetado para ser particularmente eficaz com séries temporais que apresentam fortes efeitos sazonais e múltiplos períodos de sazonalidade histórica, sendo robusto a dados faltantes e mudanças de tendência.

O modelo Prophet pode ser representado pela Equação (2-12).

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{2-12}$$

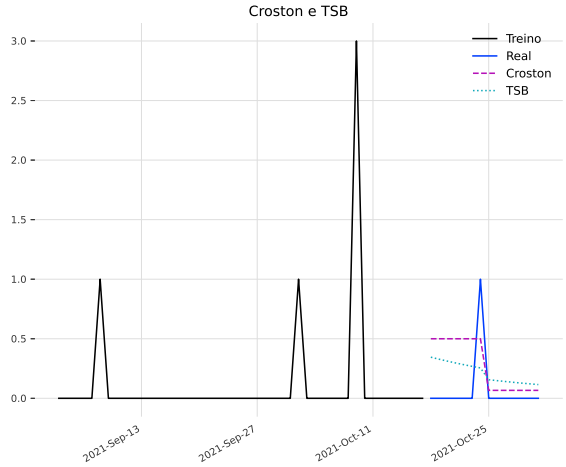


Figura 2.4: Predição utilizando Croston e TSB

Em que $y(t)$ é o valor da série no tempo t. Os componentes são: $g(t)$ representa a função de tendência, que modela mudanças não periódicas; $s(t)$ representa as mudanças periódicas (sazonalidades, como semanal e anual); $h(t)$ representa os efeitos de feriados ou eventos específicos que ocorrem em datas potencialmente irregulares; e ϵ_t é o termo de erro, que representa mudanças idiossincráticas não acomodadas pelo modelo, assumido como normalmente distribuído.

A função de **tendência** $g(t)$ é modelada primariamente de duas formas: um modelo de crescimento logístico saturante ou um modelo de tendência linear por partes. O modelo logístico é definido pela Equação (2-13).

$$g(t) = \frac{C}{1 + e^{-k(t-m)}} \tag{2-13}$$

Em que C é a capacidade de carga (saturação máxima), k é a taxa de crescimento e m é um parâmetro de deslocamento. Para permitir mudanças na tendência, o Prophet detecta automaticamente (ou permite especificar) *changepoints*, pontos no tempo em que a taxa de crescimento k pode mudar. Se δ_j é a mudança na taxa no *changepoint* j no tempo s_j, e definindo um vetor de ajustes $\boldsymbol{a}(t) \in \{0,1\}^S$ tal que $a_j(t) = 1$ se $t \geq s_j$ e 0 caso contrário (em que S é o número de *changepoints*), a taxa em qualquer tempo t é $k + \boldsymbol{a}(t)^T \boldsymbol{\delta}$. A tendência é então modelada com a taxa ajustada. Um modelo linear simples é usado se o crescimento não for saturante.

A componente de **sazonalidade** $s(t)$ é modelada utilizando séries de Fourier para fornecer um modelo flexível de efeitos periódicos [39]. Uma sazonalidade de período P (por exemplo, $P = 365.25$ para anual, $P = 7$ para semanal) é demonstrada na Equação (2-14).

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right) \tag{2-14}$$

Em que a_n e b_n são os parâmetros a serem estimados. O número de termos N (a ordem da série de Fourier) é um parâmetro que determina a flexibilidade da sazonalidade; valores maiores permitem ajustar padrões sazonais que mudam mais rapidamente.

A componente de **feriados** $h(t)$ é crucial, pois feriados e eventos especiais não seguem, em geral, um ciclo periódico regular (como a Páscoa, que varia de data) e podem ter efeitos significativos. O Prophet permite ao usuário fornecer uma lista personalizada de eventos passados e futuros. Para cada feriado i, seja D_i o conjunto de datas passadas e futuras para este feriado. O modelo cria um regressor indicador para cada feriado, $Z_i(t) = 1$ se $t \in D_i$ e 0 caso contrário. O efeito total dos feriados é então modelado como:

$$h(t) = \sum_{i=1}^{L} \kappa_i Z_i(t) = \mathbf{Z}(t)\boldsymbol{\kappa} \tag{2-15}$$

em que L é o número total de feriados considerados, κ_i é o parâmetro que representa o efeito aditivo do feriado i, $\mathbf{Z}(t) = [Z_1(t), \ldots, Z_L(t)]$ e $\boldsymbol{\kappa} = [\kappa_1, \ldots, \kappa_L]^T$. Além disso, para capturar efeitos que podem durar mais que um dia (por exemplo, um aumento nas vendas nos dias anteriores ao Natal), o Prophet permite incluir um intervalo de dias (*window*) em torno de cada feriado. Isso é feito incluindo colunas adicionais na matriz $\mathbf{Z}(t)$ que representam os dias anteriores e posteriores ao feriado, cada um com seu próprio parâmetro κ a ser estimado. O modelo assume que os efeitos dos feriados são independentes. Apesar de ser destinado principalmente a feriados esta parcela também é utilizada para eventos significativos, como por exemplo, o efeito do *Super Bowl* para a população norte-americana. Em ambos os casos, as datas não possuem sazonalidade definida, porém são previsíveis e conhecidas a priori. Além disso, os feriados e eventos dependem significativamente da localização, por exemplo, o efeito do *Super Bowl* no Brasil não possui tanto efeito quanto nos Estados Unidos. Muitas bibliotecas que implementam o Prophet já incluem calendários de feriados para diversos países.

Conforme mencionado originalmente, o Prophet demonstra robustez a *outliers*, valores faltantes e mudanças abruptas na série, características comuns no varejo. Sua capacidade de incorporar explicitamente sazonalidades múltiplas e efeitos de feriados complexos o torna uma ferramenta valiosa neste domínio. A Figura 2.5 ilustra um exemplo de predição utilizando este modelo.

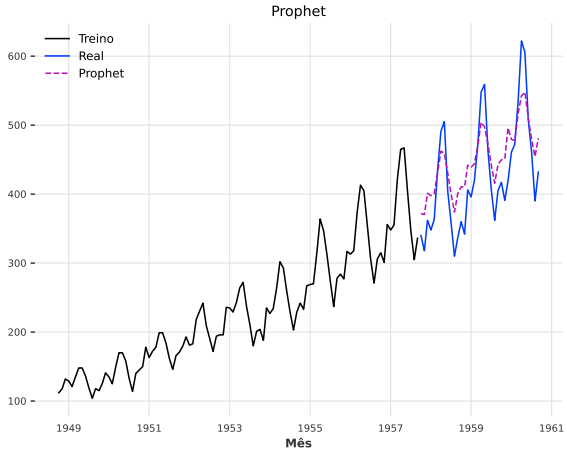


Figura 2.5: Predição utilizando Prophet

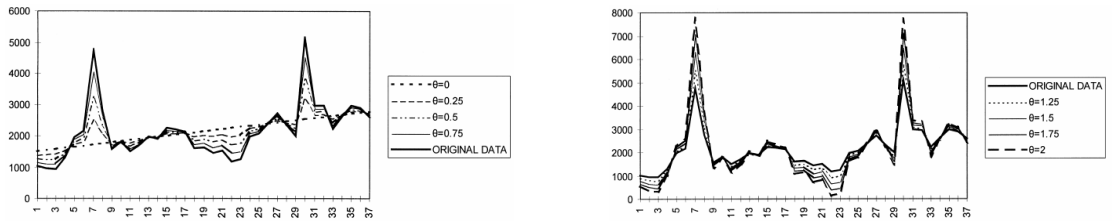
2.6 Theta

O método Theta, desenvolvido por Assimakopoulos e Nikolopoulos [4], emergiu como uma abordagem de previsão robusta e influente após demonstrar resultados notáveis na competição M3 [66]. A M3 foi uma das maiores e mais abrangentes competições de previsão de séries temporais realizadas até então, envolvendo milhares de séries de diversas naturezas (microeconômicas, industriais, macroeconômicas, financeiras, demográficas, etc.) e múltiplos horizontes de previsão. Sua importância reside no fato de ter estabelecido *benchmarks* rigorosos para a avaliação comparativa de métodos de previsão.

Neste contexto competitivo, o método Theta destacou-se por seu desempenho consistentemente superior em diferentes horizontes de previsão e métricas de erro [66], superando muitos modelos mais complexos da época. Este desempenho surpreendente conferiu grande popularidade ao método e o estabeleceu como uma referência importante na área. Subsequentemente, Hyndman e Billah [45] demonstraram que o método Theta original é, na verdade, equivalente a uma *Single Exponential Smoothing* (SES) com *drift*, em que o parâmetro de *drift* é derivado de uma maneira específica – metade da tendência linear ajustada aos dados históricos.

A formulação do Theta baseia-se na decomposição conceitual da série temporal original em duas novas linhas, denominadas “linhas Theta”. Originalmente, os autores propuseram $\theta = 2$ e $\theta = 0$, partindo da premissa de que um único método de suavização não consegue extrair toda a informação contida na série temporal. Com esta decomposição, busca-se capturar e extrapolar separadamente diferentes aspectos da série. A variável θ é aplicada na diferença de segunda ordem dos dados. Valores de θ inferiores a 1 (como $\theta = 0$, que corresponde a uma regressão linear ajustada aos dados) tendem a amortecer a curvatura local e focar na tendência de longo prazo, como ilustrado na Figura 2.6(a). Em

contrapartida, valores de θ superiores a 1 (como o θ = 2 original) exacerbam as variações locais e focam nos padrões de curto prazo, conforme Figura 2.6(b). A previsão final é então uma combinação das extrapolações dessas linhas Theta. A Figura 2.7 ilustra um exemplo de desempenho desse modelo.



(a) θ *inferior a 1.* (b) θ *superior a 1.*

Figura 2.6: Predição da M3 utilizando o modelo Theta. Fonte [4].

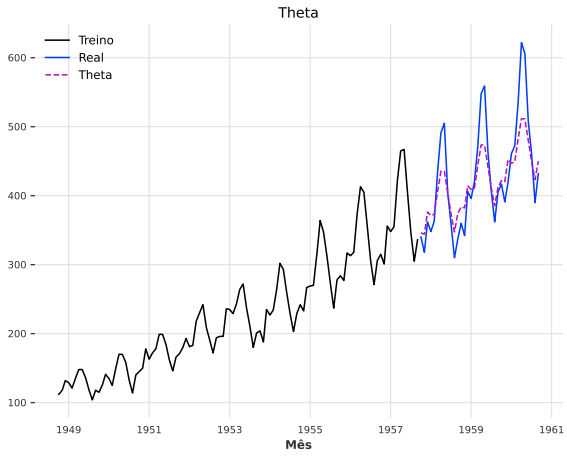


Figura 2.7: Predição utilizando Theta

2.7 LightGBM

O LightGBM (*Light Gradient Boosting Machine*), proposto por Ke et al. [52], representa uma evolução significativa na família dos algoritmos de *Gradient boosting decision tree* (GBDT) (*Gradient Boosting Decision Tree*). Embora compartilhe o princípio fundamental de construir um *ensemble* de árvores de decisão de forma sequencial, em que cada nova árvore corrige os erros residuais das anteriores, o LightGBM introduz otimizações algorítmicas que lhe conferem vantagens substanciais em termos de velocidade de treinamento e eficiência no uso de memória, especialmente em conjuntos de dados de grande escala. Seu notável desempenho foi consolidado na competição M5, em que domi-

nou as primeiras posições, superando inclusive abordagens baseadas em *Deep Learning* para a previsão de vendas no varejo [67].

A eficiência do LightGBM deriva principalmente de duas técnicas inovadoras: *Gradient-based One-Side Sampling* (GOSS) (*Gradient-based One-Side Sampling*) e *Exclusive Feature Bundling* (EFB) (*Exclusive Feature Bundling*). O *Gradient-based One-Side Sampling* (GOSS) aborda o gargalo computacional no cálculo dos ganhos de informação para cada possível divisão (*split*) em uma árvore. A intuição é que instâncias com gradientes pequenos (ou seja, erros pequenos) já estão bem treinadas e contribuem menos para o ganho de informação. O *Gradient-based One-Side Sampling* (GOSS), portanto, mantém todas as instâncias com gradientes grandes e realiza uma amostragem aleatória nas instâncias com gradientes pequenos. Ao calcular o ganho de informação, os gradientes das instâncias amostradas são reponderados para compensar a amostragem, garantindo que a distribuição dos dados não seja alterada significativamente, mas reduzindo drasticamente o custo computacional.

A técnica *Exclusive Feature Bundling* (EFB), por sua vez, foca-se na redução da dimensionalidade do espaço de *features*, particularmente eficaz em dados esparsos, comuns em problemas de varejo com muitas variáveis categóricas ou *lags*. A ideia central é que muitas *features* são mutuamente exclusivas, ou quase, significando que raramente assumem valores não nulos simultaneamente (por exemplo, *features* codificadas via *one-hot encoding*). O *Exclusive Feature Bundling* (EFB) agrupa essas *features* mutuamente exclusivas em um único "pacote" (*bundle*), tratando-as como uma única *feature* durante a construção da árvore. Um algoritmo de coloração de grafos é usado para identificar os grupos ótimos de *features* a serem agrupadas, preservando a informação original ao deslocar os intervalos de valores das *features* dentro do pacote. Isso reduz o número de *features* a serem consideradas em cada divisão, acelerando o treinamento. Adicionalmente, o LightGBM emprega uma estratégia de crescimento de árvore *leaf-wise* (baseada na folha com maior perda) em vez da tradicional *level-wise*, o que tende a convergir mais rapidamente para uma boa solução, embora exija cuidado com a regularização para evitar *overfitting*.

A aplicação do LightGBM, um modelo intrinsecamente tabular, para a previsão de séries temporais requer uma etapa crucial de conversão: a transformação da sequência temporal em um formato tabular por meio da engenharia de *features*. Este processo, detalhado na Seção 5.2, é fundamental para o sucesso do modelo. As *features* tipicamente incluem:

- ***Lags* da variável alvo:** Valores passados da série ($y_{t-1}, y_{t-2}, \ldots$).
- **Estatísticas móveis:** Média, desvio padrão, mínimo, máximo, etc., calculados sobre janelas deslizantes de *lags* (y_{t-k} a y_{t-1}).

- **Características de data/calendário:** Dia da semana, semana do ano, mês, ano, indicadores de feriados, dia do mês, etc.
- **Variáveis exógenas:** Informações externas que podem influenciar a série, como promoções, indicadores econômicos, eventos especiais (se disponíveis e relevantes).

Após esta transformação, o problema de previsão de séries temporais é efetivamente convertido em um problema de regressão tabular supervisionado, em que o objetivo é prever y_t (ou y_{t+h} para previsão multi-passos) com base nas *features* criadas para o tempo t. O LightGBM pode ser treinado como um modelo global, utilizando dados de múltiplas séries e incorporando identificadores de série como *features* categóricas para capturar padrões comuns (*cross-learning*), ou como modelos locais para séries individuais ou grupos homogêneos.

Como em outros modelos baseados em árvores, o LightGBM fornece métricas de importância de *features*, que podem oferecer *insights* sobre quais *lags*, componentes sazonais ouáveis exógenas são mais preditivos. No entanto, seu desempenho é altamente sensível à escolha dos hiperparâmetros. Parâmetros como o número de folhas (*num_leaves*), a taxa de aprendizado (*learning_rate*), a fração de *features* considerada por árvore (*feature_fraction*), a fração de dados usada por iteração (*bagging_fraction*), e parâmetros de regularização (como *lambda_l1*, *lambda_l2*, *min_gain_to_split*) devem ser cuidadosamente ajustados. Dada a complexidade e interdependência desses parâmetros, a otimização de hiperparâmetros (*Hyperparameter Optimization* (HPO)), conforme explorada na Seção 5.3 e nos resultados do Capítulo 6, torna-se uma etapa indispensável para extrair o máximo desempenho do LightGBM em aplicações práticas. A Figura 2.8 ilustra um exemplo de predição utilizando este modelo.

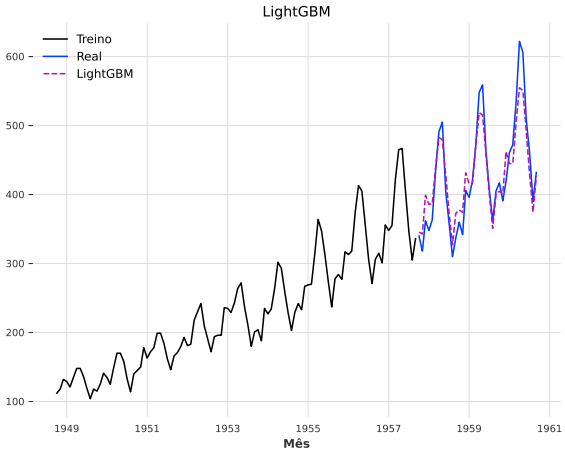


Figura 2.8: Predição utilizando LightGBM

2.8 DeepAR

O DeepAR, proposto por Salinas et al. da Amazon AI [79], representa uma abordagem inovadora para a previsão de séries temporais, superando uma limitação fundamental dos modelos estatísticos tradicionais: a necessidade de treinar um modelo distinto para cada série individual. Em cenários como o varejo, em que milhares de séries temporais relacionadas (por exemplo, vendas de diferentes produtos) existem, o DeepAR introduz o conceito de modelo global, treinando uma única rede neural em todo o conjunto de séries temporais correlacionadas. Esta abordagem permite o *cross-learning*, em que o modelo aprende padrões (como sazonalidade ou efeitos de eventos) a partir de séries com históricos mais longos e os transfere para séries com dados mais esparsos ou curtos (*cold start problem*).

A arquitetura do DeepAR é fundamentada em uma rede neural recorrente (*Recurrent Neural Network* (RNN)), tipicamente utilizando células *Long Short-Term Memory* (LSTM), devido à sua capacidade de capturar dependências temporais de longo prazo. O modelo opera de forma autorregressiva: para prever o valor da série no tempo t, y_t, ele utiliza como entrada os valores passados da própria série ($y_{t-k}, \ldots, y_{t-1}$, em que k é o tamanho do contexto ou *lookback window*) e um conjunto de covariáveis (*features*) x_t relevantes para aquele instante.

Uma característica distintiva do DeepAR é sua natureza probabilística. Em vez de prever um único valor pontual para o futuro, o modelo aprende os parâmetros de uma distribuição de probabilidade condicional sobre o valor futuro, $P(y_t | y_{t-k:t-1}, x_t)$. A escolha da distribuição de verossimilhança (*likelihood function*) depende da natureza dos dados. Para dados de contagem, como vendas, a distribuição Binomial Negativa é frequentemente usada para lidar com a presença de zeros (intermitência) e a variância que aumenta com a média. Para dados contínuos, uma distribuição Gaussiana pode ser apropriada. O modelo, então, não retorna uma previsão única, mas sim os parâmetros dessa distribuição (por exemplo, média e desvio padrão para a Gaussiana; média e parâmetro de dispersão para a Binomial Negativa). As previsões futuras são geradas por amostragem de Monte Carlo a partir da distribuição aprendida, passo a passo no horizonte de previsão. Isso permite não apenas obter uma previsão mediana ou média, mas também gerar intervalos de predição (quantis), fornecendo uma estimativa da incerteza associada à previsão.

A capacidade do *DeepAR* de incorporar **covariáveis (*features*)** é fundamental para sua aplicação prática, permitindo que o modelo condicione suas previsões em informações adicionais relevantes. Essas covariáveis podem ser categorizadas de acordo com sua disponibilidade temporal. Primeiramente, existem as *covariáveis conhecidas no futuro*, cujos valores ao longo do horizonte de previsão já são determinados no momento

em que a previsão é gerada; características de calendário (como dia da semana, mês ou indicadores de feriados) e eventos planejados são exemplos comuns desta categoria. Em segundo lugar, há as *covariáveis desconhecidas no futuro*, como preço ou informações sobre promoções futuras, cujos valores futuros não estão disponíveis; estas são tipicamente incorporadas ao modelo utilizando seus valores passados (*lag*) ou por meio de previsões separadas. Finalmente, o modelo pode utilizar *covariáveis estáticas (time-independent)*, que são características que permanecem constantes ao longo do tempo para uma série específica, como a categoria do produto ou a localização da loja. Estas covariáveis estáticas são particularmente importantes para modelos globais como o DeepAR, pois permitem à rede aprender comportamentos distintos para diferentes grupos de séries temporais. Todas essas covariáveis, em conjunto com os *lags* da própria série alvo, são alimentadas na RNN em cada passo de tempo para informar e refinar a previsão da distribuição futura.

Embora o DeepAR tenha demonstrado melhorias significativas em relação a modelos estatísticos em diversos conjuntos de dados e tenha sido uma arquitetura influente, competições como a M5 mostraram que modelos baseados em árvores como o LightGBM, com engenharia de *features* robusta, podem superá-lo em certos cenários de varejo, destacando o contínuo *trade-off* entre diferentes classes de modelos. A Figura 2.9 ilustra um exemplo de predição com *DeepAR*.

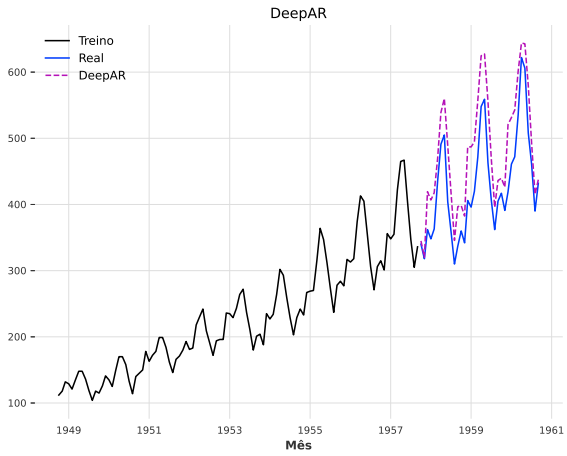


Figura 2.9: Predição utilizando DeepAR

2.9 *Long Short-Term Memory* (LSTM)

As Redes Neurais Recorrentes Longo-Curto Prazo, ou *Long Short-Term Memory* (LSTM) (*Long Short-Term Memory*), propostas por Hochreiter e Schmidhuber [41], foram desenvolvidas para superar uma limitação fundamental das RNNs tradicionais: o

problema do desaparecimento (ou explosão) do gradiente durante o treinamento (*back-propagation*). Este problema impedia as RNNs simples de aprenderem dependências de longo prazo em sequências, uma capacidade crucial para a modelagem eficaz de muitas séries temporais. A arquitetura LSTM aborda isso introduzindo uma unidade de memória explícita (o estado da célula) e mecanismos de controle chamados portões (*gates*) que regulam o fluxo de informação. Sua modelagem passou por melhorias ao longo do tempo, como as sugeridas em [34] [35] e [36], sendo a última a mais popular dada sua implementação bidirecional.

A unidade LSTM processa uma sequência de entrada $x = (x_1, x_2, \ldots, x_T)$ mantendo dois vetores de estado que são transmitidos temporalmente: o estado oculto h_t, representando a memória de curto prazo, e o estado da célula C_t, representando a memória de longo prazo. A dinâmica de atualização entre os passos $t-1$ e t é governada por um conjunto coordenado de operações, começando pelo **Portão de Esquecimento (*Forget Gate*)**. Este portão, denotado por f_t, determina heuristicamente qual fração da informação contida no estado da célula anterior, C_{t-1}, deve ser descartada. Sua ativação é calculada aplicando uma função sigmoide à combinação linear da entrada atual x_t e do estado oculto anterior h_{t-1}, conforme a Equação (2-16).

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2-16}$$

Em que σ denota a função sigmoide, W_f e b_f são os pesos e *bias* específicos deste portão, e $[h_{t-1}, x_t]$ representa a concatenação. Valores de f_t próximos a zero favorecem o esquecimento, enquanto valores próximos a um favorecem a retenção da memória preexistente.

Sequencialmente, o **Portão de Entrada (*Input Gate*)** decide qual nova informação será incorporada ao estado da célula. Este processo envolve duas sub-etapas. Primeiramente, um portão sigmoide i_t determina quais posições do estado da célula devem ser atualizadas, utilizando a mesma combinação de h_{t-1} e x_t, mas com pesos W_i, b_i distintos, como na Equação (2-17). Simultaneamente, uma camada com ativação tangente hiperbólica (tanh) gera um vetor de valores candidatos $\tilde{C}_t$, representando a nova informação a ser potencialmente adicionada, conforme a Equação (2-18), com seus próprios pesos W_C, b_C.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2-17}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2-18}$$

A **Atualização do Estado da Célula** C_t ocorre então pela combinação ponderada da memória antiga e da nova informação candidata. O estado anterior C_{t-1} é multiplicado elemento a elemento pelo portão de esquecimento f_t, e o resultado é somado

ao produto elemento a elemento do portão de entrada i_t com os valores candidatos $\tilde{C}_t$, resultando no novo estado da célula C_t, como formalizado na Equação (2-19).

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2-19}$$

Finalmente, o **Portão de Saída (*Output Gate*)** controla qual parte do estado da célula C_t será exposta como o estado oculto h_t para o passo de tempo atual e como entrada para o próximo passo. Este portão também opera em duas fases. Primeiro, calcula-se uma ativação sigmoide o_t a partir de h_{t-1} e x_t (com pesos W_o, b_o), conforme a Equação (2-20), para determinar quais partes do estado da célula serão liberadas. Em seguida, o estado da célula C_t é processado por uma função tanh (escalando os valores para o intervalo $[-1,1]$) e o resultado é multiplicado elemento a elemento por o_t, gerando o novo estado oculto h_t, como descrito na Equação (2-21).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2-20}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2-21}$$

Este estado oculto h_t serve como a saída da unidade LSTM no tempo t, frequentemente conectado a camadas subsequentes para tarefas como classificação ou regressão, e também é passado adiante como h_t para o cálculo no tempo $t+1$.

A aplicação da LSTM para previsão de séries temporais segue um *pipeline* específico. Primeiramente, a série temporal univariada ou multivariada é transformada em sequências de entrada e saída. Uma janela deslizante (*lookback window*) de tamanho k é usada para criar sequências de entrada $(y_{t-k}, \ldots, y_{t-1})$ para prever um ou mais valores futuros $(y_t, \ldots, y_{t+h-1})$, em que h é o horizonte de previsão. A LSTM processa a sequência de entrada passo a passo, atualizando seus estados C_t e h_t. O estado oculto final h_T (ou uma combinação dos estados ocultos da sequência) é tipicamente passado por uma camada densa (totalmente conectada) com uma função de ativação apropriada (frequentemente linear para problemas de regressão) para gerar a previsão desejada.

Assim como o DeepAR, as LSTMs podem incorporar ***features*** para melhorar a previsão. Covariáveis conhecidas no futuro (como calendário), desconhecidas no futuro (como preço, usadas com *lag*) e estáticas (como categoria de produto) podem ser concatenadas com o valor da série temporal y_t em cada passo de tempo t, formando um vetor de entrada x_t mais rico.

O desempenho da LSTM é sensível a diversos **hiperparâmetros**, que precisam ser cuidadosamente ajustados. Os mais comuns incluem: o número de unidades (neurônios) LSTM na camada oculta, o número de camadas LSTM empilhadas (*stacked LSTMs*), a taxa de aprendizado (*learning rate*) do otimizador, o tamanho da sequência de entrada

(*sequence length* ou *lookback window*), o tamanho do lote (*batch size*) durante o treinamento, e a taxa de *dropout* para regularização. Variações arquiteturais como LSTMs bidirecionais [35] também podem ser consideradas, embora sejam mais comuns em processamento de linguagem natural do que em previsão pura, em que a causalidade temporal é estrita.

