UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Rafael Luiz Xavier

# Sistema Autonômico de Autoescalonamento Proativo Baseado em LSTM Aplicado em Sistemas Legados On-premises

São João del-Rei

2025

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Rafael Luiz Xavier

# Sistema Autonômico de Autoescalonamento Proativo Baseado em LSTM Aplicado em Sistemas Legados On-premises

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São João del-Rei – UFSJ, como requisito parcial para a obtenção do título de Mestre.

Orientador: Elder José Reioli Cirilo

Universidade Federal de São João del-Rei – UFSJ

Mestrado em Ciência da Computação

São João del-Rei

2025

Ficha catalográfica elaborada pela Divisão de Biblioteca (DIBIB)
e Núcleo de Tecnologia da Informação (NTINF) da UFSJ,
com os dados fornecidos pelo(a) autor(a)

X23s

Xavier, Rafael Luiz.
Sistema Autonômico de Autoescalonamento Proativo Baseado em LSTM Aplicado em Sistemas Legados On premises / Rafael Luiz Xavier ; orientador Elder José Reioli Cirilo. -- São João del-Rei, 2025.
100 p.

Dissertação (Mestrado - Ciência da Computação) -- Universidade Federal de São João del-Rei, 2025.

1. Computação Autonômica. 2. Autoescalonamento Proativo. 3. Previsão de Carga de Trabalho. 4. Análise de Séries Temporais. 5. Redes LSTM. I. Cirilo, Elder José Reioli, orient. II. Título.

Rafael Luiz Xavier

# Sistema Autonômico de Autoescalonamento Proativo Baseado em LSTM Aplicado em Sistemas Legados On-premises

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São João del-Rei – UFSJ, como requisito parcial para a obtenção do título de Mestre.

São João del-Rei, 22 de janeiro de 2025:

---

**Elder José Reioli Cirilo**
Orientador

---

**João Pedro Hallack Sansão**
Universidade Federal de São João del-Rei

---

**Baldoino Fonseca dos Santos Neto**
Universidade Federal de Alagoas

São João del-Rei
2025

*Para Alice e Laura, com todo o meu amor.*

# Agradecimentos

Primeiramente, agradeço a Deus pela força e perseverança durante esta jornada acadêmica.

Ao meu orientador, Prof. Dr. Elder José Reioli Cirilo, por sua orientação, dedicação e contribuições valiosas que foram fundamentais para o desenvolvimento desta pesquisa. Seu conhecimento e experiência foram essenciais para elevar a qualidade deste trabalho.

À Universidade Federal de São João Del Rei - UFSJ, por proporcionar um ambiente de excelência acadêmica e toda a infraestrutura necessária para o desenvolvimento desta pesquisa.

Aos membros da banca examinadora, pela disposição em avaliar este trabalho e pelas contribuições que certamente o enriquecerão.

Ao Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG, por disponibilizar os dados e recursos necessários para a realização dos experimentos, e especialmente ao colega Marco Túlio Alves Nolasco Rodrigues pelo apoio e colaboração no tema de pesquisa.

Aos professores do programa de pós-graduação em Ciência da Computação, pelos ensinamentos e discussões que contribuíram para minha formação acadêmica.

Aos meus colegas de mestrado, em especial Marco Alves Otaviano Botelho, pelos estudos, troca de experiências e momentos compartilhados durante esta jornada.

Aos meus pais José e Sandra e à minha irmã Fernanda pela minha educação, formação e apoio incondicional. Sem vocês, esta conquista não seria possível.

À minha esposa Flávia, por seu amor, paciência, compreensão nos momentos de ausência e apoio durante todo este percurso.

Às minhas filhas Alice e Laura, que são minha fonte de inspiração e motivação.

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho e para meu crescimento acadêmico e profissional.

# Resumo

**Contexto**: A crescente repatriação de dados para data centers locais alerta para os desafios e altos custos de manutenção de sistemas legados na nuvem, direcionando esforços para soluções que aprimoram a escalabilidade e eficiência de sistemas on-premises. **Problema**: As limitações dos sistemas de autoescalonamento reativos, que frequentemente atuam apenas após a degradação do desempenho, destacam a necessidade de abordagens proativas. Sistemas voltados ao público que geralmente possuem cargas de trabalho não lineares requerem técnicas que antecipem e respondam dinamicamente às variações de carga para prevenir violações de SLA. **Solução Proposta**: Este trabalho apresenta (1) um modelo preditivo de carga de trabalho para requisições de clientes em sistemas legados voltados ao público utilizando análise de séries temporais multirresolução e redes LSTM, e (2) um sistema autonômico de autoescalonamento proativo implementando o ciclo de controle MAPE-K para alocação automatizada de recursos. **Método**: Esta pesquisa é conduzida como um estudo de caso exploratório-descritivo em duas etapas: primeira, comparando diferentes modelos de previsão utilizando dados benchmark e dados reais de um sistema legado, e segunda, implementando e avaliando um sistema autonômico de autoescalonamento proativo em ambiente de produção. **Resultados**: O modelo LSTM superou métodos estatísticos tradicionais em múltiplas resoluções temporais, com melhores resultados com resolução de 10 minutos. O sistema de autoescalonamento implementado alcançou uma redução de 37,9% na utilização de recursos em comparação com o escalonamento manual, mantendo o desempenho. O modelo WFLS-LSTM demonstrou desempenho robusto em produção (RMSE: 1041, MAPE: 19,84%) e antecipou com sucesso 62,5% dos eventos de pico de carga. **Contribuições**: A pesquisa demonstra tanto a viabilidade dos modelos LSTM para prever cargas de trabalho não lineares em sistemas legados voltados ao público quanto a efetividade de sistemas autonômicos de autoescalonamento proativo em ambientes on-premises, fornecendo uma solução para dimensionamento automático de recursos. **Impacto**: A abordagem combinada de previsão eficiente de carga de trabalho e escalonamento autonômico promove uso mais inteligente de recursos em data centers locais, reduzindo significativamente custos enquanto garante níveis de qualidade e disponibilidade em sistemas legados.

**Palavras-chave**: Computação Autonômica; Sistemas Legados; Redes LSTM; Autoescalonamento Proativo; Previsão de Carga de Trabalho; Análise de Séries Temporais; MAPE-K; Otimização de Recursos; Infraestrutura On-premises.

# Abstract

**Context**: The growing repatriation of data to local data centers highlights the challenges and high costs of maintaining legacy systems in the cloud, directing efforts toward solutions that enhance the scalability and efficiency of on-premises systems. **Problem**: The limitations of reactive auto-scaling systems, which often act only after performance degradation, highlight the need for proactive approaches. Public-facing systems that typically have non-linear workloads require techniques that dynamically anticipate and respond to load variations to prevent SLA violations. **Proposed Solution**: This work presents (1) a predictive workload model for client requests in public-facing legacy systems using multi-resolution time series analysis and LSTM networks, and (2) a proactive autonomic auto-scaling system implementing the MAPE-K control loop for automated resource allocation. **Method**: This research is conducted as an exploratory-descriptive case study in two stages: first, comparing different prediction models using benchmark data and real data from a legacy system, and second, implementing and evaluating a proactive autonomic auto-scaling system in a production environment. **Results**: The LSTM model outperformed traditional statistical methods across multiple temporal resolutions, with best results at 10-minute resolution. The implemented auto-scaling system achieved a 37.9% reduction in resource utilization compared to manual scaling while maintaining performance. The WFLS-LSTM model demonstrated robust production performance (RMSE: 1041, MAPE: 19.84%) and successfully anticipated 62.5% of peak load events. **Contributions**: The research demonstrates both the viability of LSTM models for predicting non-linear workloads in public-facing legacy systems and the effectiveness of proactive autonomic auto-scaling systems in on-premises environments, providing a solution for automatic resource sizing. **Impact**: The combined approach of efficient workload prediction and autonomic scaling promotes smarter resource use in local data centers, significantly reducing costs while ensuring quality and availability levels in legacy systems.

**Keywords**: Autonomic Computing; Legacy Systems; LSTM Networks; Proactive Auto-scaling; Workload Prediction; Time Series Analysis; MAPE-K; Resource Optimization; On-premises Infrastructure.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| ACF | Função de Autocorrelação (Autocorrelation Function) |
| ADF | Teste de Dickey-Fuller Aumentado (Augmented Dickey-Fuller) |
| Adam | Método de Otimização Adaptativa do Momento |
| API | Interface de Programação de Aplicações (Application Programming Interface) |
| AR | Modelo Autorregressivo (Autoregressive) |
| ARIMA | Modelo Autorregressivo Integrado de Médias Móveis (AutoRegressive Integrated Moving Average) |
| ARMA | Modelo Autorregressivo de Médias Móveis (AutoRegressive Moving Average) |
| BPTT | Retropropagação Através do Tempo (Backpropagation Through Time) |
| CECs | Carrosséis de Erro Constante |
| CNN | Rede Neural Convolucional (Convolutional Neural Network) |
| CPU | Unidade Central de Processamento (Central Processing Unit) |
| DE | Algoritmo de Evolução Diferencial (Differential Evolution) |
| DNA | Ácido Desoxirribonucleico |
| ETS | Suavização Exponencial (Exponential Smoothing) |
| GA | Algoritmo Genético (Genetic Algorithm) |
| GRU | Unidade Recorrente Gated (Gated Recurrent Unit) |
| HTTP | Protocolo de Transferência de Hipertexto (Hypertext Transfer Protocol) |
| IaaS | Infraestrutura como Serviço (Infrastructure as a Service) |
| IQR | Intervalo Interquartil (Interquartile Range) |
| KPSS | Teste de Kwiatkowski-Phillips-Schmidt-Shin |
| LR | Regressão Linear (Linear Regression) |

| | |
|---|---|
| LSTM | Long Short-Term Memory |
| MA | Média Móvel (Moving Average) |
| MAPE | Erro Absoluto Percentual Médio (Mean Absolute Percentage Error) |
| MAPE-K | Ciclo de Monitoramento, Análise, Planejamento, Execução e Conhecimento |
| MAE | Erro Médio Absoluto (Mean Absolute Error) |
| MLP | Perceptron Multicamada (Multilayer Perceptron) |
| MSE | Erro Quadrático Médio (Mean Squared Error) |
| NASA | Administração Nacional da Aeronáutica e Espaço (National Aeronautics and Space Administration) |
| Naïve | Modelo Simples Baseado na Última Observação |
| PSO | Otimização por Enxame de Partículas (Particle Swarm Optimization) |
| QoS | Qualidade de Serviço (Quality of Service) |
| RNN | Rede Neural Recorrente (Recurrent Neural Network) |
| ReLU | Unidade Linear Retificada (Rectified Linear Unit) |
| REST | Transferência de Estado Representacional (Representational State Transfer) |
| RESTful | Implementação que segue os princípios REST (REST compliant) |
| RMSE | Raiz do Erro Quadrático Médio (Root Mean Squared Error) |
| SaDE | Evolução Diferencial com Estratégias Adaptativas (Self-adaptive Differential Evolution) |
| SLA | Acordo de Nível de Serviço (Service Level Agreement) |
| SVM | Máquina de Vetores de Suporte (Support Vector Machine) |
| SVR | Regressão por Vetores de Suporte (Support Vector Regression) |
| TI | Tecnologia da Informação |
| URLs | Localizadores Uniformes de Recursos (Uniform Resource Locators) |
| VAR | Modelo Autorregressivo Vetorial (Vector AutoRegression) |

| | |
|---|---|
| VMs | Máquinas Virtuais |
| Weka | Ferramenta de Software para Análise de Dados e Mineração de Dados |
| WWW | Rede Mundial de Computadores (World Wide Web) |

# Sumário

# 1 Introdução

## 1.1 Contextualização

Nos últimos anos, a computação em nuvem consolidou-se como uma tecnologia eficaz, proativa e amplamente adotada pelas empresas para otimizar suas operações (RADHIKA; Sudha Sadasivam, 2021). Esta tecnologia fornece vários benefícios, como confiabilidade, flexibilidade, segurança, serviços escaláveis, economia de energia, estabilidade e desempenho (CHANG, 2022). Apesar do crescimento notável na adoção de serviços de computação em nuvem em diversos setores governamentais, conforme indicado pela Pesquisa Sobre o Uso das Tecnologias de Informação e Comunicação no Setor Público Brasileiro (NIC.BR, 2022), algumas organizações não realizaram a migração total ou parcial para a nuvem.

Um fenômeno recente observado é o movimento conhecido como "repatriação de nuvem", no qual empresas optam por transferir suas cargas de trabalho da nuvem de volta para suas infraestruturas locais (on-premises). Estudos recentes apontam uma tendência crescente desta repatriação. De acordo com a 451 Research[1], mais de dois terços das empresas que migraram para a nuvem estão retornando para sistemas on-premise, impulsionadas por fatores como custos elevados, necessidades de armazenamento, políticas internas e maior controle (MURUGESAN, 2024).

Adicionalmente, uma pesquisa da Citrix[2] com 350 líderes de negócios e TI dos Estados Unidos revelou que 93% das empresas participaram de projetos de repatriação de nuvem nos últimos três anos, com 25% realocando metade ou mais de suas cargas de trabalho. A principal motivação foi o custo, com 43% dos entrevistados relatando que a nuvem se mostrou mais cara do que o previsto inicialmente(LINTHICUM, 2024). Neste contexto, muitos fornecedores de nuvem oferecem o escalonamento automático como um recurso comumente desejado(RADHIKA; Sudha Sadasivam, 2021). No entanto, para infraestruturas locais, as organizações que precisam desse recurso devem implementá-lo por conta própria.

A escalabilidade é uma característica essencial para aplicações e serviços na internet(PATIBANDLA, 2012). Em um mundo cada vez mais conectado, as demandas por esses sistemas estão crescendo rapidamente. A demanda por recursos em aplicações e serviços na internet pode variar de forma significativa, devido a uma variedade de fatores, incluindo eventos sazonais, lançamentos de novos produtos ou serviços, divulgação de

[1] https://451research.com/
[2] https://www.citrix.com/

informações e acontecimentos inesperados.

Essas variações na demanda, quando decorrentes de picos de alto número de acessos, podem causar problemas de desempenho, como lentidão ou indisponibilidade, perda de receita e clientes insatisfeitos. Para garantir a disponibilidade e o desempenho destes sistemas frente ao crescimento da demanda, esses sistemas precisam ser capazes de ampliar sua capacidade de lidar com maior quantidade de dados(REIS; HOUSLEY, 2023).

O escalonamento automático permite que as aplicações ajustem automaticamente sua capacidade de recursos de acordo com a demanda. Esta característica não só garante a conformidade com os acordos de nível de serviço (SLAs), mas também contribui para a redução dos custos operacionais, uma vez que os recursos são provisionados de forma mais eficiente, o que minimiza a diferença entre os recursos provisionados e consumidos(POZDNIAKOVA, 2018).

Estratégias de escalonamento automático podem ser reativas, ajustando-se à demanda atual, ou proativas, prevendo demandas futuras. Nas abordagens reativas, o tempo de reação pode ser insuficiente, resultando em sobrecargas antes da alocação adequada de recursos (SILVA, 2021), principalmente em situações de picos repentinos de tráfego (NUNES, 2021). As estratégias proativas procuram evitar a deterioração do sistema tentando se antecipar à disponibilização de recursos antes da sobrecarga (NUNES, 2021).

Enquanto a nuvem oferece uma plataforma escalável para implantar aplicações modernas, como aquelas que usam serviços serverless e contêineres (LINTHICUM, 2024), os custos associados à migração de sistemas legados frequentemente não são vantajosos para as organizações, dado o alto custo de manter um sistema que não foi concebido de acordo com os padrões de design para aplicações nativas da nuvem.

A necessidade de cumprir acordos de nível de serviço e reduzir custos não se restringe apenas a sistemas modernos baseados em nuvem, mas também a implantações de aplicações tradicionais e legadas em ambientes on-premises. A implementação de uma metodologia de escalonamento automático proativo para esses sistemas pode contribuir significativamente para alcançar esses objetivos, além de reduzir consideravelmente a carga de trabalho da equipe de operações quanto à necessidade de intervenções manuais, cálculos, estimativas e monitoramento constante da carga de trabalho.

Os logs de acesso dos servidores web que hospedam esses sistemas e aplicações podem ser considerados a principal fonte de informação para a construção de modelos de previsão de carga de trabalho para subsidiar metodologias proativas de escalonamento automático, pois esses registros contêm informações históricas completas e precisas sobre o comportamento de navegação dos usuários da aplicação (HUSIN, 2013).

Estes dados históricos, também chamados de séries temporais, são coleções de observações realizadas sequencialmente no tempo(CHATFIELD, 2016), onde cada ponto

de dados está associado a uma marcação de tempo específica. Esses pontos de dados podem representar observações em intervalos regulares ou irregulares, como segundos, minutos, horas, dias, semanas, meses ou anos. Existem vários modelos de previsão de séries temporais para realizar as previsões de carga de trabalho com base no histórico de dados de monitoramento. No entanto, é desafiador determinar qual é o melhor modelo de previsão de séries temporais a ser utilizado em cada caso(MESSIAS, 2016).

Neste contexto, a análise de séries temporais pode ser utilizada como base da metodologia de criação do sistema de predição de carga de trabalho, procurando compreender o mecanismo gerador da série e determinando o tipo de modelo mais adequado a ser utilizado.

Estes desafios tornam-se especialmente relevantes em aplicações voltadas para o público, que enfrentam variações intensas na demanda, podendo sofrer quedas de desempenho significativas durante picos repentinos de carga. A performance da aplicação pode deteriorar rapidamente caso a estratégia de dimensionamento de recursos ou os métodos de escalonamento automático não consigam lidar com as variações eficientemente, podendo resultar em violações nos tempos de resposta estabelecidos em Acordos de Nível de Serviço (SLA). Além disso, quando o serviço não está em uso, é possível reduzir os custos desativando parte desnecessária do provisionamento de recursos.

## 1.2 Objetivo

Diante desse contexto, o objetivo deste trabalho é desenvolver uma abordagem para incorporar autoescalonamento autonômico proativo em aplicações legadas on-premises. Conforme destacado por (MESSIAS, 2016), o principal problema das técnicas de autoescalonamento reativas é o tempo de reação, pois essas técnicas frequentemente atuam apenas após a deterioração do sistema, devido ao atraso entre a solicitação e a disponibilização de um recurso, como uma máquina virtual, por exemplo. Além disso, (LORIDO-BOTRáN, 2014) aponta que sistemas reativos enfrentam dificuldades para lidar com aumentos abruptos no tráfego, como ocorre em promoções especiais ou no efeito Slashdot.

A abordagem se baseia na análise de séries temporais, provenientes de dados reais, para entender os padrões de carga de trabalho do sistema e construir um modelo capaz de prever essa carga. Diante da previsão, o sistema pode melhorar a eficiência operacional e reduzir o uso desnecessário de recursos, ajustando o provisionamento de forma inteligente de acordo com o comportamento da carga.

O modelo será treinado a partir de uma base de dados real e privada de requisições HTTP do Sistema de Gestão dos Processos Seletivos da COPEVE/CEFET-MG. A análise de séries temporais permitirá identificar características essenciais da carga, como sazonalidade, tendência e estacionariedade, que são fatores importantes para definir o

modelo preditivo mais adequado.

A análise de séries temporais também inclui a investigação de padrões não lineares, uma vez que, com base na experiência histórica do sistema, há evidências de picos repentinos de carga, especialmente em eventos críticos como inscrições e divulgações de resultados. Esses picos indicam a presença de padrões complexos que podem se diferenciar dos comportamentos lineares típicos de algumas séries temporais, tornando essencial a utilização de técnicas que possam capturar essa variabilidade e imprevisibilidade. Conforme (TSAY; CHEN, 2018) modelos lineares são mais fáceis de usar e podem fornecer boas aproximações em muitas aplicações, mas, por outro lado, séries temporais empíricas provavelmente são não lineares.

São avaliados modelos tradicionais de previsão estatística, como ARIMA (AutoRegressive Integrated Moving Average) e ETS (Error Trend and Seasonality ou exponential smoothing) e a contribuição do LSTM (Long Short-Term Memory) como técnica avançada de machine learning/deep learning aplicada neste contexto. A viabilidade dos modelos em fornecer previsões rápidas e precisas em tempo real será avaliada, considerando a necessidade de otimização de recursos e a manutenção da qualidade de serviço. Essa avaliação será fundamental para determinar se as técnicas de machine learning e deep learning são adequadas não apenas para prever a carga de trabalho, mas também para se integrarem eficientemente em um sistema de autoescalonamento que opere de maneira eficiente quando necessárias previsões periódicas em tempo real.

Este trabalho propõe a incorporação de autoescalonamento fundamentado no ciclo MAPE-K (Monitor, Analyze, Plan, Execute - Knowledge) da computação autonômica, adaptado para atender às demandas específicas do ambiente estudado. O sistema opera com base nas seguintes etapas:

- **Monitor (Monitorar):** realiza a coleta contínua de métricas do sistema de produção, como carga de trabalho, tempo de resposta e uso de recursos.
- **Analyze (Analisar):** aplica modelos preditivos aos dados coletados para antecipar variações na demanda de recursos.
- **Plan (Planejar):** define políticas de escalonamento com base nos dados das etapas anteriores.
- **Execute (Executar):** realiza o provisionamento ou desprovisionamento de recursos conforme planejado.
- **Knowledge (Conhecimento):** mantém uma base histórica de informações sobre métricas, previsões e políticas geradas.

Embora o sistema de autoescalonamento não tenha sido instalado permanentemente na infraestrutura do data center objeto do estudo de caso, sua avaliação foi realizada com acesso direto ao ambiente de produção do sistema da COPEVE através de conexão VPN segura. Os testes foram conduzidos com privilégios administrativos que permitiram ao sistema de autoescalonamento atuar diretamente sobre os recursos do ambiente de produção. A validação incluiu tanto cargas de trabalho reais quanto cargas sintéticas controladas, permitindo avaliar a eficácia das políticas propostas na otimização de recursos e manutenção dos níveis de serviço.

Portanto, os esforços deste trabalho concentraram-se em responder às seguintes questões de pesquisa (QP):

- **QP1: Qual o impacto da granularidade temporal na eficácia dos modelos preditivos?** Esta questão investiga como a resolução das séries temporais afeta a precisão e a eficiência dos modelos preditivos usados para autoescala de sistemas legados voltados para o público. Diversos trabalhos utilizaram diferentes resoluções para criar esses modelos, mas apenas (MESSIAS, 2016) justificou sua escolha. O autor usou intervalos de 1 segundo agregados em valores máximos por hora, pois o período mínimo de cobrança de provedores de infraestrutura em nuvem é geralmente é de uma hora, visando a otimização de custos. Outros trabalhos, não apresentaram uma justificativa clara para a escolha: (KUMAR; SINGH, 2018), utilizaram resolução de 1 minuto, enquanto (SINGH, 2018) e (KUMAR K. GANGADHARA RAO, 2021) optaram por 10 minutos. Já (SINGH, 2021) reamostrou os dados em intervalos de 1 minuto.

  Em data centers locais, como não existem restrições de cobrança quanto a intervalos de reconfiguração, investigar a resolução temporal mais adequada pode resultar em melhorias no desempenho dos modelos de previsão.

- **Qual o impacto do LSTM para previsão de cargas de trabalho baseadas em quantidade de requisições?** Esta questão investiga se o uso de técnicas de machine learning e deep learning, como o LSTM, pode melhorar a precisão dos modelos preditivos, especialmente ao lidar com dados que apresentam características de não linearidade, como por exemplo, “mudanças repentinas, variância (condicional) evoluindo no tempo (volatilidade) e irreversibilidade no tempo” (MORETTIN; TOLOI, 2018). - um comportamento típico de sistemas voltados ao público.

  (JANARDHANAN; BARRETT, 2017) aponta que modelos LSTM tiveram um desempenho mais consistente devido à sua capacidade de aprender dados não lineares muito melhor do que os modelos ARIMA no contexto de previsão de carga de CPU, pois em essência, o LSTM é um modelo de série temporal não linear.

Além disso, avalia-se a capacidade desses modelos de gerar previsões com boa aproximação e em tempo hábil para serem aplicadas em cenários onde a previsão deve ocorrer em tempo real, como em sistemas de escalonamento automático.

- **QP 3: Qual o impacto do sistema de autoescalonamento autonômico proativo?** Nesta questão de pesquisa, é investigado se as escolhas fundamentais do trabalho, como o uso de modelos baseados em LSTM e de séries temporais de quantidade de requisições é suficiente para construir uma metodologia eficiente de autoescala de aplicações. A quantidade de requisições HTTP reflete diretamente o volume de clientes conectados e a demanda sobre os recursos computacionais. A simplificação do sistema de escalonamento com base nesta métrica pode facilitar a implementação de sistemas de autoescalonamento proativos em data centers locais que não contam com os recursos de autoescalonamento nativos como ocorre em grandes provedores de nuvem.

## 1.3 Organização do Texto

Este trabalho está organizado em sete capítulos. O segundo capítulo apresenta o referencial teórico no qual o projeto se fundamenta, abordando conceitos como autoescalonamento, computação autonômica, análise de séries temporais, redes neurais recorrentes e LSTM. O terceiro capítulo detalha a metodologia de previsão de carga baseada em LSTM, incluindo bases de dados, pré-processamento, análise temporal, otimização e avaliação do modelo. O quarto capítulo descreve o sistema de autoescalonamento autonômico proativo, seus módulos e resultados do estudo de caso em que foi aplicado. O quinto capítulo apresenta os trabalhos relacionados. O sexto capítulo indica direções para trabalhos futuros e o sétimo capítulo traz as conclusões do trabalho.

# 2 Fundamentação Teórica

## 2.1 Auto Escalonamento

O autoescalonamento é uma técnica que permite que sistemas se ajustem automaticamente à demanda, adicionando ou removendo recursos conforme necessário (ERL, 2013).