Por sua capacidade de modelar dependências complexas e de longo prazo, a LSTM tornou-se uma ferramenta padrão em *Deep Learning* para dados sequenciais, apresentando bons resultados em séries temporais. A Figura 2.10 ilustra a arquitetura interna de um bloco LSTM, e a Figura 2.11 mostra um exemplo de predição.

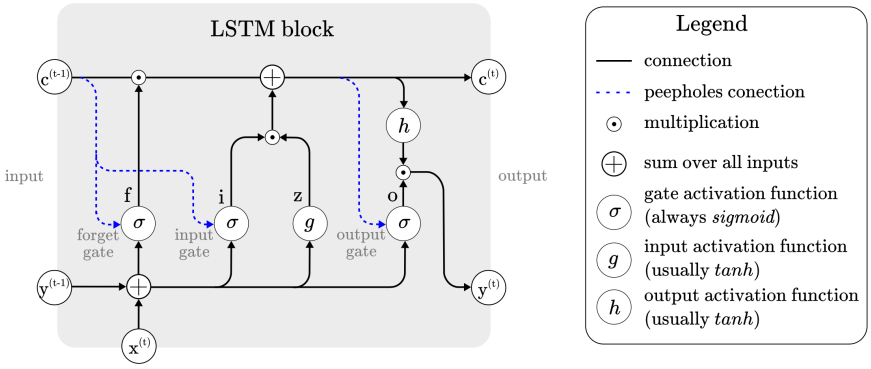


Figura 2.10: Arquitetura de um bloco LSTM [94].

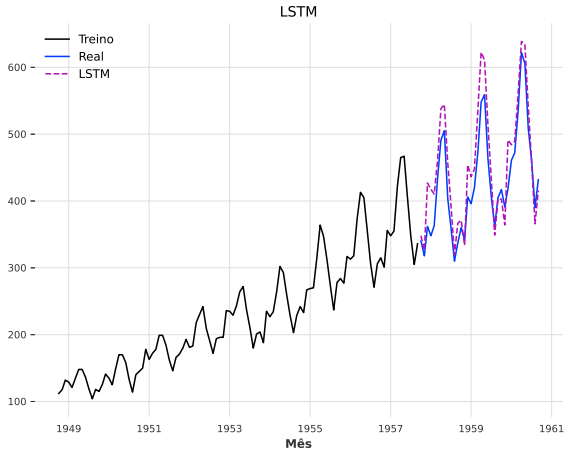


Figura 2.11: Predição utilizando LSTM

2.10 *Temporal Fusion Transformer* (TFT)

O *Temporal Fusion Transformer* (TFT) (*Temporal Fusion Transformer*), proposto por Lim et al. [60], representa uma arquitetura de *Deep Learning* híbrida projetada

especificamente para a previsão de séries temporais multi-horizonte, buscando alcançar alta acurácia enquanto oferece mecanismos para interpretabilidade. Diferentemente de abordagens puramente sequenciais como LSTMs ou puramente baseadas em atenção, o TFT integra componentes especializados para processar diferentes tipos de entradas comuns em problemas de séries temporais do mundo real e utiliza mecanismos de atenção para aprender relações temporais complexas em múltiplas escalas.

A arquitetura do TFT é notavelmente multi-componente, orquestrada para lidar simultaneamente com variáveis estáticas (metadados), variáveis conhecidas no futuro e variáveis observadas no passado. As variáveis estáticas (como identificador do produto, categoria ou localização da loja) são processadas por redes de codificação separadas (tipicamente *embeddings* seguidos por redes densas) para gerar vetores de contexto. Estes vetores são então utilizados para condicionar o processamento das variáveis dependentes do tempo por meio de Redes Residuais com Portão (*Gated Residual Networks* (GRN)), um bloco de construção fundamental no TFT que aplica mecanismos de portão (semelhantes aos da LSTM) para controlar o fluxo de informação e permitir saltos de conexão (*skip connections*).

As variáveis dependentes do tempo são divididas em duas categorias principais, cada uma processada inicialmente por codificadores sequenciais (frequentemente LSTMs) para capturar padrões locais. Variáveis conhecidas no futuro (*Known Future Inputs*) incluem informações de calendário (dia da semana, mês, feriados) ou eventos planejados (promoções futuras). O processamento separado destas variáveis é crucial para a previsão multi-horizonte. Variáveis observadas no passado (*Observed Inputs*), compostas pelos valores passados da série alvo e outras variáveis exógenas cujos valores futuros não são conhecidos no momento da previsão (como preço ou indicadores econômicos passados).

Uma inovação chave do TFT é o uso de redes de seleção de variáveis aplicadas a cada tipo de entrada dependente do tempo. Estas redes, baseadas em GRN e na atenção sensível à entidade (*entity-aware attention*), aprendem a ponderar a importância das diferentes *features* em cada passo de tempo, permitindo ao modelo focar nas informações mais relevantes e oferecendo um grau de interpretabilidade ao expor quais variáveis foram mais influentes.

O coração do TFT reside no seu mecanismo de auto-atenção multi-cabeça (*multi-head self-attention*) sensível ao tempo. Similar aos Transformers originais, a auto-atenção calcula representações contextuais para cada passo de tempo, ponderando a influência de todos os outros passos de tempo. A formulação conceitual segue $\mathit{Attention}(Q, K, V) = \mathit{softmax}(\frac{QK^T}{\sqrt{d_k}})V$, em que Q, K, e V são projeções lineares das saídas dos codificadores LSTM e d_k é a dimensão das chaves (K), servindo como fator de normalização. No TFT, este mecanismo é adaptado para respeitar a causalidade temporal (impedindo que passos

futuros influenciem passos passados) e para aprender dependências de longo prazo que podem ser difíceis de capturar apenas com RNNs. A saída da camada de atenção é então processada por outra GRN antes de ser passada para a camada final.

Finalmente, a camada de predição utiliza as representações contextuais ricas geradas pela atenção para produzir previsões quantílicas para múltiplos passos de tempo futuros simultaneamente. Ao prever quantis específicos (como $p10, p50, p90$), o TFT fornece não apenas uma previsão pontual (mediana, $p50$), mas também uma estimativa da incerteza da previsão, crucial para a tomada de decisão no varejo. A Figura 2.12 ilustra esquematicamente a arquitetura geral do TFT.

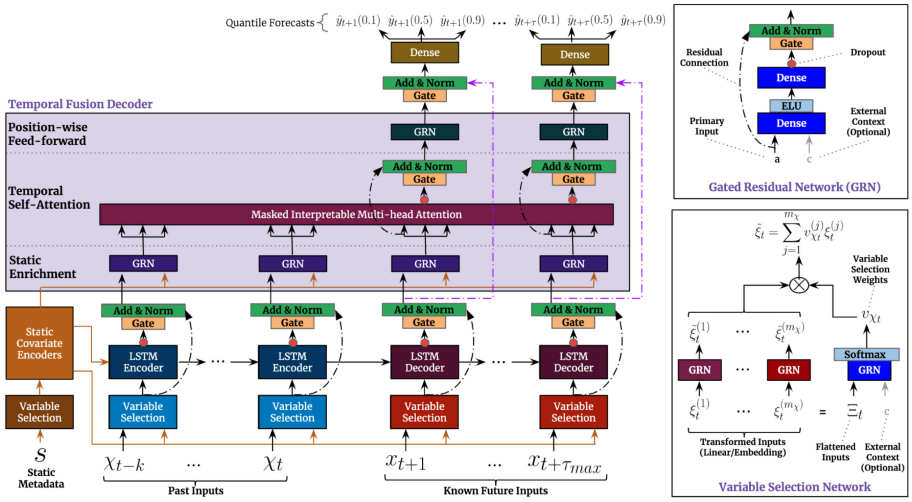


Figura 2.12: Visão geral da arquitetura do Temporal Fusion Transformer (TFT) [60]

A aplicação do TFT, como outros modelos de *Deep Learning*, requer uma cuidadosa engenharia de *features* para preparar as entradas estáticas, conhecidas no futuro e observadas no passado. No entanto, sua capacidade intrínseca de seleção de variáveis pode aliviar parte desse fardo. O desempenho do modelo é fortemente dependente da otimização de seus hiperparâmetros, incluindo a dimensionalidade das camadas ocultas e dos *embeddings*, o número de camadas LSTM, o número de cabeças de atenção, as taxas de *dropout* para regularização, a taxa de aprendizado e o tamanho do histórico de entrada (*lookback window*). Apesar de sua complexidade, o TFT estabeleceu o estado da arte em diversos *benchmarks* de séries temporais ao demonstrar a capacidade de aprender padrões complexos e fornecer previsões interpretáveis e calibradas. A Figura 2.13 ilustra um exemplo do desempenho preditivo deste modelo.

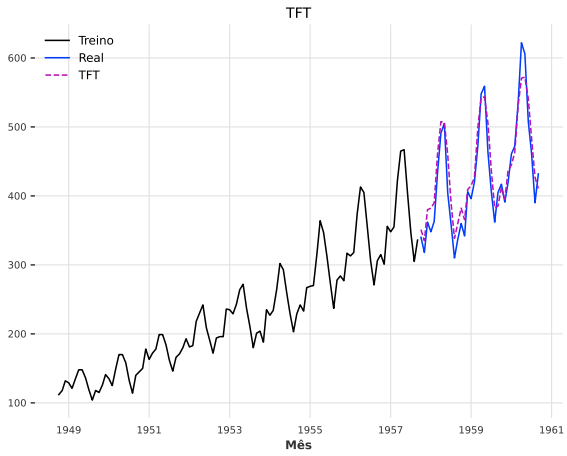


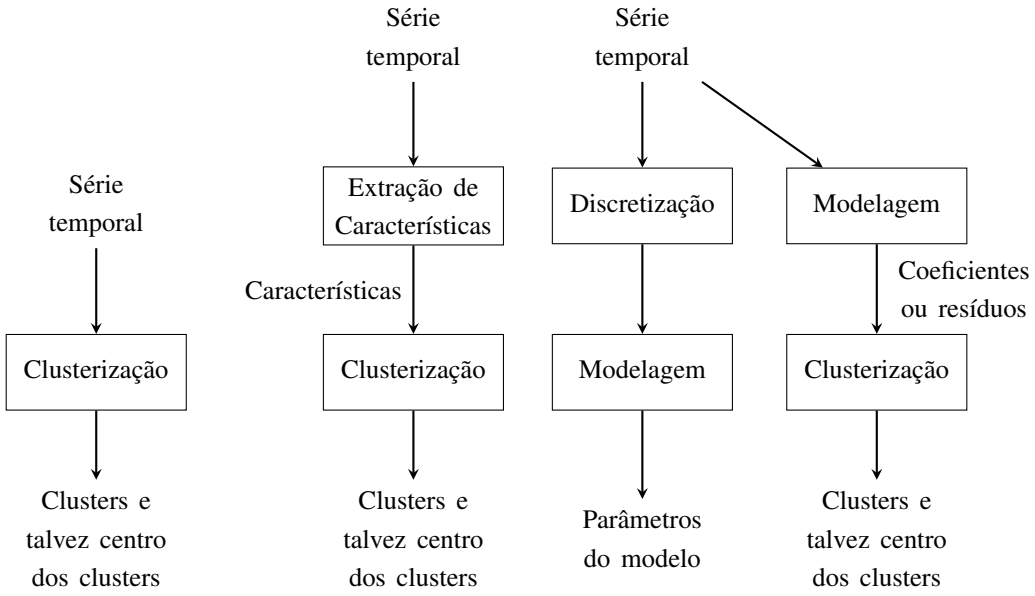
Figura 2.13: Predição utilizando TFT

CAPÍTULO 3

Clusterização

O principal objetivo da clusterização é, a partir de um conjunto de dados não rotulados, obter subconjuntos do original em que haja o máximo de separação entre eles e mínimo de separação entre os dados intra conjuntos. Dessa maneira, é esperado que os dados sejam separados em conjuntos que compartilham das mesmas características.

A clusterização de séries temporais apresenta desafios únicos, visto que métricas de distância tradicionais, como a euclidiana, falham em capturar dependências temporais, desalinhamentos de fase ou similaridade morfológica. Conforme a taxonomia fundamental proposta por Liao [59] e ilustrada na Figura 3.1, as metodologias para endereçar este problema podem ser categorizadas em três paradigmas principais.



(a) *Baseado em dados brutos* (b) *Baseado em características* (c) *Baseado em modelo*

Figura 3.1: Metodologias de clusterização para séries temporais. Adaptado de [59]

O primeiro paradigma é a abordagem *raw-data-based* (baseada em dados brutos), apresentada na Figura 3.1(a). Nela, a clusterização opera diretamente sobre as sequências temporais. Esta abordagem exige o uso de medidas de dissimilaridade elásticas

e robustas a distorções, como o DTW, que são capazes de alinhar e comparar séries que exibem padrões morfológicos semelhantes, mas que ocorrem em fases ou velocidades distintas [59].

O segundo paradigma é a abordagem *feature-based* (baseada em características), conforme a Figura 3.1(b). Em vez de comparar as séries ponto a ponto, esta metodologia primeiro transforma cada série temporal em um vetor de características (*features*) estáticas. Estas características podem ser descritores estatísticos (como média, variância, entropia) ou coeficientes de decomposição (como Fourier) [59]. A clusterização, então, frequentemente utiliza métricas de distância convencionais. A etapa de engenharia de *features*, detalhada na Seção 5.2, é um exemplo prático de parte dessa abordagem.

O terceiro paradigma, e o mais abstrato, é a abordagem *model-based* (baseada em modelo), ilustrada na Figura 3.1(c). A premissa aqui é que séries temporais similares são geradas pelo mesmo processo estocástico subjacente. Esta metodologia ajusta um modelo paramétrico (como um modelo ARIMA ou, no contexto desta tese, um Autômato Finito Probabilístico, PFSA) a cada série. A clusterização não ocorre nos dados brutos, mas sim nos parâmetros (coeficientes) ou nos resíduos desses modelos [59], agrupando séries que compartilham a mesma "física" estatística. A métrica `LikelihoodDistance`, explorada nesta tese e detalhada na Seção 3.3, é um exemplo sofisticado desta abordagem.

Os trabalhos de classificação de demanda são uma grande evidência de que há vantagens em aplicar técnicas de clusterização para previsão de demanda do ramo varejista. Por meio de métricas de variância e intermitência, extraídas do perfil da série temporal, quatro grandes perfis de demanda são sugeridos. Esses perfis de demanda, dentre outras vantagens, oferecem insumos para seleção dos melhores modelos preditivos.

3.1 Classificação de Demanda

A caracterização de perfil de demanda também foi abordada por outros autores, sendo que os mais importantes têm abordagens quanto: à participação no custo-volume [33] conhecida como ABC, dividindo-a em 3 categorias (alta, média e baixa), porém sem definir métricas para cada classe; uma segunda abordagem, sugerida em [69], denominada FSN, que utiliza a velocidade de variação da demanda, também dividida em três categorias (rápida, devagar e nenhuma); na metodologia XYZ [80] a classificação é realizada segundo a variação no consumo; já na HML [14], em que o valor do produto como parâmetro de classificação, e na VED [15] a classificação é quanto à importância no processo, seja vital, essencial ou desejada.

No contexto varejista, um dos maiores desafios reside na previsão de demanda, na qual uma característica presente é a intermitência dos valores, ou seja, não há vendas de um determinado produto naquela estampa de tempo. Não só se notou a importância

de observar se há valores nulos na série, mas também qual a duração desse intervalo dos valores nulos. Ambos os comportamentos citados têm sua origem em [49] e [86], e permitem a caracterização das demandas com base em duas variáveis: *Average Demand Interval* (ADI), que representa o intervalo médio entre duas demandas; e *Coefficient of Variation* (CV), que mede a variação das quantidades, conforme definidos pelas Equações (3-1) e (3-2) respectivamente.

$$ADI = \frac{\text{Total de Períodos}}{\text{Quantidade de valores não nulos}} \tag{3-1}$$

$$CV = \frac{\text{Desvio padrão da demanda}}{\text{Média da demanda}} \tag{3-2}$$

Tabela 3.1: Classificação de Demanda [78]

Tipo	ADI	CV^2	Variação em	
			Quantidade	Intervalo
Smooth	$< 1,32$	$< 0,49$	Baixa	Baixa
Intermittent	$\geq 1,32$	$< 0,49$	Baixa	Elevada
Erratic	$< 1,32$	$\geq 0,49$	Elevada	Baixa
Lumpy	$\geq 1,32$	$\geq 0,49$	Elevada	Elevada

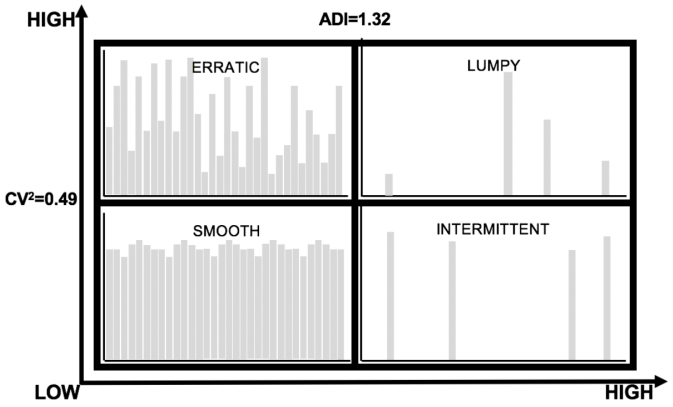


Figura 3.2: Padrões de classificação de demanda [78].

Com base nessas duas variáveis é possível classificar a demanda em quatro categorias: *Smooth*, *Intermittent*, *Erratic* e *Lumpy*, melhor representadas pela Tabela 3.1 e pela Figura 3.2.

Em [78], é sugerida a abordagem preditiva por duas distintas formas de tratamento. Para os quadrantes com menor impacto de intermitência, ou seja, *Smooth* e *Erratic*, sugerem-se técnicas de regressão. Já para os quadrantes *Lumpy* e *Intermittent*, indicam-se técnicas de classificação.

Com modelos possuindo melhor desempenho em determinadas classes, utilizando as métricas de ADI e CV, é possível aplicar métodos de clusterização que visem identificar outras características, visando segregar grupos de séries temporais por afinidade, por determinado modelo ou classe de modelos.

3.2 Algoritmos de Clusterização Particionais: K-means e *Self-Organizing Maps* (SOM)

Para organizar as séries temporais em grupos homogêneos, esta tese emprega dois algoritmos particionais canônicos: K-means e Mapas Auto-Organizáveis (*Self-Organizing Maps* (SOM)). É crucial compreender que ambos os algoritmos são, em essência, *frameworks* iterativos cujo comportamento é definido pela métrica de dissimilaridade utilizada para comparar os pontos de dados.

O K-means, introduzido por [65], é um algoritmo de refinamento iterativo que visa particionar n observações em k *clusters*, minimizando a variância intra-cluster (soma dos quadrados das distâncias). Seu processo alterna entre duas etapas: (1) atribuir cada ponto de dado ao *cluster* cujo centroide é o mais próximo, e (2) recalcular os centroides de cada *cluster* com base nos pontos recém-atribuídos. A definição de "mais próximo" e a forma como o centroide é calculado dependem inteiramente da métrica de distância subjacente.

O *Self-Organizing Maps* (SOM), ou Mapa de Kohonen [54], é uma rede neural não supervisionada que reduz a dimensionalidade dos dados enquanto preserva sua estrutura topológica. O SOM projeta os dados de entrada de alta dimensão em um mapa de baixa dimensão (geralmente 2D), em que neurônios (nós) representam os protótipos dos *clusters*. Durante o treinamento, para cada vetor de entrada, a Unidade de Melhor Correspondência (*Best Matching Unit* (BMU)) é identificada, ou seja, o neurônio cujos pesos são mais similares à entrada. O BMU e seus neurônios vizinhos no mapa têm seus pesos ajustados para se assemelharem mais ao vetor de entrada. Novamente, a identificação do BMU é um passo inteiramente dependente da métrica de dissimilaridade utilizada.

Embora a distância Euclidiana seja a métrica padrão para ambos os algoritmos em dados tabulares, ela é fundamentalmente inadequada para séries temporais, especialmente as intermitentes, pois falha em capturar padrões morfológicos ou similaridade de processo [59]. Portanto, a seleção da métrica é a decisão metodológica mais importante.

3.3 Métricas de Dissimilaridade para Séries Temporais

A literatura nesta área é comumente categorizada em duas filosofias principais para medir a similaridade em séries temporais: *shape-based* (baseada na forma) e *model-based* (baseada em modelo) [59].

3.3.1 Abordagem *Shape-Based*: *k-Shape* e a Métrica SBD

A abordagem *shape-based* agrupa séries que possuem morfologia visual similar. O principal expoente desta classe é o algoritmo *k-Shape* [76].

O *k-Shape* não deve ser confundido com o K-means padrão. Conforme descrito em [76], o *k-Shape* é um algoritmo de refinamento iterativo (similar ao K-means), mas que difere em dois aspectos cruciais: ele utiliza uma métrica de distância específica, a *Shape-Based Distance* (SBD), e emprega um método de computação de centroides distinto, projetado para extrair uma "forma" média representativa das séries no *cluster*. A SBD é uma medida normalizada de correlação cruzada, que torna a métrica invariante a deslocamento de fase e escala. Isso permite ao *k-Shape* identificar séries com padrões geométricos idênticos, mesmo que ocorram em momentos ligeiramente diferentes.

3.3.2 Abordagem *Model-Based*: `LikelihoodDistance` (*TimeSmash*)

Para dados de varejo dominados por intermitência, a "forma" é ruidosa ou irrelevante. A abordagem *model-based* torna-se, assim, teoricamente superior. Esta abordagem ignora a forma superficial e agrupa séries que foram geradas pelo mesmo processo estocástico subjacente.

A métrica `LikelihoodDistance`, disponibilizada pelo pacote *TimeSmash* [43], é a principal ferramenta *model-based* utilizada nesta tese. Este processo opera em duas fases principais. Primeiramente, na Inferência do Modelo, um conjunto de modelos de Autômatos Finitos Probabilísticos (PFSA) é inferido a partir do conjunto de dados. Cada PFSA representa uma "gramática" estocástica que descreve a dinâmica de uma série, definindo a probabilidade de transitar entre estados (como "sem venda" e "com venda") e emitir um símbolo. A segunda fase é a Extração de Características. Nela, em vez de retornar uma matriz de distância $N \times N$, o método transforma cada série temporal S_i em um vetor de características (*embedding*). Este vetor é composto pelo *log-likelihood* (log-verossimilhança) de que a série S_i tenha sido gerada por cada um dos modelos PFSA inferidos. Todo este *pipeline*, desde a inferência do PFSA até o uso da convergência de *log-likelihood* como um *embedding* de divergência, é a base do método proposto em [43].

O resultado é um espaço de características abstrato em que a distância (agora sim, Euclidiana) entre dois vetores representa a similaridade entre os processos geradores das séries originais.

3.4 Metodologias Híbridas Adotadas na Tese

As seções anteriores definiram os algoritmos (`K-means`, SOM) e as métricas (SBD, `LikelihoodDistance`). Esta tese, no entanto, implementa combinações híbridas específicas para alcançar seus objetivos, conforme detalhado na Seção 5 e avaliadas no Capítulo 6.

3.4.1 *TimeSmash*: K-means no Espaço de Características *Model-Based*

A metodologia referida nesta tese como *TimeSmash* (conforme o Capítulo 6) representa a aplicação do algoritmo K-means padrão (da biblioteca *scikit-learn*) sobre o espaço de características gerado pela métrica `LikelihoodDistance`.

O processo metodológico segue o *pipeline* da abordagem *model-based*, desdobrando-se em três etapas:

1. Executa-se a Inferência do Modelo, na qual a `LikelihoodDistance` é ajustada ao conjunto de dados de treino para inferir os modelos PFSA subjacentes que melhor capturam a dinâmica estocástica das séries;
2. Transformação do Espaço de Características: o *dataset* original de séries temporais é então projetado neste espaço de modelos, resultando em um novo *dataset* tabular em que cada série é representada por um vetor de características (um *embedding*) composto pelo *log-likelihood* (log-verossimilhança) de que aquela série tenha sido gerada por cada um dos modelos PFSA inferidos;
3. O algoritmo `K-means` é aplicado a esta matriz de características resultante. Neste ponto, o problema é efetivamente reduzido a uma clusterização tabular padrão, em que a distância Euclidiana é implicitamente utilizada sobre os *embeddings* para agrupar séries que compartilham processos geradores similares.

Portanto, a implementação *TimeSmash* desta tese utiliza `K-means` como algoritmo de particionamento e `LikelihoodDistance` como o método de extração de características.

3.4.2 *Self-Organizing Maps* (SOM)-LikelihoodDistance: A Contribuição Proposta

De forma análoga, a principal contribuição metodológica desta tese, referida como SOM, é a hibridização do algoritmo *Self-Organizing Maps* (SOM) com a métrica `LikelihoodDistance` [43].

O processo é conceitualmente semelhante ao `K-means`:

1. Os vetores de características (*embeddings*) *model-based* são gerados pela `LikelihoodDistance`.
2. O algoritmo SOM é então treinado usando esses vetores como entrada.

A vantagem desta abordagem é que ela combina o poder da extração de características *model-based* (ideal para dados intermitentes) com a capacidade de mapeamento topológico e visualização do SOM, permitindo a geração dos mapas de similaridade de modelos vistos no Capítulo 6 (Figuras 6.7 a C.3).

CAPÍTULO 4

Aprendizado de Máquina Automatizado (AutoML)

A crescente complexidade e escala dos problemas de previsão, como os encontrados no varejo, impulsionaram a adoção de modelos de ML. No entanto, o desenvolvimento de *pipelines* de ML de alto desempenho é um processo notório por seu alto custo, complexidade e dependência de especialistas com profundo conhecimento de domínio. Este processo manual envolve múltiplas etapas, incluindo pré-processamento de dados, engenharia de *features*, seleção de algoritmos e otimização de hiperparâmetros (HPO). O sucesso de todo o empreendimento depende criticamente das decisões tomadas em cada uma dessas fases.

Neste contexto, o Aprendizado de Máquina Automatizado, ou AutoML, emergiu como um campo de pesquisa fundamental que visa automatizar o ciclo de vida completo da modelagem preditiva. O objetivo do AutoML é tornar as técnicas de ML mais acessíveis a não especialistas (um processo de "democratização") e acelerar a obtenção de resultados por especialistas, automatizando as tarefas mais laboriosas e propensas a erro [44]. Sistemas de AutoML buscam substituir a busca manual, muitas vezes heurística, por um processo sistemático e computacionalmente eficiente para encontrar o melhor *pipeline* de modelagem para um determinado *dataset* e tarefa [44].

4.1 O Problema CASH: O Cerne do AutoML

No núcleo da maioria dos sistemas de AutoML reside o problema formalizado como *Combined Algorithm Selection and Hyperparameter Optimization* (CASH) [92]. Um *pipeline* de ML é definido não apenas pelos hiperparâmetros de um único modelo, mas também pela escolha do próprio algoritmo de aprendizado e, frequentemente, pelas etapas de pré-processamento e extração de características. O problema CASH, portanto, busca otimizar simultaneamente a seleção de um algoritmo A de um conjunto de algoritmos possíveis $\mathcal{A}$, e a configuração de seus hiperparâmetros λ de um espaço de configura-

ção Λ_A. O objetivo é encontrar a combinação A^*, λ^* que minimiza o erro de validação $\mathcal{L}$ em um conjunto de dados de validação $\mathcal{D}_{valid}$, após o treinamento em $\mathcal{D}_{train}$.

Resolver este problema de otimização combinada é extremamente desafiador devido ao espaço de busca vasto, heterogêneo e condicional (em que os hiperparâmetros de um ARIMA são irrelevantes se o algoritmo selecionado for o LightGBM). Para navegar neste espaço complexo, sistemas de AutoML empregam técnicas sofisticadas de otimização, distanciando-se de buscas manuais ou em grade (*grid search*).

4.2 Componentes do *Pipeline* de AutoML

Um sistema de AutoML robusto automatiza um *pipeline* de múltiplos estágios, cada um apresentando seus próprios desafios.

A primeira etapa, a Engenharia de *Features*, é frequentemente o passo mais crítico para o sucesso de modelos tabulares. No contexto de séries temporais, isso envolve a extração de características relevantes, como *lags*, estatísticas móveis e atributos baseados em calendário. Ferramentas como o TSFRESH [23], automatizam este processo extraindo e filtrando sistematicamente centenas de características, conforme detalhado na Seção 5.2.

A segunda etapa é a Otimização de Hiperparâmetros (HPO). Conforme explorado na Seção 5.3, modelos como o LightGBM possuem dezenas de hiperparâmetros que impactam drasticamente seu desempenho. A otimização manual é impraticável. Sistemas de AutoML empregam métodos de otimização baseados em modelos, como a Otimização Bayesiana (utilizada pelo GPyOpt) ou estimadores baseados em árvores, como o *tree-structured Parzen estimator* (TPE) (utilizado pelo HyperOpt), para encontrar configurações de alto desempenho de forma mais eficiente do que buscas aleatórias [12].

A terceira etapa é a Seleção de Modelos e *Ensembling*. Dado um *pool* de modelos (como os descritos no Capítulo 2), o sistema deve decidir qual modelo ou combinação de modelos usar. Abordagens de *ensembling* (combinação) frequentemente superam qualquer modelo individual. Metodologias avançadas, como o FFORMA [70], utilizam as próprias características das séries temporais (extraídas de forma semelhante ao TSFRESH) para aprender pesos ótimos para combinar as previsões de diferentes modelos, demonstrando a força de abordagens baseadas em meta-aprendizagem.

4.3 Desafios do AutoML para Séries Temporais

A aplicação do AutoML à previsão de séries temporais introduz complexidades únicas, não presentes em problemas de ML tabular padrão. Os dados não são independentes e identicamente distribuídos (i.i.d.), exigindo esquemas de validação cruzada que

respeitem a ordem temporal. Além disso, o fenômeno do *concept drift*, em que as propriedades estatísticas dos dados mudam ao longo do tempo, exige que os *pipelines* sejam reavaliados e potencialmente retreinados continuamente, exacerbando o custo computacional.

A competição M5 [67] forneceu uma evidência clara de que o sucesso na previsão de varejo em larga escala não vem de um único modelo superior, mas de um *pipeline* robusto, em que a engenharia de *features* e o ajuste de modelos de *Gradient Boosting* (como o LightGBM) foram as estratégias vencedoras. Automatizar este *pipeline* é o objetivo de sistemas de AutoML para séries temporais, como o AutoGluon-TimeSeries [84].

Contudo, um desafio permanece: o custo computacional. Em um cenário de varejo com milhares de SKUs, aplicar um processo completo de AutoML (testando múltiplos modelos e otimizando hiperparâmetros) para cada série individual é computacionalmente proibitivo. Por outro lado, um único modelo global pode falhar em capturar as dinâmicas distintas de subconjuntos de produtos, como os dados intermitentes.

É nesta interseção de desafios que a metodologia proposta no Capítulo 5 se insere. A literatura recente tem explorado a abordagem de *cluster-then-forecast* como um mecanismo para otimizar essa troca [101]. Ao agrupar séries com dinâmicas homogêneas, pode-se aplicar estratégias de AutoML de forma mais direcionada. Esta tese avança essa ideia ao propor uma arquitetura de AutoML que utiliza uma forma inovadora de clusterização *model-based* (Seção 3.4.2) como um passo de meta-aprendizagem, visando reduzir drasticamente o espaço de busca para a seleção de modelos (Seção 5.5) e, assim, endereçar o gargalo de MLOps sem sacrificar a acurácia.

CAPÍTULO 5

Metodologia Proposta

Há diversas abordagens para seleção de modelos, no entanto esta tese dedica-se a um modelo de AutoML cujo *pipeline* é composto por diversas etapas: uma primeira etapa de pré-processamento para tratamento básico dos dados, tais como valores faltantes, *outliers*, dentre outros; uma fase de engenharia de *features*, pela qual são obtidas informações da série temporal propriamente dita, sendo parte fundamental para uma comparação entre as diversas séries temporais; uma terceira etapa em que as séries são clusterizadas utilizando como entrada somente o valor a ser predito; seguida da seleção de modelos. Por fim, os modelos designados a cada série temporal são treinados. Dessa maneira, não só se busca a obtenção de melhores predições, quando comparadas a um único modelo, mas também a otimização de recursos. A Figura 5.1 ilustra o *pipeline* utilizado.

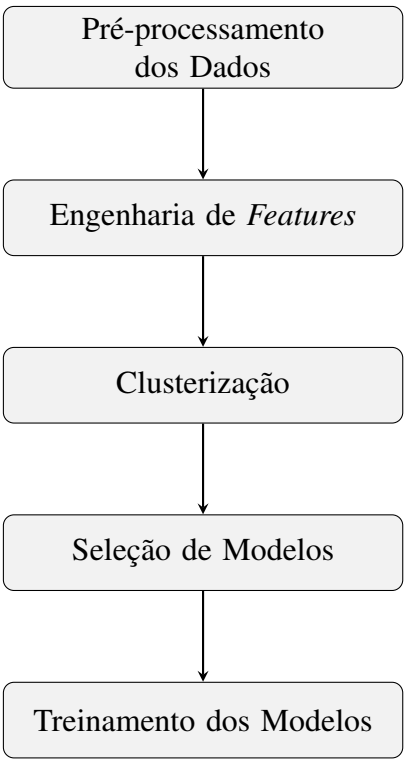


Figura 5.1: Representação gráfica do *pipeline* de AutoML proposto.

Com o uso de AutoML é possível a obtenção de resultados comparáveis aos de modelos ajustados por especialistas, porém, de forma automática e sem a necessidade, *a priori*, de conhecimento especializado. Dessa forma, o próprio mecanismo de autoaprendizagem é capaz de se ajustar às peculiaridades em cada conjunto de dados. O principal foco deste trabalho foi em séries temporais do ramo varejista, portanto, séries com grande relevância e ampla abrangência. Suas peculiaridades de intermitência, influência de eventos futuros, tais como uma promoção de *Black Friday*, sazonalidade, tendência, *confounders*, entre outros, são aspectos de grande influência nas séries e adicionam camadas de desafios relevantes aos modelos preditivos.

Nesta tese, foram analisados dados reais de lojas do ramo farmacêutico, fornecidos por meio de um projeto firmado com o Centro de Excelência em Inteligência Artificial (CEIA). O time é composto por três equipes, com as seguintes responsabilidades: engenharia de *features*, modelagem e interpretabilidade. Para o desenvolvimento deste trabalho, foram utilizadas as implementações das duas primeiras equipes, sendo o autor desta tese o líder da equipe de modelagem.

5.1 Pré-processamento de dados

Em *datasets* reais há diversos aspectos que devem ser levados em consideração antes do seu emprego propriamente dito. O processo de obtenção de dados está sujeito a fatores que podem distorcer a realidade dos valores inseridos, sendo esta fase uma das mais importantes para a obtenção de bons resultados.

No varejo, a ausência de dados de venda de um produto pode ter múltiplos significados. Por um lado, pode indicar um problema técnico, como um banco de dados corrompido ou uma falha na extração das informações. Por outro, a falta de registro pode ser um dado real: o produto simplesmente não vendeu, seja por falta de estoque ou por baixa procura. Embora não tão frequentes no setor varejista, até mesmo valores negativos podem ocorrer, sendo a devolução de um material como a razão mais provável.

Da mesma forma, picos de venda também exigem uma análise cuidadosa. Um aumento repentino pode ser resultado de um evento real, como a alta demanda por máscaras durante a pandemia de COVID-19, ou apenas um erro de digitação no lançamento dos dados (por exemplo, a adição de um zero a mais).

É importante entender que a venda intermitente — caracterizada por produtos que vendem esporadicamente — é uma característica comum em grandes varejistas com um catálogo extenso. Esta intermitência é um padrão de venda real e não deve ser confundida com a ausência de informação causada por falhas técnicas.

Em uma plataforma de AutoML, necessita-se de uma fase não muito invasiva e mais generalista sobre os dados, uma vez que certos pressupostos podem ser aplicados

somente a uma ou a um pequeno grupo de séries temporais. No pré-processamento implementado, os aspectos observados são: conformidade de tipos de valores, séries com baixo valor de receita, baixo valor de vendas e com poucos dados.

Na conformidade de tipos de valores, cada característica é formatada de modo a se enquadrar como do tipo *string*, *integer*, *datetime* ou *float*. Esta padronização é de extrema relevância quando da extração de características, evitando, por exemplo, que o código do produto seja tratado como do tipo numérico e utilizado para obtenção de métricas como média ou variância, que devem influenciar negativamente os modelos preditivos.

Dentre os principais objetivos de predição para o mercado varejista constam o controle de estoque, fluxo de caixa e lucro. Portanto, é possível descartar séries que não se enquadram nestas características, evitando o gasto computacional com séries que não se enquadram no principal objetivo. Para a questão de fluxo de caixa e/ou lucro, são descartadas as séries que, ao longo do horizonte, não ultrapassaram o valor pré-definido pelo usuário. Para auxiliar a parte de logística, a variável de quantidade de vendas é observada, e novamente são descartados produtos que não ultrapassam o limiar pré-estabelecido. Por fim, para que haja informações históricas suficientes para os modelos preditivos, séries com poucas amostras temporais também são ignoradas.

Especificamente nos dados estudados nesta tese, foram estudadas 4 lojas que totalizam 641.266 produtos ou séries, conjunto que, após o descarte explicado anteriormente, resultou em 9.619 séries e 4.471.968 observações.

Optou-se por preencher os valores faltantes de quantidade vendida e receita com 0. Esta decisão é fundamentada pelo fato de que no *dataset* utilizado, grande parte dos produtos vendidos apresentam características de intermitência com valores de ADI acima de 1,32, fato também observado por Johnston et al. [50].

Para esta etapa, foi utilizado majoritariamente um algoritmo proposto pela contratante do projeto.

5.2 Engenharia de *features*

Uma das fases mais importantes, principalmente para os modelos mais complexos, baseados em ML, é a engenharia de *features*. É nela que são extraídas características da série temporal e que agregarão informações adicionais aos modelos. Com estes atributos, as predições podem alcançar melhores valores, tendo em vista a maior quantidade de informação para gerar conhecimento.

Os modelos estatísticos, em sua grande maioria, são univariados, ou seja, não foram concebidos para utilizar particularidades extras, sendo observada somente a característica a ser predita. Neste caso, a engenharia de *features* não agregará informação. No

caso proposto, esta fase aplica-se somente aos modelos LightGBM, LSTM, DeepAR e TFT.

Além do uso nos modelos preditivos, sua participação será fundamental em outra fase do *pipeline*: a clusterização. Nesta fase, as características obtidas comporão o processo decisório para o agrupamento das séries temporais.

Existem alguns *frameworks* desenvolvidos para a extração de *features*, no entanto optou-se pelo uso do TSFRESH apresentado em [23]. Nele, são calculadas até 794 características, porém, para evitar que atributos irrelevantes sejam retornados, o que poderia causar *overfitting* aos modelos, faz parte do processo uma fase de seleção de *features*, na qual somente são retornadas as características após um teste de hipótese estatístico.

A Figura 5.2 exemplifica o processo de extração, ilustrando atributos elementares que a biblioteca TSFRESH é capaz de computar. Nela, observam-se descritores estatísticos que sumarizam a distribuição de valores (Média, Mediana, Mínimo e Máximo) e características morfológicas, como, por exemplo, a contagem de picos (*Number Peaks*), que informam sobre a estrutura da série.

Contudo, o escopo de funcionalidades do TSFRESH é consideravelmente mais amplo, permitindo uma caracterização exaustiva das séries temporais. Além de descritores estatísticos avançados, como momentos de ordem superior, o repertório completo de atributos inclui métricas de dependência temporal para quantificar tendências e autocorrelações, bem como atributos no domínio da frequência, para a análise de periodicidade. A caracterização é finalizada com estimadores de complexidade e dinâmica não-linear, como a entropia, que buscam definir a previsibilidade do processo gerador dos dados.

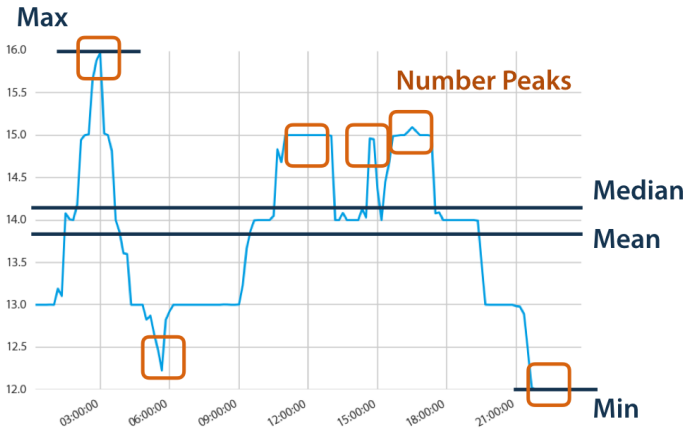


Figura 5.2: Exemplo de características extraídas pelo TSFRESH [22].

5.3 Implementação dos modelos

Foram implementados os 10 modelos apresentados no Capítulo 2. Este portfólio de estimadores foi deliberadamente construído para ser heterogêneo, abrangendo diferentes classes de abordagens. O intuito desta heterogeneidade é garantir que o *pipeline* de AutoML disponha de um repertório robusto, capaz de selecionar um modelo adequado para as distintas dinâmicas (como intermitência, sazonalidade ou tendência) inerentes a cada série temporal. Para a melhor organização desta tese, este portfólio foi agrupado nas três categorias distintas que representam esses paradigmas: Estatísticos, LightGBM e Neurais.

Os modelos são avaliados nas métricas: SMAPE, RMSSE e MASE, e, na Seção 6.1, cada uma dessas métricas será detalhada. Além disso, o tempo computacional também é observado, uma vez que é parte relevante no processo do AutoML.

A definição da estratégia de particionamento dos dados é uma etapa metodológica crítica na avaliação de modelos de séries temporais. Diferentemente de problemas tabulares padrão, nos quais a validação cruzada aleatória (*k-fold*) é comum, a natureza sequencial e dependente do tempo dos dados exige uma divisão cronológica estrita para evitar o vazamento de dados (*data leakage*). O vazamento ocorreria se informações do futuro fossem inadvertidamente utilizadas durante o treinamento ou validação, levando a uma avaliação de desempenho excessivamente otimista e metodologicamente falha [47].

Para garantir uma avaliação robusta que simule um cenário de produção real, em que apenas o passado é conhecido, esta tese adota uma abordagem de partição temporal de origem fixa, conforme referenciado no Capítulo 6. O objetivo é prever um horizonte de 14 dias. Desta forma, o conjunto de dados é dividido em três blocos contíguos e mutuamente exclusivos: o conjunto de teste é rigorosamente definido como os últimos 14 dias do histórico de vendas; o conjunto de validação é composto pelos 14 dias imediatamente anteriores ao conjunto de teste; e o conjunto de treino compreende todos os dados restantes anteriores ao período de validação. O conjunto de treino é usado para o ajuste primário dos modelos; o conjunto de validação é empregado para a otimização de hiperparâmetros (HPO); e o conjunto de teste é reservado exclusivamente para a avaliação final e imparcial do desempenho do *pipeline* completo.

5.3.1 Estatísticos

Este grupo é composto por 6 modelos: *Naive*, suavização exponencial, ARIMA, TSB, Prophet e Theta. Para estes modelos, a fase de engenharia de *features* não é utilizada, com exceção ao Prophet, que utiliza o recurso de informação de um calendário de eventos. Além disso, por serem capazes de fazer predições de séries univariadas, os modelos deste grupo recebem dados referentes a apenas uma série temporal por vez. Isso limita

o modelo a obter informações somente daquela série temporal, sem utilizar informações correlacionadas às outras séries temporais.

Na modelagem destes modelos foi utilizada a biblioteca Darts [40], que fornece uma maneira simplificada para uso dos diversos modelos, salvo para o modelo TSB, implementado pelo autor.

A correta identificação da estacionariedade de uma série temporal é um passo metodológico crítico, sendo um requisito fundamental para a escolha do modelo de previsão [30]. O principal desafio reside no diagnóstico correto, e é por isso que os testes ADF e KPSS são frequentemente usados em conjunto. A diferença fundamental entre eles está em suas hipóteses nulas opostas [51]. O teste ADF tem como hipótese nula (H_0) que a série possui uma raiz unitária (sendo, portanto, não-estacionária) [9]. Em contraste, o teste KPSS, proposto por Kwiatkowski et al. (1992), é um teste de estacionariedade [51]. Sua hipótese nula (H_0) é que a série é estacionária (ou estacionária em torno de uma tendência determinística) [56, 51]. O impacto no processo de previsão é direto: uma falha em diagnosticar corretamente a natureza da série (por exemplo, confundir uma série estacionária em tendência com uma série que possui raiz unitária) leva à má especificação do modelo, o que, por sua vez, pode comprometer significativamente a acurácia da previsão, principalmente de modelos estatísticos. No caso estudado, 96% são consideradas estacionárias quando se aplica o teste ADF e 65% quando se aplica o teste KPSS.

Por se tratar de modelos estatísticos, alguns cuidados são levados em consideração antes do efetivo uso dos modelos. Primeiro, verifica-se a sazonalidade da série, em que o ACF é calculado e passa pelo teste de hipótese homocedasticidade de Bartlett [8]. Esta informação de sazonalidade é de extrema relevância e aplicada aos modelos *Naive*, suavização exponencial e ARIMA.

Na implementação do ARIMA é utilizado o algoritmo AutoARIMA e ainda são realizados os testes KPSS e ADF, além de ser adotado o maior valor para nortear o parâmetro d, já elucidado na Seção 2.3. De forma análoga, o parâmetro D é obtido pelo maior valor entre os testes OCSB e CH. O algoritmo fará uma otimização de parâmetros a fim de encontrar a melhor combinação dos parâmetros deste modelo, limitados à ordem 5, ou seja, $p+q+P+Q <= 5$.

5.3.2 LightGBM

O LightGBM passou por uma investigação mais profunda, uma vez que seus resultados estão constantemente entre os melhores, contudo, estes resultados são fortemente impactados pelos parâmetros adotados. Os dados oriundos da fase de engenharia de *features* são incluídos no *dataset* com vistas a agregar informação entre as séries e

obter melhores resultados. Para este modelo, optou-se pela implementação por meio da biblioteca oficial do LightGBM [52].

Visto que o objetivo é obter uma plataforma de AutoML, a fixação de parâmetros não é desejável. Por isso, foram testadas 5 diferentes bibliotecas de otimização de parâmetros ou HPO, que encontram a melhor combinação de parâmetros por meio de diferentes métodos de busca, dentro de um espaço definido pelo usuário. As cinco bibliotecas testadas foram: HyperOpt [13] e HpBandSter [31] que utilizam TPE; *Sequential Model Algorithm Configuration* (SMAC) [61] que utiliza RF; e GPyOpt [5] e Scikit-Optimize que utilizam processos gaussianos.

Cada biblioteca de otimização possui uma forma distinta para entrada dos dados, por isso, foi desenvolvida uma plataforma comum, que traduz o espaço de busca definido pelo usuário para as diferentes bibliotecas.

Na documentação oficial do LightGBM existem dezenas de parâmetros, porém, neste estudo, 15 foram explorados, sendo: *num_leaves*, *colsample_bytree*, *learning_rate*, *min_data_in_leaf*, *bagging_fraction*, *neg_bagging_fraction*, *bagging_freq*, *feature_fraction*, *min_data_per_group*, *max_cat_threshold*, *cat_l2*, *cat_smooth*, *refit_decay_rate*, *cegb_tradeoff* e *num_iteration*.

Embora o processo de HPO, detalhado na Seção 5.3, seja projetado para encontrar os valores ótimos, a seleção deste conjunto específico de 15 parâmetros não é arbitrária. Ela reflete um entendimento dos mecanismos que governam o desempenho do LightGBM e podem ser agrupados conceitualmente.

Um primeiro grupo de parâmetros gerencia a complexidade estrutural e o ajuste do modelo. O hiperparâmetro *num_leaves* é o principal controlador da complexidade da árvore; valores maiores permitem que o modelo capture padrões mais complexos, mas aumentam o risco de *overfitting*. Para mitigar isso, *min_data_in_leaf* atua como um regularizador, impedindo que o modelo crie folhas (segmentações) baseadas em poucas amostras, forçando-o a aprender padrões mais gerais. O parâmetro *num_iteration* define o número de árvores (ou rodadas de *boosting*) a serem construídas, controlando o ajuste total do *ensemble*.

Um segundo grupo de parâmetros foca na regularização por meio de amostragem, uma técnica essencial para a robustez do LightGBM. Os parâmetros *bagging_fraction* e *bagging_freq* controlam a frequência e a fração de dados a serem amostrados (*bagging*), o que introduz variabilidade e reduz o *overfitting*. De forma análoga, *feature_fraction* e *colsample_bytree* realizam a amostragem no nível das colunas, selecionando um subconjunto de *features* para construir cada árvore. Isso é particularmente útil em *datasets* com alta dimensionalidade, como os gerados pela engenharia de *features* (Seção 5.2). O *neg_bagging_fraction* é um controle adicional para lidar com dados desbalanceados.

O controle do processo de aprendizado é governado pelo *learning_rate* (taxa de aprendizado), que dimensiona a contribuição de cada nova árvore ao *ensemble*. Um valor baixo requer mais iterações (*num_iteration*), mas geralmente leva a um resultado mais robusto. O *refit_decay_rate* é um parâmetro avançado, tipicamente usado em cenários de atualização de modelos, controlando como os pesos das árvores decaem durante um reajuste.

Um conjunto de parâmetros é dedicado a uma das maiores vantagens do LightGBM: seu tratamento nativo de variáveis categóricas, o que é de extrema relevância para os dados de varejo (como SKUs, dias da semana ou eventos). Os parâmetros *max_cat_threshold*, *min_data_per_group*, *cat_l2* e *cat_smooth* trabalham em conjunto para controlar como o modelo agrupa e regulariza as características categóricas. Eles ajudam a evitar o *overfitting* em categorias com baixa representatividade e a gerenciar a cardinalidade, otimizando as divisões de árvore sem a necessidade de pré-processamento manual como o *one-hot encoding*.

Finalmente, o parâmetro *cegb_tradeoff* está ligado à otimização de custo-benefício (*Cost-Efficient Gradient Boosting*). Ele introduz uma penalidade por avaliar divisões em diferentes *features*, incentivando o modelo a reutilizar *features* e, assim, otimizando o *trade-off* entre acurácia e o custo computacional do treinamento, um tema central desta tese.

5.3.3 Neurais

Constam neste grupo os modelos: LSTM, DeepAR e TFT e, analogamente ao modelo LightGBM, também são passadas aos modelos as características extraídas da fase de engenharia de *features*. Na implementação desses modelos, foi utilizado o pacote pytorch-forecasting [10], cuja adaptação para aplicação aos casos estudados foi realizada pelo coordenador no projeto.

5.4 Clusterização

Os modelos de clusterização citados no Capítulo 3 foram utilizados em função da sua capacidade de identificar padrões na seleção de modelos de predição que podem ser mais adequados para aquele grupo de séries.

Modelos como o FEDOT [71] ou FFORMA [70] são capazes de obter melhores resultados que o melhor modelo de predição de séries temporais, e é neste sentido que a clusterização pode ser vantajosa, indicando os principais modelos a serem aproveitados para a predição final. A composição de vários modelos mostrou-se vantajosa do ponto de vista de erro, porém vem com a contrapartida de que é necessário um elevado custo com-

putacional para alcançá-los. Dependendo do volume financeiro envolvido nas predições, pequenos ganhos nas métricas de acurácia fazem com que o custo computacional seja compensado, no entanto, para o caso do varejo, em que há milhares de produtos e lojas, este custo computacional pode inviabilizar a aplicação de modelos mais complexos. Não é somente o custo que pode ser elevado, mas o tempo de processamento de modelos mais complexos pode vir a tornar-se impeditivo. Especialmente no varejo, em que há milhares de produtos por loja, muitos deles com baixa ou nenhuma venda por vários dias, modelos de predição mais simples, tendem a obter resultados equivalentes e até melhores que os neurais e baseados em ML.

Após passar as séries temporais pelos modelos de clusterização, utilizando todo o conjunto de treino, cada série recebe a atribuição de um *cluster*. Neste ponto, é pertinente justificar a seleção do número de *clusters* (k), um hiperparâmetro fundamental.

É notável que a metodologia não emprega técnicas heurísticas tradicionais para a estimativa de k, como o método do "cotovelo" (*elbow*) ou a análise de silhueta. Tal omissão é deliberada e fundamenta-se no objetivo desta tese, que diverge da clusterização exploratória padrão. Métodos de validade interna, como o cotovelo, otimizam métricas de coesão e separação geométrica (como a variância intra-cluster). Contudo, o objetivo deste *pipeline* de AutoML não é encontrar os agrupamentos mais compactos, mas sim particionar o espaço de problemas de forma a otimizar a acurácia da previsão e a eficiência computacional da seleção de modelos subsequente. A otimização da separação geométrica não garante a otimização do erro de previsão.

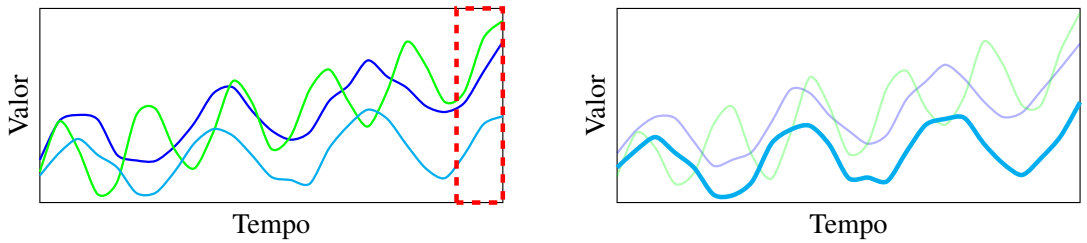
Portanto, o número de *clusters* k é tratado como um hiperparâmetro estratégico da arquitetura. A decisão de fixar $k = 10$ para os métodos *k-Shape* e *TimeSmash* foi alinhada à quantidade de modelos preditivos no *pool* (dez). Esta escolha visa estabelecer um balanço metodológico: k deve ser suficientemente grande para permitir a especialização (separando séries com dinâmicas distintas que podem se beneficiar de modelos diferentes), mas suficientemente pequeno para garantir que cada *cluster* mantenha uma população estatisticamente robusta. Um k muito pequeno (como $k = 2$) agruparia séries excessivamente heterogêneas, enquanto um k muito grande (como $k > 100$) resultaria em *clusters* esparsos, tornando a abordagem de seleção por amostragem (detalhada na Seção 5.5) estatisticamente inviável.

Para o SOM, a dimensão do mapa foi definida heuristicamente por $\text{tamanho} = \sqrt{\frac{n_{produtos}}{100}}$, com o mesmo intuito de garantir uma quantidade suficiente de itens por neurônio, ao mesmo tempo que se preserva a granularidade topológica.