### 2.1.1 Tipos de Auto Escalonamento

O tipo de escalonamento define como o auto-escalonador decide o método para provisionar recursos e qual combinação de recursos é provisionada para a aplicação. Dependendo do ambiente específico, o escalonamento pode ser realizado verticalmente, horizontalmente ou em ambos (IMDOUKH, 2020)

#### 2.1.1.1 Vertical

O autoescalonamento vertical é geralmente usado quando uma mesma instância precisa de uma quantidade maior de recursos por um curto período, para atender às demandas do serviço . Isso envolve ajustar a quantidade de CPU, memória e armazenamento alocados para um servidor físico, uma máquina virtual ou contêiner existente. O Kubernetes oferece um mecanismo de autoescalonamento vertical chamado VPA (Vertical Pod Autoscaler). Ele ajusta automaticamente os recursos solicitados com base no uso real, garantindo que cada pod tenha os recursos necessários, sem desperdício. (NUNES, 2021)

#### 2.1.1.2 Horizontal

No contexto do escalonamento horizontal, que é predominantemente utilizado em ambientes de computação em nuvem em produção, instâncias ou réplicas de um mesmo serviço são adicionadas ou removidas conforme a demanda. É geralmente usado quando há um aumento ou diminuição no número de solicitações de serviço, geralmente desencadeado por uma demanda maior ou menor de um determinado serviço. Uma aplicação IoT, por exemplo, pode ter sua demanda aumentada após a entrada de um ou mais dispositivos conectados simultaneamente. Nesse caso, sem autoescalonamento horizontal, o serviço pode ficar sobrecarregado com o número de solicitações, reduzindo o desempenho ou até causando instabilidade do serviço. Uma maneira de resolver esse problema sem autoescalonamento horizontal seria estimar o número de réplicas necessárias para satisfazer o pico de carga máximo, sempre adicionando recursos suficientes para satisfazer essa condição. Nesse caso, ocorrerá desperdício de recursos, pois o serviço não estará utili-

zando o recurso alocado o tempo todo. Em um ambiente de nuvem, pode ser caro, pois cobra pelo volume de recursos utilizados. O Kubernetes também oferece um mecanismo de autoescalonamento horizontal chamado HPA (Horizontal Pod Autoscaler) e depende da configuração manual de alguns valores de limite, como a utilização alvo da CPU, o número mínimo e máximo de pods (NUNES, 2021).

#### 2.1.1.3 Híbrido

Técnicas de escalonamento híbrido colhem os benefícios do controle fino de recursos do escalonamento vertical e da alta disponibilidade do escalonamento horizontal. Isso torna o escalonamento híbrido uma solução promissora para autoescalonamento eficaz (KWAN, 2019).

### 2.1.2 Estratégias de Auto escalonamento

Outro fator importante no auto-escalonamento de microsserviços é o modo de escalonamento, que pode ser reativo ou proativo. Essa definição foi proposta por Lorido-Botran et al. (LORIDO-BOTRáN, 2014) a partir da perspectiva de auto-escalonamento de Infraestrutura como Serviço - IaaS em nuvem (NUNES, 2021).

### 2.1.3 Reativas

São estratégias de autoescalonamento que respondem exclusivamente ao estado atual do sistema. Sistemas reativos podem enfrentar dificuldades em lidar com aumentos abruptos no tráfego, como em promoções especiais ou o efeito Slashdot (LORIDO-BOTRáN, 2014).

### 2.1.4 Pró-ativas

São estratégias de autoescalonamento que adotam técnicas avançadas para antecipar demandas futuras e realizar o provisionamento de recursos com antecedência. A antecipação é essencial devido ao inevitável atraso entre a execução de uma ação de autoescalabilidade, como adicionar um servidor, e sua efetivação – um processo que pode levar vários minutos, desde a alocação de um servidor físico até a inicialização do sistema operacional e da aplicação (LORIDO-BOTRáN, 2014).

## 2.2 Computação Autonômica

A computação autonômica, introduzida por Paul Horn da IBM em 2001, estabelece um paralelo com os sistemas biológicos autorregulados, especialmente o sistema nervoso

autônomo que regula funções vitais sem necessidade de controle consciente (KEPHART; CHESS, 2003).

A essência da computação autonômica reside na capacidade de autogestão, objetivando liberar os administradores de sistema das complexidades operacionais cotidianas, enquanto mantém a performance otimizada continuamente. Os sistemas autonômicos são projetados para se adaptarem dinamicamente a mudanças em componentes, cargas de trabalho, demandas e condições externas, além de responderem efetivamente a falhas de hardware ou software, sejam essas acidentais ou maliciosas (KEPHART; CHESS, 2003).

Sistemas de software autonômicos incorporam controles que podem ser divididos em quatro áreas funcionais de auto-gerenciamento (IBM, 2005):

- **Autoconfiguração (Self-Configuration)** Implementa a configuração automática seguindo políticas de alto nível. A incorporação de novos componentes ocorre de maneira transparente e natural ao sistema, com adaptação dinâmica dos componentes existentes.
- **Auto-otimização (Self-Optimization)** Realiza monitoramento contínuo para identificar oportunidades de otimização de desempenho e custos. O sistema experimenta e ajusta parâmetros autonomamente, desenvolvendo capacidade de decisão sobre gestão de recursos e funções.
- **Autorrecuperação (Self-Healing)** Engloba mecanismos sofisticados para detecção, diagnóstico e correção de falhas. Utiliza análise de logs e monitoramento proativo para identificar problemas, aplicando correções e patches automaticamente quando necessário.
- **Autoproteção (Self-Protection)** Implementa defesa proativa em duas dimensões: proteção contra ataques maliciosos e falhas em cascata, e antecipação de problemas através de análise preditiva, estabelecendo medidas preventivas e mitigadoras.

Para implementar essas capacidades de autogerenciamento, os sistemas autonômicos utilizam um mecanismo de controle baseado em feedback. O gerenciamento autonômico implementa um loop de controle inteligente através de funções que compartilham conhecimento entre si. Este loop deve possuir um método automatizado para coletar os detalhes necessários do sistema, analisar esses detalhes para determinar se algo precisa mudar, criar um plano ou sequência de ações que especifica as mudanças necessárias e executar essas ações. Esse loop é conhecido como ciclo de adaptação MAPE-K (Figura 1), que inclui os processos de Monitoramento (Monitor), Análise (Analyse), Planejamento (Plan) e uma Base de Conhecimento (Knowledge Base) compartilhada (NGUYEN, 2015).

- **Função de Monitoramento** Fornece os mecanismos que coletam, agregam, filtram e reportam detalhes (como métricas e topologias) coletados de um recurso gerenciado.
- **Função de Análise** Fornece os mecanismos que correlacionam e modelam situações complexas (por exemplo, previsão de séries temporais e modelos de filas). Esses mecanismos permitem que o gerenciador autonômico aprenda sobre o ambiente de TI e ajude a prever situações futuras.
- **Função de Planejamento** Fornece os mecanismos que constroem as ações necessárias para atingir metas e objetivos. O mecanismo de planejamento usa informações de política para guiar seu trabalho.
- **Função de Execução** Fornece os mecanismos que controlam a execução de um plano considerando atualizações dinâmicas.

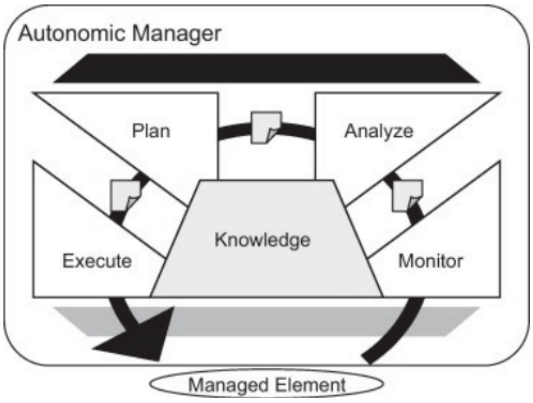


Figura 1 – Ciclo MAPE-K da Computação Autonômica. Fonte: (MOHALIK, 2017)

## 2.3 Análise de Séries Temporais

Uma série temporal é uma coleção de observações realizadas sequencialmente no tempo onde cada ponto de dados está associado a uma marcação de tempo específica. Esses pontos de dados podem representar observações em intervalos regulares ou irregulares, como segundos, minutos, horas, dias, semanas, meses ou anos. Exemplos de uso ocorrem em uma variedade de áreas, desde economia até engenharia e a análise de séries temporais constitui uma importante área da estatística (CHATFIELD, 2016).

Os valores que uma série temporal pode assumir são variados, podendo ser números discretos, reais, conjuntos de valores ou até mesmo letras. Geralmente, os pontos na série temporal são distribuídos igualmente ao longo do tempo. Uma série temporal é **univariada** se cada ponto da série contiver apenas uma variável. No entanto, se em cada instante de tempo houver mais de uma variável então a série temporal será categorizada

como **multivariada**. Uma série temporal geralmente é denotada por uma sequência de letras onde o subscrito denota o passo de tempo, com um índice maior indicando um ponto cronologicamente posterior: $X_1, X_2, X_3, \ldots, X_n$ (HABEN, 2023).

A Análise de Séries Temporais consiste em extrair informações relevantes e estatísticas significativas a partir de dados organizados cronologicamente. Essa abordagem visa tanto diagnosticar o comportamento passado como antecipar o comportamento futuro (NIELSEN, 2019).

### 2.3.1 Distribuição

Segundo (TUKEY, 1977), o Box plot ou Box-and-whisker plots é uma representação gráfica que consiste em uma caixa retangular que engloba o intervalo interquartil (IQR), com uma linha vertical representando a mediana. Além disso, são traçados "whisker" (ou "bigodes") representando os valores mínimo e máximo dos dados. Essa visualização fornece uma maneira concisa e intuitiva de identificar a dispersão, assimetria e possíveis outliers nos dados.

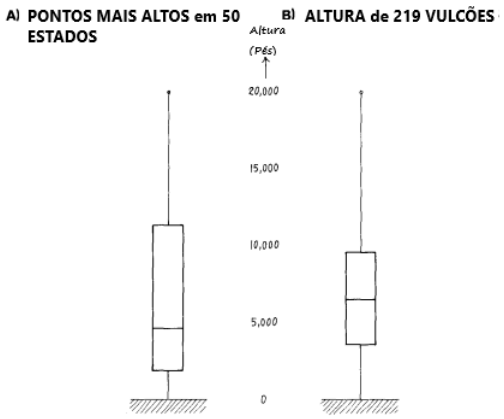


Figura 2 – Gráfico de box-plot. Adaptado de: (TUKEY, 1977)

### 2.3.2 Decomposição

A decomposição de séries temporais visa separar uma série temporal não estacionária $y_t$ em componentes determinísticos não estacionários (como tendência $T_t$ e sazonalidade $S_t$) e um componente estocástico residual $R_t$, permitindo assim análises mais detalhadas e previsões. Os componentes determinísticos são previsíveis e contribuem para a tarefa de previsão por meio de suas estimativas ou por extrapolação. Melhorar a precisão da previsão requer também uma análise do componente estocástico restante (DAMA; SINOQUET, 2021)

Estes componentes podem ser somados para se realizar uma decomposição **aditiva**:

$$y_t = S_t + T_t + R_t$$

Ou os componentes podem ser multiplicados ao se fazer uma decomposição **multiplicativa**:

$$y_t = S_t * T_t * R_t$$

A decomposição aditiva é a mais adequada quando a magnitude das flutuações sazonais, ou a variação em torno do ciclo de tendência, não varia com o nível da série temporal. Por outro lado, se a variação no padrão sazonal, ou a flutuação em torno do ciclo de tendência, parecer ser proporcional ao nível da série temporal, então a decomposição multiplicativa é mais apropriada. (HYNDMAN; ATHANASOPOULOS, 2021).

#### 2.3.2.1 Tendência

A tendência representa a direção geral dos dados ao longo do tempo, indicando se a série está crescendo, decrescendo ou mantendo-se estável. A tendência são alterações macroscópicas gerais nos dados, sendo a mais comum uma tendência linear, onde existe um crescimento linear e gradual na série temporal (HABEN, 2023). Por exemplo, a série temporal que representa as vendas de um produto pode apresentar um crescimento ao longo do tempo, indicando a expansão do mercado.

#### 2.3.2.2 Sazonalidade

A sazonalidade refere-se a padrões recorrentes e periódicos que ocorrem em uma série temporal em intervalos regulares, podendo ser diários, semanais, mensais ou em qualquer período definido. Frequentemente, são chamados de padrões cíclicos (HABEN, 2023). Por exemplo, o número de vendas de um produto pode aumentar durante a temporada de férias ou o número de atendimentos em um hospital pode crescer durante a temporada de gripe.

#### 2.3.2.3 Componente residual

O componente residual representa variações aleatórias ou não explicadas pela tendência e sazonalidade, contendo informações que não podem ser atribuídas a nenhuma outra componente específica.

### 2.3.3 Estacionariedade

Uma série temporal é considerada estacionária quando suas propriedades estatísticas permanecem constantes ao longo do tempo. Existem dois tipos principais de estacionariedade:

- **Estacionariedade de primeira ordem**: A média e a variância da série temporal são constantes ao longo do tempo.

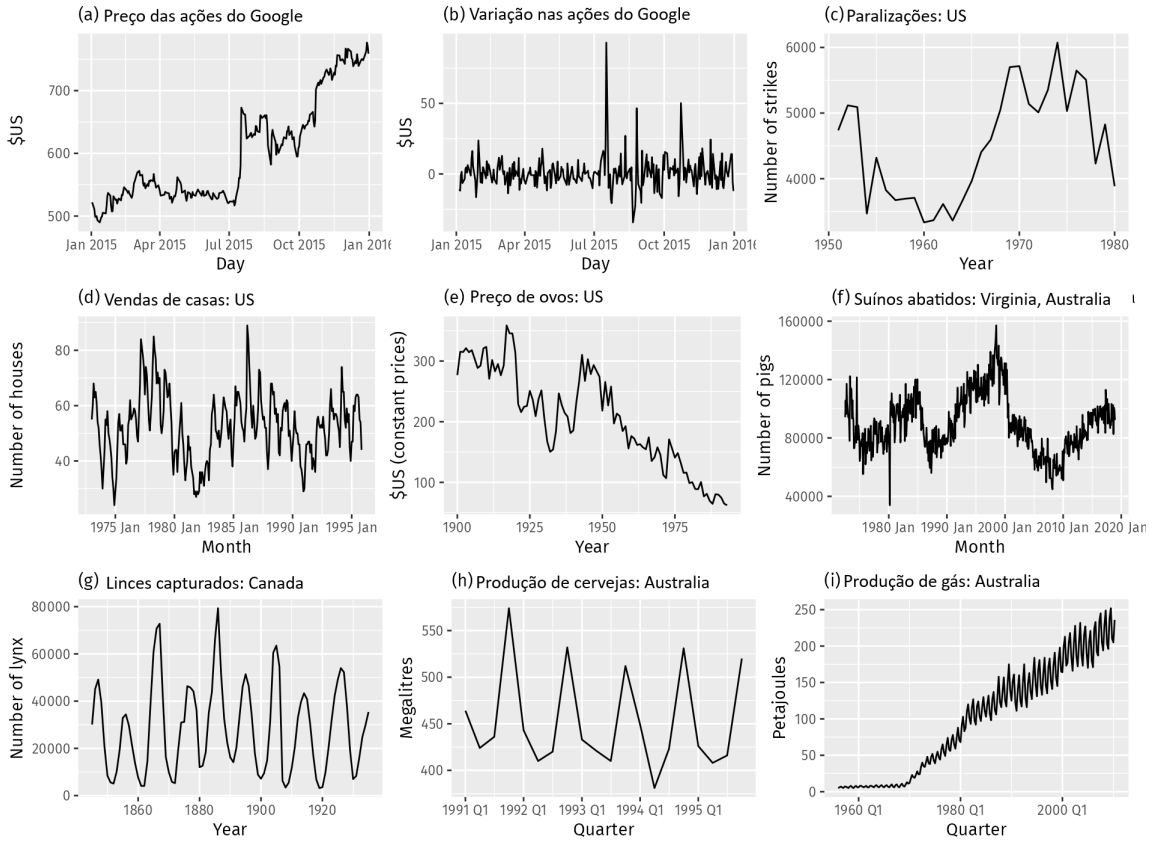


Figura 3 – Quais séries são estacionárias?. Fonte: HYNDMAN, ATHANASOPOULOS 2021

- **Estacionariedade de segunda ordem**: Além da média e da variância, a autocovariância da série temporal também é constante ao longo do tempo.

A interpretação de séries temporais pode ser complexa, especialmente quando há comportamento cíclico. Uma série é estacionária se possui ciclos, mas sem tendência ou sazonalidade, resultando em uma previsão incerta de onde ocorrem picos ou quedas. Geralmente, séries temporais estacionárias não apresentam padrões previsíveis a longo prazo, exibindo gráficos temporais com uma tendência horizontal aproximada e variância constante, apesar da possível presença de comportamento cíclico (HYNDMAN; ATHANASOPOULOS, 2021).

A Figura 3 mostra exemplos de séries temporais estacionárias e não estacionárias. As séries (d), (h) e (i) possuem sazonalidades óbvias e, portanto, não são estacionárias; as séries (a), (c), (e), (f) e (i) possuem tendência e mudança de níveis, sendo também não estacionárias; as séries (b) e (g) são estacionárias, sendo que os fortes ciclos nesta última são aperiódicos, não sendo previsível, no longo prazo, o momento destes ciclos.

Séries temporais estacionárias são mais fáceis de analisar e prever em comparação com séries temporais não estacionárias. Isso ocorre porque as propriedades estatísticas permanecem inalteradas, permitindo a aplicação consistente de modelos matemáticos ao

longo do tempo.

A estacionariedade pode ser verificada visualmente, por meio de um correlograma que representa a Função de Autocorrelação (ACF).

$$\rho(h) = \frac{\text{Cov}(X_t, X_{t+h})}{\sqrt{\text{Var}(X_t)\text{Var}(X_{t+h})}} \quad \text{(Função de Autocorrelação)}, \tag{2.1}$$

No correlograma, observa-se uma rápida diminuição até zero para processos estacionários, enquanto para processos não estacionários, o decaimento é mais lento. (DAMA; SINOQUET, 2021). A figura 4 demonstra este comportamento.

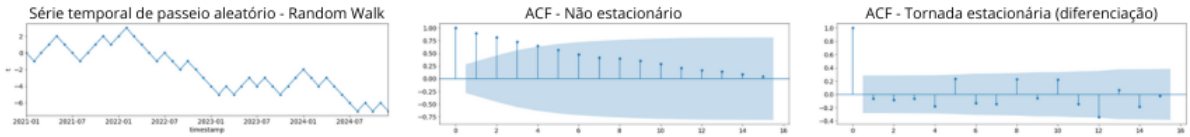


Figura 4 – Correlogramas da série temporal de passeio aleatório e de sua diferenciação

Entretanto, a análise visual por si só pode não ser suficiente para concluir sobre a estacionaridade da série. Para confirmar essa característica, é comum recorrer a testes estatísticos de raiz unitária. O teste de Kwiatkowski-Phillips-Schmidt-Shin (KPSS) (KWIATKOWSKI, 1992) verifica a hipótese nula de que a série é estacionária em torno de uma tendência determinística, testando contra a alternativa de não estacionariedade (passeio aleatório). Já o teste de Dickey-Fuller Aumentado (ADF) (DICKEY; FULLER, 1979) avalia a estacionaridade em torno de uma tendência linear, assumindo uma autocorrelação de ordem $p$ (DAMA; SINOQUET, 2021).

O resultado do teste ADF inclui a estatística do teste ADF, o $p - valor$ e valores críticos para diferentes níveis de significância.

As hipóteses para o teste de Dickey-Fuller aumentado (ADF) são:

- Hipótese nula (H0): A série temporal não é estacionária porque existe uma raiz unitária (se $p - valor > 0,05$)
- Hipótese alternativa (H1): A série temporal é estacionária porque não há raiz unitária (se $p - valor \leq 0,05$)

A série temporal é estacionária se pudermos rejeitar a hipótese nula do teste ADF:

- Se o $p - valor \leq 0,05$
- Se a estatística de teste for mais extrema que os valores críticos

As hipóteses para o teste de Kwiatkowski-Phillips-Schmidt-Shin (KPSS) são:

- Hipótese nula (H0): A série temporal é estacionária porque não há raiz unitária (se o $p - valor > 0,05$)
- Hipótese alternativa (H1): A série temporal não é estacionária porque há uma raiz unitária (se o $p - valor \leq 0,05$)

Quanto mais positiva a estatística, maior a probabilidade de rejeitarmos a hipótese nula (temos uma série temporal não estacionária). A série temporal é estacionária se não rejeitarmos a hipótese nula do teste KPSS:

- Se o $p - valor > 0,05$
- Se a estatística de teste for menos extrema que os valores críticos

#### 2.3.3.1 Diferenciação

A técnica de diferenciação é frequentemente empregada para eliminar tendências e sazonalidade em séries temporais, tornando-as estacionárias. Essa abordagem torna a análise da série mais apropriada, aprimora o desempenho de modelos de previsão temporal e facilita a detecção de mudanças estruturais.

A diferenciação em séries temporais é o processo de subtrair valores sucessivos de uma série temporal e pode ser calculada através desta fórmula:

$$y'_t = y_t - y_{t-1}$$

Essa fórmula expressa a primeira diferença de uma série temporal, onde $y'_t$ é a primeira diferença no tempo $t$, $y_t$ é o valor da série temporal no tempo $t$, e $y_{t-1}$ é o valor da série temporal no tempo $t-1$. Ocasionalmente, os dados diferenciados não parecerão estacionários, podendo ser necessário diferenciar os dados uma segunda vez para obter uma série estacionária; a este procedimento é dado o nome de diferenciação de segunda ordem (HYNDMAN; ATHANASOPOULOS, 2021)

### 2.3.4 Autocorrelação

A autocorrelação descreve como as mudanças na série temporal em um ponto se relacionam com a série temporal em pontos defasados (ou mais antigos) da série temporal, ou seja, é uma medida da relação entre os valores de uma série temporal com seus valores passados. Encontrar correlações nos dados é uma parte importante da identificação de quais valores históricos podem ser importantes para estimar pontos futuros (HABEN, 2023).

A autocorrelação é calculada usando a seguinte fórmula:

$$r(k) = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2} \tag{2.2}$$

onde:

$r(k)$ é a autocorrelação de ordem $k$, $T$ é o comprimento da série temporal, $y_t$ é o valor da série temporal no tempo $t$ e $\bar{y}$ é a média da série temporal. Portanto, os coeficientes de autocorrelação, por exemplo, $r(1)$ mede a relação entre $yt$ e $yt - 1$, $r(2)$ mede a relação entre $yt$ e $yt - 2$, e assim por diante. Os coeficientes de autocorrelação compõem a função de autocorrelação ou ACF - Auto Correlation Function (HYNDMAN; ATHANASOPOULOS, 2021).

A autocorrelação pode ser usada para identificar tendências e sazonalidade em séries temporais. Quando os dados têm uma tendência, as autocorrelações para pequenas defasagens tendem a ser grandes e positivas; quando os dados são sazonais, as autocorrelações serão maiores para as defasagens sazonais (em múltiplos do período sazonal) do que para outras defasagens e quando os dados possuem tendência e sazonalidade, há uma combinação dos dois efeitos. (HYNDMAN; ATHANASOPOULOS, 2021).

A figura 5 representa essas características por meio do gráfico da função de autocorrelação da série temporal das vendas mensais de medicamentos antidiabéticos na Austrália, as linhas azuis tracejadas indicam se as correlações são significativamente diferentes de zero:

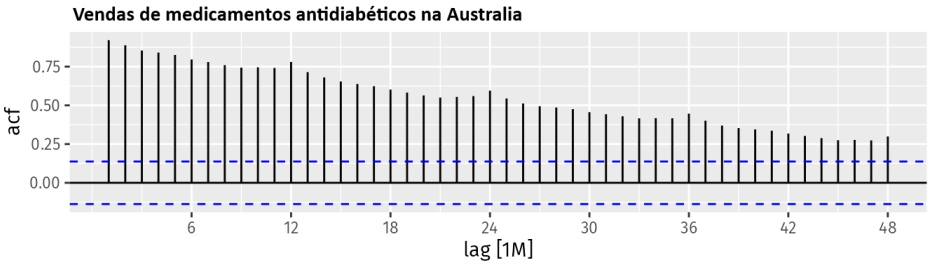


Figura 5 – Gráficos de Função de Autocorrelação - Vendas mensais de medicamentos antidiabéticos na Austrália. Fonte: HYNDMAN, ATHANASOPOULOS 2021

### 2.3.5 Complexidade e Previsibilidade de Series Temporais

A entropia é uma medida da desordem ou complexidade de um sistema. Em análise de séries temporais, a entropia pode ser usada para medir a complexidade de uma série temporal. Os métodos de entropia mais comumente usados para séries temporais são a Entropia de Amostra (Sample Entropy) e a Entropia Aproximada (Approximate Entropy).

Baseada na entropia aproximada, a entropia de amostra apresenta maior robustez, é menos dependente do comprimento das séries temporais e consegue manter uma

consistência eficaz em diferentes configurações de parâmetros. Isso permite uma distinção eficiente da complexidade entre diferentes séries temporais (WANG; ZHAO, 2019).

A Entropia de Amostra pode ser calculada da seguinte forma:

1. Transformação da serie temporal original $x_i$ no vetor $X(i)$ através da seguinte fórmula:
$$X(i) = [x_i, x_{i+1}, \ldots, x_{i+(m-1)}], \quad i = 1, 2, \ldots, N - m + 1 \tag{2.3}$$
onde, m é a dimensão de incorporação e N é o comprimento da série temporal.

2. para $i \neq j$ , calcule a distância máxima entre $X(i)$ e $X(j)$ , conforme a fórmula:
$$d_{ij} = \max(x_{i+k} - x_{j+k}), k = 0, 1, 2, \ldots, m - 1 \tag{2.4}$$

3. Dado o valor de tolerância $r$, para cada $i$, conte o número $num(d_{ij} < r)$ e calcule seu quociente $B_i^m(r)$ com o número total do vetor conforme a formula:
$$B_i^m(r) = num(d_{ij} < r)/(N - m), 1 \leq j \leq N - m, j \neq i \tag{2.5}$$

4. $B^m(r)$ pode ser calculado da seguinte forma:
$$B^m(r) = \frac{1}{N - m + 1} \sum_{i=1}^{N-m+1} B_i^m(r) \tag{2.6}$$

5. $m = m + 1$, repita as etapas 1-4 e calcule $B^{m+1}(r)$.

6. A Entropia de Amostra $SE(m, r)$ é calculada de acordo com a fórmula:
$$SE\,(m, r) = \lim_{N \to \infty} [-\ln \left( B^{m+1}\,(r)\,/B^m\,(r) \right)] \tag{2.7}$$
podendo ser aproximada para aplicações práticas de acordo com esta fórmula:
$$SE\,(m, r, N) = -\ln[B^{m+1}\,(r)\,/B^m\,(r)] \tag{2.8}$$

### 2.3.6 Modelagem e Previsão de Séries Temporais

A modelagem de séries temporais apresenta uma variedade de abordagens para previsão, sendo que a escolha do modelo mais adequado para cada caso constitui um desafio significativo (MESSIAS, 2016). Esta seleção depende principalmente de testes empíricos, já que não existe uma regra única. É preciso balancear dois fatores: a precisão das previsões e a complexidade do modelo, considerando sempre as características específicas dos dados (XUAN, 2022).

Nas últimas décadas, o campo diversificou significativamente seus métodos e abordagens. Os modelos estatísticos clássicos incluem ARIMA (Autoregressive Integrated Moving Average) (BOX, 2015), ETS (Error, Trend, Seasonality) (HYNDMAN, 2008), VAR (Vector Autoregression) (SIMS, 1980) entre outros, que mantêm-se fundamentais por seus pressupostos estatísticos bem definidos e alta interpretabilidade dos resultados.

Com o avanço da computação e o surgimento de grandes volumes de dados, técnicas de Machine Learning, desde a regressão linear (DRAPER; SMITH, 1998) até métodos mais complexos como Random Forests (BREIMAN, 2001), XGBoost (CHEN; GUESTRIN, 2016) e Support Vector Regression (SVR) (CORTES; VAPNIK, 1995), expandiram as possibilidades de análise, oferecendo capacidade adicional de capturar relações temporais complexas. O campo foi ainda enriquecido com arquiteturas de Deep Learning, começando com Redes Neurais Recorrentes (RNN) e suas variantes como LSTM (Long Short-Term Memory) (HOCHREITER; SCHMIDHUBER, 1997) e GRU (Gated Recurrent Unit) (CHO, 2014), que se destacam na modelagem de dependências temporais de longo prazo (HOCHREITER; SCHMIDHUBER, 1997). Mais recentemente, arquiteturas baseadas em mecanismos de atenção, como os Transformers (VASWANI, 2017), têm demonstrado resultados promissores ao processar eficientemente sequências longas, enquanto Redes Neurais Convolucionais (CNN) (LECUN, 1989), tradicionalmente projetadas para conjuntos de dados baseados em imagens, são adaptadas para lidar com séries temporais usando convoluções causais (LIM; ZOHREN, 2021).

Projetar uma previsão de série temporal para um caso específico geralmente envolve cinco etapas. A primeira etapa é o pré-processamento dos dados, transformando os dados brutos em uma forma adequada para o método de previsão. A segunda etapa é a engenharia de características, que busca extrair características ocultas das séries temporais ou identificar informações exógenas úteis para o método de previsão. Cada método de previsão tem hiperparâmetros que devem ser definidos, então a terceira etapa envolve a otimização desses hiperparâmetros para melhorar a precisão da previsão. Com os métodos de previsão candidatos e hiperparâmetros otimizados, a quarta etapa aborda a seleção do método mais adequado para garantir precisão. Finalmente, a quinta etapa busca aumentar a robustez da previsão por meio de um conjunto de previsões, agrupando múltiplas previsões de diferentes modelos para evitar resultados imprecisos (MEISENBACHER, 2022).

## 2.4 Redes Neurais Recorrentes - RNN

As Redes Neurais Recorrentes (RNNs) (RUMELHART, 1986) são uma classe de redes neurais artificiais inspirada na conectividade cíclica dos neurônios no cérebro, utilizam loops de função iterativos para armazenar informações, permitindo que uma “memó-

ria" das entradas anteriores persista no estado interno da rede, o que influencia a saída (GRAVES, 2012).

A redes neurais recorrentes são projetadas especificamente para lidar com dados sequenciais, onde a ordem dos elementos é importante. Dados como áudio, textos, DNA, sinais vitais e preços de ações da bolsa de valores são exemplos de dados sequenciais. Em outras palavras, uma rede neural recorrente é uma rede neural especializada em processar sequências de valores do tipo $x^{(1)}, x^{(2)}, \ldots, x^{(\tau)}$. As redes recorrentes possuem a capacidade de lidar com sequências consideravelmente mais extensas do que seria viável para redes que não contam com especialização baseada em sequência. Além disso, a maioria das redes recorrentes é capaz de processar sequências de comprimento variável (GOODFELLOW, 2016).

A figura 6 mostra a arquitetura de uma RNN:

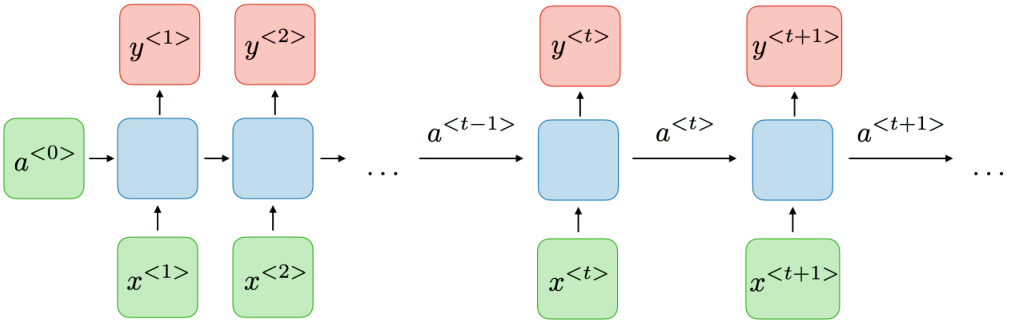


Figura 6 – Arquitetura de uma RNN tradicional. Adaptado de: https://stanford.edu/~shervine/

Para cada intervalo de tempo $t$, a ativação $a^{<t>}$ e a saída $y^{<t>}$ são expressas da seguinte forma:

$$a^{<t>} = g1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$
$$y^{<t>} = g2(W_{ya}a^{<t>} + b_y)$$

onde $W_{ax}$ representa a matriz de pesos para a entrada, $W_{aa}$ a matriz de pesos para atualização do estado interno, $W_{ya}$ a matriz de pesos para a saída, $b_a$ o bias de ativação do estado interno, $b_y$ o bias de saída e $g1$, $g2$ são funções de ativação.

A figura 7 mostra a estrutura interna de uma unidade RNN.

**Forward Pass**

O processo de treinamento de um modelo em uma Rede Neural Recorrente (RNN) inicia na etapa de **Forward Pass** ou passagem direta, esta etapa é análoga à de um Perceptron de Múltiplas Camadas (MLP) com uma única camada oculta. A diferença reside no fato de que as ativações da camada oculta são influenciadas tanto pela entrada externa atual quanto pelas ativações da camada oculta no passo de tempo anterior. Para

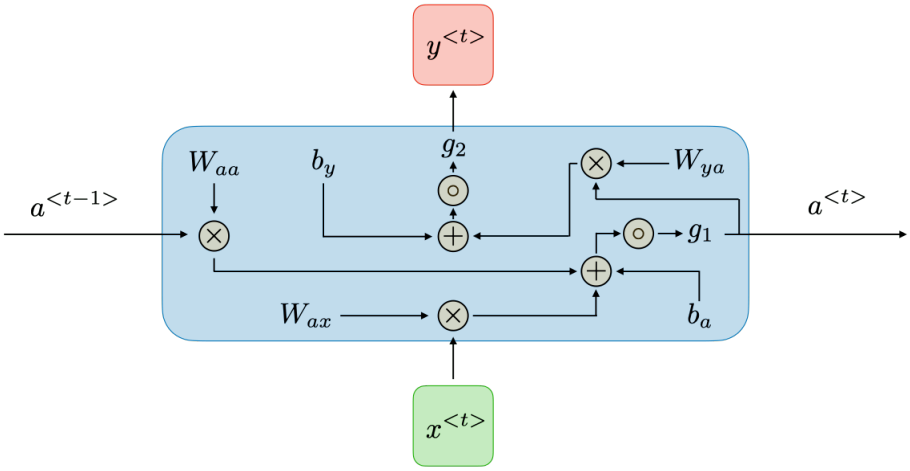


Figura 7 – Estrutura interna de uma unidade RNN. Adaptado de: https://stanford.edu/~shervine/

as unidades ocultas, aplicamos funções de ativação não lineares e diferenciáveis da mesma forma que em um MLP (GRAVES, 2012).

**Loss Function**

Após a etapa do Forward Pass, a função de perda (Loss Function) é calculada. A função de perda calcula as discrepâncias entre as saídas com os valores reais. O objetivo do treinamento é minimizar esta função de perda. No caso de uma rede neural recorrente, a função de perda $\mathcal{L}$ de todos os passos de tempo é definida com base na perda em cada passo de tempo da seguinte forma:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{t=1}^{T} \mathcal{L}(\hat{\mathbf{y}}_t, \mathbf{y}_t)$$

**Backward Pass**

Na etapa de Backward Pass, geralmente é utilizado o algoritmo **Backpropagation through time - BPTT** por sua simplicidade e eficiência computacional. O BPTT, semelhante à retro-propagação padrão, utiliza a regra da cadeia, com a diferença de que, para redes recorrentes, a função de perda depende não apenas da influência da camada oculta na saída, mas também da influência da camada oculta no próximo passo de tempo.

A retro-propagação é realizada em cada ponto no tempo. No passo de tempo $T$, a derivada da perda $\mathcal{L}$ em relação à matriz de pesos $\mathbf{W}$ é expressa da seguinte forma:

$$\frac{\partial \mathcal{L}^{(T)}}{\partial w} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial W}$$

Onde, T: Representa o número total de etapas de tempo na sua sequência; w: Representa um parâmetro específico na rede RNN, como o peso de uma conexão; $\mathcal{L}(T)$: Representa

a perda total da rede, calculada como a soma das perdas individuais em cada etapa de tempo (t) W: Representa uma matriz de pesos ou bias da rede.

No entanto, as RNNs tradicionais enfrentam um problema conhecido como o "problema do desvanecimento e explosão do gradiente". Isso ocorre porque, durante o treinamento, os gradientes que são propagados de volta na rede podem se tornar muito pequenos (desvanecimento) ou muito grandes (explosão), o que torna o aprendizado de dependências temporais de longo prazo extremamente difícil.

Devido a esse problema, as RNNs tradicionais são limitadas em sua capacidade de lembrar informações de sequências muito longas. Isso as torna inadequadas para tarefas que exigem uma compreensão de contexto de longo prazo, como tradução automática ou análise de texto longo.

## 2.5 Long Short Term Memory - LSTM

Long Short Term Memory - LSTM (HOCHREITER; SCHMIDHUBER, 1997) é um tipo de rede neural recorrente que se destaca por sua habilidade de aprender dependências de longo prazo em sequências de dados. O LSTM não é afetado pelo problema do desvanecimento ou explosão do gradiente, como ocorre nas RNNs tradicionais.

A arquitetura inicial do LSTM introduziu a ideia de CECs – "Carrosséis de Erro Constante" (CECs) dentro de unidades especiais chamadas células, criando auto-loops para produzir caminhos nos quais o gradiente pode fluir por longas durações. A ativação dos CECs é chamada de estado da célula e unidades de portas multiplicativas aprendem a abrir e fechar o acesso às células. Posteriormente, foi verificado que este primeiro modelo enfrentava dificuldades para aprender a processar corretamente séries temporais muito longas ou contínuas que não foram previamente segmentadas em subsequências de treinamento apropriadas, com inícios e finais claramente definidos. Um fluxo contínuo de entrada podia resultar no crescimento ilimitado dos valores internos das células. Foi então proposta por (GERS, 2000) uma solução: criação de "portões de esquecimento" adaptativos projetados para aprender a redefinir blocos de memória uma vez que seus conteúdos estão desatualizados e, portanto, inúteis.

O funcionamento do modelo LSTM é baseado em células de memória que armazenam informações e as atualizam de acordo com os dados de entrada e as decisões tomadas pelo próprio modelo. Cada célula é composta por três portões (gates) que controlam o fluxo de informação: o portão de entrada (input gate), o portão de esquecimento (forget gate) e o portão de saída (output gate). A figura 8 exibe estas estruturas.

**Portão do esquecimento**

O portão de esquecimento decide qual a informação deve ser descartada da célula

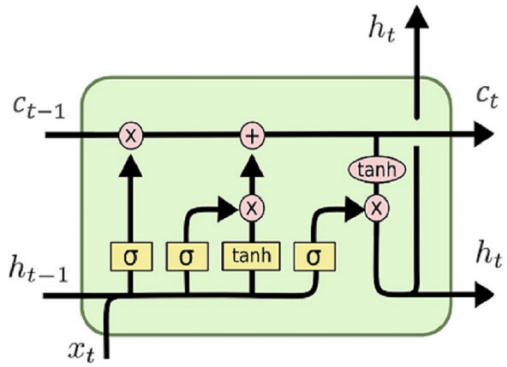


Figura 8 – Arquitetura da célula LSTM. Fonte: DOI: 10.7717/peerj-cs.1001/fig-1

de memória, permitindo que o modelo esqueça informações desnecessárias e mantenha apenas as informações mais relevantes. A fórmula a seguir descreve o cálculo:

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

$x_t$ é a entrada no tempo $t$, $h_{t-1}$ é a saída da célula anterior, e $\sigma$ é a função sigmoide. Se a saída da porta de esquecimento for 1, a informação é mantida no estado da célula. Em seguida, a função sigmoide cria um vetor contendo possíveis novos valores.

**Portão de entrada**

O portão de entrada decide qual informação será adicionada à célula de memória, com base na entrada atual e na informação armazenada anteriormente.

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

O novo vetor de valores candidatos ($C'_t$) é criado pela seguinte equação:

$$C'_t = \tanh(W_C[h_{t-1}, x_t] + b_c)$$

Agora, o antigo estado da célula $C_{t-1}$ é atualizado para o novo estado da célula ($C_t$):

$$C_t = f_t * C_{t-1} + i_t * C'_t$$

**Portão de saída**

Finalmente, decidimos a saída da rede, que é baseada no estado da célula. Primeiro, a camada sigmoide é utilizada para decidir quais partes do estado da célula serão utilizadas,

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

em seguida, a função tangente hiperbólica (tanh) é aplicada ao estado da célula, multiplicado por essa camada sigmoide:

$$h_t = o_t * \tanh(C_t)$$

## 2.6 Trabalhos Relacionados

### 2.6.1 Considerações iniciais

Existem vários modelos de previsão de séries temporais para calcular as previsões de carga de trabalho com base no histórico de dados de monitoramento. No entanto, é desafiador determinar qual é o melhor modelo de previsão de séries temporais a ser utilizado em cada caso(MESSIAS, 2016).

Nesse capítulo serão apresentados estudos que trabalharam com conjuntos de dados específicos de carga de trabalho por requisições HTTP e também bases reais que estão disponíveis publicamente na Internet. Além disso, algumas técnicas utilizadas neste trabalho foram baseadas em trabalhos apresentados neste capítulo.

### 2.6.2 Trabalhos

Messias et al (MESSIAS, 2016) afirmam que a maioria dos estudos anteriores sobre métodos de previsão focam apenas em métodos individuais para avaliar o desempenho das previsões e defendem que a solução mais eficaz pode não ser um modelo isolado, mas sim uma combinação de modelos. Citam também o problema do tempo de reação nas técnicas reativas para autoescalonamento, que frequentemente reagem após a deterioração do sistema devido ao atraso entre solicitação de um recurso (máquina virtual) e sua disponibilização. Já as técnicas proativas, a análise de séries temporais com base em modelos estatísticos clássicos não oferece um que é o melhor em todos os casos.

No trabalho, apresentam uma abordagem de previsão adaptativa que utiliza algoritmos genéticos para combinar diferentes modelos de previsão de séries temporais sem exigir uma fase prévia de treinamento, fazendo com que o modelo se ajuste continuamente à chegada de novos dados. Afirmam que a vantagem desta técnica é que pode adaptar-se a novos tipos de cargas de trabalho.

Utilizaram cinco modelos estatísticos de previsão: modelo Ingênuo (Naive), modelo Autorregressivo (AR); modelo Autorregressivo de Médias Móveis (ARMA); modelo Autorregressivo Integrado de Médias Móveis (ARIMA); e modelo de Suavização Exponencial Estendida (ETS)

Foi utilizado três conjuntos de dados de logs reais de servidores Web (Nasa, FIFA World Cup e ClarkNet Traces) e o pré-processamento considerando a série temporal ori-

ginal com resolução de 1 segundo.

A arquitetura do sistema de autoescalonamento é dividida em duas fases, a primeira é responsável pelo modelo de previsão da demanda e a segunda fase representa o módulo responsável por calcular a alocação de recursos mínima para suportar a demanda. A figura 9 apresenta a arquitetura proposta pelos autores.

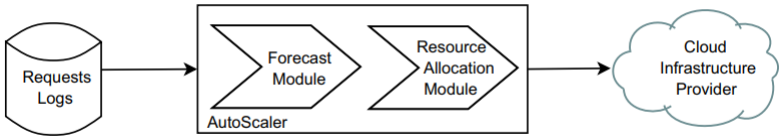


Figura 9 – Arquitetura do auto escalonador. Fonte: (MESSIAS, 2016)

O Sistema autoescalonador ajusta dinamicamente a aplicação web em resposta a mudanças na quantidade de solicitações, em intervalos fixos de 1 hora. Este intervalo de reconfiguração visa otimizar os custos, já que muitos provedores de infraestrutura de nuvem consideram 1 hora como o período mínimo de cobrança.

Concluem que o método é suficientemente genérico para alcançar bons resultados de autoescalonamento de aplicações web na nuvem. Demonstraram que, quanto maior a taxa de processamento do servidor e menor o tempo de resposta, menor será o erro na alocação de recursos, concluindo que, ao diminuir a taxa de processamento do servidor e o tempo máximo de resposta (SLA), a precisão da previsão se torna mais importante. Também definem que a métrica utilizada, MEI, se mostrou a melhor opção para avaliar a elasticidade da técnica de autoescalonamento, sendo menos sensível a outliers.

O trabalho se relaciona com o elaborado, pois propõe também uma arquitetura de autoescalonamento baseada em duas etapas e se diferencia por não utilizar modelo proativo de autoescalonamento baseado em inteligência artificial.

Kumar e Singh (KUMAR; SINGH, 2018) apresentaram um modelo de previsão de carga utilizando redes neurais e evolução diferencial autoadaptável em abordagem de aprendizado supervisionado para treinar o modelo. O modelo aprende e extrai padrões da carga de trabalho, sendo treinado com uma abordagem evolutiva para minimizar o efeito da escolha inicial da solução. O algoritmo evolutivo explora o espaço em diversas direções, reduzindo a necessidade de ajuste de parâmetros. Conforme os autores, as previsões geradas pelo modelo podem ser empregadas para aprimorar decisões de escalonamento de recursos. Nos testes com conjuntos de dados de dois servidores, o modelo superou abordagens baseadas em média, máximo e retropropagação de redes neurais, apresentando uma significativa redução no erro quadrático médio de previsão. Os experimentos foram conduzidos nos conjuntos de dados de acesso HTTP dos servidores NASA e Saskatchewan para diferentes períodos de previsão.

Durante a etapa de pré-processamento, são extraídas as requisições e agregadas em intervalos de 1 minuto. Em seguida, os dados são normalizados e utilizados como entrada na rede neural de três camadas (camada de entrada, camada oculta e camada de saída). 60% dos dados foram utilizados como dados de treinamento, enquanto 40% dos dados foram usados para teste. A precisão foi avaliada e medida usando a métrica Raiz do Erro Médio Quadrático (RMSE). Durante o treinamento, utilizou-se a técnica de Evolução Diferencial Autoadaptável(Self Adaptive Differential Evolution - SaDE); esta variante do Differential Evolution - DE demonstrou eficácia em competições de algoritmos evolutivos. De acordo com os autores, o SaDE foi escolhido por sua simplicidade, facilidade de uso e desempenho superior a outras abordagens baseadas em população, como PSO, GA e outras.

A figura 10 apresenta o fluxo de trabalho da abordagem proposta para previsão de carga de trabalho.

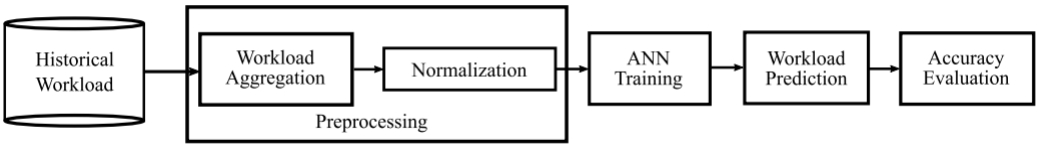


Figura 10 – Fluxo de trabalho da previsão de carga de trabalho. Fonte: (KUMAR; SINGH, 2018)

Os autores concluem que, comparando os resultados com um modelo baseado no algoritmo de retro-propagação, observou-se melhorias significativas. O modelo proposto conseguiu reduzir o erro em até 168 vezes nas previsões da carga do servidor de Saskatchewan quando o intervalo de previsão foi de 5 minutos, com um erro mínimo de previsão de 0,001, reduzido em comparação com o algoritmo de retro-propagação.

Singh, Gupta e Jyoti.(SINGH, 2019) apresentaram uma abordagem que realiza primeiramente a classificação do padrão de carga de trabalho e assim seleciona o modelo de previsão de curto prazo. Trabalharam com os modelos de regressão linear (LR), ARIMA e regressão de vetor de suporte (SVR) que são selecionados de acordo com o padrão de carga. Afirmam que a previsão de curto prazo é altamente adequada para as aplicações web em ambiente de nuvem. O estudo utilizou as bases de dados Clarknet Traces e NASA e realizou o pré-processamento, reduzindo a amostragem da série temporal original de requisições por segundo para 10 minutos.

A Arquitetura do modelo desenvolvido para realizar o provisionamento de recursos conta com vários componentes com finalidades específicas, por exemplo: o módulo de controle de admissão aceita ou rejeita as solicitações do usuário de acordo com o estado atual do sistema. O provisionador de aplicações recebe as solicitações aceitas pelo módulo de controle de admissão que realiza o agendamento para as Máquinas Virtuais (VMs) com

a capacidade para processá-las. O módulo também recebe informações do Modelador de Desempenho sobre as VMs necessárias. Se a capacidade desejada e a capacidade real das VMs diferirem em número, o módulo realiza ação automática de dimensionamento para cima ou para baixo. Há ainda o Modelador de Desempenho que prevê a demanda com base no Módulo de Previsão de Carga de Trabalho e no feedback de desempenho das VMs do módulo de Provisionador de Aplicações.

A figura 11 exibe as características da arquitetura do modelo.

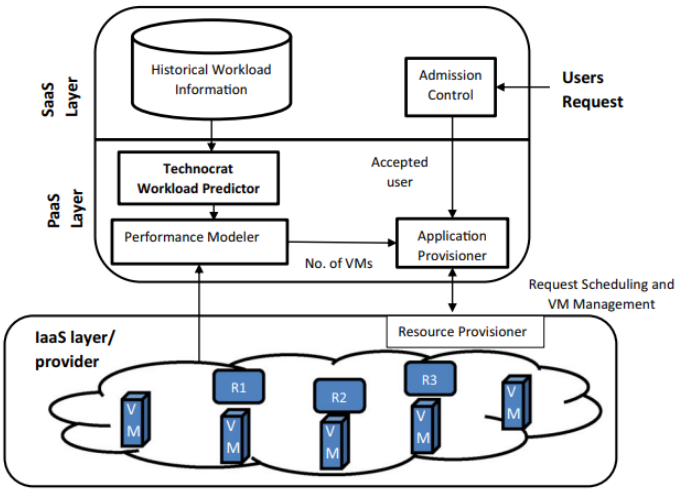


Figura 11 – Arquitetura do provisionador de nuvem. Fonte: (SINGH, 2019)

A figura 12 exibe as características do método. A previsão é baseada no estudo sazonal/não sazonal da série temporal que, por sua vez, depende da classificação da série temporal.

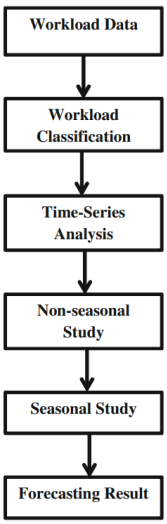


Figura 12 – Metodologia de previsão genérico. Fonte: (SINGH, 2019)

O trabalho também se relaciona com o elaborado, pois propõe uma arquitetura de

autoescalonamento para aplicações envolvendo cargas de trabalho por requisições HTTP e se diferencia pois não utilizaram modelo proativo de autoescalonamento baseado em inteligência artificial e sim em modelos estatísticos convencionais.

Singh et al. (SINGH, 2021) propuseram uma metodologia híbrida para realizar o autoescalonamento de recursos na nuvem de forma tanto reativa quanto proativa. Com base na metodologia de previsão realizada no trabalho anterior, o TASM,(SINGH, 2019) foi adicionado um componente que utiliza regras baseadas em limiares para realização da abordagem reativa de autoescalonamento. O trabalho também utilizou o ciclo Monitorar-Analisar-Planejar-Executar (MAPE) A abordagem foi testada com as bases de dados reais, ClarkNet Traces e NASA re-amostrada para um intervalo de 1 minuto. Os autores concluem que a técnica proposta alcançou uma redução de 14% nos custos, uma melhoria significativa no tempo de resposta, na violação do acordo de nível de serviço (SLA) e proporciona consistência na utilização da CPU.

Kumar et al.(KUMAR K. GANGADHARA RAO, 2021) conduziram um estudo comparativo aplicando Regressão Linear (LR), K-Nearest Neighbors (KNN), Máquina de Vetores de Suporte (SVM), ARMA, ARIMA e Support Vector Regression (SVR) para aplicações web, a fim de selecionar o algoritmo mais adequado conforme as características da carga de trabalho. Também utilizaram a base de dados Clarknet traces e NASA.

Realizaram o pré-processamento considerando o número de solicitações a cada 10 minutos e, durante esta fase, os dados foram preparados e analisados tanto horizontal quanto verticalmente por meio de técnicas de pré-processamento no Weka. No processo de seleção de características do Weka, cada subconjunto de valores foi testado com o algoritmo de aprendizado de máquina específico. O subconjunto que apresentou melhor desempenho, multidimensionalidade e adequação aos dados foi escolhido para as previsões. Essa estratégia resultou na redução do número de características consideradas de 18 para cinco, englobando tempo de espera, tempo de execução, número de processadores designados, tempo médio de CPU utilizado e memória utilizada. A amostragem horizontal incluiu a aplicação de validação cruzada.