Dessa forma, a avaliação da "qualidade" da clusterização nesta tese não é medida por sua coesão interna, mas sim pela sua eficácia na tarefa final: o desempenho do erro de previsão e o ganho computacional obtido pela arquitetura de seleção de modelos que ela habilita, conforme avaliado empiricamente no Capítulo 6.

5.5 Seleção de modelos

De forma a evitar o elevado custo computacional associado à aplicação de múltiplos modelos de predição, foi adotada uma estratégia de seleção de modelos baseada nos *clusters* gerados na fase anterior. Esta abordagem permite identificar quais modelos são mais adequados para cada grupo de séries temporais, otimizando assim o processo de predição.



(a) *Janela reduzida* (b) *Conjunto reduzido*

Figura 5.3: Exemplos das metodologias de seleção

Duas abordagens distintas foram implementadas, com vista ao processo de seleção de modelos. A primeira é a utilização de uma amostra recente (janelas reduzidas) de cada série temporal. Para isso, os conjuntos de treino são limitados aos valores mais recentes de 30, 60 e 120 dias. Quanto maior a janela, maior é o contexto para os modelos, porém há também um maior tempo computacional para treino. Um exemplo desta metodologia é ilustrado pela Figura 5.3(a). Nela, destaca-se, em vermelho, os dados mais recentes que serão utilizados para a seleção dos modelos.

Na Figura 5.4 é ilustrada a distribuição da duração das séries nas lojas estudadas nesta tese. Nota-se que a grande maioria das séries possui pelo menos 120 dias de histórico. A moda das séries supera 480 dias, ou seja, 4 vezes mais dados que a maior janela, sendo as janelas propostas adequadas para serem tratadas como amostras.

Enquanto a clusterização busca identificar os perfis de comportamento das séries temporais, a escolha pela janela reduzida busca obter as informações mais valiosas para a predição de curto prazo. A combinação destas duas estratégias visa maximizar a eficiência do processo de predição, reduzindo o custo computacional, sem comprometer a qualidade das previsões. Segue abaixo a metodologia da primeira abordagem, também ilustrada pela Figura 5.6:

- Extração de uma amostra de cada série temporal do conjunto de treino, contando com uma janela fixa de 30, 60 ou 120 dias mais recentes do conjunto de treino;
- Treino de todos os modelos de predição;
- Para cada série, há a seleção dos 3 melhores modelos com base nas métricas de desempenho;

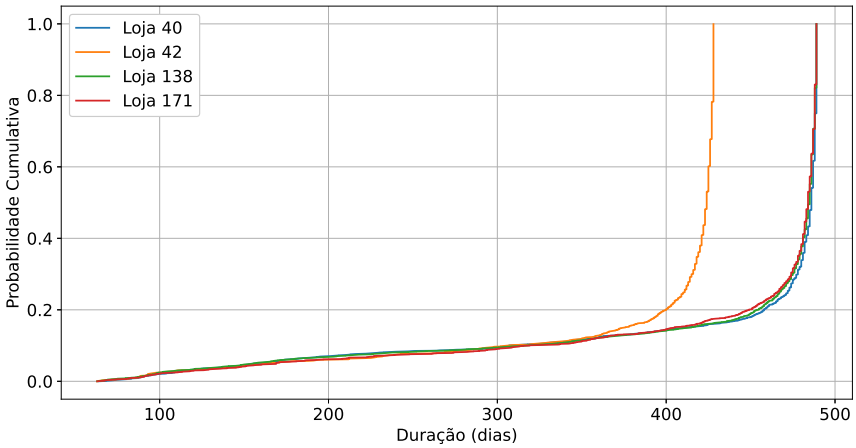


Figura 5.4: Distribuição cumulativa das durações de cada série por loja

- Para cada *cluster*, há a seleção dos 2 modelos com menor erro;
- Designação de modelo para cada série com base no menor erro dentre os modelos disponíveis para seu respectivo *cluster*.

Padrões de consumo, preços e estratégias promocionais alteram-se ao longo do tempo, fazendo com que o histórico distante perca relevância preditiva para o futuro imediato. Ao focar o treinamento de seleção apenas nos 30, 60 ou 120 dias mais recentes, esta metodologia prioriza a agilidade e a adaptação a regimes de mercado recentes.

A segunda abordagem utiliza um percentual de amostras de produtos de cada *cluster* e, então, treinam-se os modelos na totalidade dos dados do conjunto de treino. Um exemplo desta metodologia é ilustrado pela Figura 5.3(b). Nela, destaca-se a curva em azul como sendo uma amostra de série temporal do *cluster*; neste caso, somente ela será utilizada para seleção dos modelos, as demais séries terão o modelo designado por ela. A metodologia é descrita abaixo.

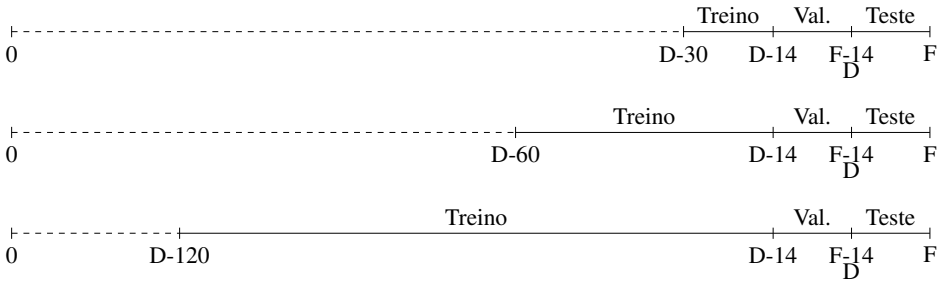
- Extração de uma amostra aleatória de séries temporais de cada *cluster*;
- Treino de todos os modelos de predição;
- Para cada *cluster*, seleção do modelo com menor erro;
- Designação de modelo para cada série com base no menor erro dentre os modelos disponíveis para seu respectivo *cluster*.

A segunda abordagem, de “conjunto reduzido”, opera sob uma premissa metodológica distinta, mais alinhada à meta-aprendizagem e à eficiência computacional. Esta estratégia assume que a etapa de clusterização (Capítulo 3) foi bem-sucedida em agrupar séries com dinâmicas estruturais homogêneas. Sob esta hipótese de homogeneidade *intra-cluster*, o treinamento de modelos no histórico completo de uma pequena amostra

percentual de séries pode servir como um *proxy* estatisticamente robusto para o comportamento de todo o *cluster*. O objetivo aqui não é primariamente capturar o *drift* recente (embora os dados recentes estejam incluídos), mas sim identificar o modelo que melhor se ajusta, apostando na robustez da clusterização para generalizar esta escolha para todos os membros do grupo.

Enquanto a metodologia de janela reduzida busca reduzir o treinamento reduzindo a quantidade de dados de todas as séries temporais encurtando o tamanho de cada série (duração) e limitando-se somente aos dados mais recentes; a metodologia de conjunto reduzido busca a redução mediante a utilização de apenas algumas séries completas (todo o período) do *cluster*.

Para questões de divisão do conjunto de dados, o processo segue o mesmo fluxo do treinamento, ou seja, para a janela reduzida teríamos o conjunto de teste com os últimos 14 dias da série, validação com 14 dias anteriores ao primeiro dia do conjunto de teste e, por fim, o conjunto de treino sendo os 30, 60 ou 120 dias anteriores ao primeiro dia do conjunto de teste. A Figura 5.5(a) ilustra esta divisão. Já para a metodologia de conjunto reduzido não há alteração no *split*, conforme demonstrado na Figura 5.5(b).



(a) *Janela reduzida*

(b) *Conjunto reduzido*

Figura 5.5: Exemplos das metodologias de *split*

O treino dos modelos nesta etapa de seleção segue o mesmo processo de treino dos modelos na janela completa, incluindo a otimização de hiperparâmetros para o LightGBM. A diferença está no fato de que o conjunto de dados é significativamente menor, o que reduz o tempo necessário para o treinamento e a avaliação dos modelos.

O desempenho será observado por meio da comparação de tempo computacional e erros entre as opções com clusterização e o melhor modelo para a série ou para a loja.

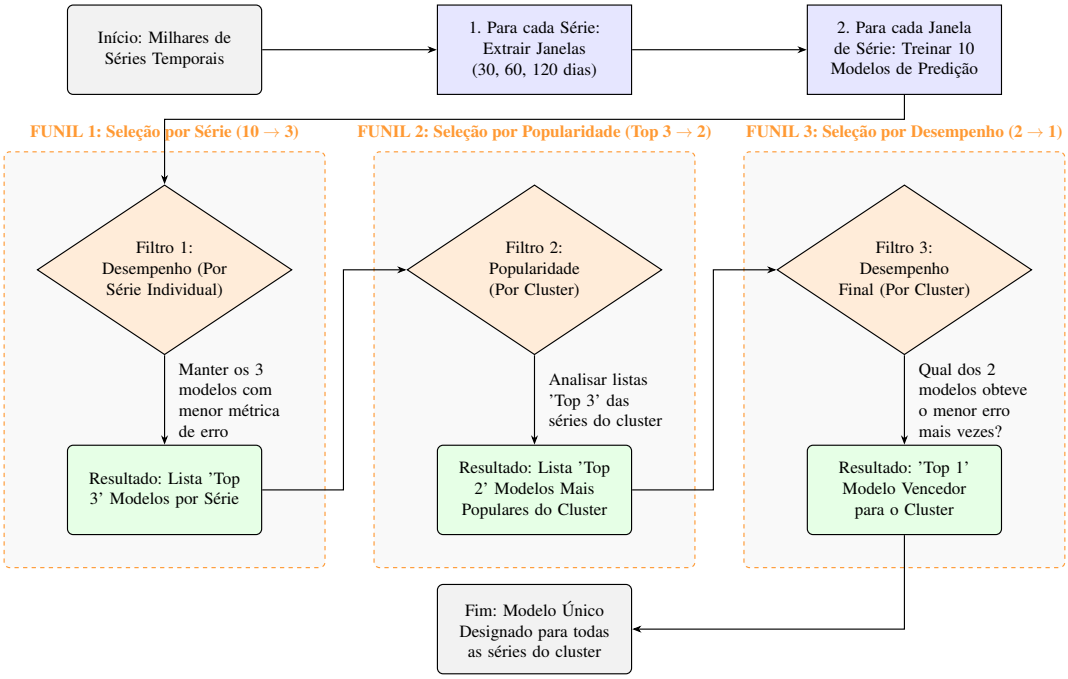


Figura 5.6: *Pipeline* para janelas reduzidas

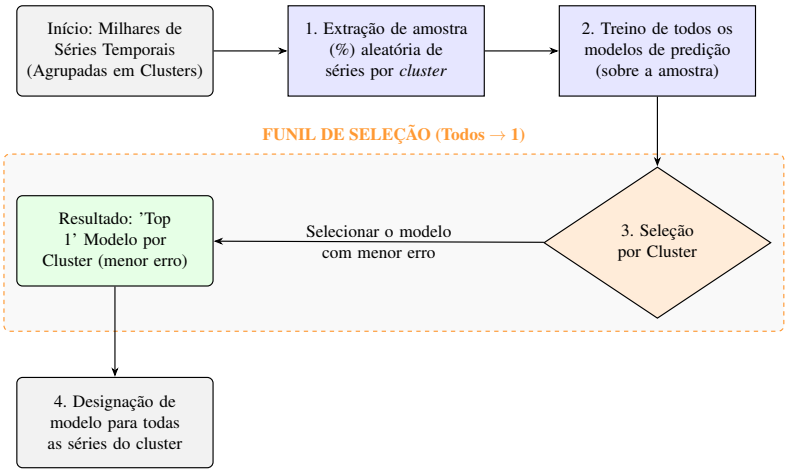


Figura 5.7: *Pipeline* para conjunto reduzido

CAPÍTULO 6

Resultados

Neste capítulo, são apresentados alguns resultados obtidos por meio dos estudos realizados, com ênfase nas apurações dos modelos preditivos e no processo de seleção de modelos por clusterização. É também explorada a questão da classificação de demanda, abordada na Seção 3.1.

6.1 Desempenho dos Modelos

Para a avaliação do desempenho dos modelos foram utilizados os dados reais de quatro lojas do ramo farmacêutico. As lojas foram escolhidas observando o tamanho, em quantidade de produtos, sendo então adotadas as lojas 40, 42, 138 e 171.

O objetivo da avaliação de desempenho, em conformidade com a metodologia descrita no Capítulo 5, é prever a quantidade vendida no horizonte de 14 dias. Para tanto, foi empregada uma estratégia de partição temporal de origem fixa. Esta abordagem é metodologicamente crucial para simular um cenário de implantação real e, fundamentalmente, para garantir a prevenção de vazamento de dados (*data leakage*), o que invalidaria os resultados. Os dados foram divididos em três blocos cronológicos contíguos: o conjunto de teste, compreendendo os últimos 14 dias do histórico; o conjunto de validação, composto pelos 14 dias imediatamente antecedentes; e o conjunto de treino, contendo todos os dados restantes anteriores ao período de validação.

O conjunto de treino é, portanto, o único com tamanho variável, refletindo a natureza diversa do ciclo de vida dos produtos. No entanto, conforme demonstrado na Seção 5.5, a maioria das séries possui um histórico robusto, com a moda superando 480 dias. Garante-se também que as séries que chegam a esta etapa de modelagem já foram filtradas pelo pré-processamento (Seção 5.1), possuindo dados suficientes para o treinamento.

É relevante destacar que este protocolo de avaliação não emprega uma validação cruzada com janelas deslizantes (*rolling window*) ou expansíveis, que avaliariam o desempenho médio de múltiplos retreinamentos. Em vez disso, ao usar uma origem fixa, avalia-se um único e rigoroso cenário: dado todo o histórico disponível até um ponto de

corte, qual é o desempenho do modelo na previsão dos próximos 14 dias. Desta forma, os 14 valores previstos são todos baseados no mesmo conjunto de treino estático. Uma consequência esperada e natural desta abordagem de previsão multi-passo é a acumulação de incerteza, na qual a acurácia tende a diminuir à medida que o horizonte de previsão avança (isto é, a previsão para o 14º dia é intrinsecamente mais incerta do que a previsão para o 1º dia).

Adicionalmente, para permitir uma análise de desempenho estratificada, o portfólio de produtos foi segmentado em seis faixas de vendas. Esta classificação foi determinada pela quantidade total vendida por cada produto durante o período de teste de 14 dias. A Figura 6.1 apresenta a distribuição da contagem de produtos de cada loja (40, 42, 138 e 171) por essas faixas. Os intervalos, detalhados na legenda, utilizam a notação matemática de conjuntos, na qual os colchetes [,] significam respectivamente inclusivo e exclusivo.

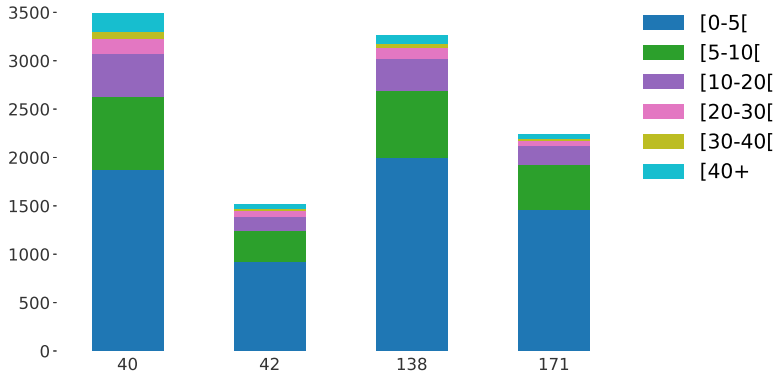


Figura 6.1: Quantidade de produtos por faixa de vendas

Nestas duas semanas, em todas as lojas, predominaram os produtos com baixa quantidade de vendas, sendo que, na maioria dos dias, não houve venda (ou seja, os valores foram nulos). Este fato já era esperado, uma vez que lojas do ramo varejista trabalham com uma grande variedade de produtos. Contudo, isso impõe um desafio ainda maior aos modelos preditivos.

Foram treinados 10 modelos preditivos, e suas avaliações foram feitas segundo quatro métricas distintas: RMSE, SMAPE, RMSSE e MASE.

O *Mean Absolute Percentage Error* (MAPE), definido pela Equação (6-1), traz boa interpretabilidade e possibilita a comparação tanto entre lojas como entre modelos, no entanto, sua formulação é falha quando os valores reais são zero ou na sua marginalidade. Assim sendo, poucos casos podem distorcer a análise de toda uma loja.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{|Y_i|} \tag{6-1}$$

onde n representa o número de observações no horizonte de previsão.

As métricas propostas minimizam esses efeitos, mas em contrapartida, acabam perdendo um pouco da interpretabilidade. O SMAPE, definido pela Equação (6-2), é uma variação do MAPE, e possui as vantagens semelhantes ao MAPE. Apesar de, com este modelo, se retirar a indeterminação quando o valor real Y é nulo, ainda peca quando tanto o valor previsto ($\hat{Y}$) quanto o valor real (Y) são nulos. Visto que se busca minimizar a diferença entre $\hat{Y}$ e Y, portanto é esperado que $\hat{Y}$ seja zero quando Y é zero.

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{(|Y_i| + |\hat{Y}_i|)/2} \tag{6-2}$$

Já o MASE, definido pela Equação (6-3), praticamente elimina o problema da indeterminação. Sua formulação só permite indeterminação quando toda a série de treino possui o mesmo valor. Sua vantagem advém do fator de escala entre o numerador, representado pelo erro absoluto do modelo a ser avaliado, e o denominador, representado pelo erro absoluto da previsão *Naive* no conjunto de treino (onde T é o tamanho do conjunto de treino). O MASE pode ser interpretado como o quão boa é a previsão comparada com uma previsão que simplesmente repete o último valor.

$$MASE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{\frac{1}{T-1}\sum_{t=2}^{T}|Y_t - Y_{t-1}|} \tag{6-3}$$

As métricas detalhadas até agora possuem fatores de escala, e permitem comparações tanto entre modelos como entre lojas, mas não englobam a noção de escala do erro. Por exemplo, errar 1% de um produto com venda de 10 unidades tem impacto diferente de errar 1% em 1.000 unidades. O RMSE, definido pela Equação (6-4), utiliza a raiz da média do erro quadrático e, por ser dependente de escala, lojas com maior volume de vendas tendem a ter um RMSE maior. Isso dificulta a comparação de um mesmo modelo entre lojas. Para mitigar este problema, o RMSSE vem sendo amplamente utilizado na previsão de séries temporais no varejo. O RMSSE, como definido na Equação (6-5), é uma variação do RMSE, que escala, assim como o MASE, o erro do modelo pelo erro de uma previsão ingênua (*Naive*) no conjunto de treino, permitindo comparações justas entre lojas com diferentes volumes de vendas.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{6-4}$$

$$RMSSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{Y}_i}{\frac{1}{T-1}\sum_{t=2}^{T}|Y_t - Y_{t-1}|}\right)^2} \tag{6-5}$$

A utilização de um portfólio de quatro métricas distintas (RMSE, SMAPE,

RMSSE e MASE) não representa uma redundância, mas sim uma necessidade metodológica para uma avaliação holística. Não existe uma única métrica de erro "verdadeira"; cada uma quantifica um aspecto diferente do desempenho do modelo, respondendo a diferentes questões de negócio e estatística. A escolha de qual métrica priorizar depende do objetivo da previsão e da natureza dos dados. O RMSE (6-4), por exemplo, é dependente de escala e penaliza erros grandes de forma quadrática, sendo crucial para a gestão de inventário, em que o custo de um grande erro em um item de alto volume é substancial. Em contrapartida, métricas de erro percentual, como o SMAPE (6-2), são independentes de escala, permitindo a comparação do desempenho relativo entre produtos de volumes distintos.

O desafio central desta tese, e a principal razão para a multiplicidade de métricas, reside na natureza do *dataset*, que é dominado pela demanda intermitente (conforme ilustrado na Figura 6.1). Como já apontado, dados com valores reais nulos invalidam métricas percentuais puras como o MAPE (6-1) e introduzem artefatos no SMAPE (6-2), em que um modelo simples como o *Naive* pode apresentar um erro artificialmente baixo. Para superar esta limitação, as métricas escaladas (MASE (6-3) e RMSSE (6-5)) são essenciais. Elas são robustas a valores nulos e também independentes de escala, pois normalizam o erro do modelo pelo erro de um *benchmark Naive* no conjunto de treino. O RMSSE (6-5), em particular, foi a métrica de avaliação principal da competição M5 precisely por esta combinação de robustez à intermitência e sensibilidade a erros de grande magnitude (devido ao erro quadrático). Portanto, a análise conjunta destas quatro métricas é a única forma de avaliar fidedignamente o desempenho dos modelos neste *dataset* complexo, compreendendo simultaneamente o erro de magnitude (RMSE), o erro percentual enviesado (SMAPE) e o erro de habilidade relativa (RMSSE/MASE).

Para fundamentar a escolha da biblioteca de HPO a ser utilizada no *pipeline* de AutoML, foi conduzido um experimento preliminar. Este experimento não visa "criar" hiperparâmetros, mas sim avaliar a *eficiência* de diferentes *frameworks* de otimização (GPyOpt, HpBandSter, HyperOpt, Scikit-Optimize/skopt e SMAC) na busca pelo conjunto ótimo de parâmetros do LightGBM, detalhados na Seção 5.3. O experimento foi executado nas lojas 42 e 184. Inicialmente, realizou-se uma busca exploratória com 200 iterações na loja 42 para determinar o ponto de convergência. Conforme ilustrado na Figura 6.2 (eixo X por *Number of iterations*), observou-se que todas as bibliotecas atingem um platô de erro, sem melhorias expressivas no RMSE, por volta da 40ª iteração. A Figura 6.3 complementa esta análise, mostrando a mesma convergência em relação ao tempo computacional total (*Overall Time*).

O processo de otimização de hiperparâmetros é reconhecidamente uma das etapas mais onerosas do *pipeline* de ML. A análise inicial (Figura 6.2) demonstrou que o benefício marginal de iterações adicionais após a 40ª é praticamente nulo para esta

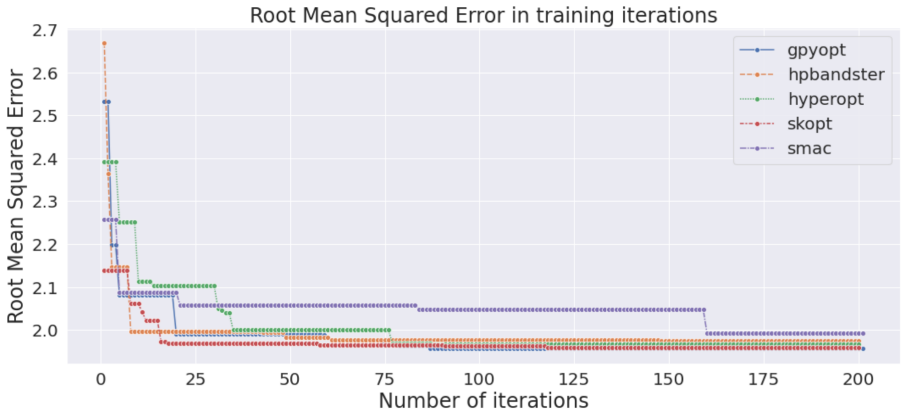


Figura 6.2: Comparação da evolução do erro ao longo das iterações entre as diferentes bibliotecas de HPO na loja 42. Eixo X representa o número de iterações. Eixo Y o RMSE.

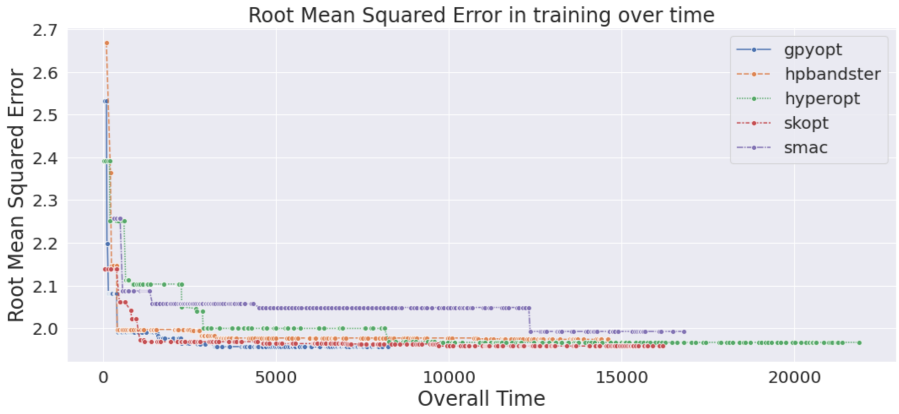


Figura 6.3: Comparação da evolução do erro ao longo do tempo entre as diferentes bibliotecas de HPO na loja 42. Eixo X representa o tempo total. Eixo Y o RMSE.

tarefa. A métrica de seleção mais relevante não é o erro final (pois são equivalentes), mas sim a eficiência computacional — o tempo de processamento (*Overall Time*) necessário para atingir esse vale. Para avaliar esta eficiência, um novo experimento, limitado a 40 iterações, foi conduzido nas lojas 42 e 184, com os resultados plotados contra o tempo computacional nas Figuras 6.4 e 6.5, respectivamente.

A análise das Figuras 6.4 e 6.5 corrobora a escolha. Em ambos os cenários de teste (lojas 42 e 184), a biblioteca GPyOpt consistentemente alcançou o menor RMSE no menor tempo computacional. Enquanto outras bibliotecas, como HyperOpt ou SMAC, eventualmente atingem um patamar de erro similar, elas o fazem com um custo de tempo significativamente maior. Dado que o objetivo da arquitetura de AutoML desta tese é precisamente otimizar o *trade-off* entre acurácia e custo computacional, a GPyOpt demonstrou ser a ferramenta de HPO mais alinhada a este objetivo, sendo, portanto, a escolhida para todos os experimentos subsequentes de otimização do LightGBM.

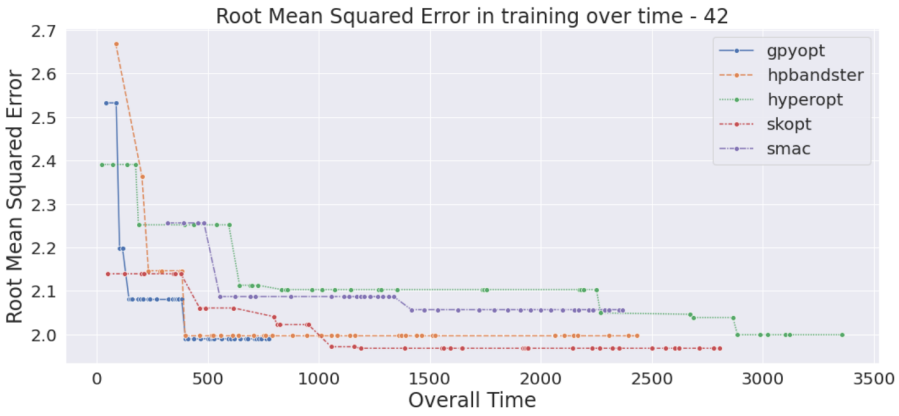


Figura 6.4: Comparação da evolução do erro ao longo do tempo para 40 iterações entre as diferentes bibliotecas de HPO para a loja 42. Eixo X representa o tempo total. Eixo Y o RMSE.