Os autores concluem que os modelos de Regressão Linear (LR) e ARIMA apresentaram melhorias significativas para o conjunto de dados da NASA, enquanto KNN e ARIMA mostram melhorias significativas para o conjunto de dados da ClarkNet.

### 2.6.3 Considerações Finais

Nesse capítulo foram apresentados os trabalhos que se relacionam diretamente, ou indiretamente, à pesquisa desenvolvida. Foram apresentados os trabalhos que têm como objetivo a autoescalabilidade de recursos com base em previsão de carga de trabalho do tipo requisições HTTP.

# 3 Previsão de Carga de Trabalho de Sistemas Legados Baseado em LSTM

Este capítulo apresenta um estudo exploratório-descritivo com o objetivo de desenvolver um modelo de previsão de carga de trabalho com base no volume de requisições para sistemas legados implantados em data centers locais, denominado *WFLS-LSTM (Workload Forecasting for Legacy Systems - LSTM)*[1]. Utilizamos uma abordagem quantitativa que combina análise estatística de séries temporais e métricas de avaliação de modelos de previsão. O desenvolvimento, implementação e comparação do modelo de previsão proposto foram documentados de forma sistemática, analisando séries temporais provenientes de um conjunto de dados de referência (benchmark) e de um sistema real.

As questões de pesquisa propostas (Seção 1.2) visam explorar diferentes aspectos do autoescalonamento de sistemas legados voltados para o público em data centers locais. O primeiro objetivo, relacionado à QP 1, investiga a influência da resolução temporal e a precisão das previsões neste contexto. A segunda questão, QP2, concentra-se em comparar diferentes modelos preditivos, incluindo técnicas de aprendizado profundo, como LSTM, com modelos tradicionais, para avaliar suas vantagens na previsão de carga de trabalho baseada em requisições HTTP aplicadas neste contexto.

A figura 13 exibe a metodologia adotada neste trabalho que inclui várias etapas: inicialmente, a coleta de dados e a análise da série temporal em diferentes resoluções, visando responder à QP1. Em seguida, o desenvolvimento do modelo de previsão com a otimização dos hiperparâmetros e a avaliação de sua precisão, comparando-o com outros métodos de previsão e trabalhos relacionados 2.6, para responder à QP2.

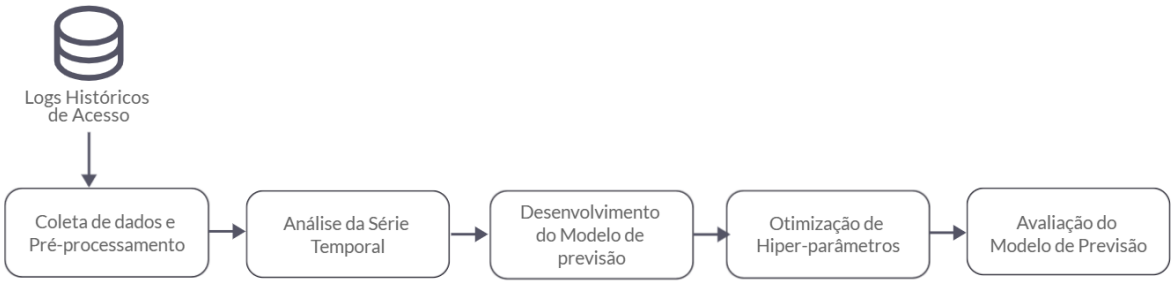


Figura 13 – Metodologia

[1] Código e dados em: https://doi.org/10.5281/zenodo.14504203

## 3.1 Bases de Dados

As bases de dados se tratam de arquivos de log de dois conjuntos de dados distintos: o Clarknet Traces, utilizado como benchmark, e os registros de um sistema legado real voltado para o público, implantado em ambiente on-premises - Sistema de Gerência de Processos Seletivos COPEVE.

Os dados são extraídos de logs de acesso de servidores web Apache, onde cada linha contém informações detalhadas sobre as requisições, conforme exemplificado na Figura 14. O formato inclui: endereço IP de origem da requisição, informação temporal, início da requisição HTTP, código de status HTTP retornado, quantidade de bytes enviados, informações de referência, dados sobre o navegador e o tempo necessário para atender à requisição. Esta estrutura possibilita uma análise minuciosa da demanda e do desempenho da aplicação, permitindo identificar padrões de utilização e potenciais pontos de sobrecarga ao longo do tempo.

### 3.1.1 Conjunto de dados de benchmark - Clarknet Traces

Consiste em duas semanas de registros de todas as solicitações HTTP para o servidor www do ClarkNet, um provedor de acesso à Internet para a área metropolitana de Baltimore-Washington DC, disponível publicamente (CLARKNET, 1995). A série temporal resultante contém 1.209.511 observações de contagem de requisições por segundo, com valores variando de 0 a 45 e média de 3 requisições por segundo.

### 3.1.2 Conjunto de dados empírico - COPEVE

O segundo conjunto foi obtido através da recuperação de dados do servidor de produção e infraestrutura de backup. Após a coleta, os dados foram organizados e anonimizados para proteger informações sensíveis como endereços IP, parâmetros de URL do método GET e origem das requisições. Este conjunto abrange aproximadamente 3 meses de registros do servidor balanceador de carga da aplicação, resultando em uma série temporal de 8.875.129 observações. A série apresenta valores entre 0 e 900 requisições por segundo, com média de 3, sendo aproximadamente sete vezes maior que a série de benchmark e com um intervalo de valores 20 vezes superior.

## 3.2 Coleta e pré-processamento dos dados

Os arquivos de log para os dados privados da COPEVE contêm dados sensíveis que precisam ser anonimizados, dados como, por exemplo: endereço de IP do cliente, dados URL e origem da requisição.

```
172.16.0.40 - - [03/May/2024:22:41:51 -0300]
"GET /home/ HTTP/1.1" 200 16323
"https://www.google.com/" "Mozilla/5.0 (Linux;
Android 10; K) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/124.0.0.0 Mobile Safari/537.36"
116486
```

Figura 14 – Linha de um log de acesso de um servidor web Apache.

Após a coleta e anonimização dos dados, os arquivos de logs foram manipulados para gerar a série temporal com a contagem das requisições por segundo ao servidor web para os dois conjuntos de dados. Para realizar esta manipulação dos arquivos de log, foi utilizada a biblioteca de análise de dados Python chamada Pandas (PANDAS, 2020). Os arquivos de texto com os registros de acesso ao servidor foram convertidos na estrutura de dados primária do pandas: o dataframe que representa dados tabulares bidimensionais, mutáveis em tamanho e potencialmente heterogêneos. A estrutura de dados também contém eixos rotulados (linhas e colunas).

A seguir é demonstrado o algoritmo que agregou as requisições para a geração da série temporal e do dataframe utilizando como base as funções pd.Grouper() e size() do Pandas:

**Algorithm 1** Geração da série temporal a partir dos dados de log do servidor

▷ Agrupamento de requisições e contagem por segundo
2: *req_count* ← final_copeve_data.groupby(pd.Grouper(key='timestamp', freq='S')).size()
▷ Criação do DataFrame a partir da contagem de requisições
4: *df_req_count* ← req_count.reset_index().rename(columns=0: 'num_requests')

As séries geradas inicialmente com resolução de segundos foram reamostradas em outros períodos: agregando as requisições que ocorreram nos intervalos de um minuto (KUMAR; SINGH, 2018), dez minutos (SINGH, 2018)(SINGH, 2021) e outra reamostragem com a quantidade máxima de requisições que ocorreram em um segundo no intervalo de uma hora, conforme realizado por Messias et al (MESSIAS, 2016). Os valores ausentes (zeros) foram preenchidos com o valor 1 para o correto cálculo da métrica MAPE na fase de avaliação.

## 3.3 Análise da série temporal

A análise em múltiplas resoluções da série temporal, conforme abordado na Questão de Pesquisa 1 (QP1), visa identificar a periodicidade mais adequada dos dados para

otimizar o desempenho de sistemas de autoescalonamento implantados em data centers locais, diferenciando-se de estudos focados em ambientes de nuvem.

Além disso, a análise de séries temporais tem como objetivo compreender o mecanismo subjacente que gera os dados, identificar padrões periódicos, descrever o comportamento e prever valores futuros. Entender a dinâmica da série permite determinar transformações necessárias e escolher os parâmetros mais apropriados, especialmente para modelos estatísticos tradicionais. Nessa fase, são considerados aspectos como sazonalidade, autocorrelação, estacionariedade e entropia, com o intuito de extrair informações que possam contribuir para a construção dos modelos de previsão.

No caso da série temporal do Sistema da COPEVE, a análise busca verificar semelhanças com as características observadas na série temporal da ClarkNet. A identificação de características semelhantes da carga de trabalho dos dois conjuntos de dados pode indicar o uso do mesmo método de previsão.

## 3.4 Desenvolvimento do modelo e otimização dos hiper-parâmetros

Esta etapa consiste na implementação da arquitetura do modelo de machine learning para a análise e predição da carga de trabalho dos serviços. Será utilizado o algoritmo LSTM (Long Short-Term Memory), um tipo de rede neural recorrente - RNN que possui a capacidade de processar e aprender com sequências temporais. Atualmente, as Redes Neurais Recorrentes (RNNs) têm-se mostrado uma ferramenta poderosa para análise de séries temporais, seja para tarefas de previsão ou de classificação (LANCIANO, 2021). A rede LSTM será treinada com os dados históricos pré-processados, a fim de construir um modelo capaz de prever a carga de trabalho futura com base nas séries temporais obtidas pelos arquivos de logs de acesso ao serviço. A seguir são descritos os hiperparâmetros e detalhes da arquitetura do modelo:

### 3.4.1 Função de Perda

O modelo resultante utiliza a função de perda MSE (Mean Squared Error) ou Erro Quadrático Médio. Esta função é útil na detecção de valores discrepantes, permitindo a atribuição de pesos maiores a esses pontos. O quadrado associado ao MSE amplia o erro quando o modelo gera uma única previsão muito ruim, tornando-o sensível a outliers. (CHICCO, 2021) A escolha do MSE também se justifica no contexto deste trabalho, pois é desejável a detecção de valores discrepantes, que podem representar os picos de utilização do serviço.

### 3.4.2 Otimizador

Os otimizadores desempenham um papel fundamental no aprendizado profundo, ajustando variáveis da rede neural, como pesos e taxas de aprendizado, com o objetivo de minimizar as funções de perda. Neste trabalho, utilizou-se o algoritmo Adam (Adaptive Moment Estimation), que combina as vantagens dos métodos de momento e RMSProp, integrando o gradiente médio e o gradiente quadrático acumulado para aprimorar a eficiência das atualizações dos pesos. O algoritmo Adam utiliza estimativas corrigidas de viés para ambos os componentes, como descrito em (ARIFF; ISMAIL, 2023).

O algoritmo também se destaca por adaptar dinamicamente as taxas de aprendizado para cada parâmetro da rede, reduzindo a necessidade de ajustes manuais de hiperparâmetros (KINGMA; BA, 2017). A configuração utilizada neste modelo empregou uma taxa de aprendizado (*learning rate*) de 0.00015, definida após testes sistemáticos de *grid search*, o que resultou em estabilidade e boa convergência durante o treinamento.

### 3.4.3 Early Stop

Também foi configurada a parada antecipada, ou Early Stop, que é uma estratégia onde o modelo é interrompido quando o erro no conjunto de validação não melhora por um tempo, em vez de atingir o mínimo local do erro de validação. O Early Stop é uma técnica que tem se mostrado bastante eficaz para prevenir o ajuste excessivo dos dados de treinamento ou Overfitting(TIAN; ZHANG, 2022). O valor configurado para a parada antecipada foi de 20 épocas de treinamento.

### 3.4.4 Arquitetura do Modelo

A arquitetura proposta para a rede neural recorrente do tipo LSTM consiste em uma camada LSTM com 64 unidades, seguida por uma camada de dropout com uma taxa de 0.2 (20% das unidades são desativadas aleatoriamente durante o treinamento). Em seguida, há uma camada densa com uma unidade e uma função de ativação ReLU (Rectified Linear Unit). Esta camada foi definida com a função de ativação ReLU pois, durante os testes iniciais, algumas previsões tiveram valores negativos, o que não é possível neste contexto. A função ReLU mostra que se a entrada $x$ for um número negativo, a saída é igual a 0 (Bai, Yuhan, 2022).

$$ReLU(x) = max(0, x)$$

$$ReLU(x) = \begin{cases} 0 & \text{se } x \leq 0 \\ x & \text{se } x > 0 \end{cases}$$

O tamanho da janela foi definido como 48 para as séries temporais de resolução de minuto e 10 minutos e 12 para a série por hora. Este parâmetro é o número de observações anteriores que são consideradas como entrada para o modelo; em outras palavras, ele determina quantos passos de tempo anteriores são utilizados para prever o próximo valor na sequência. A quantidade de parâmetros em cada camada é apresentada na tabela 1.

Tabela 1 – Detalhes da arquitetura do modelo.

| Camada (tipo) | Formato da Saída | Parâmetros |
|---|---|---|
| LSTM | (None, 64) | 39,680 |
| Dropout | (None, 64) | 0 |
| Dense | (None, 1) | 65 |
| Total | | 39,745 |

A figura 15 representa graficamente o modelo desenvolvido.

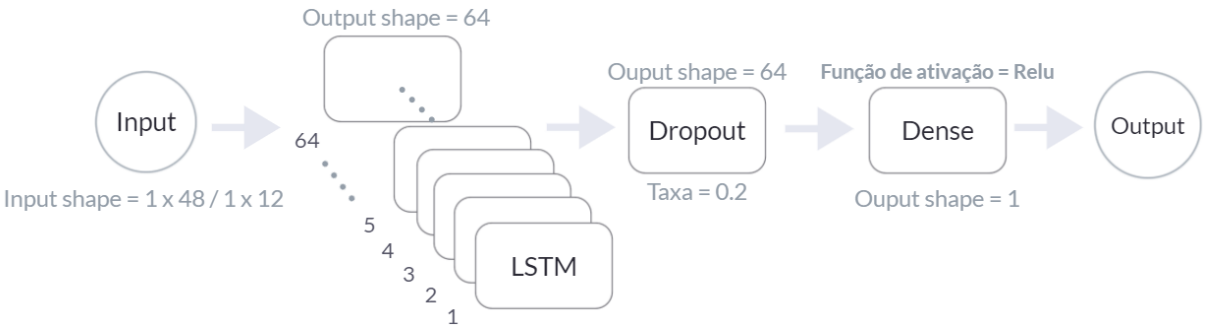


Figura 15 – Modelo

### 3.4.5 Grid Search

Foi utilizada a técnica de pesquisa em grade, ou Grid Search, para encontrar os hiperparâmetros que otimizam o treinamento do modelo. A pesquisa em grade é um método exaustivo que explora todas as combinações possíveis de hiperparâmetros para avaliar seu desempenho e determinar a configuração que oferece a melhor performance(OGUNSANYA, 2023). A tabela 2 mostra os hiperparâmetros e valores testados.

### 3.4.6 Particionamento dos Dados

O conjunto de dados foi particionado em 80% para treinamento e 20% para teste, uma proporção escolhida para garantir uma quantidade suficiente de dados para o treinamento do modelo, ao mesmo tempo em que preserva uma amostra representativa de dados de teste para avaliar sua capacidade de generalização. A figura 16 ilustra o particionamento aplicado à série temporal.

Tabela 2 – Hiperparâmetros e Valores testados

| Hiperparâmetro | Valores |
|---|---|
| Taxa de aprendizado | 0.00015, 0.001, 0.01 |
| Unidades 1ª camada | 64, 240, 256, 512 |
| Unidades 2ª camada | 0, 32, 64, 128 |
| Unidades 3ª camada | 0, 16, 32 |
| Unidades 4ª camada | 0, 8, 16 |
| Taxa de dropout | 0.2, 0.3 |
| Função de perda | mean_squared_error<br>mean_squared_logarithmic_error<br>huber |

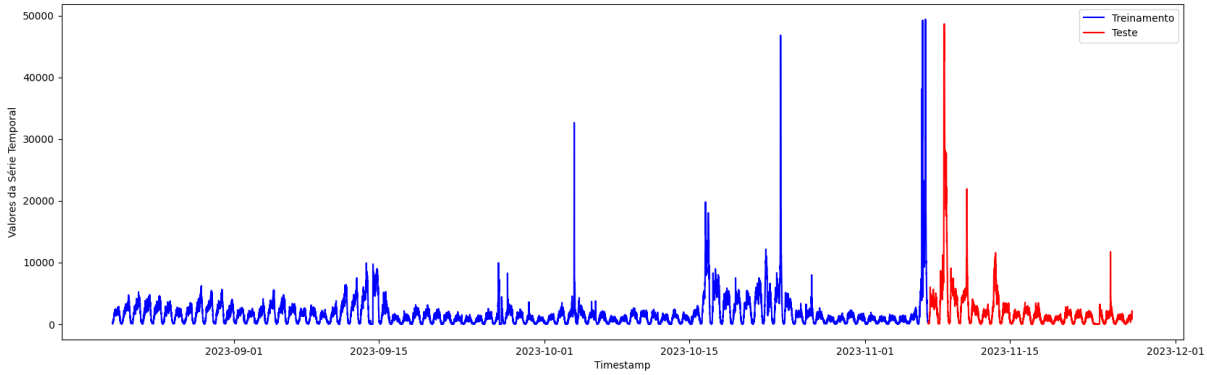


Figura 16 – Particionamento do conjunto de dados: 80% para treinamento e 20% para teste.

Além desse particionamento tradicional, foi realizado um estudo comparativo para identificar a técnica de particionamento mais adequada ao conjunto de dados em questão. Foram empregadas técnicas de validação cruzada, ou cross-validation, específicas para séries temporais, utilizando as bibliotecas `TimeSeriesSplit` e `BlockingTimeSeriesSplit`. Essas abordagens são amplamente recomendadas para manter a estrutura sequencial dos dados.

A figura 17 apresenta o particionamento gerado pela técnica `TimeSeriesSplit`, enquanto a figura 18 ilustra o particionamento obtido com `BlockingTimeSeriesSplit`.

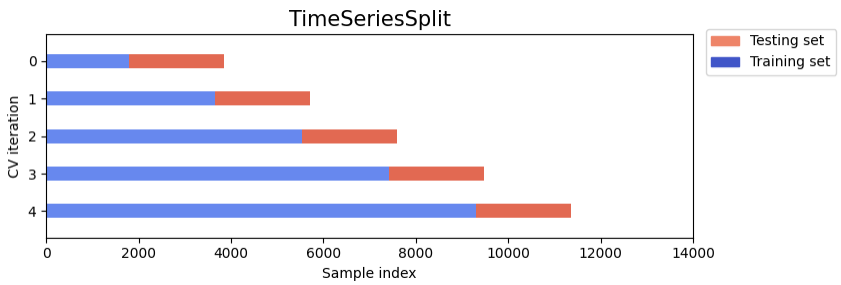


Figura 17 – Particionamento utilizando validação cruzada: `TimeSeriesSplit`.

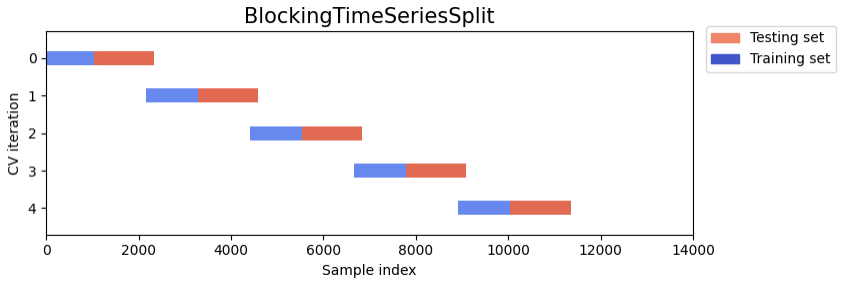


Figura 18 – Particionamento utilizando validação cruzada: `BlockingTimeSeriesSplit`.

## 3.5 Avaliação do Modelo

A Avaliação do modelo consiste em verificar se as previsões possuem boa acurácia em relação aos conjuntos de treinamento e também em dados de teste que não foram utilizados para treinamento. As métricas RMSE (Root Mean Squared Error), MSE (Mean Squared Error), MAPE(Mean Absolute Percentage Error) e MAE (Mean Absolute Error) foram utilizadas para avaliar a precisão do modelo de previsão da carga de trabalho.

São métricas comumente utilizadas em problemas de aprendizado de máquina e análise de séries temporais. Essas métricas também foram utilizadas nos trabalhos relacionados que usaram a base de requisições HTTP de benchmark Clarknet Traces.

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

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \times 100$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|$$

Quanto menor o valor das métricas, melhor pode ser considerado o modelo. O parâmetro $n$ denota o número de observações do passado até o início da série temporal. $Yi$ é a carga de trabalho real no período, e $\hat{Y}i$ representa a carga de trabalho prevista.

A performance do modelo também é avaliada em termos de sua capacidade de realizar previsões diretas para diferentes horizontes (6, 12, 48 e 144 passos à frente), permitindo maior flexibilidade na criação de políticas de escalonamento ao identificar tendências futuras.

Além disso, analisa-se o tempo necessário para ajustar ou treinar os modelos, bem como o tempo de previsão para todo o conjunto de teste. Esse aspecto é fundamental

para verificar a viabilidade da aplicação em cenários que exigem previsões periódicas em tempo real, como em sistemas de escalonamento automático.

Para ajudar a responder a questão de pesquisa 1 (QP1), a arquitetura do modelo proposto é utilizada para treinamento e validação da série temporal baseada no conjunto de dados de benchmark Clarknet e comparados os resultados com trabalhos que utilizaram a mesma base.

## 3.6 Configuração dos experimentos

Os experimentos foram conduzidos em um ambiente computacional utilizando um sistema equipado com um processador Intel Core i7-1165G7 de 4 núcleos e 8 threads, com uma frequência base de 2,80 GHz e uma frequência de turbo máxima de 4.7 GHz. O sistema foi equipado com 20 GB de memória RAM DDR4 e uma placa gráfica GPU Intel(R) Iris(R) Xe com 10 GB de memória compartilhada.

Para o treinamento dos modelos de aprendizado profundo, empregamos o framework TensorFlow versão 2.16.1, executado sem aceleração por GPU, em um ambiente de desenvolvimento Python 3.10.11 no sistema operacional Windows 11.

O treinamento dos modelos foi realizado em um único computador. O tempo total de treinamento para cada modelo foi registrado utilizando a função de registro de tempo do framework TensorFlow.

## 3.7 Resultados

Nesta seção, são detalhadas as configurações do ambiente experimental, incluindo especificações do hardware e software utilizados. Além disso, é apresentada uma análise dos resultados obtidos ao aplicar a metodologia para o desenvolvimento do modelo de previsão e aplicação no sistema de escalonamento automático.

### 3.7.1 QP 1: Qual o impacto da granularidade temporal na eficácia dos modelos preditivos?

Para responder a esta questão de pesquisa, foi realizada a análise da série temporal e realizados experimentos com a aplicação do modelo desenvolvido no conjunto de dados de benchmark Clarknet. Após a coleta e pré-processamento dos dados conforme descrito na subseção 3.2 da metodologia, seguem abaixo os resultados obtidos a partir da análise da série temporal:

A série com a contagem de requisições por segundo possui 1.209.511 observações, a escala dos valores na série é de 0 a 45, sendo o valor 3 a média. A figura 19 representa

graficamente as séries temporais geradas a partir dos dados de requisições HTTP do servidor. A visualização temporal é uma das formas mais simples e rápidas de representar dados de séries temporais (DAMA; SINOQUET, 2021)


Figura 19 – Serie temporal Clarknet Traces em diferentes resoluções

A **distribuição** dos dados ao longo do tempo é visualizada através de quatro gráficos de distribuição em diferentes escalas temporais: mensal, semanal, diária e por hora do dia. A figura 20 demonstra este comportamento através da análise da distribuição dos dados ao longo do tempo pelos gráficos box plot. A análise visual do conjunto de dados demonstra uma quantidade maior de requisições nos dias de semana e quantidade menor nos finais de semana. O horário de 10 às 17h mostra um aumento nas requisições no gráfico diário. Cabe ressaltar que os outliers do gráfico de boxplot são dados importantes para o modelo de aprendizado, pois se tratam de valores nos quais há de fato um uso mais intenso na aplicação. Logo, é necessário que o modelo de previsão de carga capture-os para que o sistema de autoescalonamento aumente o provisionamento de recursos para evitar queda no tempo de resposta da aplicação.

A figura 21 exibe a **decomposição** da série temporal de resolução de 10 minutos como exemplo. A análise do gráfico de decomposição da série mostra a sazonalidade diária e variabilidade na tendência. Os resíduos mostram períodos de alta variabilidade assim como ocorrem com a tendência.

A **sazonalidade** foi testada por meio do gráfico de autocorrelação, figura 28, que apresentou picos fora da faixa de confiança nos lags múltiplos de 144 (144 períodos de 10

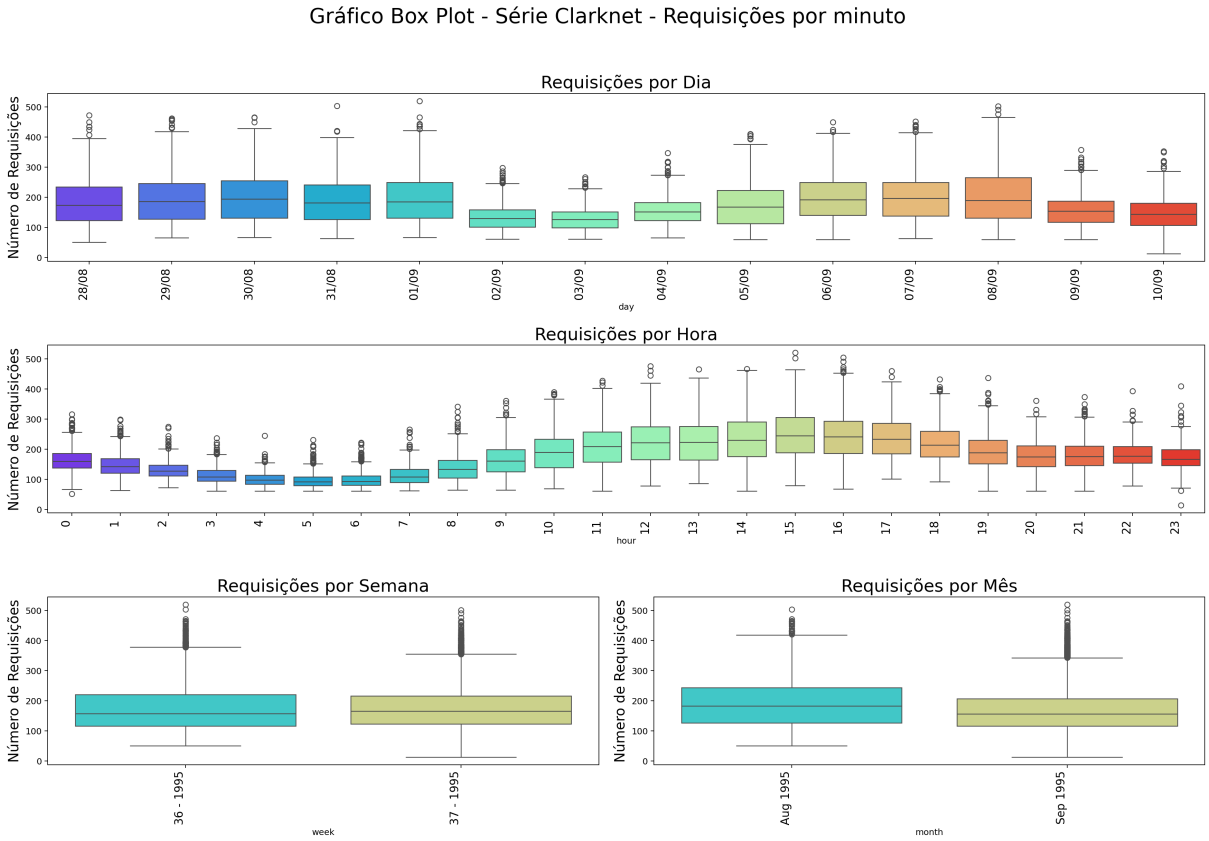


Figura 20 – Distribuição dos dados de requisições para a série Clarknet Traces - 10 min.

minutos são 24 horas), indicando uma forte autocorrelação dos dados em relação a este período.

Em relação à **estacionariedade**, a análise gráfica por meio do gráfico ACF para as séries temporais de 1 minuto e 10 minutos apresentaram um decaimento lento indicando não estacionariedade, já a série com resolução de uma hora teve um caimento mais rápido no sexto lag, podendo indicar algum grau de estacionariedade. A figura 23 demonstra este comportamento para a série Clarknet Traces. Entretanto, os testes estatísticos ADF e KPSS, tabelas 3, 4 e 5, descrevem um resultado alternativo para as séries temporais com resolução de 1 minuto, 10 minutos e 1 hora, respectivamente.

Conforme a tabela 11 para a série temporal com resolução de 1 minuto, a estatística do teste ADF (-4.4355) é significativamente menor do que os valores críticos em todos os níveis de significância; o valor p é extremamente pequeno e menor do que qualquer nível usual de significância (0.01, 0.05, ou 0.10). Isto reforça a evidência para rejeitar a hipótese nula de que a série possui uma raiz unitária (ou seja, não é estacionária) e, portanto, concluir que a série é estacionária. Além disto, o teste KPSS também sugere que a série é estacionária com um grau de confiança razoável, pois, além do $valor - p > 0.05$, a estatística do teste é 0.3989, que é menor que os valores críticos para os níveis de

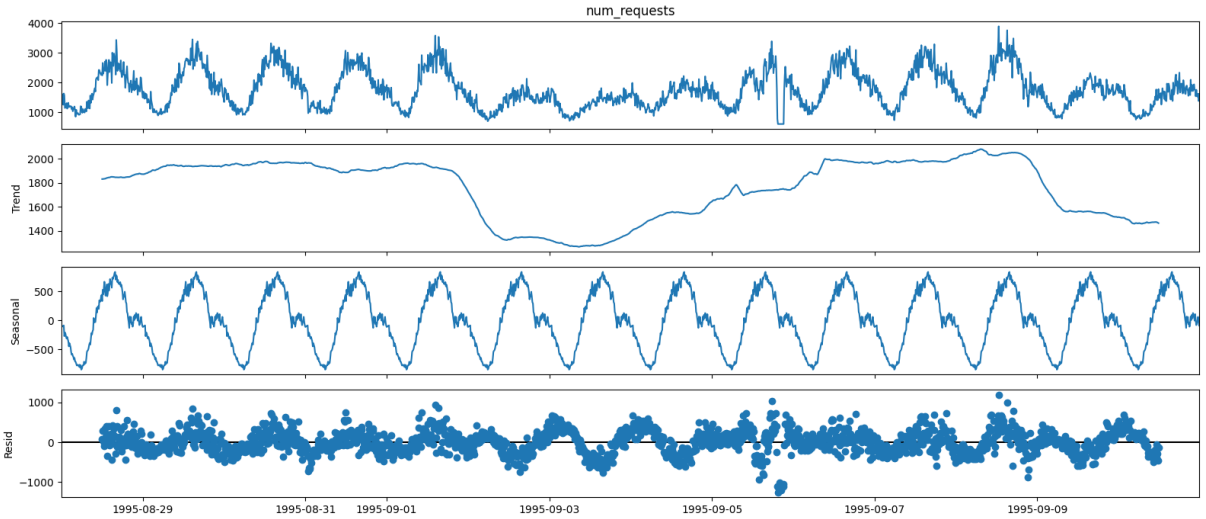


Figura 21 – Decomposição da série temporal de 10 min.

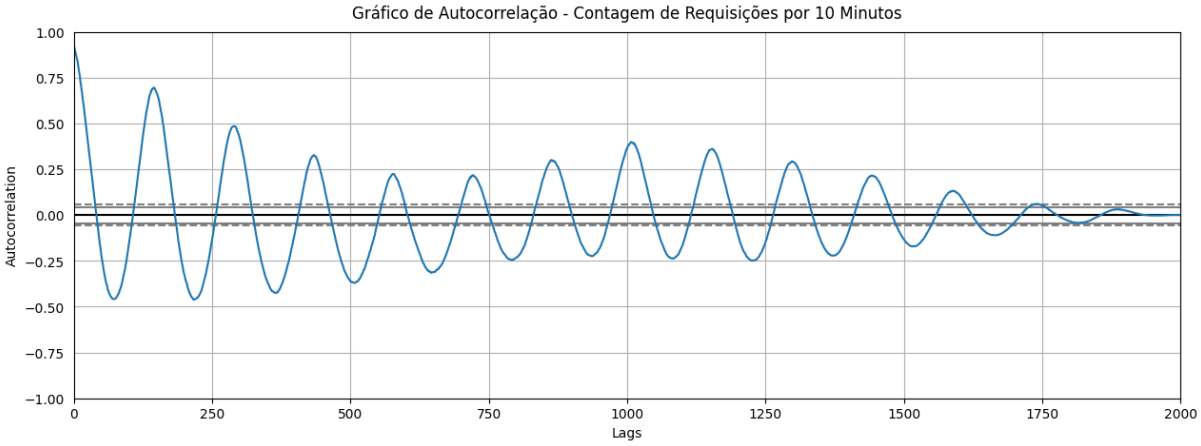


Figura 22 – Sazonalidade através do gráfico de autocorrelação

significância de 1% e 5%, mas um pouco acima do valor crítico para 10% .

Tabela 3 – Resultados dos Testes de Estacionariedade - Série 1 minuto

| **Teste** | **Estatística** | **Valor p** | **Valores Críticos 1%, 5%, 10%** | Resultado |
|---|---|---|---|---|
| ADF | −4.4355 | 0.00026 | −3.43, −2.86, −2.57 | Estacionária |
| KPSS | 0.3989 | 0.0776 | 0.74, 0.46, 0.35 | Estacionária |

Para a série temporal com resolução de 10 minutos, de acordo com os resultados exibidos na tabela 12, a estatística do teste (-5.706) também é menor do que os valores críticos em todos os níveis (1%, 5%, e 10%); O $valor - p$ é extremamente pequeno, indicando que podemos rejeitar a hipótese nula de que a série tem uma raiz unitária, podendo concluir que a série é estacionária. Para o teste KPSS, a estatística do teste

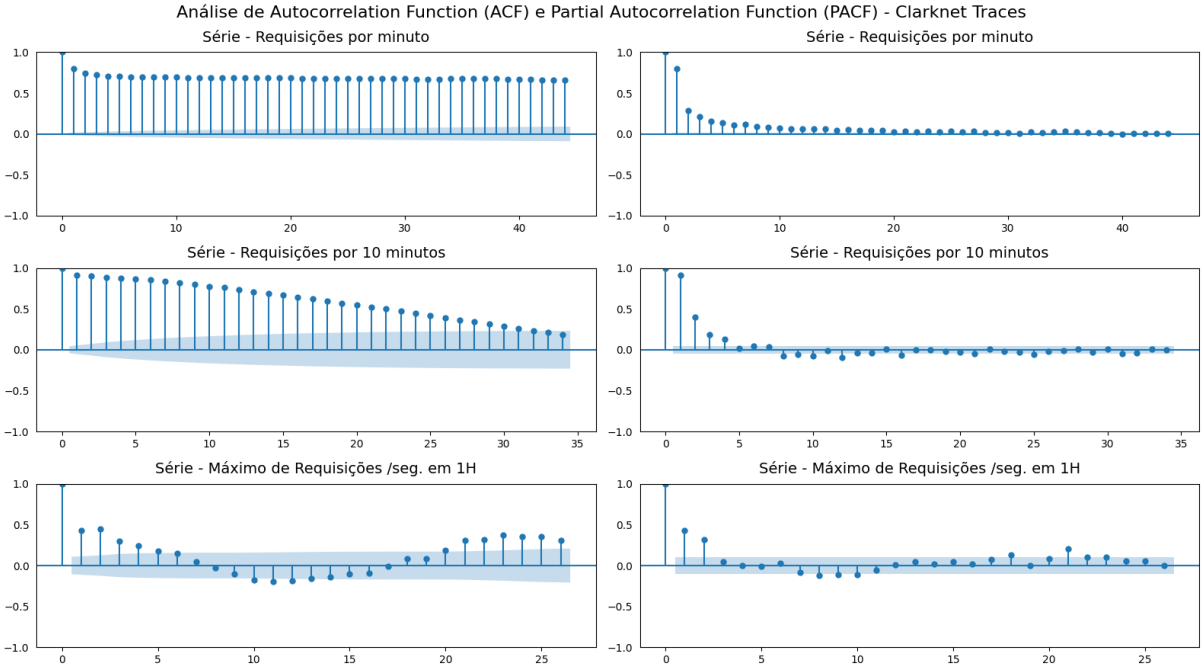


Figura 23 – Autocorrelação - Clarknet em diferentes resoluções

(0.1423) é menor do que os valores críticos para todos os níveis de significância (1%, 5% e 10%), além disso, o p-value é maior que 0.05, indicando que a hipótese nula não é rejeitada, portanto, podemos concluir que a série é estacionária.

Tabela 4 – Resultados dos Testes de Estacionariedade - Série 10 minutos

| **Teste** | **Estatística** | **Valor p** | **Valores Críticos 1%, 5%, 10%** | Resultado |
|---|---|---|---|---|
| ADF | -5.7069 | $7.45 \times 10^{-7}$ | $-3.43, -2.86, -2.57$ | Estacionária |
| KPSS | 0.1423 | 0.1 | 0.739, 0.463, 0.347 | Estacionária |

Para a série temporal com resolução de 1h, a estatística do teste ADF (-6.5129) é menor que os valores críticos e o $valor - p$ é muito pequeno, reforçando que a hipótese nula pode ser rejeitada, podendo concluir que a série é estacionária. Para o teste KPSS, a estatística (0.1362) é menor que os valores críticos e o $valor - p$ é maior que $0,05$, sugerindo que a série é estacionária.

Tabela 5 – Resultados dos Testes de Estacionariedade - Série 1 hora

| **Teste** | **Estatística** | **Valor p** | **Valores Críticos 1%, 5%, 10%** | Resultado |
|---|---|---|---|---|
| ADF | $-6.5129$ | $1.09 \times 10^{-8}$ | $-3.45, -2.87, -2.57$ | Estacionária |
| KPSS | 0.1362 | 0.1 | 0.739, 0.463, 0.347 | Estacionária |

A medida de avaliação de **complexidade** - Sample Entropy, das séries temporais em diferentes resoluções foi calculada e os resultados exibidos na tabela 6 indicam que a série temporal em intervalos de 10 minutos possui a menor complexidade e a série por hora a maior.

Tabela 6 – Resultados de Sample Entropy em Diferentes Intervalos Temporais

| **Resolução** | **Sample Entropy** |
|---|---|
| Por minuto | 1.2733 |
| Por 10 minutos | 0.9202 |
| Por hora | 1.5431 |

#### 3.7.1.1 Considerações sobre a Análise da Série Temporal

Embora os testes estatísticos indiquem que as séries temporais apresentam estacionariedade de tendência constante (ADF) e em termos de autocorrelação de primeira ordem (KPSS) para todas as resoluções analisadas, a observação visual revela a existência de sazonalidade, um elemento que sugere não estacionariedade. Além disso, os gráficos ACF das séries de 1 minuto e 10 minutos demonstram um comportamento de decaimento lento, o que reforça a possibilidade de não estacionariedade. Também são identificáveis, através da análise da distribuição dos dados e da decomposição, uma tendência variável e picos súbitos, sugerindo a presença de padrões não lineares.

Esses resultados indicam a complexidade do comportamento dessas séries temporais obtidas empiricamente, o que torna promissor o uso de modelos de aprendizado de máquina e aprendizado profundo, como o LSTM, em comparação com métodos estatísticos tradicionais. Conforme destacado por (JANARDHANAN; BARRETT, 2017), o LSTM, sendo um modelo não linear para séries temporais, mostrou desempenho superior na previsão de carga de trabalho baseada no uso de CPUs.

#### 3.7.1.2 Avaliação do Modelo

Nesta seção, é apresentada uma análise comparativa entre o modelo desenvolvido com base em LSTM, os modelos estatísticos tradicionais, como ARIMA e ETS, e outros modelos de aprendizado de máquina, como SVR, SVM, KNN e Regressão Linear, com resultados obtidos em trabalhos relacionados que utilizaram o mesmo conjunto de dados.

A tabela 7 mostra a comparação entre os modelos com base nas métricas MAE, MSE, RMSE e MAPE para a série temporal com resolução de 1 minuto. O desempenho do modelo para esta série foi melhor que o modelo comparado em relação a todas as métricas avaliadas, possuindo uma melhoria de 8,77% na métrica RMSE e 18,02% na MAPE.

A tabela 8 mostra a comparação entre os modelos para a série temporal com resolução de 10 minutos. O modelo proposto apresentou um desempenho superior em

Tabela 7 – Comparação - série de resolução / min

| Modelo | MAE | MSE | RMSE | MAPE |
|---|---|---|---|---|
| LR[1] | - | - | 64.37 | 34.52 |
| SVR[1] | – | - | 64.67 | 32.95 |
| AR[1] | - | - | 54.60 | 30.42 |
| MA[1] | – | - | 59.24 | 33.14 |
| ARMA[1] | - | - | 59.25 | 32.98 |
| ARIMA[1] | - | - | 51.24 | 26.64 |
| TASM[1] | - | - | 42.24 | 20.63 |
| WFLS-LSTM* | **28.99** | **1483.87** | **38.52** | **16.90** |

[1] Resultados de (KUMAR; SINGH, 2018); * Resultados deste trabalho.

relação aos demais modelos ao utilizar as métricas RMSE, MSE e MAPE. Comparado ao TASM(SINGH, 2018), que foi identificado como o modelo de melhor desempenho nos dois estudos, o modelo proposto registrou uma melhoria de 4.65% nas métricas RMSE e MSE, 15.24% para MAPE e 0.94% em relação à métrica MAE.

Tabela 8 – Comparação - série de resolução / 10min

| Modelo | MAE | MSE | RMSE | MAPE |
|---|---|---|---|---|
| KNN[1] | 210.45 | 70265.12 | 250.81 | 12.45 |
| LR[1] | 265.23 | 79638.23 | 295.14 | 15.24 |
| SVM[1] | 250.63 | 95325.48 | 320.15 | 18.26 |
| ARMA[1] | 220.30 | 86257.26 | 250.45 | 15.69 |
| ARIMA[1] | 168.26 | 58234.18 | 235.72 | 11.52 |
| Naïve[2] | 192.75 | 65561.45 | 256.04 | 12.87 |
| LR[2] | 207.63 | 79569.25 | 282.08 | 14.52 |
| AR[2] | 178.81 | 57862.46 | 240.54 | 12.41 |
| MA[2] | 197.15 | 69282.43 | 263.21 | 13.78 |
| ARMA[2] | 219.24 | 81878.24 | 286.14 | 15.19 |
| ARIMA[2] | 181.19 | 58415.94 | 241.69 | 12.47 |
| SVR[2] | 230.65 | 93,023.06 | 304.99 | 16.12 |
| TASM[2] | 152.40 | 45911.11 | 214.26 | 10.72 |
| WFLS-LSTM* | **150.97** | **41747.34** | **204.32** | **9.09** |

[1] Resultados de (KUMAR K. GANGADHARA RAO, 2021); [2] Resultados de (SINGH, 2018); * Resultados deste trabalho.

A tabela 9 mostra a comparação entre os modelos treinados com a série temporal resolução de uma hora. A avaliação do desempenho do modelo proposto demonstrou uma melhora de 7.94% em relação à métrica RMSE do modelo comparado e 2.30% para a métrica MAE. Já para a métrica MAPE, houve um desempenho 3.40% inferior.

Tabela 9 – Comparação - série de resolução /hora

| Modelo | MAE | MSE | RMSE | MAPE |
|---|---|---|---|---|
| Naïve[1] | 5.37 | - | 7.41 | 28.01 |
| AR[1] | 4.46 | - | 6.44 | 22.90 |
| ARIMA (1, 1)[1] | 4.29 | - | 6.16 | **22.07** |
| ARIMA[1] | 4.51 | - | 6.33 | 23.96 |
| ETS[1] | 4.45 | - | 6.24 | 23.24 |
| GA[1] | 4.35 | - | 6.17 | 22.40 |
| WFLS-LSTM* | **4.25** | **32.21** | **5.68** | 22.82 |

[1] Resultados de (MESSIAS, 2016); * Resultados deste trabalho.

A figura 24 exibe as séries temporais em diferentes resoluções, sendo a original em azul, a série prevista durante o treinamento em laranja e a série prevista na etapa de testes na cor verde.

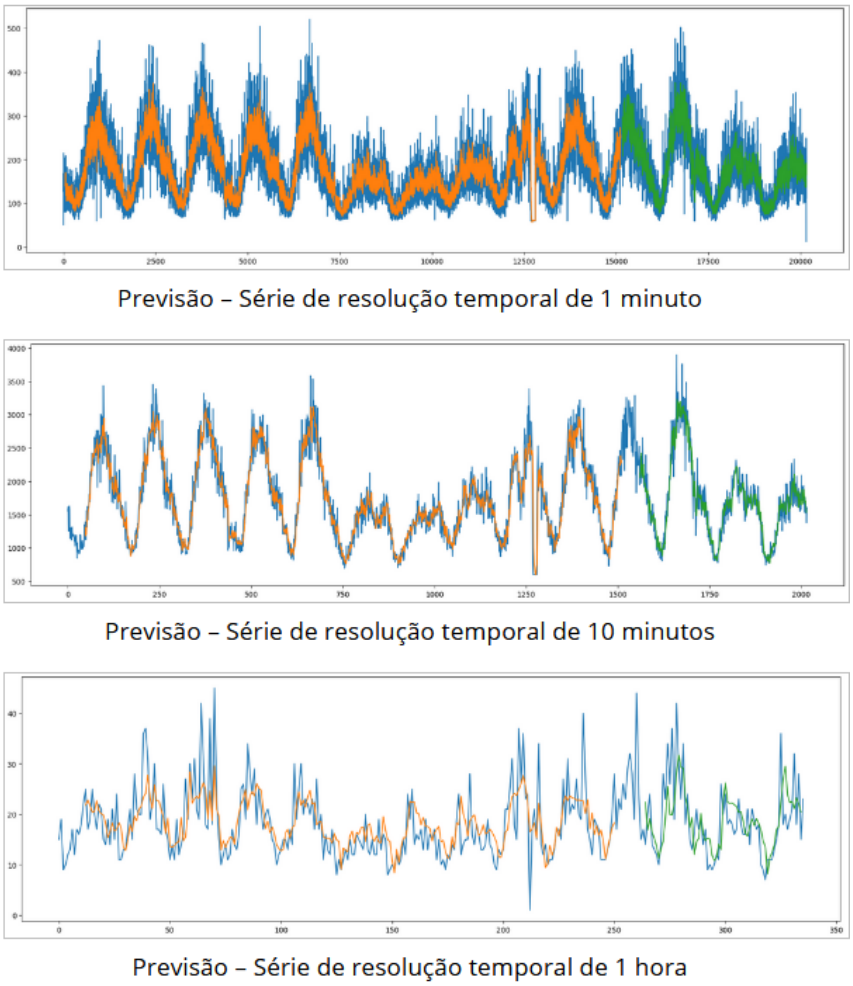


Figura 24 – Previsão - Série de resolução /hora

A tabela 10 mostra o tempo utilizado para treinamento e previsão das séries nas

diferentes resoluções:

Tabela 10 – Tempo de treinamento e previsão do modelo LSTM - Clarknet Traces

| Resolução (série) | Treinamento (segundos) | Previsão (segundos) |
|---|---|---|
| 1 min | 444.19 | 3.46 |
| 10 min | 208.95 | 0.65 |
| 1 hora | 35.77 | 0.44 |

#### 3.7.1.3 Discussão

Diante dos resultados, e em resposta à **QP1** - *Qual o impacto da granularidade temporal na eficácia dos modelos preditivos?* de acordo com as métricas avaliadas, podemos afirmar que o modelo LSTM obteve resultados com uma diferença mais expressiva na série temporal com resolução de 1 minuto, seguida pela série de 10 minutos e por último a série com resolução de 1 hora. Porém, a análise da série temporal indica a série com resolução de 10 minutos a de menor complexidade e a avaliação do modelo indica que o tempo de treinamento e previsão é mais razoável para se aplicar em contextos de previsões em tempo real, uma vez que a série com resolução de um minuto exibe um tempo consideravelmente maior tanto para treinamento quanto para produzir previsões (quanto maior o conjunto de dados, maior será o tempo gasto para essas atividades). Já a série de resolução de 1 hora, apesar de possuir um tempo de treinamento e previsão menor, obteve resultados menos expressivos em relação a outros modelos estatísticos tradicionais e de aprendizado de máquina. Em cenários onde não há limitações para reconfiguração de infraestrutura, como em ambientes On-Premises (em contraste com ambientes em nuvem), a resolução de 1 hora pode ser insuficiente para capturar e agir em variações de curto prazo.

Desta forma, a série com resolução de 10 minutos se destacou por oferecer uma combinação equilibrada entre precisão e eficiência de processamento, se mostrando mais viável para aplicação em previsões de carga de trabalho em tempo real, como acontece em sistemas de autoescalonamento proativos.

### 3.7.2 QP 2: Qual o impacto do LSTM para previsão de cargas de trabalho baseadas em quantidade de requisições?

Para responder a esta questão de pesquisa, foi realizada a análise da série temporal e realizados experimentos com a aplicação do modelo desenvolvido no conjunto de dados do Sistema de Gerência de Processos Seletivos da COPEVE.

A figura 25 representa graficamente as séries temporais geradas a partir dos dados de requisições HTTP do servidor.

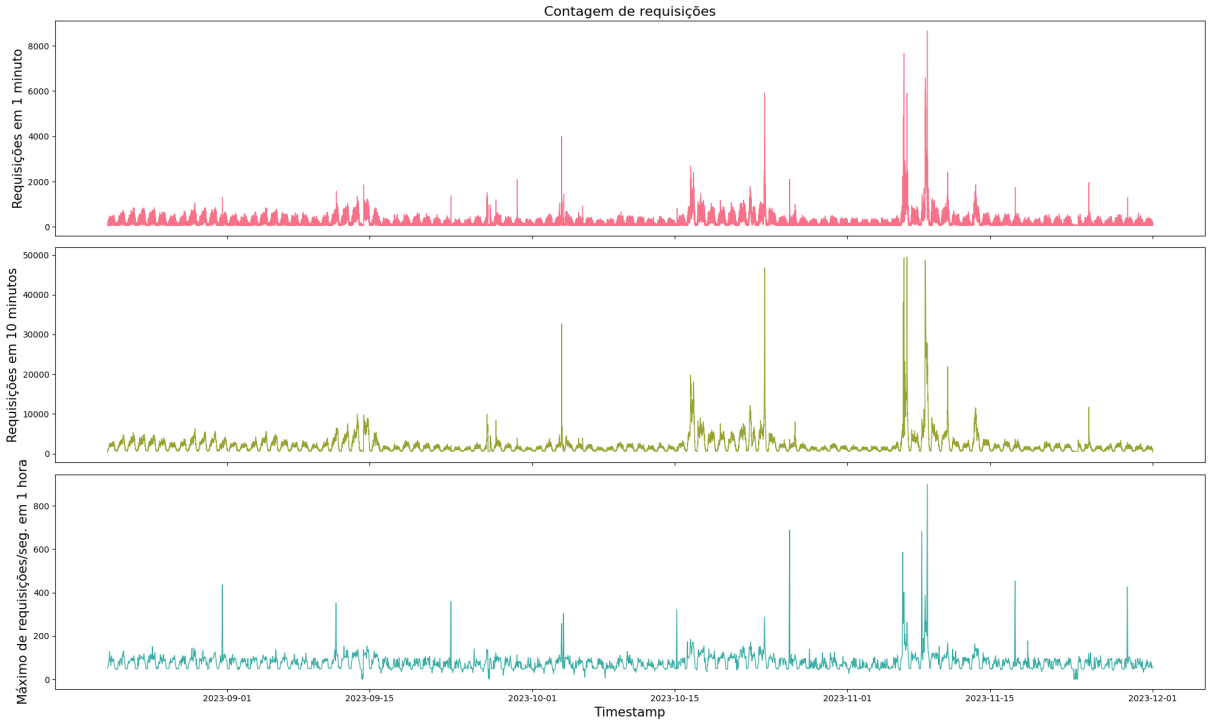


Figura 25 – Serie Temporal Copeve access log em diferentes resoluções

A **distribuição** analisada através dos gráficos de box-plot, 26, demonstra uma quantidade maior de requisições nos dias de semana e quantidade menor nos finais de semana. No gráfico de horário, a partir das 11h da manhã, há picos de acesso que se acentuam até as 13h, decaindo com menor intensidade até as 19h. Às 20h, inicia um novo ciclo de alta na taxa de requisições.