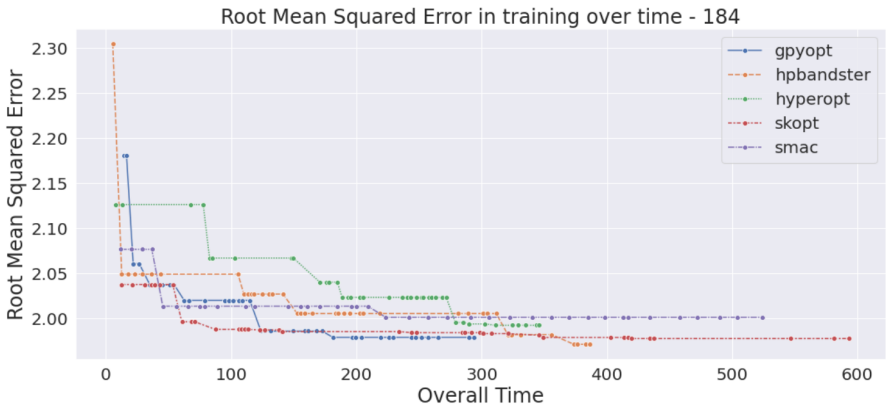


Figura 6.5: Comparação da evolução do erro ao longo do tempo para 40 iterações entre as diferentes bibliotecas de HPO para a loja 184. Eixo X representa o tempo total. Eixo Y o RMSE.

6.1.1 Desempenho da seleção dos modelos

A etapa de seleção de modelos, detalhada na Seção 5.5, é fundamental e objetiva otimizar o processo de predição de modelos, aliando custo computacional à minimização de erros.

Esta etapa depende do resultado da clusterização e foram testados três modelos: *k-Shape*, *TimeSmash* e SOM. O *k-Shape* tem sua abordagem baseada na forma das séries temporais, enquanto o *TimeSmash* utiliza uma abordagem baseada nos processos estocásticos que geram os dados. Na implementação do SOM, em substituição à distância euclidiana, foi utilizada a `LikelihoodDistance` e, tanto quanto foi possível apurar, esta abordagem é inédita.

O tempo computacional para geração dos *clusters* é apresentado na Tabela 6.1, e, para questões das avaliações de custo computacional, o *hardware* utilizado foi um servidor com 60x Intel Xeon E5-2698 v4 @ 2.2GHz. Para os modelos neurais, LightGBM

e *k-Shape*, que possuem implementações que se aproveitam do uso de GPU, foi utilizada 1x NVIDIA Tesla V100-SXM2-32GB. Na implementação dos modelos estatísticos, foi empregado o *multiprocessing*, tirando assim melhor proveito do *hardware* disponível.

Tabela 6.1: Tempo Computacional

Loja	Modelo	Tempo Computacional (s)
40	kshape	2148
42	kshape	998
138	kshape	1737
171	kshape	1167
40	timesmash	28
42	timesmash	8
138	timesmash	22
171	timesmash	14
40	som	5414
42	som	374
138	som	7673
171	som	1763

Uma análise da Tabela 6.1 revela as assinaturas de custo distintas de cada metodologia de clusterização. O *TimeSmash* é o mais eficiente, com tempos de execução na ordem de segundos, pois sua complexidade reside na inferência de um conjunto fixo de modelos PFSA e na subsequente projeção das séries em um espaço de *log-likelihood*. Em contraste, o *k-Shape* apresenta um custo significativamente maior, pois requer cálculos iterativos da SBD (uma medida baseada em correlação cruzada) sobre o comprimento total das séries. A abordagem proposta com SOM apresenta, deliberadamente, o maior custo computacional, um artefato direto das 30 iterações de otimização Bayesiana necessárias para ajustar os hiperparâmetros da rede e organizar o mapa topológico. Este custo inicial, embora elevado, deve ser interpretado como um investimento de capital metodológico, cuja viabilidade é avaliada em função dos ganhos de eficiência obtidos no *pipeline* de seleção recorrente.

Desempenho da seleção pelo método de janela reduzida

Em uma primeira abordagem, foram testadas três janelas reduzidas para o treino dos modelos: 30, 60 e 120 dias. Para cada uma dessas janelas, ao final do treino dos modelos, foram apuradas as métricas de erro. A seleção dos três melhores modelos por série e a seleção dos modelos que fazem parte do *pool* do *cluster* levaram em consideração sua métrica correspondente. Dessa forma, o mesmo *cluster* pode possuir um *pool* diferente de modelos a depender da métrica que se busca minimizar.

O desempenho de custo computacional total do *pipeline* de "janela reduzida" é apresentado de forma representativa na Tabela 6.2 para a Loja 40. O tempo total reportado

agrega o custo da clusterização (Tabela 6.1), o custo do treinamento de seleção (todos os modelos nas janelas reduzidas) e o custo do treinamento final (apenas os modelos selecionados no histórico completo). Uma observação importante é que as arquiteturas de seleção de modelos propostas alcançaram ganhos computacionais expressivos em relação ao método Empírico. O modelo Empírico é obtido por meio do melhor modelo dentre os existentes. Na Loja 40 (Tabela 6.2), por exemplo, o *benchmark* Empírico (treinar todos os 10 modelos) exigiu 21.264 segundos. A abordagem som_30, mesmo incluindo o custo elevado de clusterização (5.414s), completou o *pipeline* em 8.713 segundos, uma redução de 59% no custo computacional total.

A análise aprofundada do *trade-off* financeiro revela a estratégia de amortização. O custo da clusterização (ex: 5.414s para o SOM na Loja 40) é um custo fixo, pago uma única vez para mapear as séries. O custo da seleção (ex: $8.713s - 5.414s = 3.299s$) é o custo recorrente, executado a cada ciclo de retreinamento para atualizar a escolha do "melhor" modelo. Este custo recorrente de $\approx 3.300s$ é substancialmente inferior ao custo do *benchmark* Empírico (21.264s). Portanto, a metodologia proposta demonstra que um investimento computacional inicial na clusterização desbloqueia uma redução de custo operacional recorrente superior a 80% (3.300s vs 21.264s), validando a viabilidade econômica da arquitetura. Os resultados detalhados para as lojas 42, 138 e 171, que corroboram este mesmo padrão de ganho computacional, encontram-se disponíveis no Apêndice A, Tabelas A.1 a A.3.

A Tabela 6.3 detalha o resultado do processo de seleção de modelos para a Loja 40. A análise revela um padrão bipolar claro: a seleção é massivamente dominada pelo LightGBM e pelo *Naive*, modelos que representam extremos opostos de complexidade e custo computacional. Este achado por si só valida a necessidade da arquitetura de AutoML, pois demonstra que uma abordagem de modelo único (como usar apenas o LightGBM) falharia em capturar o desempenho do *Naive*, especialmente para a métrica SMAPE, em que o *Naive* apresenta forte participação (ex: 693 de 3245 seleções no *k-Shape*-60).

Uma segunda observação, crucial para a validação da eficiência do *pipeline*, é a estabilidade da seleção em relação ao tamanho da janela. O perfil de dominância (a divisão entre LightGBM e *Naive*) permanece notavelmente estável entre as janelas de 30, 60 e 120 dias. Por exemplo, na Tabela 6.3 (*k-Shape*, SMAPE), a seleção do LightGBM e do *Naive* é de (2415, 633) para 30 dias e (2415, 631) para 120 dias, uma distribuição quase idêntica. Isso é uma evidência robusta de que a janela de 30 dias, embora computacionalmente mais barata, é suficiente para capturar as informações necessárias para a seleção de modelos, validando o uso da janela mais curta para otimizar o custo de seleção recorrente. Este padrão bipolar e a estabilidade da janela mantêm-se consistentes, conforme detalhado nas tabelas das demais lojas, disponíveis no Apêndice A (Tabelas A.4 a A.6).

Tabela 6.2: Custo computacional dos modelos, loja 40.

Modelo	**Custo Computacional (s)**
AutoArima	1505
Croston	608
DeepAR	7268
ExponentialSmoothing	118
LightGBM	877
Naive	67
Prophet	3713
RNN	5808
TSB	789
Theta	512
Empírico	21264

Modelo	**Custo Computacional (s)**		
	MASE	**RMSSE**	**SMAPE**
kshape_30	5423	5975	5386
kshape_60	7901	9123	7882
kshape_120	14932	16163	14979
som_30	8713	9231	8675
som_60	11308	12213	11194
som_120	18570	19498	18235
timesmash_30	3334	3830	3292
timesmash_60	5865	6887	5846
timesmash_120	13240	13992	12828

Desempenho da seleção pelo método de conjunto reduzido

A segunda abordagem de seleção, o método de "conjunto reduzido", parte de uma premissa de meta-aprendizagem (conforme Seção 5.5): assume-se que a etapa de clusterização foi eficaz em agrupar séries com dinâmicas homogêneas. Sob esta hipótese, uma pequena amostra percentual de séries de um *cluster* pode atuar como um *proxy* estatisticamente robusto para todo o grupo, permitindo que o modelo ótimo, determinado sobre essa amostra, seja generalizado para todas as séries do *cluster*. O objetivo desta análise é, portanto, duplo: determinar a sensibilidade da seleção do modelo ao tamanho dessa amostra; e validar a hipótese fundamental de que a clusterização SOM com `LikelihoodDistance` de fato agrupa os modelos de forma topológica e estatisticamente significativa.

Para avaliar a sensibilidade ao tamanho da amostra, foi conduzido um experimento incrementando o percentual de séries amostradas. A Figura 6.6 apresenta a evolução do erro SMAPE agregado (eixo Y) conforme o percentual de amostragem (eixo X) aumenta. Uma análise do painel da Loja 40 revela um achado central: a seleção do modelo ótimo é notavelmente robusta mesmo com percentuais de amostragem extremamente baixos. Para as três metodologias de clusterização (*k-Shape*, *TimeSmash* e SOM),

Tabela 6.3: Tabela de dominância dos modelos nos clusters, loja 40.

Modelo	Janela	Métrica	AA	CR	ES	LGB	NA	PR	TSB	TT
kshape	30	MASE	2			3140	75		28	
kshape	30	RMSSE		30	1	2925			289	
kshape	30	SMAPE	2	191	4	2415	633			
kshape	60	MASE				3150	86		6	3
kshape	60	RMSSE	66	21		3146			12	
kshape	60	SMAPE		74	2	2475	693		1	
kshape	120	MASE	1			3152	78			14
kshape	120	RMSSE	66	21		3148			10	
kshape	120	SMAPE	1	196	2	2415	631			
som	30	MASE		6		3153	78		8	
som	30	RMSSE	7	41		2970			227	
som	30	SMAPE		219		2384	642			
som	60	MASE		2		3156	82	1	4	
som	60	RMSSE	46	51		3125			23	
som	60	SMAPE		135		2437	673			
som	120	MASE				3157	75	4		9
som	120	RMSSE	51	12	3	3153		9	17	
som	120	SMAPE		144		2431	670			
timesmash	30	MASE				3155	78		12	
timesmash	30	RMSSE		31		2929			285	
timesmash	30	SMAPE		235		2379	631			
timesmash	60	MASE		1		3159	78		7	
timesmash	60	RMSSE	53	27		3157			8	
timesmash	60	SMAPE		235		2379	631			
timesmash	120	MASE				3151	73	4		17
timesmash	120	RMSSE	62	13		3170				
timesmash	120	SMAPE		95		2456	694			

Legenda: AA=AutoArima, CR=Croston, ES=Exponential Smoothing, LGB=LightGBM, NA=Naive, PR=Prophet, TT=Theta.

o SMAPE agregado atinge um platô quase imediato. O erro obtido com apenas 5% de amostragem é virtualmente indistinguível daquele obtido com 100%. Isso sugere que a dominância de um modelo específico dentro de cada *cluster* (conforme Tabela 6.3) é tão pronunciada que é capturada rapidamente, e amostras adicionais apenas reconfirmam esta escolha. Este resultado valida a abordagem como computacionalmente viável, pois um percentual mínimo de amostragem é suficiente para uma seleção estável. Para as métricas RMSSE e MASE, que demonstram uma convergência similar, os resultados encontram-se no Apêndice B nas Figuras B.1 e B.2.

Validada a eficiência da amostragem, passa-se à validação da própria hipótese de clusterização, que é a contribuição inédita desta tese. A questão fundamental é se a abordagem SOM combinada com a métrica `LikelihoodDistance` é capaz de separar espacialmente os modelos ótimos. A Figura 6.7 ilustra o mapa de SOM treinado para a

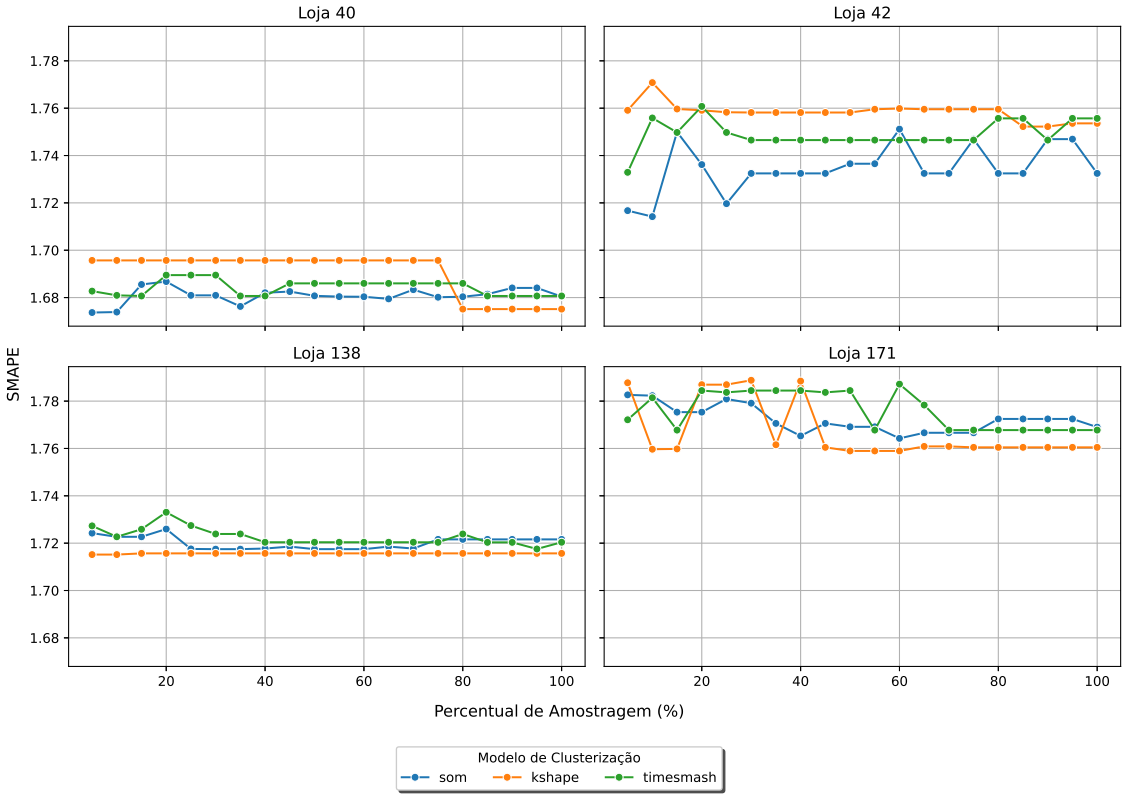


Figura 6.6: Evolução do SMAPE pelo percentual de amostragem

Loja 40, em que cada neurônio é colorido de acordo com o modelo de melhor desempenho (o *benchmark* Empírico) para as séries ali alocadas.

Focando na métrica SMAPE (painel superior esquerdo da Figura 6.7), a hipótese é visualmente confirmada. O mapa não apresenta uma distribuição aleatória de modelos, mas sim uma clara concentração topológica. Notavelmente, os dois modelos dominantes (conforme Tabela 6.3), *Naive* e LightGBM, segregam-se em regiões distintas. O LightGBM concentra-se na porção inferior direita do mapa, enquanto o *Naive* ocupa majoritariamente a região oposta (superior esquerda). Isso evidencia que a métrica `LikelihoodDistance` (uma abordagem *model-based*) agrupou com sucesso séries que compartilham características estocásticas que as tornam mais adequadas a um modelo simples ou a um modelo complexo. Os mapas para as demais lojas (42, 138 e 171), que exibem separação topológica similar, estão disponíveis no Apêndice C nas Figuras C.1 a C.3.

Enquanto a Figura 6.7 fornece uma confirmação visual da segregação topológica, a análise dos centróides é necessária para quantificar objetivamente a tendência central dessa separação. A Figura 6.8 executa esta quantificação ao calcular e plotar a localização média (o "centro de massa") de todas as séries associadas a cada modelo vencedor no mapa treinado. Este passo é metodologicamente fundamental por duas razões: primeiro,

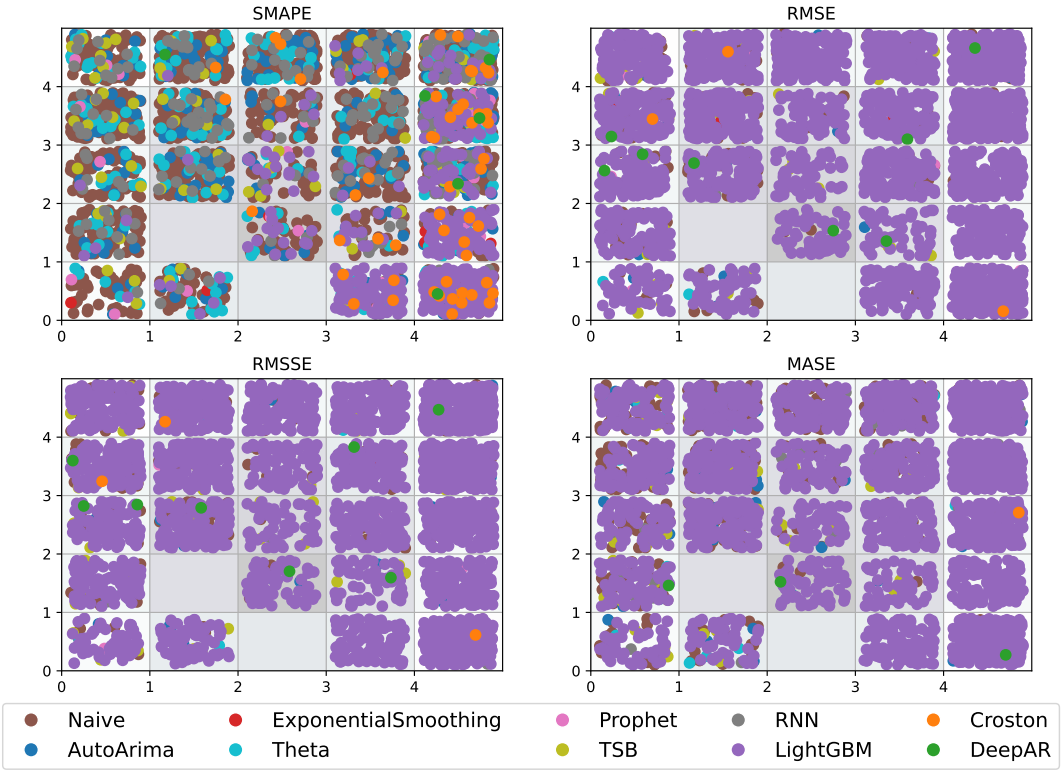


Figura 6.7: Mapa SOM para loja 40

ele transforma a observação visual subjetiva em uma medida objetiva da distância entre os grupos de modelos; segundo, ele serve como a base quantitativa para o teste estatístico subsequente (PERMANOVA), que avalia formalmente se a distância entre os centróides dos grupos é estatisticamente maior do que a variação dentro de cada grupo. A análise do painel SMAPE (superior esquerdo) confirma a eficácia da separação: a distância entre os centróides dos modelos dominantes é maximizada, com o centróide do LightGBM e o do *Naive* situando-se em quadrantes opostos do mapa. Isso demonstra que a clusterização SOM, quando combinada com a métrica `LikelihoodDistance`, é de fato eficiente em particionar o espaço de problemas. Os gráficos de centróides para as demais lojas estão no Apêndice C nas Figuras C.1 a C.3.

Para formalizar esta observação visual, a separação foi testada estatisticamente. Foi aplicado o teste PERMANOVA (Análise de Variância Multivariada Permutacional), um teste não paramétrico que opera sobre matrizes de distância e não assume normalidade multivariada, para cada loja e métrica. Em todos os cenários avaliados, foram obtidos *p-values* inferiores a 0,05, levando à rejeição da hipótese nula. Isso confirma que a separação dos modelos no mapa SOM não é um artefato aleatório, mas sim uma diferença estatisticamente significativa na localização dos produtos, com base no seu melhor modelo preditivo.

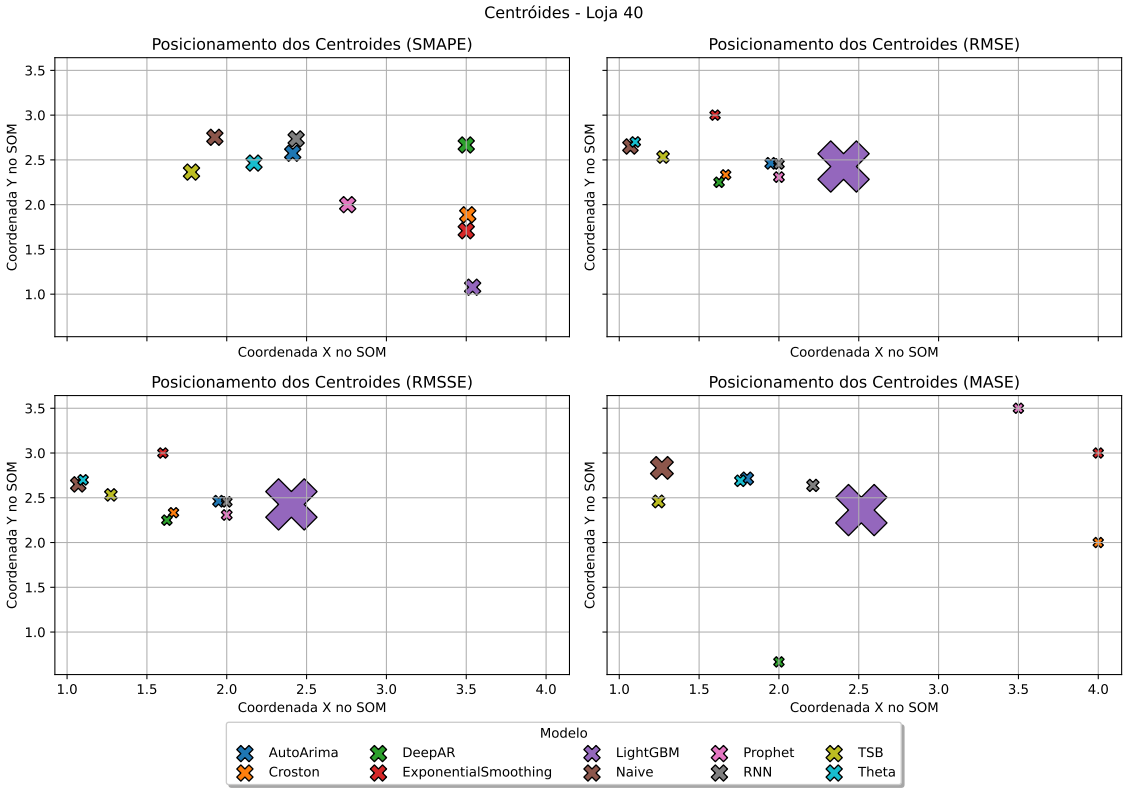


Figura 6.8: Mapa SOM para loja 40

6.1.2 Análise dos Erros

Esta seção consolida a avaliação de desempenho de todas as metodologias, comparando os 10 modelos clássicos (descritos no Capítulo 2) com as arquiteturas de seleção de modelos baseadas em clusterização (detalhadas na Seção 5.5). O modelo "Empírico" é introduzido como o *benchmark* de acurácia, representando o limite inferior teórico do erro, pois é constituído pelo resultado do melhor modelo para cada produto individual, calculado após o treinamento de todos os 10 modelos em todo o conjunto de dados.

A Tabela 6.4 apresenta os resultados agregados de desempenho para a Loja 40, servindo como nosso estudo de caso principal. Uma análise dos modelos clássicos revela um padrão bipolar. Para as métricas escaladas RMSSE e MASE, o LightGBM apresenta uma dominância clara, alcançando (0.4472) e (0.5655) respectivamente. Este resultado é esperado e corrobora os achados da competição M5, na qual o LightGBM demonstrou superioridade sobre outras classes de modelos para dados de varejo, devido à sua capacidade de lidar com a alta dimensionalidade da engenharia de *features* e sua robustez a irregularidades.

Em contrapartida, na métrica SMAPE, o modelo *Naive* apresenta o melhor desempenho (0.7758) entre os modelos clássicos, superando o LightGBM (1.5287) por uma

margem substancial. Esta anomalia não deve ser interpretada como uma superioridade generalizada do *Naive*, mas sim como um artefato metodológico conhecido da métrica SMAPE quando aplicada a dados de demanda intermitente, que predominam neste *dataset* (conforme Figura 6.1). Quando o valor real Y_i é zero (um evento frequente), o *Naive* (que prevê o último valor, frequentemente zero) alcança um erro de 0%, enquanto um modelo mais sofisticado que prevê um valor pequeno ϵ é desproporcionalmente penalizado.

Ainda na Tabela 6.4, a análise do desempenho das arquiteturas de clusterização propostas (*k-Shape*, *TimeSmash*, SOM) revela a principal conclusão desta tese. Focando na métrica RMSSE, o *benchmark* Empírico (limite teórico) é 0.4435 e o melhor modelo único (LightGBM) alcança 0.4472. As arquiteturas de seleção, como a timesmash_120 (0.4522) e a som_120 (0.4532), atingem um nível de erro quase idêntico ao melhor modelo. Elas sacrificam uma quantidade marginal de acurácia (aproximadamente 1.3% de perda em relação ao LightGBM) para obter uma economia massiva no custo computacional recorrente (conforme analisado na Tabela 6.2). Esta é a validação do *trade-off* central da arquitetura de AutoML. As tabelas de erro detalhadas para as lojas 42, 138 e 171, que confirmam esta mesma dinâmica, são apresentadas no Apêndice D.

Tabela 6.4: Métricas médias por modelo na loja 40.

Model	SMAPE	RMSSE	MASE
Empírico	0.6093	0.4435	0.5555
AutoArima	1.4758	0.8452	1.2725
Croston	1.5366	0.9437	1.7994
DeepAR	1.6940	2.6038	3.0633
ExponentialSmoothing	1.6294	0.7153	1.0799
LightGBM	1.5287	**0.4472**	**0.5655**
Naive	**0.7758**	0.9273	1.3057
Prophet	1.6155	0.7244	1.0877
RNN	1.4273	1.4209	1.6423
TSB	1.6097	0.7179	1.1126
Theta	**1.3324**	0.7126	0.8564
kshape_30	1.3998	0.4707	0.5880
kshape_60	1.3951	0.4540	0.5889
kshape_120	1.4002	0.4537	0.5862
timesmash_30	1.3996	0.4708	0.5878
timesmash_60	1.3996	0.4532	0.5873
timesmash_120	1.3954	**0.4522**	0.5811
som_30	1.3979	0.4687	0.5876
som_60	1.3966	0.4561	0.5878
som_120	1.3982	0.4532	**0.5804**

Obs.: **Menor valor**, **Segundo menor valor**

Na Figura 6.9 ilustra-se o *trade-off* para a metodologia de "conjunto reduzido"

(amostragem). O eixo Y representa o ganho de tempo computacional (maior é melhor) e o eixo X o erro relativo ao *benchmark* (próximo de 100% é melhor). A região ideal é o canto superior esquerdo. Observa-se que a clusterização *k-Shape* oferece cenários de baixo risco, com acurácia próxima a 100%, mas ganhos de tempo discretos. O *TimeSmash* apresenta um cenário de alto ganho de tempo (superior a 17.5s), mas com uma penalidade de acurácia (erro 8% maior). A metodologia proposta, SOM, demonstra o balanço mais estratégico: alcança ganhos de tempo computacional significativos (entre 7.5s e 12.5s) enquanto mantém a acurácia em níveis competitivos.