Além disso, no gráfico mensal, observa-se uma tendência de crescimento nos valores identificados como outliers neste gráfico que aumentam ao longo dos meses até novembro, no qual há eventos importantes nos editais de vestibulares. Tanto no gráfico semanal quanto no diário, esta tendência de crescimento é evidente, mas também é perceptível uma queda nas três últimas semanas observadas. A figura 26 demonstra este comportamento através da análise da distribuição dos dados ao longo do tempo pelos gráficos box plot.

A figura 26 representa esta análise para a série temporal empírica do sistema legado.

No gráfico mensal, observa-se uma tendência de crescimento nos valores mais comuns dentro de cada caixa, destacando-se os picos (identificados como outliers neste gráfico) que aumentam ao longo dos meses até novembro, no qual há eventos importantes nos editais de vestibulares. Tanto no gráfico semanal quanto no diário, esta tendência de crescimento é evidente, mas também é perceptível uma queda nas três últimas semanas observadas. Observa-se também uma maior quantidade de requisições nos dias normais de semana, com menor atividade nos finais de semana. Durante os dias úteis, os picos de

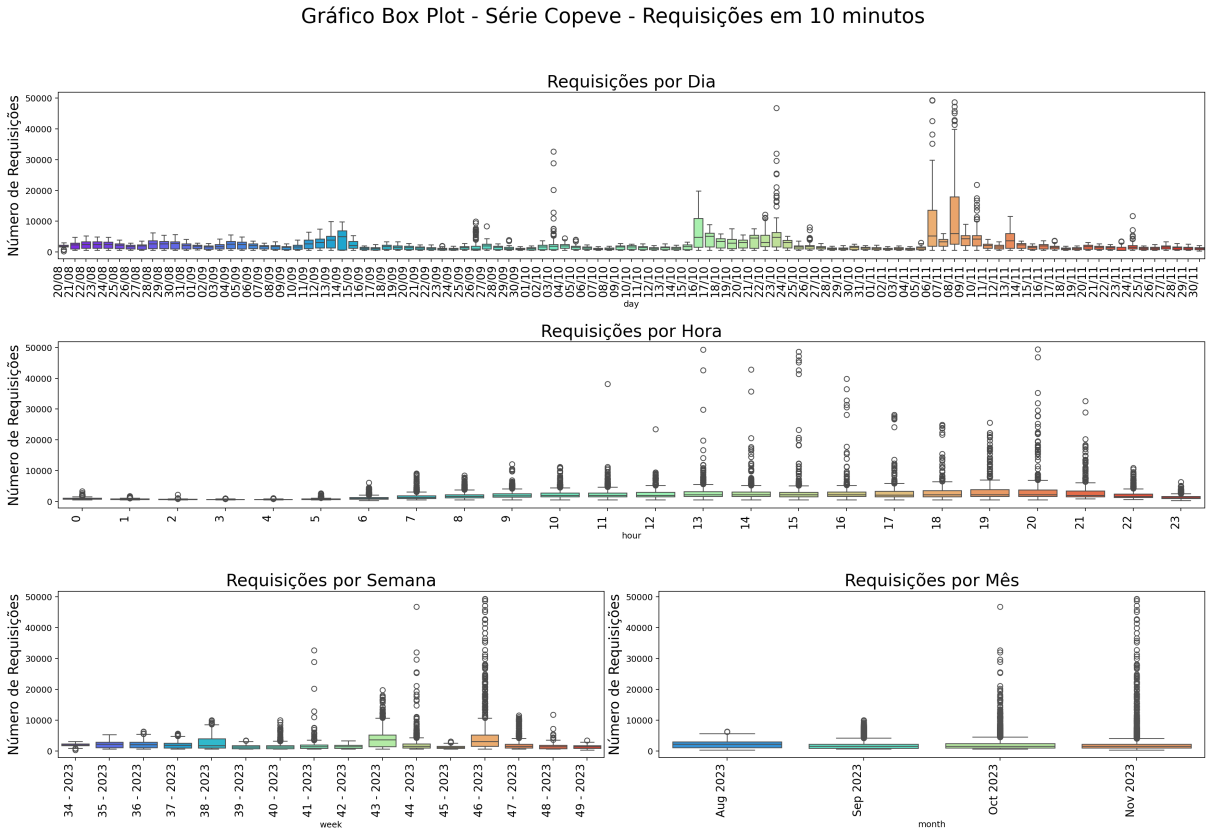


Figura 26 – Distribuição dos dados de requisições

acesso se iniciam a partir das 11h da manhã, atingindo o ápice por volta das 13h até as 15h, e diminuindo gradualmente até as 19h. Às 20h, inicia-se um novo ciclo de aumento na taxa de requisições.

A análise do gráfico de **decomposição** da série mostra a sazonalidade diária e variabilidade na tendência. Os resíduos mostram períodos de alta variabilidade, assim como ocorre com a tendência. Esta análise sugere que a série não é estacionária.

A **sazonalidade** foi testada por meio do gráfico de autocorrelação, figura 28, que apresentou picos fora da faixa de confiança nos lags múltiplos de 144 (144 períodos de 10 minutos são 24 horas), indicando uma forte autocorrelação dos dados em relação a este período.

A análise gráfica por meio do gráfico ACF para as séries temporais 1 minuto e 10 minutos apresentaram um decaimento lento indicando não estacionariedade, já a série com resolução de uma hora teve um caimento mais rápido no sexto lag, podendo indicar algum grau de estacionariedade. figura 29 demonstra este comportamento.

Em relação aos testes estatísticos ADF e KPSS, as tabelas 11, 12 e 13 descrevem os resultados para as séries temporais com resolução de 1 minuto, 10 minutos e 1 hora, respectivamente.

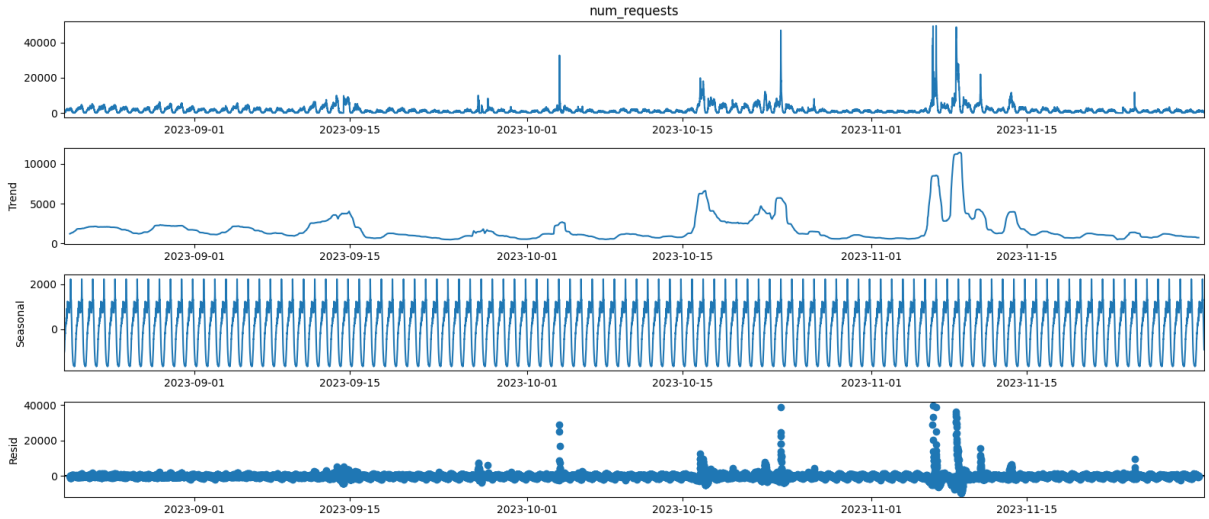


Figura 27 – Decomposição da série temporal

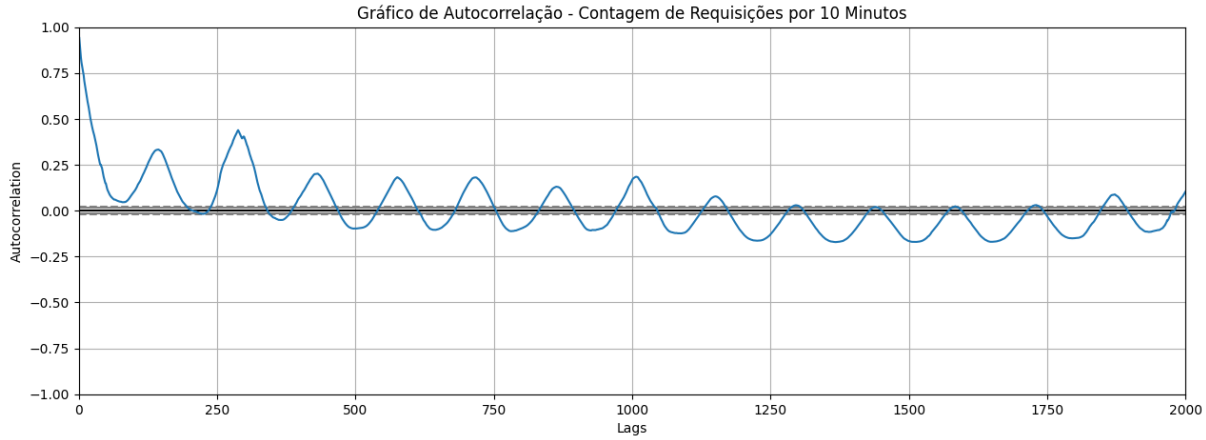


Figura 28 – Sazonalidade através do gráfico de autocorrelação

Tabela 11 – Resultados dos Testes de Estacionariedade - Série 1 minuto

| **Teste** | **Estatística** | **Valor p** | **Valores Críticos 1%, 5%, 10%** | Resultado |
|---|---|---|---|---|
| ADF | -14.9343 | $1.35 \times 10^{-27}$ | $-3.43, -2.86, -2.57$ | Estacionária |
| KPSS | 0.6714 | 0.0161 | 0.74, 0.46, 0.35 | Não Estacionária |

Conforme a tabela 11 para a série temporal com resolução de 1 minuto, a estatística do teste ADF (-14.934) é significativamente menor do que os valores críticos em todos os níveis de significância; o valor p é extremamente pequeno e menor do que qualquer nível usual de significância (0.01, 0.05, ou 0.10). Isto reforça a evidência para rejeitar a hipótese nula de que a série possui uma raiz unitária (ou seja, não é estacionária) e, portanto, concluir que a série é estacionária. Já o teste KPSS sugere que a série é não estacionária nos níveis de significância de 5% e 10%, mas estacionária no nível de 1%,

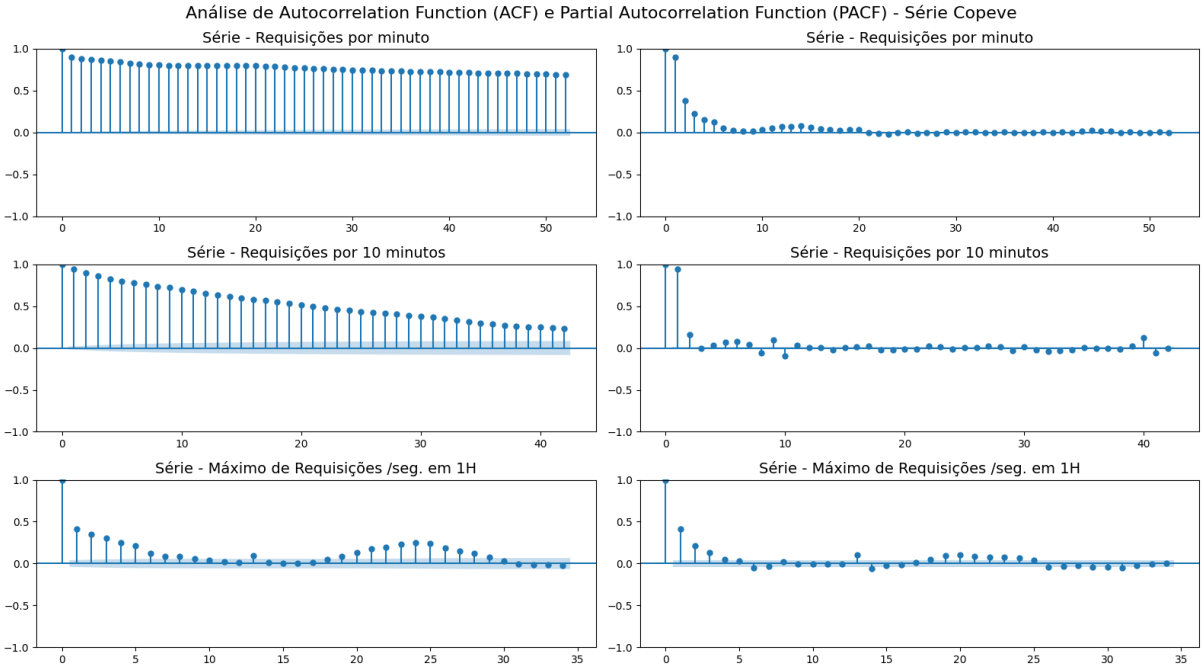


Figura 29 – Autocorrelação - Copeve em diferentes resoluções

pois a estatística do teste (0.671) é maior do que os valores críticos nos níveis de 5% e 10%, mas é menor do que o valor crítico no nível de 1%. demonstra este resultado.

Tabela 12 – Resultados dos Testes de Estacionariedade - Série 10 min.

| **Teste** | **Estatística** | **Valor p** | **Valores Críticos 1%, 5%, 10%** | Resultado |
|---|---|---|---|---|
| ADF | -12.9888 | $2.84 \times 10^{-24}$ | $-3.43, -2.86, -2.57$ | Estacionária |
| KPSS | 0.3220 | 0.1 | 0.74, 0.46, 0.35 | Estacionária |

Para a série temporal com resolução de 10 minutos, de acordo com os resultados exibidos na tabela 12, a estatística do teste (-12.989) também é significativamente menor (mais negativa) do que os valores críticos em todos os níveis (1%, 5%, e 10%); o valor p é extremamente pequeno, indicando que podemos rejeitar a hipótese nula de que a série tem uma raiz unitária, podendo concluir que a série é estacionária. Para o teste KPSS, a estatística do teste (0.322) é menor do que os valores críticos para todos os níveis de significância (1%, 5% e 10%); além disso, o p-value é maior que 0.05, a hipótese nula não é rejeitada, indicando que a série é estacionária.

Para a série temporal com resolução de 1h, a estatística do teste ADF (-5.5249) é significativamente menor que os valores críticos e o valor p é muito pequeno, reforçando que a hipótese nula pode ser rejeitada, podendo concluir que a série é estacionária. Para o teste KPSS, a estatística (0.2382) é menor que os valores críticos e o valor p é menor que 0,05 sugerindo que a série é estacionária.

Tabela 13 – Resultados dos Testes de Estacionariedade - Série 1 hora.

| **Teste** | **Estatística** | **Valor p** | **Valores Críticos 1%, 5%, 10%** | Resultado |
|---|---|---|---|---|
| ADF | -5.5249 | $1.84 \times 10^{-6}$ | $-3.43, -2.86, -2.57$ | Estacionária |
| KPSS | 0.2382 | 0.1 | 0.74, 0.46, 0.35 | Estacionária |

A tabela 14 descreve os resultados para o Sample Entropy para as séries temporais nos diferentes intervalos de tempo.

Tabela 14 – Resultados de Sample Entropy em Diferentes Intervalos Temporais

| **Intervalo Temporal** | **Sample Entropy** |
|---|---|
| Por minuto | 0.4456 |
| Por 10 minutos | 0.2266 |
| Por hora | 0.8439 |

#### 3.7.2.1 Considerações sobre a Análise da Série Temporal

Assim como ocorre na análise da série temporal do conjunto de dados Clarknet traces, os testes estatísticos ADF e KPSS indicam estacionariedade nas séries, com exceção da série de resolução de 1 minuto, para a qual o teste KPSS sugere uma possível não estacionariedade. Enquanto o teste ADF sugere estacionariedade em relação à tendência constante, o teste KPSS aponta para a estacionariedade em relação à autocorrelação de primeira ordem, exceto para a série de 1 minuto. Além disso, a análise visual também revela a presença de sazonalidade, um componente que também sugere não estacionariedade com periodicidade de 144 para a série temporal de 10 minutos.

Da mesma forma que ocorre com os dados do Clarknet, as análises de autocorrelação, através dos gráficos ACF, das séries de 1 minuto e 10 minutos, mostram um comportamento de caimento lento, reforçando a indicação de não estacionariedade e refletindo a complexidade do comportamento dessas séries temporais empíricas.

Conforme observado também na análise do conjunto de dados Clarnet Traces, a série temporal com resolução de 10 minutos do conjunto de dados da COPEVE demonstrou uma melhor previsibilidade em relação às séries por segundo e por minuto, resultando em 0.2379, indicando menor complexidade e/ou previsibilidade.

A análise do conjunto de dados do Sistema de Gerência de Concursos da COPEVE indica certa volatilidade em alguns períodos e também evidencia incertezas quanto à estacionariedade dos dados com a presença de componentes sazonais e tendência variável ao longo do tempo. Essencialmente, quando a variância muda ao longo do tempo ("volatilidade"), as séries temporais não seguem um modelo linear. Conforme (MEISENBACHER,

2022), embora a maioria dos métodos estatísticos de previsão se baseie em suposições sobre a distribuição dos dados de séries temporais, os métodos de aprendizado de máquina têm menos restrições em termos de linearidade e estacionariedade.

Como a análise demonstra propriedades muito semelhantes às encontradas na série Clarknet Traces, o mesmo modelo utilizado para avaliação da previsão será utilizado para o conjunto de dados da COPEVE. e comparados aos modelos estatísticos tradicionais, como AutoRegressive (AR) e AutoRegressive Integrated Moving Average (ARIMA), que obtiveram bons resultados nos trabalhos relacionados.

#### 3.7.2.2 Avaliação do Modelo

Ao reamostrar a série temporal empírica do sistema legado, agrupando as requisições que ocorreram em intervalos de 10 minutos, o número de observações foi reduzido de 8.875.129 para 14.793. É esperado que o valor do RMSE para este modelo seja muito mais alto comparado ao RMSE do modelo treinado com a série temporal de benchmark. Isso ocorre porque a série é aproximadamente sete vezes maior que a série de benchmark, possui uma escala de valores 20 vezes superior e apresenta picos repentinos nos dados. O RMSE é uma medida de erro que leva em consideração a diferença entre os valores previstos e os valores reais, elevando essa diferença ao quadrado. Se a série for maior em escala, as diferenças entre os valores previstos e reais também serão maiores em termos absolutos, resultando em um RMSE maior.

A abordagem de treinamento usando o particionamento convencional(80%/20%), sem a implementação da validação cruzada, resultou em um desempenho inferior para o conjunto de treinamento em comparação com as estratégias de validação cruzada. No entanto, para o conjunto de teste, o desempenho foi superior. Esse resultado para o conjunto de teste indica que a capacidade de generalização do modelo treinado pela estratégia de particionamento convencional foi melhor, sugerindo uma menor tendência ao overfitting neste caso.

A Tabela 15 apresenta a avaliação do modelo LSTM em comparação com modelos estatísticos clássicos, que obtiveram um bom desempenho nos trabalhos relacionados. O modelo LSTM demonstrou desempenho superior nas métricas MAPE e MAE, com uma redução de 11.06% no MAPE e 4.75% no MAE em relação ao modelo ETS. No entanto, nas métricas RMSE e MSE, o modelo ETS apresentou o melhor desempenho, sendo 8.90% e 17.02% melhor que o LSTM, respectivamente.

A figura 30 exibe a série do conjunto de teste e a série prevista utilizando a estratégia de particionamento convencional para o modelo LSTM. Através da análise gráfica, percebe-se um bom ajuste do padrão de sazonalidade diária e também em momentos de picos de utilização.

Tabela 15 – Avaliação do modelo

| Modelo | RMSE | MSE | MAE | MAPE |
|---|---|---|---|---|
| ETS | **850.98** | **724167.49** | 386.67 | 16.46 |
| ARIMA | 966.28 | 933702.02 | 386.90 | 19.53 |
| WFLS-LSTM | 934.17 | 872672.35 | **368.30** | **14.64** |

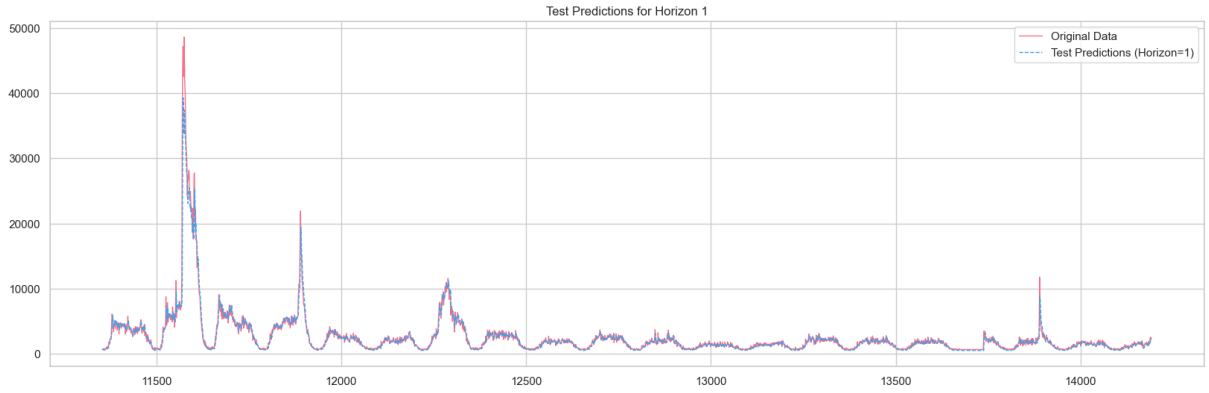


Figura 30 – Previsão no conjunto de teste - sem validação cruzada

A Tabela 16 exibe a avaliação do modelo utilizando previsões diretas (direct forecasting) para os próximos 6, 12, 48 e 144 passos, onde cada previsão é feita para um ponto futuro específico, sem depender das previsões anteriores. Devido à capacidade de capturar padrões não lineares complexos nos dados, modelo LSTM obteve desempenho superior aos modelos estatísticos clássicos na estratégia de previsão direta para todos os horizontes de previsão analisados.

Tabela 16 – Avaliação do modelo para diferentes horizontes de previsão H

| Modelo | H | RMSE | MSE | MAE | MAPE |
|---|---|---|---|---|---|
| ETS | 6 | 1906.05 | 3633032.77 | 1840.26 | 273.19 |
| | 12 | 1659.03 | 2752394.03 | 1597.76 | 230.43 |
| | 48 | 3681.50 | 13553513.67 | 3113.67 | 128.31 |
| | 144 | 3237.67 | 10482571.64 | 2770.48 | 98.22 |
| ARIMA | 6 | 337.00 | 113572.17 | 319.61 | 47.41 |
| | 12 | 473.55 | 224250.77 | 447.27 | 63.33 |
| | 48 | 1879.95 | 3534238.48 | 1472.32 | 52.29 |
| | 144 | 1751.76 | 3068676.17 | 1500.60 | 53.92 |
| WFLS-LSTM | 6 | **213.81** | **45713.88** | **185.04** | **27.98** |
| | 12 | **145.19** | **21080.05** | **125.65** | **18.05** |
| | 48 | **780.10** | **608562.09** | **529.40** | **27.22** |
| | 144 | **758.12** | **574743.36** | **552.90** | **19.70** |

Adicionalmente, foi realizada a avaliação do tempo necessário para ajuste ou treinamento dos modelos ARIMA, ETS e LSTM, juntamente com a previsão para todo o conjunto de teste. O tempo dispendido foi expresso em segundos, conforme mostrado na Tabela 17.

Tabela 17 – Tempo de ajuste/treinamento e previsão

| Modelo | Tempo (segundos) |
|---|---|
| ARIMA | 1047.16 |
| ETS | 830.27 |
| WFLS-LSTM | 668.01 |

#### 3.7.2.3 Discussão

Em resposta à **QP2** - *Qual o impacto do LSTM para previsão de cargas de trabalho baseadas em quantidade de requisições?* Na análise do conjunto de dados de benchmark, o LSTM manteve sua superioridade em praticamente todas as resoluções temporais avaliadas, exceto para as previsões da série com resolução de 1 hora, onde o ARIMA apresentou MAPE inferior. Entretanto, ao examinar os dados empíricos do sistema da COPEVE, os resultados revelaram que, embora o LSTM tenha apresentado melhor desempenho nas métricas absolutas (MAE e MAPE), os modelos ETS se destacaram nas métricas quadráticas (RMSE e MSE), que são mais sensíveis a grandes erros.

A superioridade do LSTM nas métricas absolutas como MAE e MAPE evidencia sua capacidade de reduzir erros médios, resultando em maior precisão geral das previsões. Por outro lado, seu desempenho inferior nas métricas quadráticas (RMSE e MSE) indica uma menor propensão em capturar picos abruptos e outliers com a mesma proporção que os modelos ETS, especialmente em conjuntos de dados caracterizados por alta variabilidade.

No contexto de implementação em sistemas de escalonamento automático, onde as previsões devem ser executadas periodicamente e em tempo real, o LSTM apresenta duas vantagens significativas:

- Eficiência operacional: A necessidade reduzida de retreinamento/reajuste em comparação com os modelos estatísticos analisados (ARIMA e ETS) resulta em menor custo operacional, mesmo considerando a complexidade inerente das redes neurais recorrentes.
- Capacidade de realizar previsões multi-horizonte mais precisas: possibilita previsões de tendências da carga de trabalho com boa performance, viabilizando a implementação de políticas proativas de escalonamento automático mais sofisticadas.

## 3.8 Ameaças à validade

Para garantir a validade dos resultados do nosso estudo, consideramos várias ameaças potenciais à validade e as abordamos da seguinte forma:

**Validade Interna.** A validade interna pode ser ameaçada por erros na coleta e análise de dados. Para mitigar isso, usamos fontes de dados confiáveis, obtidas a partir do canal oficial para a base de benchmark e diretamente através dos logs de acesso do servidor para o sistema real.

**Validade Externa.** Uma potencial ameaça é a limitação na generalização dos resultados para diferentes tipos de sistemas, pois o estudo está centrado em sistemas legados disponíveis publicamente na internet. No entanto, a análise das séries temporais dos dois conjuntos de dados, apesar de suas diferentes escalas (0-45 e 0-900 requisições/segundo), demonstra que eles compartilham propriedades estatísticas semelhantes, como estacionariedade e sazonalidade. Esta similaridade sugere que o modelo pode ser aplicável a outros sistemas web que exibam estes mesmos padrões, embora possa ter limitações para sistemas com padrões de uso muito diferentes, como sistemas internos corporativos ou sistemas com acesso controlado.

**Validade de Construto.** A escolha da quantidade de requisições como principal métrica pode ser vista como uma ameaça, pois não considera diretamente aspectos como o consumo de recursos. No entanto, esta simplificação oferece benefícios computacionais significativos na implementação de abordagens proativas de autoescalonamento, pois o comportamento dos usuários está diretamente relacionado ao consumo de recursos. Outras métricas como tempo de resposta e uso de recursos podem ser utilizadas em diferentes componentes do sistema de autoescalonamento, como em abordagens reativas.

# 4 Sistema de Autoescalonamento Autonômico Proativo

Este capítulo aborda a Questão de Pesquisa 3, que investiga se sistemas de autoescalonamento autonômico com capacidades proativas, baseados em modelos de previsão de carga de trabalho por quantidade de requisições, podem proporcionar uma abordagem eficiente para o dimensionamento automático de recursos em sistemas legados implantados em data centers locais. Para isso, foi conduzido um estudo explicativo com abordagem quantitativa e epistemologia positivista, utilizando a metodologia de estudo de caso em um sistema legado real no ambiente on-premises[1].

A figura 31 exibe o fluxo metodológico adotado neste trabalho incluindo as seguintes etapas: Inicialmente o sistema é implementado seguindo o modelo MAPE-K através de seus quatro componentes principais interligados por uma base de conhecimento compartilhada (Analyze, Monitor, Plan, Execute e Knowledge). Em seguida, é realizada a execução dos experimentos que envolve a preparação do ambiente com a configuração do sistema e a execução de testes de carga, e também analisando a carga real. Durante a execução, métricas são coletadas continuamente e armazenadas em logs para posterior análise. A análise dos resultados se inicia com o processamento dos logs coletados, seguida pela geração de visualizações que permitem identificar padrões de comportamento do sistema. Os resultados foram comparados com uma linha de base (baseline) para avaliar as melhorias obtidas.

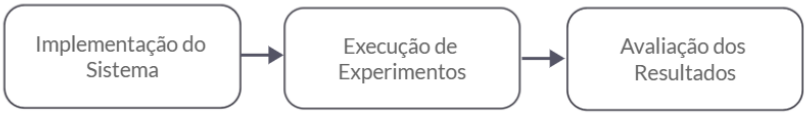


Figura 31 – Metodologia

## 4.1 Contextualização do Ambiente

O Sistema de Gestão dos Processos Seletivos da COPEVE/CEFET-MG é uma aplicação legada, voltada para o público e hospedada em um data center local. Devido à natureza do sistema e à falta de um mecanismo de escalonamento automático de recursos, as equipes de operações de TI enfrentam alguns desafios referentes à variação e

[1] Código e dados disponíveis em: https://doi.org/10.5281/zenodo.14504203

sazonalidade na carga de trabalho do sistema. Durante períodos de alta demanda, como inscrições e divulgação de resultados, os recursos são frequentemente superdimensionados para garantir que o sistema suporte o tráfego. No entanto, essa abordagem leva ao desperdício de recursos computacionais e energia, pois, em momentos de menor demanda, os recursos permanecem subutilizados.

A infraestrutura é composta por seis máquinas virtuais. Nas três primeiras, a aplicação é instalada diretamente no sistema operacional. Nas outras três, uma versão específica da aplicação, focada apenas no módulo de resultados, é instalada utilizando contêineres Docker. A figura 32 apresenta a infraestrutura na qual o ambiente da aplicação roda.

A infraestrutura utilizada é composta por seis máquinas virtuais, cada uma configurada com 8 CPUs e 8 GB de memória RAM. Nas três primeiras, a aplicação é instalada diretamente no sistema operacional. Nas outras três, uma versão específica da aplicação, focada apenas no módulo de resultados, é implantada em contêineres Docker, permitindo maior flexibilidade de escalonamento horizontal. A Figura 32 apresenta a arquitetura detalhada desta infraestrutura, evidenciando como o ambiente foi estruturado para acomodar diferentes cenários de uso.

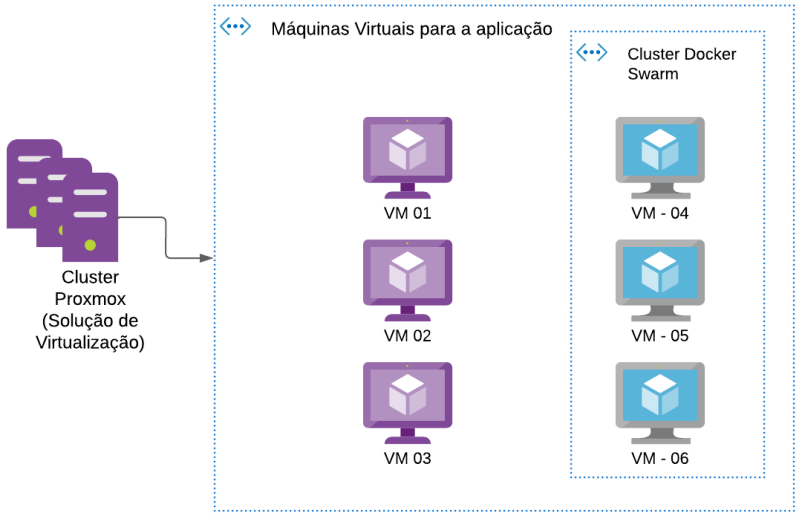


Figura 32 – Infraestrutura que roda a aplicação

Historicamente, o sistema tem sofrido com a violação dos níveis de serviço, resultando em tempos de resposta elevados e sobrecarga que provoca quedas nas instâncias de aplicação, especialmente durante eventos críticos. Esses problemas afetam tanto os usuários, que enfrentam dificuldades de acesso, quanto a instituição, que é alvo de reclamações. O dimensionamento dos recursos, por não ser automatizado, depende de estimativas manuais feitas pela equipe de operações, baseadas em dados históricos e variáveis como o número de candidatos de cada edital de processo seletivo. Contudo, essas estimativas

frequentemente falham devido a imprevistos, como prorrogações de prazos ou comportamentos inesperados de usuários.

Quando ocorre sobrecarga, o escalonamento dos recursos também é feito manualmente, seja aumentando a capacidade de CPU e memória (escalonamento vertical), seja adicionando novas máquinas virtuais (escalonamento horizontal). Embora o escalonamento horizontal seja mais eficaz, ele demanda mais tempo para ser implementado quando é realizado de forma manual. Além disso, ele é realizado apenas em momentos críticos, deixando os recursos superdimensionados na maior parte do tempo, o que resulta em custos operacionais desnecessários.

A figura 33 representa em azul os dados de tempo de resposta médio da aplicação ao longo do tempo, apresentando picos abruptos em momentos de alta carga. A linha vermelha tracejada mostra o tempo médio de resposta máximo ideal para atender aos critérios de qualidade no acesso ao sistema sem violar acordos de nível de serviço (2 segundos). Com base nos cálculos de expectativa de carga pela equipe de operações, o escalonamento manual é destacado pela linha verde, com a contagem de instâncias alocadas para o sistema em determinado período.

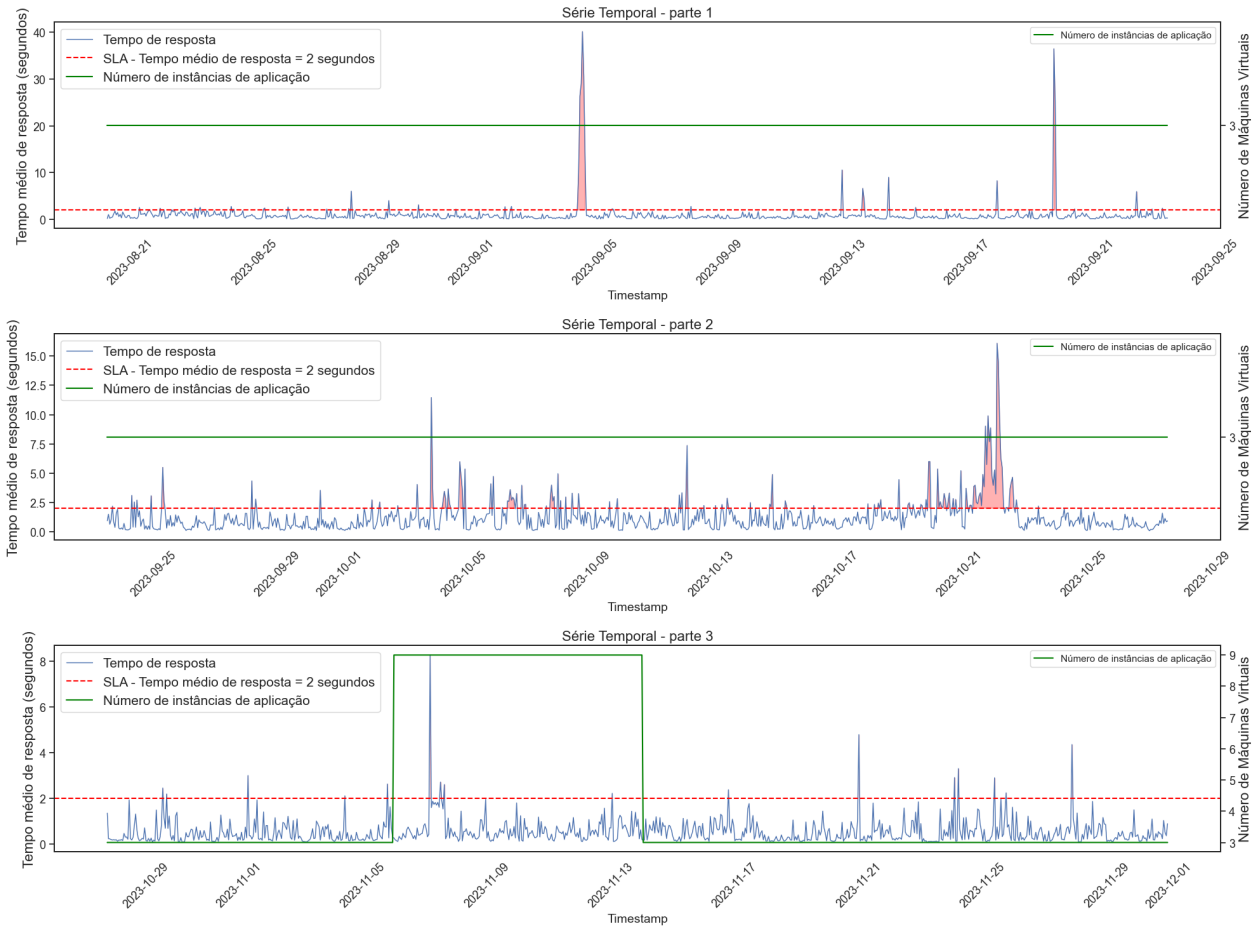


Figura 33 – Histórico do Padrão de carga e tempo de resposta da aplicação

De acordo com esses dados, observa-se que o escalonamento manual para o período de alta demanda, a partir de 06/11/2023, reduziu efetivamente as violações de nível de serviço. No entanto, ainda existem oportunidades significativas de otimização, como evidenciado pela análise dos seguintes períodos:

- **Períodos de Sobrecarga e Violações de SLA:** foram identificados diversos momentos em que o dimensionamento de recursos foi insuficiente para manter os níveis de serviço adequados, como por exemplo, em setembro, foram registrados quatro períodos críticos: entre os dias 04 e 05, 12 e 14, 18 e 20, além do dia 25. Outubro apresentou a maior concentração de incidentes, com sobrecargas significativas entre os dias 03 e 08, 11 e 12, no dia 15, e um período prolongado entre 19 e 23. Em novembro, foram observados dois períodos principais de sobrecarga: entre os dias 20 e 21, e um intervalo mais extenso entre 24 e 29. Em todas estas ocasiões, um ajuste preventivo no dimensionamento poderia ter mitigado os impactos negativos dessas demandas inesperadas.

- **Períodos de Subutilização de Recursos:** em contrapartida, a análise ambém foram identificados momentos de baixa carga nos quais a escala poderia ter sido reduzida para otimizar o uso de recursos. Por exemplo, em agosto, foi registrado um dia de baixa utilização no dia 26. Setembro apresentou múltiplos períodos de ociosidade: no dia 03, dia 05, entre 08 e 09, 11 e 12, e no dia 26. Em outubro, a subutilização foi observada no dia 01, entre os dias 16 e 17, 23 e 24, e um período mais extenso entre 26 e 28. Novembro demonstrou uma tendência mais consistente de baixa utilização, especialmente a partir do dia 09, indicando uma oportunidade sustentada para otimização de recursos.

  Durante esses períodos de baixa demanda, a redução de recursos seria adequada para a redução de custos e liberação de recursos para outros sistemas.

A adoção de uma metodologia de escalonamento automático poderia mitigar esses problemas ao ajustar dinamicamente os recursos conforme a demanda. Isso permitiria a liberação automática de recursos durante períodos de baixa utilização, otimizando o consumo de energia e recursos computacionais. Nos momentos de alta demanda, o sistema aumentaria automaticamente sua capacidade, mantendo a qualidade do serviço, independentemente do horário ou da intervenção manual da equipe de operações. Como resultado, haveria uma redução no esforço da equipe para monitorar a carga de trabalho e ajustar os recursos, liberando-os para se concentrar em atividades mais estratégicas e melhorando a eficiência operacional da instituição.

## 4.2 Implementação do Sistema

O sistema desenvolvido para realizar o autoescalonamento proativo implementa o ciclo MAPE-K da computação autonômica, sendo composto pelos seguintes módulos: (i) Monitor - sensores que coletam dados em tempo real sobre métricas de carga, tempo de resposta e recursos em uso do sistema; (ii) Analyze - módulo de análise que realiza previsões da carga futura e disponibiliza essas previsões através de uma API; (iii) Plan - módulo de planejamento que atua como calibrador para determinar a quantidade adequada de instâncias necessárias com base nos dados coletados; e (iv) Execute - módulo executor que efetivamente realiza as ações de escalonamento, aumentando ou diminuindo os recursos conforme planejado. A base de conhecimento (Knowledge Base) se refere ao histórico de logs, séries temporais, modelos de previsão e as políticas de escalonamento produzidas. Adicionalmente, o sistema possui um módulo para geração de cargas sintéticas, permitindo a construção das políticas de escalonamento de inicialização do sistema por meio da avaliação de diferentes cenários de carga. A figura 34 representa graficamente esta arquitetura autonômica do sistema. A seguir, é descrito com detalhes cada módulo do sistema de escalonamento proativo desenvolvido.

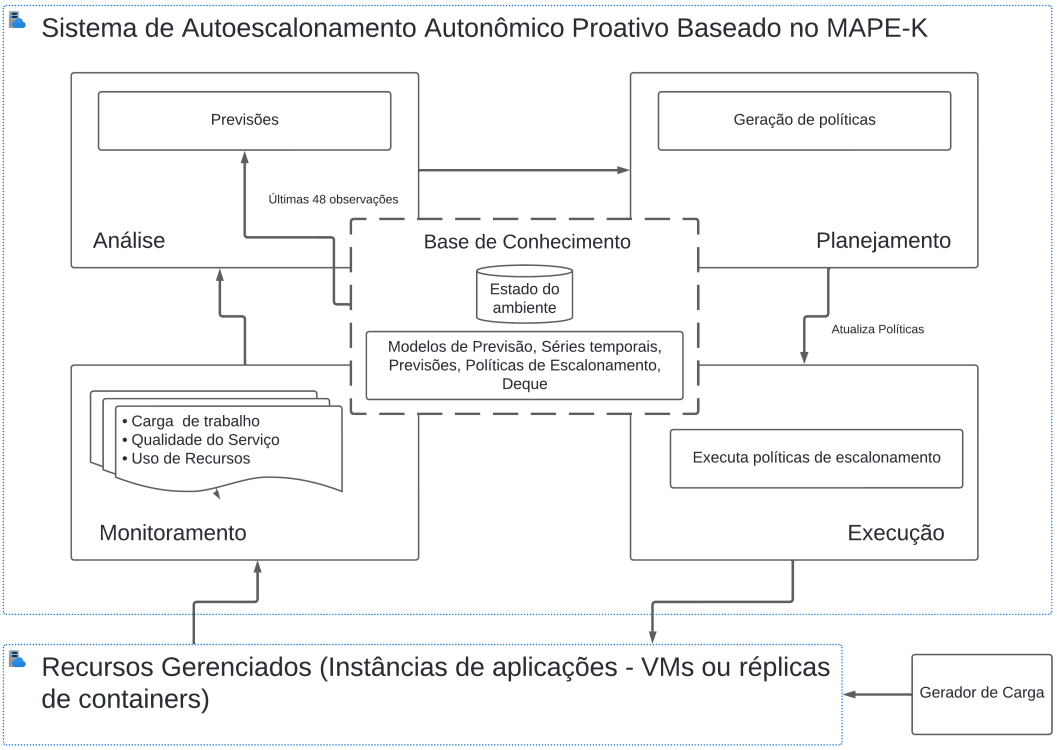


Figura 34 – Diagrama de Arquitetura do Sistema de Autoescalonamento

### 4.2.1 Módulo de Monitoramento (Monitor)

Este componente implementa a fase de monitoramento do ciclo MAPE-K, sendo responsável por coletar e agregar dados em tempo real sobre três aspectos fundamentais do sistema gerenciado: carga de trabalho (quantidade de requisições), qualidade de serviço (tempo médio de resposta) e estado dos recursos computacionais (VMs e contêineres). Essas métricas constituem os dados primários que alimentam a Knowledge Base e são fundamentais para o funcionamento do ciclo autonômico.

O Monitor implementa diferentes estratégias de coleta para cada tipo de métrica, atuando como sensores:

- **Monitoramento de carga:** Realiza requisições HTTP GET periódicas ao módulo de status do servidor Apache 2, extraindo a métrica "Total Accesses" que reflete a carga de trabalho atual do sistema.
- **Monitoramento de QoS:** Utiliza um exportador de métricas do Prometheus [2] (RABENSTEIN; VOLZ, 2015) que processa continuamente os logs de acesso do Apache para calcular o tempo médio de resposta das requisições HTTP.
- **Monitoramento de recursos:** Integra-se às APIs do Proxmox[3] e Docker Swarm[4] para coletar o estado operacional das VMs e informações sobre contêineres, incluindo o número de réplicas da aplicação.

Os dados coletados são disponibilizados através de uma API REST para consumo pelos demais componentes do ciclo MAPE-K. Esta organização permite que o sistema autonômico mantenha uma visão atualizada e precisa do ambiente gerenciado.

O algoritmo 2 detalha a implementação do monitoramento de QoS que calcula o tempo médio de resposta, uma das métricas essenciais monitoradas pelo sistema.

[2] Prometheus - Sistema de monitoramento e banco de dados de séries temporais https://prometheus.io/

[3] Proxmox Virtual Environment - Open-source server virtualization management platform https://www.proxmox.com/

[4] Docker Swarm API - Native clustering and orchestration for Docker https://docs.docker.com/engine/swarm/

**Algorithm 2** Monitoramento e Atualização da Métrica Tempo Médio de Resposta

```
1: function LER_LOG_ARQUIVO(caminho_log)
2:     Abre o arquivo de log
3:     Lê as novas entradas
4:     Retorna as entradas lidas
5: end function
6: function AGRUPAR_TEMPOS_RESPOSTA(entradas)
7:     Cria um dicionário para armazenar os tempos de resposta por intervalo
8:     for cada entrada em entradas do
9:         Extrai o timestamp e o tempo de resposta
10:        Define o intervalo de 10 minutos correspondente ao timestamp
11:        Adiciona o tempo de resposta ao dicionário no intervalo correto
12:    end for
13:    Retorna o dicionário com os tempos agrupados
14: end function
15: function CALCULAR_MEDIA_RESPOSTA(tempos_agrupados)
16:    Cria um dicionário para armazenar as médias
17:    for cada intervalo em tempos_agrupados do
18:        Calcula a média dos tempos de resposta no intervalo
19:        Armazena a média no dicionário
20:    end for
21:    Retorna o dicionário com as médias
22: end function
23: procedure MONITORAR_METRICAS
24:    while o sistema estiver em execução do
25:        entradas ← LER_LOG_ARQUIVO(caminho_log)
26:        tempos_agrupados ← AGRUPAR_TEMPOS_RESPOSTA(entradas)
27:        medias ← CALCULAR_MEDIA_RESPOSTA(tempos_agrupados)
28:        Atualiza métricas no sistema de monitoramento com as médias
29:        Aguarda o intervalo de atualização
30:    end while
31: end procedure
```

### 4.2.2 Módulo de Análise (Analyze)

O componente de análise do ciclo MAPE-K é responsável por gerar previsões de carga futura utilizando modelos preditivos. Este módulo processa os dados coletados pelo Monitor, organizando as últimas 48 observações de quantidade de requisições em uma estrutura de fila. Antes de alimentar os modelos de previsão, os dados são normalizados utilizando as propriedades de variância da série temporal de amostra armazenada na Knowledge Base.

A implementação utiliza múltiplas *threads* para garantir a execução paralela e não-bloqueante das diferentes responsabilidades do módulo: consumo dos dados do Monitor, processamento das previsões e atualização da estrutura de fila na Knowledge Base. O módulo pode gerar previsões em diferentes horizontes temporais, permitindo a elaboração

de estratégias de escalonamento mais sofisticadas, caso necessário.

Para comunicação com os demais componentes do ciclo MAPE-K, o sistema expõe uma API *REST* que disponibiliza as previsões realizadas. A seguir, o algoritmo 3 demonstra a lógica de funcionamento do módulo.

**Algorithm 3** Módulo de Analise (Previsão)

```
1:  procedure ColetarContagemRequisicoes
2:      while verdadeiro do
3:          requisiçõesAtuais ← ObterContagemRequisicoesApache()
4:          if requisiçõesAtuais é válido then
5:              diferençaRequisicoes ← requisiçõesAtuais - requisicoesAnteriores
6:              requisicoesRecentes.Adicionar(diferençaRequisicoes)
7:              SalvarDeque(requisicoesRecentes)
8:              requisicoesAnteriores ← requisiçõesAtuais
9:          end if
10:         Aguardar(600 segundos)
11:     end while
12: end procedure
13: procedure PreverRequisicoes(horizonte)
14:     if Comprimento(requisicoesRecentes) < 48 then
15:         return Erro("Dados insuficientes para previsão")
16:     end if
17:     valores ← Normalizar(requisicoesRecentes)
18:     modelo ← SelecionarModelo(horizonte)
19:     previsaoNormalizada ← modelo.Predizer(valores)
20:     previsaoDesnormalizada ← Desnormalizar(previsaoNormalizada)
21:     return previsaoDesnormalizada
22: end procedure
```

### 4.2.3 Módulo de Planejamento (Plan)

O módulo de Planejamento (Plan) do ciclo MAPE-K é responsável por determinar a quantidade adequada de instâncias da aplicação para garantir o Acordo de Nível de Serviço (SLA). Este módulo processa os dados coletados pelo Monitor e planeja políticas de configuração de recursos que serão executadas pelo módulo Executor.

O Planejador utiliza dados monitorados a cada minuto, incluindo:

- Estado dos recursos gerenciados (VMs e contêineres em execução)
- Métricas de performance (tempo médio de resposta)
- Métricas de carga (total de requisições)

Durante o planejamento, quando identificado que o tempo de resposta médio ultrapassa 2 segundos, uma violação de SLA é registrada junto com sua probabilidade de ocorrência.

O Planejador gera políticas de escalonamento para diferentes configurações de recursos. A cada 10 minutos, processa as informações coletadas e planeja uma nova política de auto-configuração. Uma política é representada por:

$$P_{\text{instances}} = (T_{\text{avg}}, R_{\text{total}}, V_{\text{count}}, C_{\text{count}}, P_{\text{violation}}) \tag{4.1}$$

Onde:

- $T_{\text{avg}}$ é o tempo de resposta médio ponderado, calculado como:

$$T_{\text{avg}} = \frac{\sum_{i=1}^{n} r_i \cdot t_i}{R_{\text{total}}} \tag{4.2}$$

  onde $r_i$ é o número de requisições no $i$-ésimo período de coleta, $t_i$ é o tempo de resposta médio desse período e $R_{\text{total}}$ é o número total de requisições.

- $R_{\text{total}}$ é o número total de requisições, calculado como:

$$R_{\text{total}} = \sum_{i=1}^{n} r_i \tag{4.3}$$

- $V_{\text{count}}$ é a quantidade de VMs em execução
- $C_{\text{count}}$ é a quantidade de réplicas de contêineres em execução
- $P_{\text{violation}}$ é a probabilidade de ocorrência de violações de SLA, calculada como a razão entre o número de violações e o número total de períodos de coleta

As políticas geradas são armazenadas na Knowledge Base para serem utilizadas pelo módulo Executor. Para calcular a quantidade total de instâncias (*instances*) na política, é aplicada a seguinte fórmula:

$$instances = \min(V_{\text{count}} - 1, 3) + C_{\text{count}} \tag{4.4}$$

Essa fórmula leva em consideração características específicas da infraestrutura estudada:

- O cluster Docker Swarm é composto por 3 VMs dedicadas para gerenciar os contêineres
- Uma dessas VMs deve permanecer sempre ativa para manter o cluster operacional

- Estas VMs não executam diretamente as aplicações, mas sim gerenciam os contêineres
- Por isso, subtrai-se 1 do $V_{\text{count}}$ ($V_{\text{count}} - 1$) para considerar apenas as VMs disponíveis para aplicações
- O $\min(..., 3)$ limita o número máximo de VMs de aplicação a 3, que é o número máximo de VMs disponíveis para aplicações normais
- $C_{\text{count}}$ representa o número de réplicas de contêineres em execução
- Estas réplicas são distribuídas através das VMs do cluster Docker Swarm
- O número de réplicas é adicionado diretamente pois cada réplica representa uma instância da aplicação

A nova política é considerada mais eficiente se ela mantiver o tempo de resposta e a probabilidade de violação dentro de limites aceitáveis. Além disso, deve otimizar o uso de recursos, garantindo que a quantidade de recursos utilizados seja menor ou igual à da política atual para uma carga de trabalho equivalente. A política também deve apresentar uma pontuação de desempenho (baseada no número de requisições e na probabilidade de violação) igual ou superior à da política anterior. Se todos esses critérios forem atendidos, a nova política para a referida quantidade de recursos é atualizada na Knowledge Base.