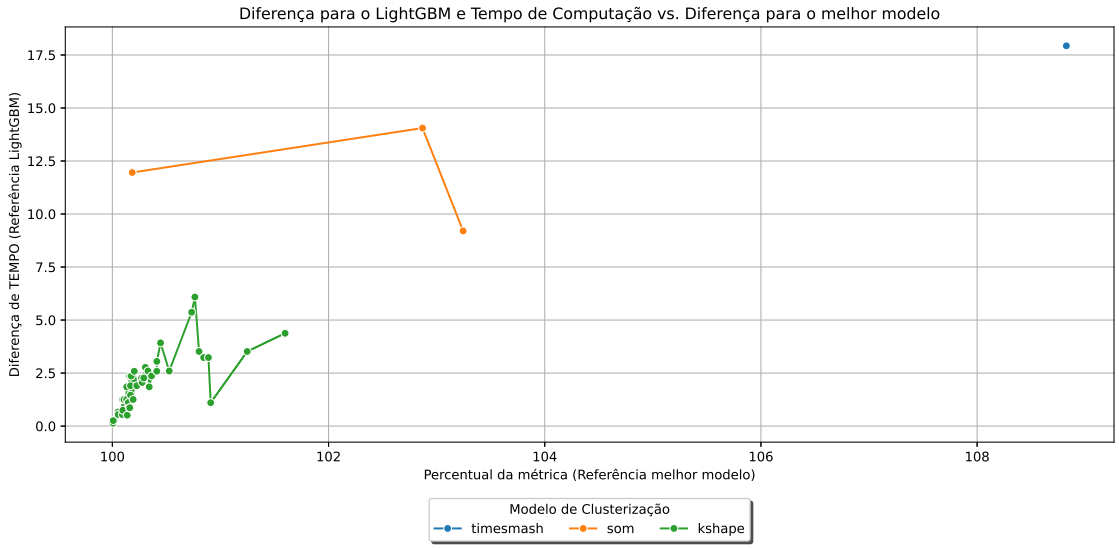


Figura 6.9: *Trade-off* entre ganho de custo computacional (segundos) e desempenho relativo da métrica (vs. melhor modelo) para as arquiteturas de seleção por amostragem.

Para aprofundar a análise de quais modelos são selecionados, a Figura 6.10 detalha a dominância do modelo vencedor (Empírico) para cada faixa de volume de vendas. Conforme esperado, a análise do SMAPE (painel superior direito) mostra uma clara diversificação: o *Naive* domina as faixas de baixo volume (intermitentes), enquanto o LightGBM prevalece em produtos com vendas mais elevadas. Nas demais métricas, como RMSSE e MASE, a soberania do LightGBM é quase total. No entanto, é crucial notar a presença de modelos simples (*Naive*) e especializados em intermitência (TSB, Croston) nas faixas de vendas mais baixas, reforçando a necessidade de um *pool* de modelos heterogêneo.

A Figura 6.11 oferece outra perspectiva sobre os mesmos dados, mostrando a distribuição do erro de cada modelo por faixa de vendas. Esta visualização confirma que modelos como Prophet e LightGBM são mais adequados para volumes de vendas elevados, enquanto o DeepAR parece adaptar-se melhor a faixas intermediárias. A uniformi-

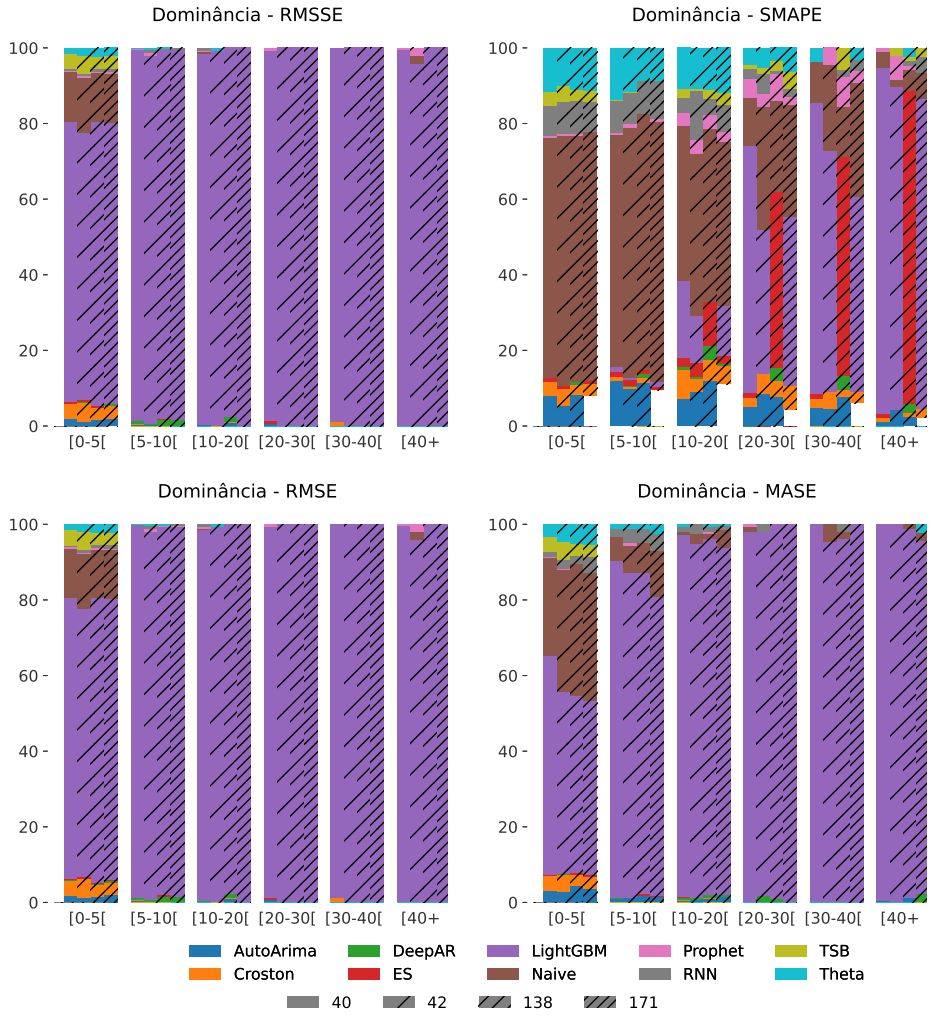


Figura 6.10: Dominância dos modelos por quantidade vendida

dade do comportamento entre as lojas (indicada pelas cores) sugere que os padrões de desempenho dos modelos são consistentes e não específicos de um estabelecimento.

Finalmente, na Tabela 6.5 quantifica-se o RMSE por modelo para cada faixa de vendas na Loja 40. Esta tabela contém a justificação econômica final para a arquitetura de AutoML proposta. Observa-se que na faixa $[0-5[$, que, conforme a Figura 6.1, contém a maioria absoluta dos produtos (1888 séries) e, portanto, representa a maior parte do custo computacional, o LightGBM é de fato o melhor modelo (RMSE 0.307). Contudo, modelos computacionalmente muito mais simples, como o Theta (0.480) e o *Exponential Smoothing* (ES) (0.517), apresentam desempenhos que, embora piores, estão na mesma ordem de magnitude.

Esta observação é a chave: para a maioria dos produtos, a arquitetura de AutoML

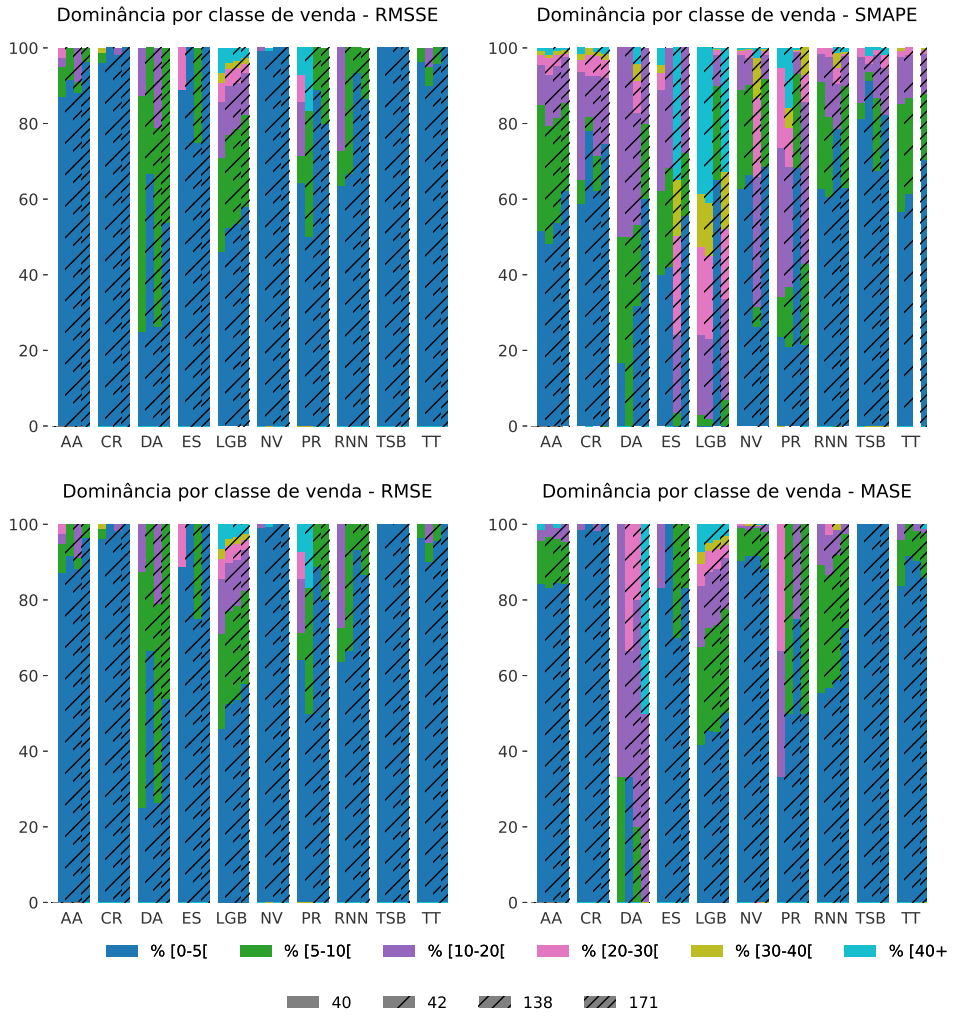


Figura 6.11: Distribuição do desempenho (contagem de seleção) dos modelos por faixa de vendas. Legenda: AA=AutoArima, CR=Croston, ES=Exponential Smoothing, LGB=LightGBM, DA=DeepAR, NA=Naive, PR=Prophet, TT=Theta.

pode optar por um modelo substancialmente mais barato (como o Theta) sofrendo uma penalidade de erro controlada (0.173 de RMSE). Esta troca estratégica de acurácia marginal por ganho computacional substancial na faixa de maior volume de produtos é o que permite que as metodologias de cluster (como som_120) alcancem um RMSSE agregado (0.453) tão competitivo quanto o LightGBM (0.447), validando a hipótese de que a seleção de modelos baseada em clusterização é uma alternativa viável para otimizar o balanço entre acurácia e desempenho. As tabelas de RMSE por faixa para as demais lojas são apresentadas no Apêndice D nas Tabelas D.4 a D.6.

Tabela 6.5: RMSE por faixa, por modelo na loja 40

faixa Model	[0-5[	[5-10[	[10-20[	[20-30[	[30-40[	[40+
AutoArima	0.773806	1.274036	2.603633	2.480542	3.085302	9.173315
Croston	1.078587	1.393861	1.858416	2.639520	3.059579	7.427489
DeepAR	8.250297	6.490900	11.810432	3.229363	14.749824	13.068140
ES	0.517911	**0.903452**	**1.454382**	**2.106175**	2.692085	8.154189
LightGBM	**0.307286**	**0.590160**	**0.936163**	**1.309801**	**1.606195**	**3.918697**
Naive	0.886648	1.460679	2.374813	2.510263	3.523664	9.621569
Prophet	0.552999	0.916693	1.473684	2.132902	2.670894	9.129718
RNN	1.669623	1.768506	2.120211	2.663938	3.584689	10.557942
TSB	0.518367	0.933345	1.489638	2.224716	**2.647840**	**6.609116**
Theta	**0.480121**	0.925836	1.484127	2.220463	2.776811	7.737842

Obs.: **Menor valor**, **Segundo menor valor**

6.2 Resultados de classificação de demanda

Uma análise final do desempenho dos modelos é realizada por meio da segmentação pela classificação de demanda (cuja metodologia foi detalhada na Seção 3.1), uma taxonomia padrão para dados de varejo. A Figura 6.12 ilustra a distribuição dos produtos nas quatro classes (*Intermittent*, *Lumpy*, *Erratic* e *Smooth*). Confirma-se que o *dataset* é massivamente dominado pela classe *Intermittent*, o perfil de demanda mais desafiador, que representa a maior fração de produtos em todas as lojas estudadas.

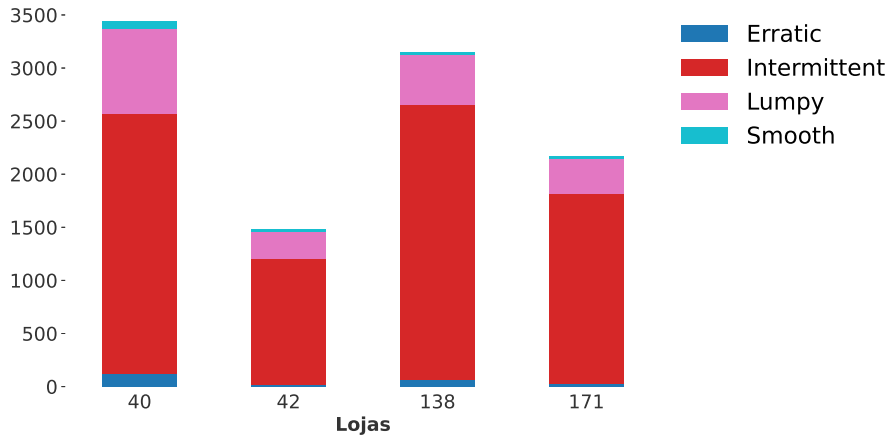


Figura 6.12: Quantidade de produtos por classificação de demanda

A Figura 6.13 detalha qual modelo (Empírico) obteve o melhor desempenho, agregado por classe de demanda e métrica. Esta visualização revela uma dicotomia clara, dependente da métrica de avaliação. Para as métricas escaladas (RMSSE e MASE) e para o RMSE, o LightGBM apresenta soberania quase total, vencendo em todas as quatro classes. Contudo, para a métrica SMAPE, o resultado inverte-se: o modelo *Naive* domina

as classes de alta intermitência (*Intermittent* e *Lumpy*), enquanto o LightGBM só assume a liderança nas classes de demanda mais regular (*Erratic* e *Smooth*). Isso reforça a análise de que o forte desempenho do *Naive* está intrinsecamente ligado aos artefatos da métrica SMAPE quando aplicada a dados esparsos.

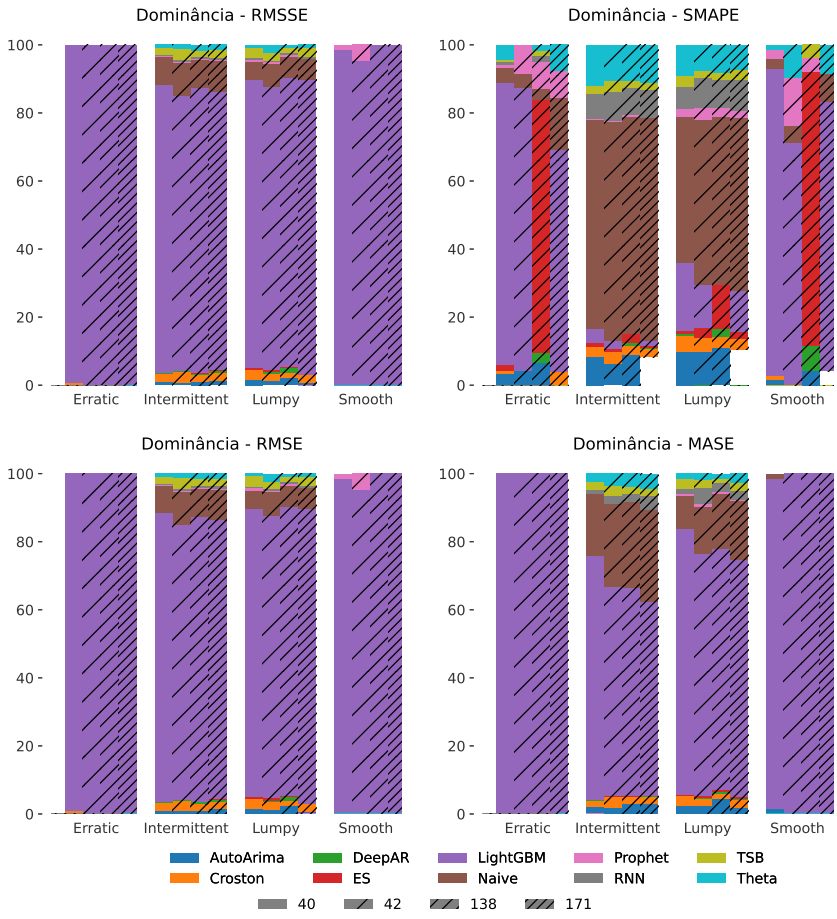


Figura 6.13: Dominância dos modelos por classificação de demanda

A Figura 6.13 demonstra a *seleção* do melhor modelo, mas não a magnitude da diferença de erro. Esta análise de custo-benefício, que é central para a tese, é fornecida pela Tabela 6.6, que detalha o RMSE de todos os modelos por classe de demanda para a Loja 40. Conforme a figura anterior, o LightGBM de fato apresenta o menor erro em todas as quatro classes. As tabelas que confirmam esta dominância do LightGBM nas demais lojas (42, 138 e 171) estão disponíveis no Apêndice D.

A análise aprofundada da Tabela 6.6 revela a principal justificativa econômica para a arquitetura de AutoML baseada em clusterização. Embora o LightGBM seja o vencedor em todas as classes, a margem de sua vitória difere drasticamente. Nas classes de demanda regular (*Erratic* e *Smooth*), o LightGBM é substancialmente superior; na

Tabela 6.6: RMSE por categoria de demanda, por modelo na loja 40

categoria Model	Intermittent	Lumpy	Erratic	Smooth
AutoArima	1.134241	3.203166	8.641169	6.370279
Croston	1.281531	3.087100	5.871641	**4.724880**
DeepAR	8.367758	8.528678	9.474228	15.876885
ES	**0.843185**	**2.304309**	8.057170	5.739392
LightGBM	**0.527104**	**1.488412**	**3.081885**	**2.989344**
Naive	1.233381	3.107259	9.275547	6.981564
Prophet	0.851181	2.375071	9.433068	6.100582
RNN	1.859536	3.020474	9.497721	9.500885
TSB	0.868964	2.379188	**5.637801**	4.774093
Theta	0.867536	2.367601	7.339498	5.491045

Obs.: **Menor valor**, **Segundo menor valor**

classe *Erratic*, seu RMSE (3.081) é quase 83% melhor que o segundo colocado, TSB (5.637). Em contrapartida, na classe *Intermittent*, que a Figura 6.12 mostra ser a classe mais populosa e, portanto, a mais cara computacionalmente, a diferença entre os melhores modelos é muito menor. O LightGBM (0.527) supera o ES (0.843) e o Theta (0.867), mas a diferença absoluta de erro é pequena.

Esta observação é a chave para o *trade-off* de custo-benefício: a arquitetura de AutoML pode, para a maioria dos produtos (os intermitentes), selecionar um modelo computacionalmente mais simples (como ES ou Theta) em troca de uma penalidade de acurácia marginal e controlada. A arquitetura pode, então, alocar o custoso LightGBM apenas para as classes de demanda regular, nas quais ele é indiscutivelmente superior, otimizando assim o balanço entre acurácia e custo computacional que é o objetivo desta tese.

CAPÍTULO 7

Conclusões

Neste trabalho foram apresentados os principais conceitos para predição de séries temporais, incluindo os modelos clássicos bem como o estado da arte. Abordou-se a característica particular para as séries temporais do ramo varejista, composta de milhares de séries, sendo a maioria com elevada participação de intermitência. Com base neste pressuposto, introduziu-se a clusterização como uma possível ferramenta viável de insumo necessária para seleção de modelos, em especial, para predições de curto prazo e para empresas cujos investimentos em predição podem ser impeditivos.

A tese foi estruturada para construir a argumentação de forma progressiva, recapitulando a jornada da pesquisa.

No Capítulo 1, apresentou-se o dilema central da previsão no varejo: a necessidade de acurácia em um cenário de alta intermitência e o custo computacional proibitivo das soluções de MLOps.

No Capítulo 2, revisou-se o *pool* de modelos preditivos que fundamentam a análise, desde os estatísticos até as arquiteturas de *Deep Learning*.

No Capítulo 3, foram investigadas as metodologias de agrupamento, introduzindo os paradigmas *shape-based* e *model-based* e detalhando a métrica `LikelihoodDistance` e o algoritmo SOM.

No Capítulo 4, formalizou-se o contexto do problema, definindo o AutoML e o desafio CASH, e justificando a necessidade de arquiteturas especializadas para mitigar o custo computacional da seleção de modelos em larga escala.

Respondendo a este desafio, no Capítulo 5 detalhou-se a arquitetura de AutoML proposta, incluindo a contribuição inédita da hibridização do SOM com a métrica `LikelihoodDistance` e as estratégias de seleção de modelos por amostragem.

No Capítulo 6, consolidou-se a validação empírica da proposta. Demonstrou-se que as arquiteturas de seleção de modelos alcançaram um balanço estratégico, obtendo uma acurácia preditiva (medida por RMSSE) estatisticamente comparável ao *benchmark*, mas com uma redução drástica no custo computacional recorrente. A análise topológica dos mapas SOM e os testes PERMANOVA validaram a hipótese de que a métrica *model-based* efetivamente segrega séries temporais por seu modelo preditivo ótimo.

A proposta de AutoML com seleção de modelos mediante o uso de clusterização demonstrou competitividade e consistência. Na abordagem de janelas reduzidas, as métricas encontradas ficaram muito próximas do melhor modelo, porém sem a necessidade de treinar todos os modelos na janela completa. A abordagem por amostragem mostrou-se mais vantajosa, uma vez que com baixos valores de amostras já é possível identificar os melhores modelos e ainda assim obter ganhos computacionais.

Os resultados obtidos revelam uma forte correlação entre a natureza do processo gerador de uma série temporal e a escolha do modelo de previsão mais adequado. A utilização da métrica `LikelihoodDistance` em conjunto com o SOM permitiu particionar o conjunto de séries de forma estatisticamente robusta. Tal evidência sugere que a similaridade estrutural entre os modelos geradores, como os Autômatos Finitos Probabilísticos (PFSA), é o fator determinante para que um mesmo algoritmo de previsão alcance a maior acurácia para um conjunto de séries temporais.

7.1 Trabalhos Futuros

Os resultados desta tese, embora robustos, abrem diversas avenidas para investigações futuras, tanto na validação da arquitetura quanto na expansão de seus componentes metodológicos.

Uma primeira extensão natural seria a validação da generalibilidade da arquitetura em outros *datasets* públicos de larga escala. A aplicação da metodologia em domínios de varejo distintos, como os explorados nas competições M5 ou VN1, ou em *benchmarks* de séries temporais canônicos, permitiria verificar se a robustez da seleção de modelos *model-based* se mantém em diferentes condições de intermitência e sazonalidade. Adicionalmente, a aplicação em domínios radicalmente distintos do varejo poderia revelar novos *insights*; a abordagem focada no processo gerador é teoricamente promissora para o setor financeiro (para clusterização de regimes de volatilidade), dados de sensores de IoT (para identificação de estados operacionais) ou em séries temporais biomédicas, em que a "forma" da série é menos importante que sua dinâmica estocástica subjacente.

Uma segunda vertente, de grande relevância prática para MLOps, seria o desenvolvimento de um ferramental de acompanhamento da deterioração dos modelos. Esta tese propôs um *pipeline* que amortiza o custo de clusterização, mas assume que os *clusters* (baseados no processo gerador) são estáticos. Uma investigação futura deveria focar em métricas para detectar o *concept drift*, como os propostos em [100] ou [62], não apenas no erro de previsão, mas na própria estabilidade dos *clusters* SOM. Isso responderia à questão operacional crítica: "quando o mapa topológico se tornou obsoleto e precisa ser retreinado?".

No âmbito da própria metodologia de clusterização, esta tese utilizou a `LikelihoodDistance` como um método de extração de características (um *embedding* de *log-likelihood*) para algoritmos que esperam entradas vetoriais (K-means e SOM). Uma investigação futura poderia explorar seu uso como uma matriz de distância $N \times N$ direta em algoritmos particionais que aceitam matrizes de dissimilaridade arbitrárias, como o K-Medoids. Além disso, a substituição do PFSA por outros modelos geradores, como Modelos Ocultos de Markov (HMMs), poderia ser investigada como base para a métrica de divergência de verossimilhança [7].

Finalmente, otimizações na própria arquitetura do SOM poderiam ser exploradas. Esta tese utilizou a topologia retangular padrão. A adoção de uma topologia hexagonal, que proporciona uma distância de vizinhança mais uniforme (seis vizinhos equidistantes em vez de quatro vizinhos e quatro diagonais), poderia resultar em um mapa topológico mais suave e uma separação de centróides (Figura 6.8) potencialmente mais clara e definida. Adicionalmente, uma integração mais profunda com o *pipeline* de AutoML poderia tratar o número de *clusters* k não como um hiperparâmetro fixo, mas como uma variável a ser otimizada dentro do mesmo processo de HPO usado para os modelos preditivos, buscando um k ótimo que minimize o erro de previsão final.

Referências Bibliográficas

[1] **DataSource.ai - Data Science Competitions For Startups & SMBs — datasource.ai**. [Accessed 04-08-2025].

[2] ALIJANI, G.; FULK, H.; OMAR, A.; TULSI, R. **Cloud computing effects on small business**. *The Entrepreneurial Executive*, 19:35–45, 01 2014.

[3] ALSHAREF, A.; AGGARWAL, K.; SONIA.; KUMAR, M.; MISHRA, A. **Review of ml and automl solutions to forecast time-series data**. *Archives of Computational Methods in Engineering*, 29(7):5297–5311, 2022.

[4] ASSIMAKOPOULOS, V.; NIKOLOPOULOS, K. **The theta model: a decomposition approach to forecasting**. *International Journal of Forecasting*, 16(4):521–530, oct 2000.

[5] AUTHORS, T. G. **GPyOpt: A bayesian optimization framework in python**. http://github.com/SheffieldML/GPyOpt, 2016.

[6] AWS PLAIN ENGLISH. **Model drift is inevitable. here's how to catch it before it costs you**. https://aws.plainenglish.io/model-drift-is-inevitable-heres-how-to-catch-it-before-it-costs-you-397ee37 2023. Acessado em: 05-08-2025.

[7] BARRETO, G. A. **Time Series Prediction with the Self-Organizing Map: A Review**, p. 135–158. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[8] BARTLETT, M. S. **Properties of sufficiency and statistical tests**. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 160(901):268–282, 1937.

[9] BAUM, C. F. **Topics in time series**. Apresentação de slides, Boston College, 2009. Disponível em: http://fmwww.bc.edu/cfb/stata/TStalkJan2009.beamer.pdf.

[10] BEITNER, J. **Introducing pytorch forecasting**. *Towards Data Science*, 2020.

[11] BENVENUTO, D.; GIOVANETTI, M.; VASSALLO, L.; ANGELETTI, S.; CICCOZZI, M. **Application of the arima model on the covid-2019 epidemic dataset**. *Data in Brief*, 29:105340, 2020.

[12] BERGSTRA, J.; BARDENET, R.; BENGIO, Y.; KÉGL, B. **Algorithms for hyper-parameter optimization**. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'11, p. 2546–2554, Red Hook, NY, USA, 2011. Curran Associates Inc.

[13] BERGSTRA, J.; YAMINS, D.; COX, D. **Making a science of model search: Hyper-parameter optimization in hundreds of dimensions for vision architectures**. In: Dasgupta, S.; McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 de **Proceedings of Machine Learning Research**, p. 115–123, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[14] BOSE, D. C. **Inventory management**. PHI Learning Pvt. Ltd., 2006.

[15] BOTTER, R.; FORTUIN, L. **Stocking strategy for service parts - a case study**. *International Journal of Operations & Production Management*, 20(6):656–674, Jan. 2000.

[16] BOX, G.; JENKINS, G. **Time Series Analysis: Forecasting and Control**. Holden-Day series in time series analysis and digital processing. Holden-Day, 1970.

[17] BREIMAN, L. **Random forests**. *Machine Learning*, 45(1):5–32, 2001.

[18] BROWN, ROBERT GOODELL, . **Statistical forecasting for inventory control** / **Robert G. Brown**. McGraw-Hill New York, 1959.

[19] CANOVA, F.; HANSEN, B. E. **Are seasonal patterns constant over time? a test for seasonal stability**. *Journal of Business & Economic Statistics*, 13(3):237–252, 1995.

[20] CHATFIELD, C. **The analysis of time series: theory and practice**. Springer, 2013.

[21] 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*, KDD '16, p. 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[22] CHRIST, M. **Disponível em: <'https://tsfresh.readthedocs.io/en/latest/text/introduction.html'> acessado em 07 de fev de 2023.**, 2023.

[23] CHRIST, M.; BRAUN, N.; NEUFFER, J.; KEMPA-LIEHR, A. W. **Time series feature extraction on basis of scalable hypothesis tests (tsfresh - a python package)**. *Neurocomputing*, 307:72–77, 2018.

[24] CHUNG, J.; GULCEHRE, C.; CHO, K.; BENGIO, Y. **Empirical evaluation of gated recurrent neural networks on sequence modeling**, 2014.

[25] COTTIER, B.; RAHMAN, R.; FATTORINI, L.; MASLEJ, N.; BESIROGLU, T.; OWEN, D. **The rising costs of training frontier ai models**, 2025.

[26] CROSTON, J. D. **Forecasting and stock control for intermittent demands**. *Journal of the Operational Research Society*, 23(3):289–303, 1972.

[27] DATASOURCE.AI. **Announcing the winners of the VN1 forecasting datathon: Advancing supply chain efficiency and reducing forecasting errors**. DataSource.ai Blog, 2024. Accessed on August 6, 2025. URL: https://www.datasource.ai/en/data-science-articles/announcing-the-winners-of-the-vn1-forecasting-datathon-advancing-supply-cha

[28] DEY, B.; ROY, B.; DATTA, S.; USTUN, T. S. **Forecasting ethanol demand in india to meet future blending targets: A comparison of arima and various regression models**. *Energy Reports*, 9:411–418, 2023. 2022 9th International Conference on Power and Energy Systems Engineering.

[29] DOUKHAN, P. **Stochastic models for time series**, volume 80. Springer, 2018.

[30] EDBROOKE, J. **Essays on Time Series Analysis**. PhD thesis, Imperial College London, 2018. Disponível em: https://www.imperial.ac.uk/media/imperial-college/faculty-of-natural-sciences/department-of-mathematics/math-finance/James_Edbrooke-EDBROOKE_JAMES_01290027.pdf.

[31] EGGENSPERGER, K.; MÜLLER, P.; MALLIK, N.; FEURER, M.; SASS, R.; KLEIN, A.; AWAD, N. H.; LINDAUER, M.; HUTTER, F. **Hpobench: A collection of reproducible multi-fidelity benchmark problems for HPO**. *CoRR*, abs/2109.06716, 2021.

[32] ETZKOWITZ, H. **The Triple Helix: University-industry-government Innovation in Action**. NetLibrary, Inc. Routledge, 2008.

[33] FLORES, B. E.; CLAY WHYBARK, D. **Multiple criteria abc analysis**. *International Journal of Operations & Production Management*, 6(3):38–46, Jan. 1986.

[34] GERS, F. A.; SCHMIDHUBER, J.; CUMMINS, F. **Learning to forget: continual prediction with lstm.** *Neural computation*, 12:2451–71, Oct 2000.

[35] GERS, F. A.; SCHMIDHUBER, J. **Recurrent nets that time and count**. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, 3:189–194 vol.3, 2000.

[36] GRAVES, A.; SCHMIDHUBER, J. **Framewise phoneme classification with bidirectional lstm and other neural network architectures**. *Neural Networks*, 18(5):602–610, 2005. IJCNN 2005.

[37] GUPTA, S. D. **Evaluating time series foundation models on noisy periodic time series**, 2025.

[38] HAQUE, M. S.; AMIN, M. S.; MIAH, J.; CAO, D. M.; SAYED, M. A.; AHMED, S. **Retail demand forecasting: A comparative study for multivariate time series**. *Journal of Mathematics and Statistics Studies*, 4(4):40–46, 2023.

[39] HARVEY, A. C.; SHEPHARD, N. **Structural time series models**, volume Vol. 11:Econometrics, p. 261–302. North Holland, Amsterdam, (edited by g.s. maddala, c.r. rao and h.d. vinod) edition, 1993.

[40] HERZEN, J.; LÃ¤SSIG, F.; PIAZZETTA, S. G.; NEUER, T.; TAFTI, L.; RAILLE, G.; POTTELBERGH, T. V.; PASIEKA, M.; SKRODZKI, A.; HUGUENIN, N.; DUMONAL, M.; KOÅ›CISZ, J.; BADER, D.; GUSSET, F.; BENHEDDI, M.; WILLIAMSON, C.; KOSINSKI, M.; PETRIK, M.; GROSCH, G. **Darts: User-friendly modern machine learning for time series**. *Journal of Machine Learning Research*, 23(124):1–6, 2022.

[41] HOCHREITER, S.; SCHMIDHUBER, J. **Long short-term memory**. *Neural Computation*, 9(8):1735–1780, nov 1997.

[42] HOLT, C. C. **Forecasting seasonals and trends by exponentially weighted moving averages**. Technical report, International Journal of Forecasting, 1957.

[43] HUANG, Y.; CHATTOPADHYAY, I. **Data smashing 2.0: Sequence likelihood (sl) divergence for fast time series comparison**, 2019.

[44] HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. **Automated Machine Learning: Methods, Systems, Challenges**. The Springer Series on Challenges in Machine Learning. Springer International Publishing, 2019.

[45] HYNDMAN, R. J.; BILLAH, B. **Unmasking the theta method**. *International Journal of Forecasting*, 19(2):287–290, 2003.

[46] HYNDMAN, R. J.; KHANDAKAR, Y. **Automatic time series forecasting: The forecast package for r**. *Journal of Statistical Software*, 27(3):1–22, 2008.

[47] HYNDMAN, R.; ATHANASOPOULOS, G. **Forecasting: Principles and Practice**. OTexts, Australia, 2nd edition, 2018.

[48] IHL GROUP. **Retail inventory distortion: The good, the bad, and the ugly**. Technical report, IHL Group, 2023. Disponível em: https://blueyonder.com/resources/retail-inventory-distortion-report.

[49] JOHNSTON, F. R.; BOYLAN, J. E. **Forecasting for items with intermittent demand**. *Journal of the Operational Research Society*, 47(1):113–121, 1996.

[50] JOHNSTON, F. R.; BOYLAN, J. E.; SHALE, E. A. **An examination of the size of orders from customers, their characterisation and the implications for inventory control of slow moving items**. *Journal of the Operational Research Society*, 54(8):833–837, Aug. 2003.

[51] KARLSSON, ADAM E LINDSTRÖM, F. **Adf versus kpss: A comparison of the two tests through a monte carlo simulation**. Master's thesis, Lund University, 2022. Disponível em: https://www.diva-portal.org/smash/get/diva2:1668033/FULLTEXT01.pdf.

[52] KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. **Lightgbm: A highly efficient gradient boosting decision tree**. In: Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[53] KOBIELA, D.; KREFTA, D.; KRÓL, W.; WEICHBROTH, P. **Arima vs lstm on nasdaq stock exchange data**. *Procedia Computer Science*, 207:3836–3845, 2022.

[54] KOHONEN, T. **Self-organized formation of topologically correct feature maps**. *Biological Cybernetics*, 43(1):59–69, 1982.

[55] KOLLTVEIT, A. B.; LI, J. **Operationalizing machine learning models - a systematic literature review**. In: *2022 IEEE/ACM 1st International Workshop on Software Engineering for Responsible Artificial Intelligence (SE4RAI)*, p. 1–8, 2022.

[56] KWIATKOWSKI, D.; PHILLIPS, P. C. B.; SCHMIDT, P.; SHIN, Y. **Testing the null hypothesis of stationarity against the alternative of a unit root**. *Journal of Econometrics*, 54(1-3):159–178, 1992.

[57] KWIATKOWSKI, D.; PHILLIPS, P. C.; SCHMIDT, P.; SHIN, Y. **Testing the null hypothesis of stationarity against the alternative of a unit root: How sure**

are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159–178, 1992.

[58] LIANG, Y.; WEN, H.; NIE, Y.; JIANG, Y.; JIN, M.; SONG, D.; PAN, S.; WEN, Q. **Foundation models for time series analysis: A tutorial and survey**. In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, p. 6555–6565. ACM, Aug. 2024.

[59] LIAO, T. W. **Clustering of time series data—a survey**. *Pattern Recognition*, 38(11):1857–1874, nov 2005.

[60] LIM, B.; ARIK, S. O.; LOEFF, N.; PFISTER, T. **Temporal fusion transformers for interpretable multi-horizon time series forecasting**. *International Journal of Forecasting*, 37:1748–1764, 2021.

[61] LINDAUER, M.; EGGENSPERGER, K.; FEURER, M.; BIEDENKAPP, A.; DENG, D.; BENJAMINS, C.; RUHKOPF, T.; SASS, R.; HUTTER, F. **Smac3: A versatile bayesian optimization package for hyperparameter optimization**. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

[62] LIU, Z.; GODAHEWA, R.; BANDARA, K.; BERGMEIR, C. **Handling concept drift in global time series forecasting**, 2023.

[63] LU, Y.; PHILLIPS, G. M.; YANG, J. **The impact of cloud computing and ai on industry dynamics and concentration**. NBER Working Papers 32811, National Bureau of Economic Research, Inc, Aug 2024.

[64] 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*, 249, 08 2015.

[65] MACQUEEN, J. **Classification and analysis of multivariate observations**. In: *5th Berkeley Symp. Math. Statist. Probability*, p. 281–297. University of California Los Angeles LA USA, 1967.

[66] MAKRIDAKIS, S.; HIBON, M. **The m3-competition: results, conclusions and implications**. *International Journal of Forecasting*, 16(4):451–476, 2000. The M3-Competition.

[67] MAKRIDAKIS, S.; SPILIOTIS, E.; ASSIMAKOPOULOS, V. **M5 accuracy competition: Results, findings, and conclusions**. *International Journal of Forecasting*, 38(4):1346–1364, 2022. Special Issue: M5 competition.

[68] McKinsey & Company. **Improving retail forecast accuracy with machine learning**, 2021. Baseado em dados de estudos da McKinsey. Disponível em: https://aws.amazon.com/blogs/architecture/improving-retail-forecast-accuracy-with-machine-learning/.

[69] Mitra, S.; Reddy, M. S.; Prince, K. **Inventory control using fsn analysis–a case study on a manufacturing industry**. *International Journal of Innovative Science, Engineering & Technology*, 2(4):322–325, 2015.

[70] Montero-Manso, P.; Athanasopoulos, G.; Hyndman, R. J.; Talagala, T. S. **Fforma: Feature-based forecast model averaging**. *International Journal of Forecasting*, 36(1):86–92, 2020. M4 Competition.

[71] Nikitin, N. O.; Vychuzhanin, P.; Sarafanov, M.; Polonskaia, I. S.; Revin, I.; Barabanova, I. V.; Maximov, G.; Kalyuzhnaya, A. V.; Boukhanovsky, A. **Automated evolutionary approach for the design of composite machine learning pipelines**. *Future Generation Computer Systems*, 2021.

[72] Nordin, N. H. M.; Bakar, A. S. A. **Forecast performance of malaysia export of goods for 2022-2023 with box-jenkins and arima model**. *International Journal of Advanced Management and Business Intelligence*, 4(1), 2023.

[73] Oreshkin, B. N.; Carpov, D.; Chapados, N.; Bengio, Y. **Meta-learning framework with applications to zero-shot time-series forecasting**, 2020.

[74] Oreshkin, B. N.; Carpov, D.; Chapados, N.; Bengio, Y. **N-BEATS: Neural basis expansion analysis for interpretable time series forecasting**. In: *International Conference on Learning Representations*, 2020.

[75] Osborn, D. R.; Chui, A. P. L.; Smith, J. P.; Birchenhall, C. R. **Seasonality and the order of integration for consumption***. *Oxford Bulletin of Economics and Statistics*, 50(4):361–377, 1988.

[76] Paparrizos, J.; Gravano, L. **k-shape: Efficient and accurate clustering of time series**. *SIGMOD Rec.*, 45(1):69–76, June 2016.

[77] Perkmann, M.; Tartari, V.; McKelvey, M.; Autio, E.; Broström, A.; D'Este, P.; Fini, R.; Geuna, A.; Grimaldi, R.; Hughes, A.; Krabel, S.; Kitson, M.; Llerena, P.; Lissoni, F.; Salter, A.; Sobrero, M. **Academic engagement and commercialisation: A review of the literature on university–industry relations**. *Research Policy*, 42(2):423–442, 2013.

[78] ROŽANEC, J. M.; FORTUNA, B.; MLADENIĆ, D. **Reframing demand forecasting: A two-fold approach for lumpy and intermittent demand**. *Sustainability*, 14(15), 2022.

[79] SALINAS, D.; FLUNKERT, V.; GASTHAUS, J. **Deepar: Probabilistic forecasting with autoregressive recurrent networks**, 2017.

[80] SCHOLZ-REITER, B.; HEGER, J.; MEINECKE, C.; BERGMANN, J. **Integration of demand forecasts in abc-xyz analysis: practical investigation at an industrial company**. *International Journal of Productivity and Performance Management*, 61(4):445–451, Jan. 2012.

[81] SCULLEY, D.; HOLT, G.; GOLOVIN, D.; DAVYDOV, E.; PHILLIPS, T.; EBNER, D.; CHAUDHARY, V.; YOUNG, M. **Machine learning: The high interest credit card of technical debt**. In: *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[82] SCULLEY, D.; HOLT, G.; GOLOVIN, D.; DAVYDOV, E.; PHILLIPS, T.; EBNER, D.; CHAUDHARY, V.; YOUNG, M.; CRESPO, J.-F.; DENNISON, D. **Hidden technical debt in machine learning systems**. In: *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, p. 2503–2511, Cambridge, MA, USA, 2015. MIT Press.

[83] SHARMA, A.; ENE, T.-D.; KUNAL, K.; LIU, M.; HASAN, Z.; REN, H. **Assessing economic viability: A comparative analysis of total cost of ownership for domain-adapted large language models versus state-of-the-art counterparts in chip design coding assistance**, 2024.

[84] SHCHUR, O.; TURKMEN, C.; ERICKSON, N.; SHEN, H.; SHIRKOV, A.; HU, T.; WANG, Y. **Autogluon-timeseries: Automl for probabilistic time series forecasting**, 2023.

[85] SUPERLINEAR. **Foundation models for forecasting: The future or folly?** Superlinear Insights, 2024. URL: https://superlinear.eu/insights/articles/foundation-models-for-forecasting-the-future-or-folly.

[86] SYNTETOS, A.; BOYLAN, J. **On the bias of intermittent demand estimates**. *International Journal of Production Economics*, 71(1):457–466, 2001. Tenth International Symposium on Inventories.

[87] SYNTETOS, A. A.; BOYLAN, J. E. **The accuracy of intermittent demand estimates**. *International Journal of Forecasting*, 21(2):303–314, 2005.

[88] TAN, M.; MERRILL, M. A.; GUPTA, V.; ALTHOFF, T.; HARTVIGSEN, T. **Are language models actually useful for time series forecasting?**, 2024.

[89] TARADE, R.; KATTI, P. **A comparative analysis for wind speed prediction**. In: *2011 International Conference on Energy, Automation and Signal*, p. 1–6. IEEE, IEEE, dec 2011.

[90] TAYLOR, S. J.; LETHAM, B. **Forecasting at scale**. *PeerJ Preprints*, 5:e3190v2, Sept. 2017.

[91] TEUNTER, R. H.; SYNTETOS, A. A.; BABAI, M. Z. **Intermittent demand: Linking forecasting to inventory obsolescence**. *European Journal of Operational Research*, 214(3):606–615, 2011.

[92] THORNTON, C.; HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. **Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms**. *CoRR*, abs/1208.3719, 2012.

[93] TULABANDHULA, T.; RUDIN, C. **Machine learning with operational costs**, 2013.

[94] VAN HOUDT, G.; MOSQUERA, C.; NÁPOLES, G. **A review on the long short-term memory model**. *Artificial Intelligence Review*, 53, 12 2020.

[95] WEI, W. W. **Multivariate time series analysis and applications**. John Wiley & Sons, 2019.

[96] WESTERGAARD, G.; ERDEN, U.; MATEO, O. A.; LAMPO, S. M.; AKINCI, T. C.; TOPSAKAL, O. **Time series forecasting utilizing automated machine learning (automl): A comparative analysis study on diverse datasets**. *Information*, 15(1), 2024.

[97] WINTERS, P. R. **Forecasting sales by exponentially weighted moving averages**. *Management science*, 6(3):324–342, 1960.

[98] ZANOTTI, M. **On the retraining frequency of global forecasting models**, 2025.

[99] ZHANG, X. **Achieving 1st place in the VN1 forecasting competition with fine-tuned Moirai model**. dev.to, 2024. Accessed on August 6, 2025. URL: https://dev.to/orange111/achieving-1st-place-in-the-vn1-forecasting-competition-with-fine-tuned-moir

[100] ZHAO, L.; SHEN, Y. **Proactive model adaptation against concept drift for online time series forecasting**. In: *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '25, p. 2020–2031. ACM, July 2025.

[101] ÁNGEL LÓPEZ ORIONA.; MANSO, P. M.; FERNÁNDEZ, J. A. V. **Time series clustering based on prediction accuracy of global forecasting models**, 2023.

APÊNDICE A

Desempenho da Clusterização - Janela Reduzida

Tabela A.1: Custo computacional dos modelos, loja 42.

Modelo	**Custo Computacional (s)**		
AutoArima	3131		
Croston	229		
DeepAR	2467		
ExponentialSmoothing	39		
LightGBM	410		
Naive	15		
Prophet	1505		
RNN	2426		
TSB	292		
Theta	155		
Empírico	10670		
Modelo	**Custo Computacional (s)**		
	MASE	**RMSSE**	**SMAPE**
kshape_30	2430	2730	2408
kshape_60	3490	6026	3498
kshape_120	5143	7648	5116
som_30	1913	2059	1809
som_60	2868	5526	2876
som_120	4515	6737	4517
timesmash_30	1465	1686	1454
timesmash_60	2516	4933	2504
timesmash_120	4290	6216	4145

Tabela A.2: Custo computacional dos modelos, loja 138.

Modelo	**Custo Computacional (s)**		
AutoArima	1744		
Croston	572		
DeepAR	6497		
ExponentialSmoothing	102		
LightGBM	868		
Naive	57		
Prophet	3038		
RNN	5148		
TSB	703		
Theta	473		
Empírico	19202		
Modelo	**Custo Computacional (s)**		
	MASE	**RMSSE**	**SMAPE**
kshape_30	4717	5340	4741
kshape_60	6926	8371	4903
kshape_120	11530	13252	11473
som_30	10785	11304	10731
som_60	12927	14280	12904
som_120	17497	18888	17446
timesmash_30	3105	3741	3037
timesmash_60	5229	6650	5239
timesmash_120	9817	11233	9809

Tabela A.3: Custo computacional dos modelos, loja 171.

Modelo	**Custo Computacional (s)**		
AutoArima	2732		
Croston	385		
DeepAR	4453		
ExponentialSmoothing	74		
LightGBM	584		
Naive	36		
Prophet	2153		
RNN	3596		
TSB	486		
Theta	322		
Empírico	14820		
Modelo	**Custo Computacional (s)**		
	MASE	**RMSSE**	**SMAPE**
kshape_30	3132	3537	3121
kshape_60	4559	6865	4579
kshape_120	7166	9695	7180
som_30	3952	4210	3733
som_60	5196	7309	5189
som_120	8123	10194	7802
timesmash_30	2278	2364	1997
timesmash_60	3430	5740	3423
timesmash_120	6048	8308	6058

Tabela A.4: Tabela de dominância dos modelos nos clusters, loja 42.

Modelo	Janela	Métrica	AA	CR	ES	LGB	NA	RNN	TSB	TT
kshape	30	MASE			1	1340	24			
kshape	30	RMSSE	1			1206			158	
kshape	30	SMAPE				1024	341			
kshape	60	MASE				1341	21			3
kshape	60	RMSSE	21			1326			18	
kshape	60	SMAPE		62		986	317			
kshape	120	MASE				1338	20			7
kshape	120	RMSSE	21			1331			13	
kshape	120	SMAPE		7		1017	341			
som	30	MASE				1340	19	5	1	
som	30	RMSSE		8		1207			150	
som	30	SMAPE		45		1002	318			
som	60	MASE				1342	21			2
som	60	RMSSE	19	12		1334				
som	60	SMAPE		45		1002	318			
som	120	MASE				1340	19			6
som	120	RMSSE	18	12		1330			5	
som	120	SMAPE		45		1002	318			
timesmash	30	MASE				1344	19		2	
timesmash	30	RMSSE		24		1207			134	
timesmash	30	SMAPE		82		983	300			
timesmash	60	MASE				1346	19			
timesmash	60	RMSSE	19	26		1317			3	
timesmash	60	SMAPE		37		1004	324			
timesmash	120	MASE				1340	14			11
timesmash	120	RMSSE	18	4	1	1314			28	
timesmash	120	SMAPE		37		1004	324			

Legenda: AA=AutoArima, CR=Croston, ES=Exponential Smoothing, LGB=LightGBM, NA=Naive, PR=Prophet, TT=Theta.

Tabela A.5: Tabela de dominância dos modelos nos clusters, loja 138.

Modelo	Janela	Métrica	AA	CR	ES	LGB	NA	PR	TSB	TT
kshape	30	MASE				2905	69		1	
kshape	30	RMSSE	1	4		2707			263	
kshape	30	SMAPE		99	6	2244	622	2	2	
kshape	60	MASE				2906	69			
kshape	60	RMSSE	57	4		2902			12	
kshape	60	SMAPE	3	5		2317	650			
kshape	120	MASE	1			2900	60			14
kshape	120	RMSSE	60			2913		2		
kshape	120	SMAPE	1	4		2318	650		2	
som	30	MASE	3	1		2909	58		4	
som	30	RMSSE	1	76		2742			156	
som	30	SMAPE		204		2201	570			
som	60	MASE				2905	62		1	7
som	60	RMSSE	52	10		2902			11	
som	60	SMAPE		110		2260	601		4	
som	120	MASE				2903	59		1	12
som	120	RMSSE	52	2	1	2905		3	12	
som	120	SMAPE		62		2293	620			
timesmash	30	MASE		2		2908	61		4	
timesmash	30	RMSSE	6	54		2745			170	
timesmash	30	SMAPE		92		2267	616			
timesmash	60	MASE				2908	64			3
timesmash	60	RMSSE	56	2		2910			7	
timesmash	60	SMAPE		92		2267	616			
timesmash	120	MASE				2903	56			16
timesmash	120	RMSSE	56			2909			10	
timesmash	120	SMAPE		92		2267	616			

Legenda: AA=AutoArima, CR=Croston, ES=Exponential Smoothing, LGB=LightGBM, NA=Naive, PR=Prophet, TT=Theta.

Tabela A.6: Tabela de dominância dos modelos nos clusters, loja 171.

Modelo	Janela	Métrica	AA	CR	ES	LGB	NA	PR	RNN	TSB	TT
kshape	30	MASE				1994	36			2	
kshape	30	RMSSE		4		1821				207	
kshape	30	SMAPE		53	4	1568	405	2			
kshape	60	MASE				1996	36				
kshape	60	RMSSE	35	6		1982				9	
kshape	60	SMAPE	3	47	3	1571	408				
kshape	120	MASE				1995	36		1		
kshape	120	RMSSE	38			1991				3	
kshape	120	SMAPE		47	3	1574	408				
som	30	MASE		1		1992	34		5		
som	30	RMSSE	6	48		1830				148	
som	30	SMAPE		68		1561	403				
som	60	MASE		1		1994	34			3	
som	60	RMSSE	32	10		1983				7	
som	60	SMAPE		68		1561	403				
som	120	MASE				1993	34		5		
som	120	RMSSE	36	1		1990				5	
som	120	SMAPE		89		1548	395				
timesmash	30	MASE		1		1991	33		7		
timesmash	30	RMSSE		52		1820				160	
timesmash	30	SMAPE		109		1536	387				
timesmash	60	MASE				1995	35			2	
timesmash	60	RMSSE	37	1		1990				4	
timesmash	60	SMAPE		36		1577	419				
timesmash	120	MASE				1996	33				3
timesmash	120	RMSSE	37	1		1994					
timesmash	120	SMAPE		109		1536	387				

Legenda: AA=AutoArima, CR=Croston, ES=Exponential Smoothing, LGB=LightGBM, NA=Naive, PR=Prophet, TT=Theta.

APÊNDICE B

Desempenho da Clusterização - Conjunto Reduzido

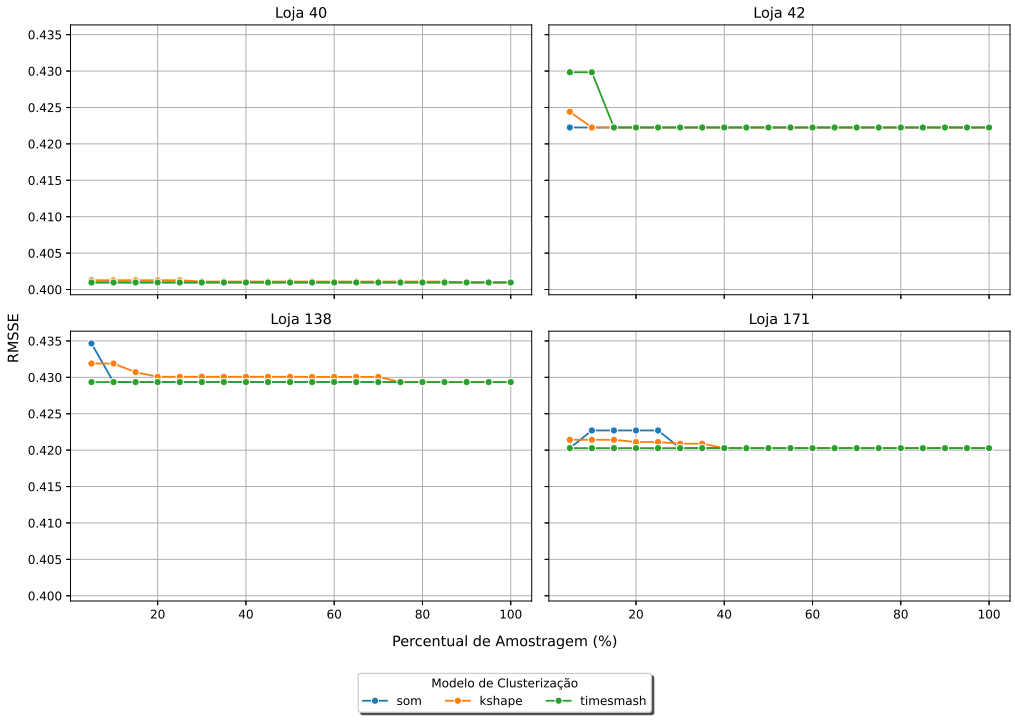


Figura B.1: Evolução do RMSSE pelo percentual de amostragem

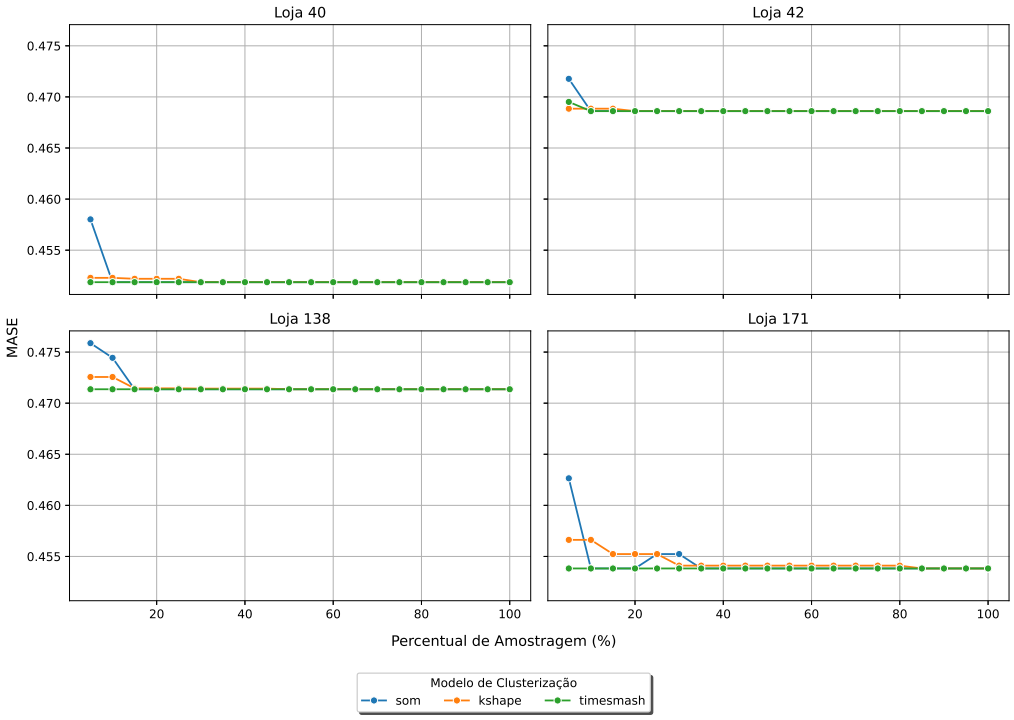


Figura B.2: Evolução do MASE pelo percentual de amostragem

APÊNDICE C

Desempenho da Clusterização - SOM

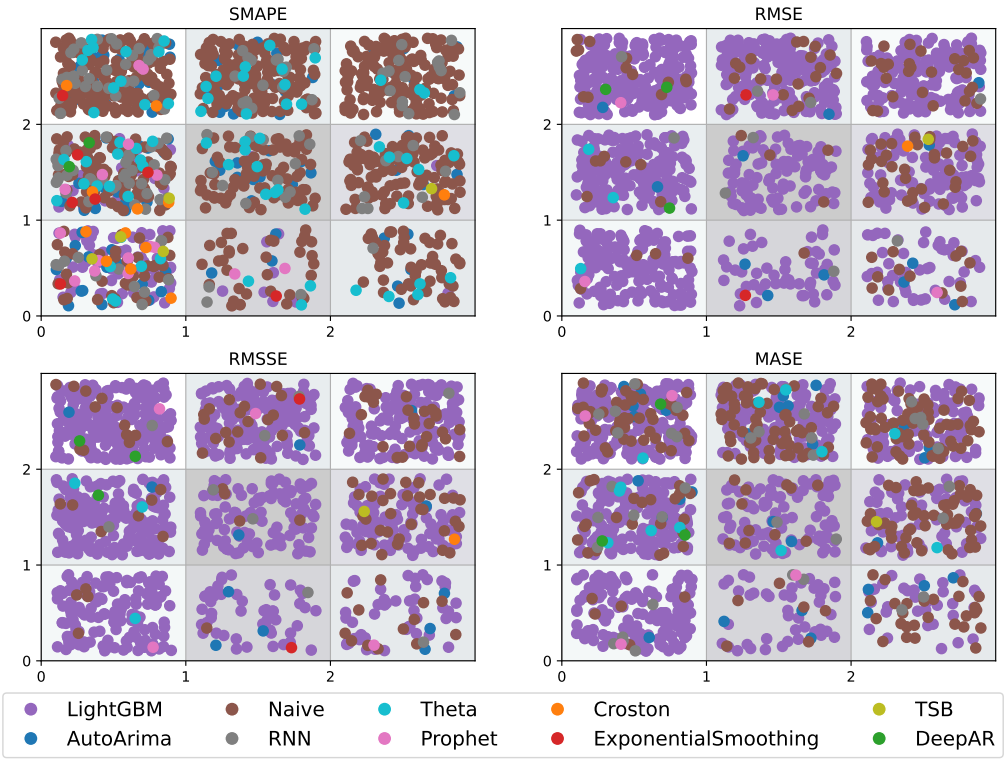


Figura C.1: Mapa SOM para loja 42

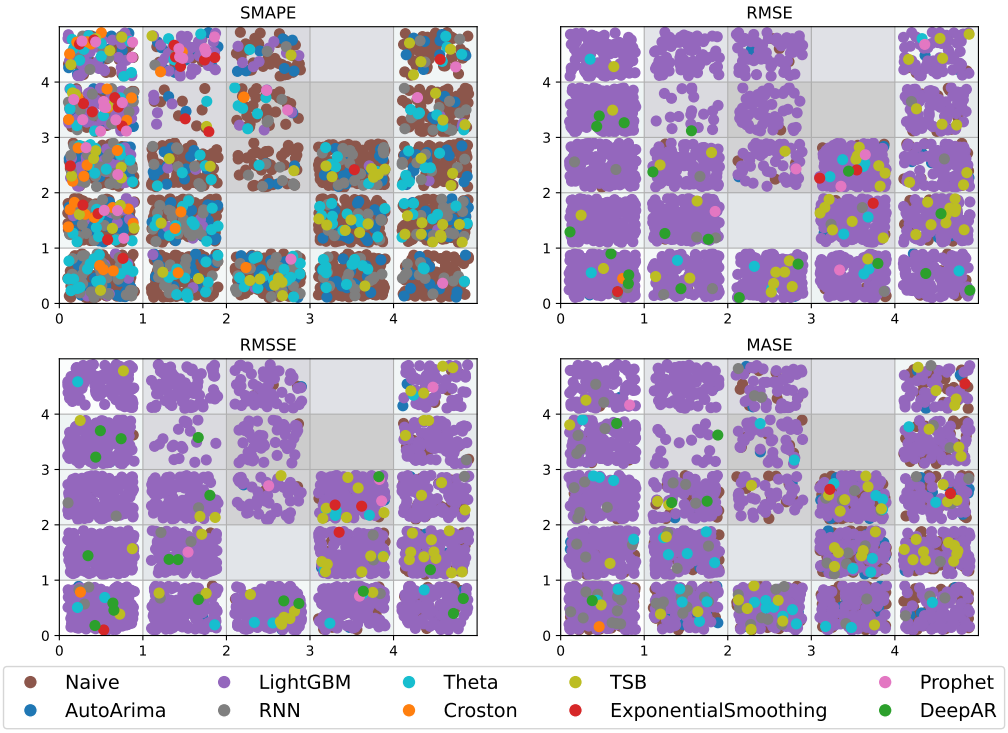


Figura C.2: Mapa SOM para loja 138

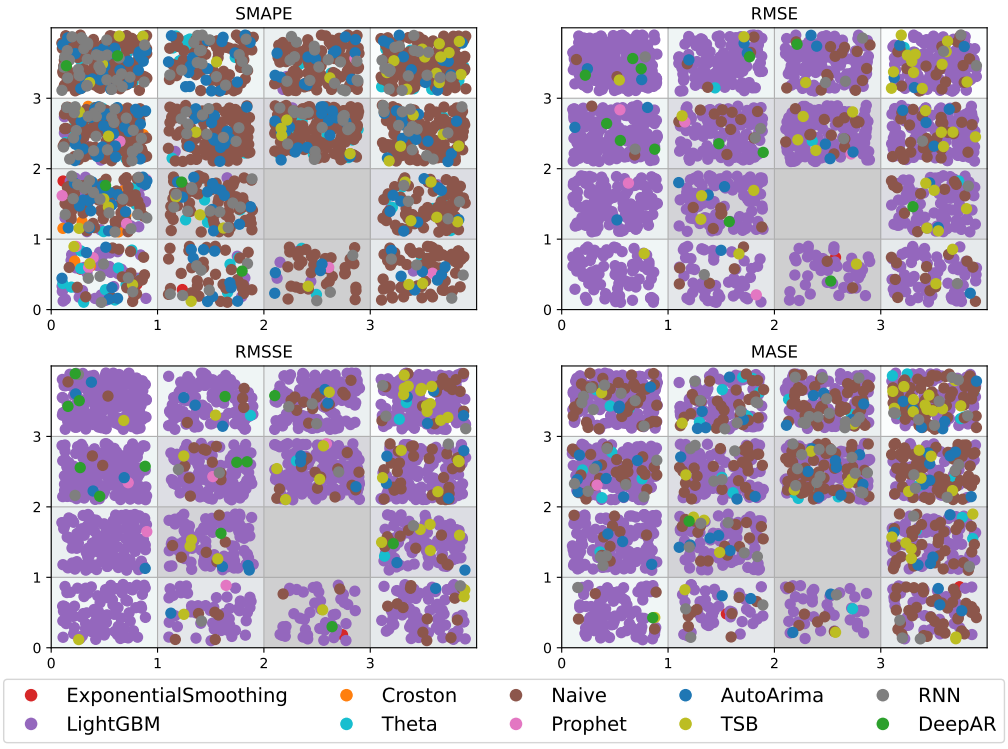


Figura C.3: Mapa SOM para loja 171

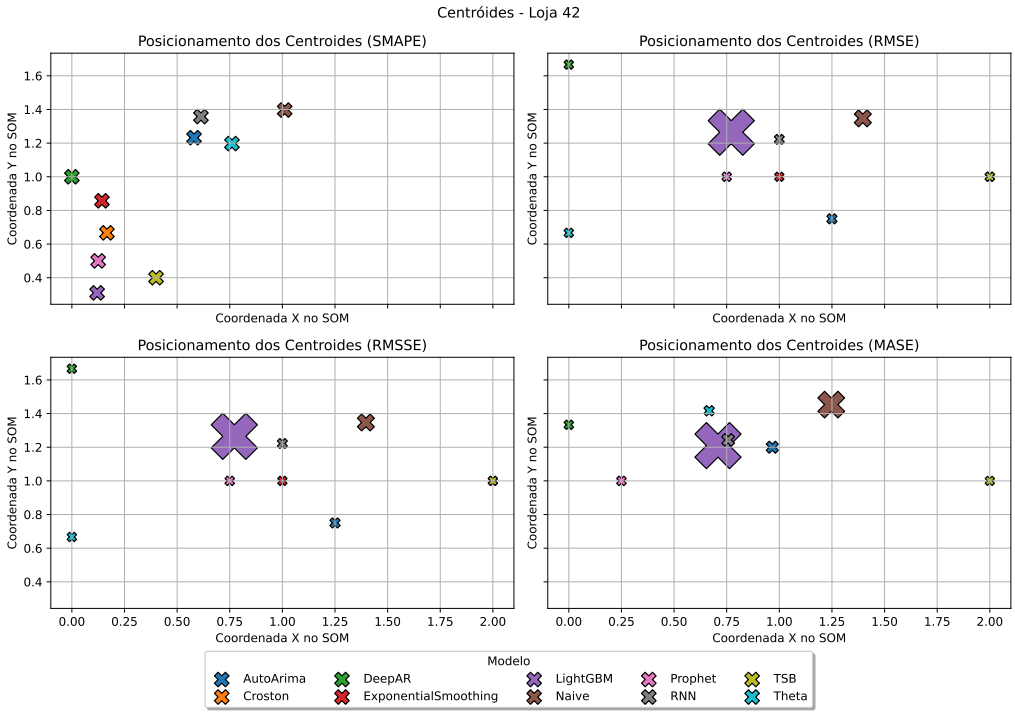


Figura C.4: Mapa SOM para loja 42

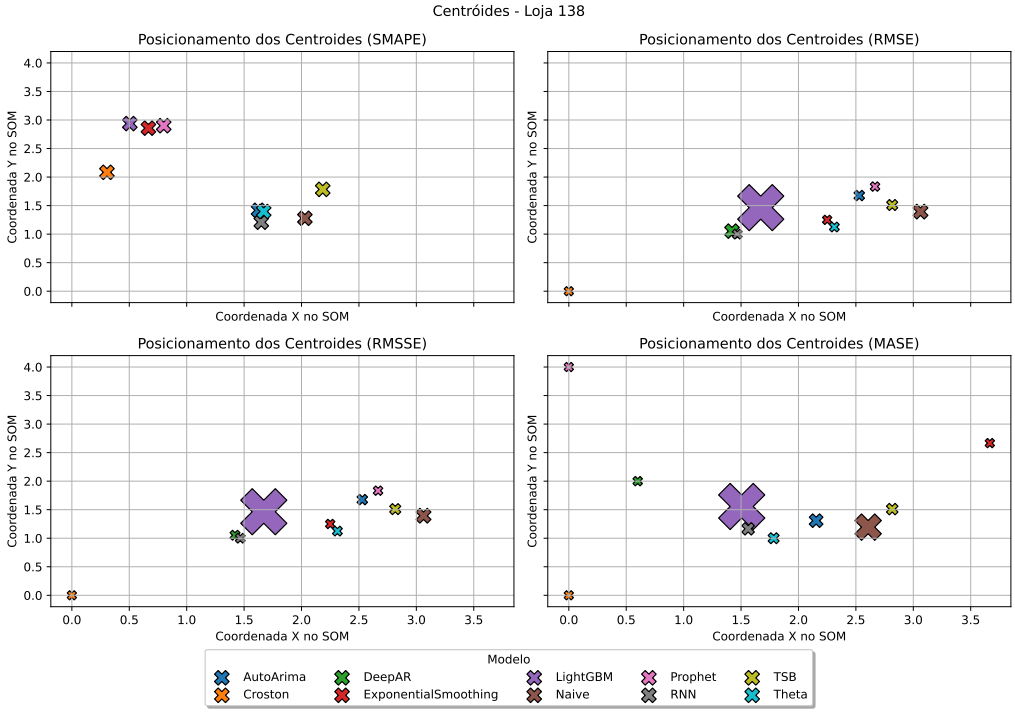


Figura C.5: Mapa SOM para loja 138

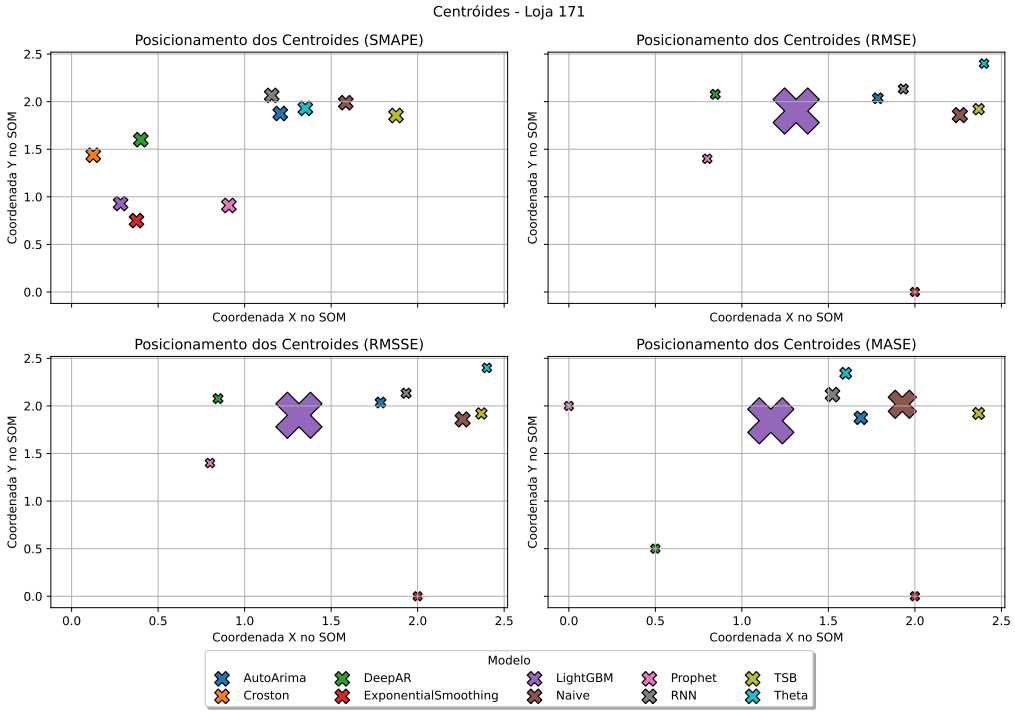


Figura C.6: Mapa SOM para loja 171

APÊNDICE D

Métricas de Acurácia

Tabela D.1: Métricas médias por modelo na loja 42.

Model	SMAPE	RMSSE	MASE
Empírico	0.5344	0.4978	0.6357
AutoArima	1.5464	0.9122	1.4102
Croston	1.6164	1.0213	2.1969
DeepAR	1.7509	1.7920	2.5269
ExponentialSmoothing	1.7124	0.7744	1.1988
LightGBM	1.6315	**0.5017**	**0.6570**
Naive	**0.6731**	0.9218	1.2367
Prophet	1.7074	0.7730	1.1866
RNN	**1.2846**	1.2370	1.4109
TSB	1.6876	0.7719	1.2345
Theta	1.3214	0.7631	0.9202
kshape_30	1.4375	0.5322	0.6710
kshape_60	1.4455	0.5087	0.6698
kshape_120	1.4379	**0.5080**	0.6701
timesmash_30	1.4423	0.5306	0.6691
timesmash_60	1.4375	0.5141	**0.6691**
timesmash_120	1.4375	0.5155	0.6714
som_30	1.4387	0.5323	0.6721
som_60	1.4387	0.5083	0.6696
som_120	1.4387	0.5088	0.6700

Obs.: **Menor valor**, **Segundo menor valor**

Tabela D.2: Métricas médias por modelo na loja 138.

Model	SMAPE	RMSSE	MASE
Empírico	0.5687	0.4624	0.5611
AutoArima	1.5149	0.8371	1.2575
Croston	1.6193	0.9591	1.8435
DeepAR	1.7446	1.5960	2.2067
ExponentialSmoothing	1.6945	0.7167	1.1003
LightGBM	1.6228	**0.4681**	**0.5746**
Naive	**0.7054**	0.8913	1.1843
Prophet	1.6945	0.7220	1.1030
RNN	1.3875	1.2020	1.3206
TSB	1.6948	0.7270	1.1336
Theta	**1.3779**	0.7129	0.8492
kshape_30	1.4844	0.4895	0.5911
kshape_60	1.4833	0.4736	0.5910
kshape_120	1.4838	0.4737	**0.5889**
timesmash_30	1.4850	0.4897	0.5895
timesmash_60	1.4850	**0.4734**	0.5905
timesmash_120	1.4850	0.4734	0.5891
som_30	1.4870	0.4899	0.5908
som_60	1.4859	0.4743	0.5899
som_120	1.4846	0.4736	0.5899

Obs.: **Menor valor**, **Segundo menor valor**

Tabela D.3: Métricas médias por modelo na loja 171.

Model	SMAPE	RMSSE	MASE
Empírico	0.5140	0.4564	0.5624
AutoArima	1.5263	0.8393	1.2812
Croston	1.6609	0.9877	1.9686
DeepAR	1.7730	1.4140	1.9546
ExponentialSmoothing	1.7437	0.7154	1.1159
LightGBM	1.6653	**0.4600**	**0.5734**
Naive	**0.6527**	0.8822	1.1965
Prophet	1.7380	0.7167	1.1050
RNN	**1.2672**	1.0069	1.0890
TSB	1.7324	0.7227	1.1435
Theta	1.3699	0.7037	0.8384
kshape_30	1.5058	0.4831	0.5923
kshape_60	1.5036	0.4665	**0.5922**
kshape_120	1.5037	0.4654	0.5923
timesmash_30	1.5053	0.4839	0.5935
timesmash_60	1.5018	0.4654	0.5923
timesmash_120	1.5053	**0.4651**	0.5924
som_30	1.5038	0.4833	0.5934
som_60	1.5038	0.4684	0.5924
som_120	1.5088	0.4654	0.5934

Obs.: **Menor valor**, **Segundo menor valor**

Tabela D.4: RMSE por faixa, por modelo na loja 42

faixa Model	[0-5[	[5-10[	[10-20[	[20-30[	[30-40[	[40+
AutoArima	1.034390	1.196944	1.988229	2.768392	4.862569	29.359416
Croston	1.620899	1.311422	1.775636	2.592533	3.527405	6.669441
DeepAR	1.873343	2.324800	3.030746	2.894123	3.890238	11.127029
ES	0.483841	0.960393	1.609373	2.470465	3.709964	7.578888
LightGBM	**0.306811**	**0.648327**	**1.071101**	**1.623630**	**2.549230**	**3.418470**
Naive	0.724584	1.257359	2.109223	2.978773	4.635254	8.852672
Prophet	0.518440	**0.953329**	**1.598729**	**2.448556**	**3.491843**	5.946146
RNN	1.058004	1.392165	1.968753	2.893783	4.065072	11.247702
TSB	0.493699	0.970270	1.643438	2.542002	3.492214	6.108933
Theta	**0.458837**	0.973915	1.645599	2.524669	3.705708	**5.819699**

Obs.: **Menor valor**, **Segundo menor valor**

Tabela D.5: RMSE por faixa, por modelo na loja 138

faixa Model	[0-5[	[5-10[	[10-20[	[20-30[	[30-40[	[40+
AutoArima	0.821404	1.147534	1.851577	4.468271	4.338968	9.704605
Croston	1.522603	1.277661	1.877637	2.900250	3.417908	6.388197
DeepAR	1.711290	1.675775	1.953148	2.692054	3.618793	10.439556
ES	0.484082	**0.884569**	**1.474019**	**2.158040**	3.042946	7.311697
LightGBM	**0.309523**	**0.604004**	**0.987040**	**1.456195**	**2.027190**	**4.251806**
Naive	0.840156	1.164211	1.890003	3.015818	4.027987	11.639670
Prophet	0.495816	0.893992	1.476652	2.706352	**2.963504**	7.459701
RNN	1.226768	1.414510	1.938382	2.727866	3.766243	10.651055
TSB	0.501825	0.964345	1.519318	2.363897	3.017995	**6.316927**
Theta	**0.451557**	0.914152	1.530068	2.205112	3.046517	6.623157

Obs.: **Menor valor**, **Segundo menor valor**

Tabela D.6: RMSE por faixa, por modelo na loja 171

faixa Model	[0-5[	[5-10[	[10-20[	[20-30[	[30-40[	[40+
AutoArima	0.730288	1.274881	2.362424	2.732280	6.098509	6.585304
Croston	1.066328	2.232962	2.434109	2.575735	6.711134	**6.061335**
DeepAR	1.521138	1.851967	2.015845	2.606556	3.966152	8.345880
ES	0.470099	0.953549	1.554871	2.351768	**3.496278**	6.132651
LightGBM	**0.301031**	**0.615973**	**1.027364**	**1.502301**	**2.367480**	**3.791482**
Naive	0.881442	1.203189	2.078306	2.636801	4.205323	6.873420
Prophet	0.474653	**0.928729**	**1.510232**	2.425718	3.550854	6.757908
RNN	0.831134	1.162219	1.822517	2.736955	4.074149	8.574252
TSB	0.484232	1.062514	1.595960	**2.269858**	3.576944	6.119282
Theta	**0.447560**	0.962799	1.565231	2.303682	3.605534	6.297966

Obs.: **Menor valor**, **Segundo menor valor**

Tabela D.7: RMSE por categoria de demanda, por modelo na loja 42

categoria Model	Intermittent	Lumpy	Erratic	Smooth
AutoArima	1.097410	2.584759	40.396004	9.491158
Croston	1.151082	3.233609	**3.419108**	7.871223
DeepAR	2.214865	2.590822	4.927369	15.107585
ES	0.869396	1.850688	8.112220	6.561584
LightGBM	**0.564154**	**1.274454**	**2.344753**	**3.773241**
Naive	1.166283	2.122835	5.761027	11.406862
Prophet	**0.864079**	1.901610	5.063131	**6.161487**
RNN	1.376638	2.278327	5.045201	15.238042
TSB	0.879050	1.848514	3.501160	7.613432
Theta	0.891205	**1.843396**	4.024686	6.706002

Obs.: **Menor valor**, **Segundo menor valor**

Tabela D.8: RMSE por categoria de demanda, por modelo na loja 138

categoria Model	Intermittent	Lumpy	Erratic	Smooth
AutoArima	1.018726	3.432277	9.119760	6.589904
Croston	1.137834	3.419134	**5.376826**	5.825473
DeepAR	1.748278	2.658175	9.745016	9.625398
ES	**0.771687**	**2.036955**	6.860778	5.864056
LightGBM	**0.514501**	**1.416028**	**3.687653**	**3.229120**
Naive	1.048377	2.840405	11.634815	8.571347
Prophet	0.776157	2.062439	7.313391	6.002311
RNN	1.417333	2.562543	9.920249	9.811522
TSB	0.801597	2.149828	5.438402	**5.159470**
Theta	0.787666	2.056810	5.626812	6.021915

Obs.: **Menor valor**, **Segundo menor valor**

Tabela D.9: RMSE por categoria de demanda, por modelo na loja 171

categoria Model	Intermittent	Lumpy	Erratic	Smooth
AutoArima	0.944080	3.147500	4.021080	5.563392
Croston	1.199752	3.788735	**3.935776**	**4.825955**
DeepAR	1.589426	2.895577	5.427855	7.909352
ES	**0.694450**	2.221419	3.945330	5.144759
LightGBM	**0.460998**	**1.416910**	**2.326369**	**3.186913**
Naive	1.001531	2.868457	4.854842	5.240611
Prophet	0.702740	**2.220388**	4.701017	5.825181
RNN	1.016604	2.557015	5.566212	8.117531
TSB	0.721381	2.276821	4.131509	4.874275
Theta	0.702534	2.282007	4.027729	4.971257

Obs.: **Menor valor**, **Segundo menor valor**