A pontuação (*score*) de uma política é calculada da seguinte forma:

$$score = max\_requests \times (1 - violation\_probability)$$

Onde:

- $max\_requests$ é o número total de requisições atendidas pela política,
- $violation\_probability$ é a probabilidade de violação do SLA, com um valor menor indicando uma política mais eficiente.

Esta fórmula reflete o equilíbrio entre o número de requisições e o risco de violação, permitindo uma avaliação comparativa entre as políticas. No entanto, se a nova política gerada para a quantidade de instâncias atual não for considerada melhor e tiver uma probabilidade de violação maior que a política persistida, a política existente deve atualizar a sua probabilidade de violação. Isso ocorre porque, se a nova política apresenta uma maior taxa de violação com a mesma quantidade de recursos, essa violação também pertence à política atual. Dessa forma, caso ocorram mais violações em níveis mais baixos de carga, a política atual tende a ser substituída por uma com um número máximo de requisições menor, mas com o mesmo número de instâncias.

Essa abordagem proporciona capacidades autonômicas ao sistema de escalonamento, como a auto-otimização das políticas de escalonamento e a auto-configuração do módulo executor. Com essas capacidades, o sistema pode se adaptar a mudanças no ambiente, como alterações na infraestrutura de virtualização ou variações nas necessidades de escalonamento vertical. O algoritmo 4 ilustra a lógica simplificada de funcionamento do módulo.

**Algorithm 4** Módulo Planejador Autonômico Simplificado

```
1: function PlanejadorAutonomico
2:     Inicializar configurações
3:     while sempre em execução do
4:         ColetarMetricas                          ▷ A cada minuto
5:         AvaliarEAjustarPoliticas                 ▷ A cada 10 minutos
6:     end while
7: end function
8: function ColetarMetricas
9:     Obter tempo médio de resposta
10:    Obter número total de requisições
11:    Verificar recursos (VMs e containers ativos)
12:    if tempo de resposta > 2 segundos then
13:        Registrar violação do SLA
14:    end if
15: end function
16: function AvaliarEAjustarPoliticas
17:    Analisar dados coletados
18:    Calcular tempo médio e probabilidade de violação
19:    Criar nova política com base nas métricas
20:    if nova política é melhor then
21:        Atualizar políticas e aplicar mudanças
22:    end if
23: end function
```

### 4.2.4 Módulo Executor (Execute)

O módulo executor implementa o módulo do sistema de escalonamento autonômico proativo responsável por executar as políticas de escalonamento geradas pelo módulo de planejamento. Operando sobre uma infraestrutura híbrida na qual pode aumentar ou reduzir instâncias de aplicações instaladas diretamente no sistema operacional de VMs ou aplicações conteinerizadas em um cluster Docker.

Conforme mostrado na figura 32 da seção 4.1, as máquinas virtuais são organizadas em dois grupos distintos: um para executar as aplicações nativas do sistema operacional e outro para hospedar o cluster Docker. O executor avalia periodicamente as previsões realizadas pelo módulo de análise e ajusta o provisionamento de recursos com base nas políticas recomendadas do módulo de planejamento.

O serviço implementa um endpoint REST que recebe notificações de atualização das políticas armazenadas na Knowledge Base pelo módulo de planejamento e, em seguida, executa o recarregamento das políticas.

A lógica de escalonamento é acionada a cada minuto por meio de um agendador, que obtém a carga futura consultando a API do módulo de análise que gera as previsões. Se a previsão de carga exceder a capacidade permitida da política atual, o serviço realiza o redimensionamento para cima, ligando máquinas virtuais ou aumentando o número de réplicas de contêineres Docker. Caso a previsão esteja abaixo do limite, o serviço reduz recursos, desligando máquinas virtuais ou diminuindo réplicas do serviço em contêineres.

As funções de escalonamento garantem que as máquinas virtuais estejam ativas antes de aumentar o número de containers, mantendo a quantidade mínima de recursos durante o processo de redução. O serviço registra informações detalhadas sobre cada ação de escalonamento para análise posterior do desempenho.

A coleta de métricas em tempo real é realizada através de chamadas à API do módulo de monitoramento, fornecendo dados sobre a quantidade de máquinas virtuais e contêineres ativos. O algoritmo 5 mostra a lógica de funcionamento deste módulo.

**Algorithm 5** Módulo Executor

```
1:  procedure EscalonadorPrincipal
2:      while verdadeiro do
3:          previsaoCarga ← ConsultarModuloPreditor()
4:          instanciasAtuais ← ConsultarModuloSensor()
5:          politicaAtual ← ConsultarKnowledgeBase(instanciasAtuais)
6:          if previsaoCarga > politicaAtual.maxRequisicoes then
7:              AumentarCapacidade()
8:          else if previsaoCarga < politicaAtual.limiteInferior then
9:              ReduzirCapacidade()
10:         end if
11:         Aguardar(1 minuto)
12:     end while
13: end procedure
14: procedure AumentarCapacidade
15:     if RecursosDisponiveis then
16:         ProvisionarNovaInstancia()
17:     end if
18: end procedure
19: procedure ReduzirCapacidade
20:     if InstanciasAcimaDoMinimo then
21:         RemoverInstancia()
22:     end if
23: end procedure
```

### 4.2.5 Gerador de Carga

Este módulo adicional é responsável por gerar carga de trabalho para testes que envolvem o módulo de planejamento (Plan). Ele utiliza o Locust[5], um framework de código aberto para testes de carga, simulando requisições HTTP para diversas URLs do sistema, com base em uma série temporal que contém dados históricos sobre a quantidade de requisições em determinado período.

O Locust opera com base em usuários simulados que geram requisições ao sistema-alvo. Cada um desses usuários executa continuamente tarefas (como realizar uma requisição HTTP) e, entre essas tarefas, aguarda por um intervalo de tempo definido pela função de espera, o *wait_time*. No caso deste módulo, a função `my_task` representa a tarefa executada por cada usuário, que envolve a execução de uma requisição HTTP para uma URL do sistema com base em um peso (de acordo com os logs do sistema, as URLs mais acessadas são configuradas com um maior peso).

Em cada instante da série temporal, este módulo verifica a taxa de requisições correspondente e ajusta dinamicamente o tempo de espera (*wait_time*) entre as requisições dos usuários simulados no Locust. Esse ajuste permite um alinhamento entre o comportamento da série temporal e a taxa de requisições, permitindo a reprodução de cargas de trabalho próximas aos cenários históricos. Para calcular o tempo de espera (*wait_time*) entre requisições no instante de tempo $t$, é utilizada a seguinte fórmula:

$$w(t) = \max\left(1, \frac{u}{r(t)}\right)$$

Onde:

- $w(t)$ é o tempo de espera (*wait_time*) no instante $t$;
- $u$ é o número de usuários simultâneos definidos no Locust;
- $r(t)$ é a taxa de requisições por segundo no instante $t$ definida na série temporal;
- $\max(1, \cdot)$ garante que o tempo de espera seja no mínimo 1 segundo.

O funcionamento do módulo é demonstrado pelo algoritmo 6:

[5] Ferramenta de teste de carga de código aberto. https://locust.io/

**Algorithm 6** Módulo de Geração de Carga

```
Carrega a série temporal
Define o índice atual timestamp para 0
Define uma lista de URLs com pesos
function CALCULATE_WAIT_TIME(rate, num_users)
    if rate > 0 and num_users > 0 then
        return max(1, num_users / rate)
    else
        return 1
    end if
end function
function CHOOSE_URL
    return Escolhe randomicamente uma URL da lista com base no peso
end function
procedure MY_TASK
    Obtém o timestamp atual
    Obtém a quantidade de requisições rate do timestamp
    Obtém num_users atual do Locust
    Calcula wait_time usando a função calculate_wait_time(rate, num_users)
    Seleciona uma URL usando a função choose_url()
    Faz a requisição HTTP a partir da URL selecionada
    Atualiza o índice do timestamp para a próxima iteração
    Aguarda o tempo definido em wait_time
end procedure
```

## 4.3 Configuração dos Experimentos

Os experimentos foram conduzidos pela execução do sistema de autoescalonamento autonômico proativo em quatro períodos distintos para avaliar diferentes cenários em contexto de produção:

- Cenário 1: 2024-11-14 09:40:00 - 2024-11-14 15:40:00 (baixa variância)
- Cenário 2: 2024-11-19 18:50:00 - 2024-11-20 02:40:00
- Cenário 3: 2024-11-20 10:50:00 - 2024-11-20 16:10:33
- Cenário 4: 2024-11-20 23:00:00 - 2024-11-21 18:40:00 (alta variabilidade)

Para avaliação do sistema, foram utilizadas as seguintes métricas:

- RMSE (Root Mean Square Error), MSE (Mean Squared Error), MAE (Mean Absolute Error) e MAPE (Mean Absolute Percentage Error) para avaliação da precisão do módulo Analyze, que realiza a previsão de carga de trabalho utilizando o WFLS-LSTM

- Redução percentual de recursos computacionais comparados através de dados históricos (base-line)
- Taxa de antecipação de picos de carga em relação à necessidade de executar redimensionamento de recursos com base nas políticas de autoescalonemanto geradas pelo módulo Plan.

## 4.4 Resultados

Esta seção apresenta os resultados obtidos com a implementação do sistema de autoescalonamento autonômico proativo em ambiente de produção.

### 4.4.1 QP 3: Qual o impacto do sistema de autoescalonamento autonômico proativo?

A análise abrange quatro aspectos principais: a evolução das políticas de escalonamento através do processo de autoajuste do sistema, a comparação do desempenho em relação à abordagem manual de escalonamento previamente utilizada, a avaliação da eficácia do modelo preditivo WFLS-LSTM com dados inéditos aplicado no módulo de análise, e a avaliação da capacidade de antecipação do sistema em picos de carga. Os resultados demonstram tanto a capacidade do sistema em otimizar a utilização de recursos quanto sua efetividade na antecipação de mudanças na demanda de carga.

#### 4.4.1.1 Comparação entre políticas inicializadas e ajustadas

A inicialização das políticas de autoescalonamento foi realizada utilizando dados reais de carga de trabalho de um dia típico e simulações com testes de carga com variância moderada. Identificou-se que a política mínima de 2 instâncias possui uma boa capacidade de atender a quantidade máxima de 3907 requisições com mínimas violações de tempo de resposta.

A partir da definição da política mínima, as demais políticas foram projetadas usando a mesma proporção, resultando em uma progressão linear representada pela linha azul na Figura 35. Após múltiplas execuções do sistema de autoescalonamento proativo autonômico, as políticas para $2, 3, 4, 5$ e 7 instâncias foram automaticamente reconfiguradas. Essas modificações visaram maximizar a quantidade de instâncias, minimizando a probabilidade de violação, resultando na linha vermelha do gráfico.

A Figura 35 ilustra comparativamente as políticas originais e as políticas ajustadas para cada quantidade de instâncias, evidenciando as adaptações realizadas pelo sistema autonômico. As políticas para 5 e 6 instâncias apresentaram probabilidade de violação

maior que zero, que apesar de não ultrapassarem o limite definido, tendem a ser substituídas ao longo do tempo por políticas com uma quantidade menor de requisições devido à penalização na comparação entre políticas no módulo de planejamento (plan).

As políticas para 8 a 5 instâncias não foram ajustadas pois, no período de execução do sistema, não foi preciso aumento do número de instâncias devido a carga de trabalho não superar a capacidade da política para 7 instâncias.

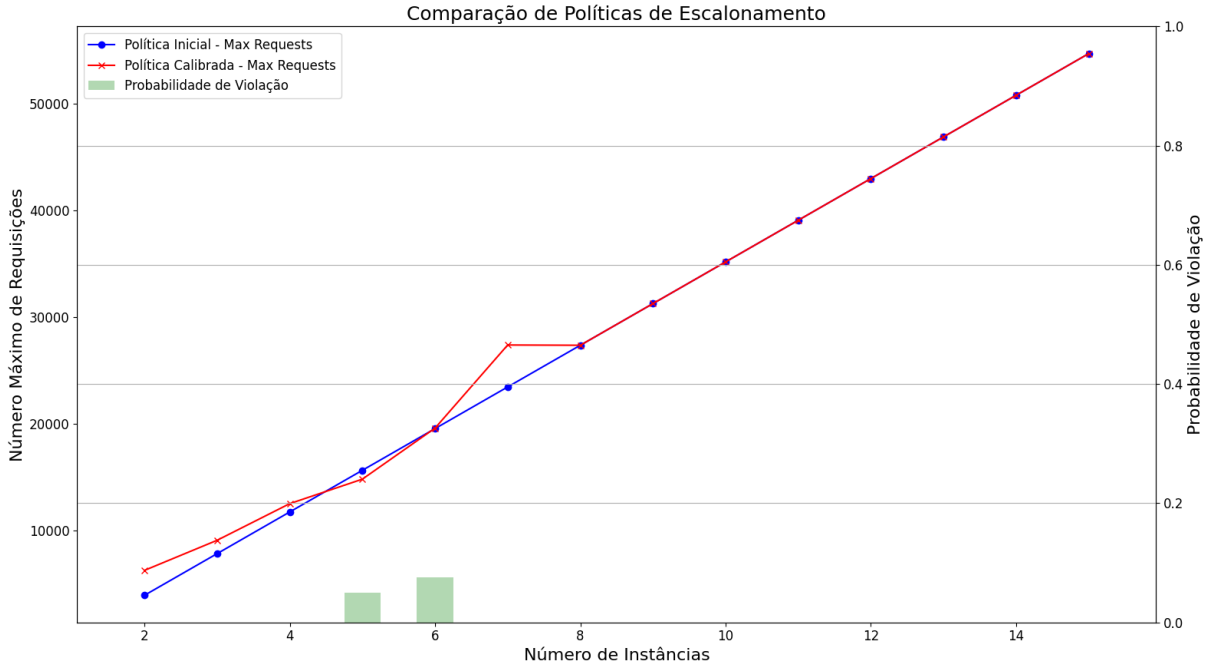


Figura 35 – Comparação entre as políticas iniciais e ajustadas

#### 4.4.1.2 Comparação com a linha de base

A Figura 36 apresenta uma análise comparativa entre as estratégias de escalonamento manual e automática ao longo do recorte temporal dos dados históricos. O gráfico demonstra três elementos principais: o número de requisições ao longo do tempo (eixo principal), o número de instâncias alocadas pelo escalonamento manual histórico e o número de instâncias sugeridas pelo sistema de autoescalonamento proposto (eixo secundário).

A análise dos resultados revela padrões distintos de alocação de recursos. O sistema de autoescalonamento proposto demonstrou maior eficiência na utilização de recursos, identificando três períodos principais onde apenas duas instâncias seriam suficientes para atender à demanda:

- 11/10 a 16/10, 25/10 a 06/11 e após 14/11: Períodos com baixa demanda

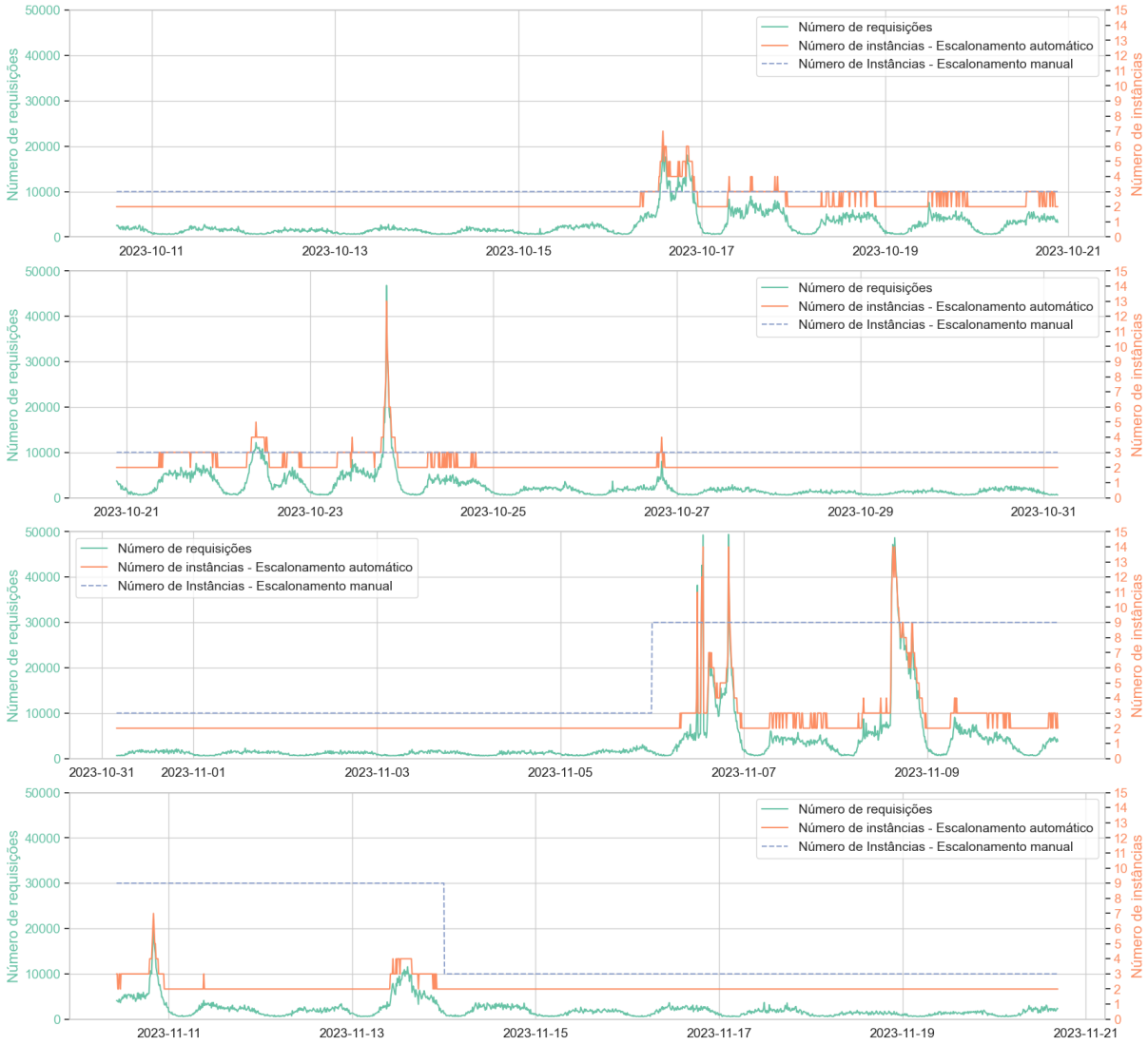


Figura 36 – Comparação entre as abordagens de escalonamento manual e automático em relação ao número de requisições

Adicionalmente, o sistema automático identificou períodos onde a infraestrutura manual estava subdimensionada, sugerindo um aumento no número de instâncias em três momentos críticos:

- 17/10: pico pontual de demanda;
- 22/10 a 24/10: período sustentado de carga moderada a alta;
- 07/10 a 09/10: intervalo de demanda elevada

A implementação do sistema de autoescalonamento resultaria em uma redução significativa de 37,9% na utilização de recursos computacionais, enquanto mantém a capacidade de resposta a picos de demanda. Esta otimização demonstra o potencial de

economia gerada através da implantação do sistema de autoescalonamento. O cálculo foi realizado conforme a equação:

$$R = \frac{\sum_{t=1}^{n}(I_{man}(t) - I_{auto}(t))}{\sum_{t=1}^{n} I_{man}(t)} \times 100 \tag{4.5}$$

Onde:

- $R$ é a redução percentual de recursos
- $I_{man}(t)$ é o número de instâncias no escalonamento manual no tempo $t$
- $I_{auto}(t)$ é o número de instâncias no escalonamento automático no tempo $t$
- $n$ é o número total de períodos analisados

#### 4.4.1.3 Avaliação do método de previsão com dados inéditos

A avaliação do modelo WFLS-LSTM aplicado ao sistema de autoescalonamento autonômico proativo com dados inéditos destaca sua aplicabilidade prática em cenários de produção. Embora as métricas dos testes em produção, (Tabela 19 - RMSE: 1041, MSE: 1252544, MAE: 728, MAPE: 19,84%) sejam menos eficientes às obtidas durante a fase de treinamento/teste do modelo (RMSE: 934,17, MSE: 872672,35, MAE: 368,30, MAPE: 14,64%), indicando um aumento nos erros, esses resultados refletem as condições mais desafiadoras de um ambiente real de produção. Este aumento nas métricas de erro pode ser atribuído aos seguintes fatores:

1. **Variabilidade dos Períodos de Teste**: Os quatro intervalos analisados apresentam características distintas de tráfego e comportamento, para avaliar o modelo em diferentes cenários operacionais. O primeiro cenário com uma variância menor até o ultimo cenário com alta variabilidade.

2. **Impacto da Inicialização do Modelo**: No início da avaliação, o modelo enfrenta limitações devido à indisponibilidade de observações suficientes para preencher completamente a janela de entrada do modelo de previsão (48 observações). Isso resulta em previsões iniciais menos precisas, com melhorias progressivas à medida que mais dados são incorporados (Figura 37).

3. **Caracterização da Série Temporal**: A série temporal utilizada na fase de treinamento/teste do modelo possui uma maior extensão e períodos mais longos de cargas estáveis. Em contrapartida, o recorte temporal analisado, com dados reais, apresenta menor tamanho e maior volatilidade. O último cenário (Tabela 18) apresenta o maior RMSE (1449), devido à maior magnitude dos valores.

4. **Carga mínima no 1º cenário**: A análise do primeiro cenário (Tabela 18) apresenta valores de cargas mais baixos em relação aos outros cenários, onde o modelo realizou previsões de valores mínimos.

Tabela 18 – Resumo das Métricas por Intervalo

| Intervalo | RMSE | MSE | MAE | MAPE |
|---|---|---|---|---|
| 2024-11-14 09:40:00 - 2024-11-14 15:40:00 | 590 | 348290 | 423 | 38% |
| 2024-11-19 18:50:00 - 2024-11-20 02:40:00 | 600 | 359899 | 401 | 13% |
| 2024-11-20 10:50:00 - 2024-11-20 16:10:33 | 749 | 561385 | 547 | 18% |
| 2024-11-20 23:00:00 - 2024-11-21 18:40:00 | 1449 | 2099981 | 1014 | 18% |

Tabela 19 – Métricas Médias Ponderadas

| Métricas Médias Ponderadas | Valores |
|---|---|
| RMSE | 1041 |
| MSE | 1252544 |
| MAE | 728 |
| MAPE | 19.84% |

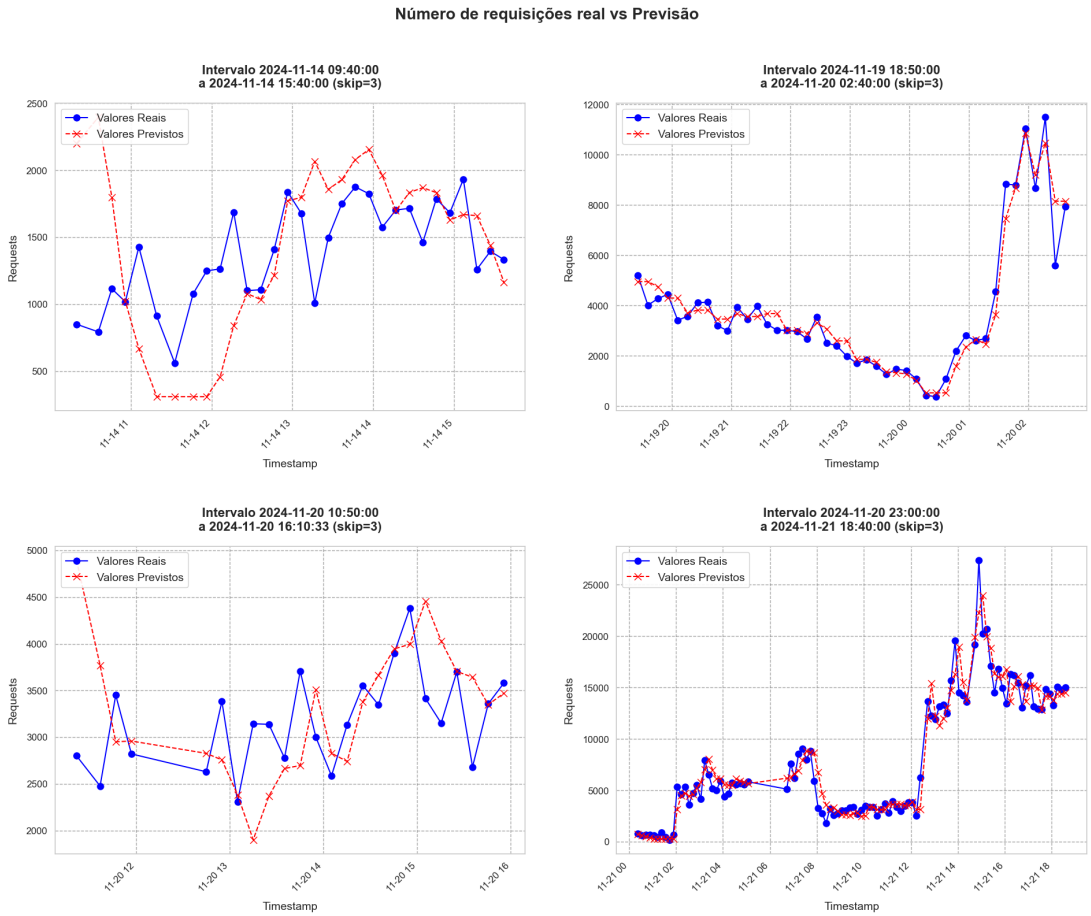


Figura 37 – Cenários - Número de requisições real vs Previsão

#### 4.4.1.4 Análise de antecipação em picos de carga

As figuras 38 e 39 apresentam a análise do comportamento do sistema de autoescalonamento autonômico proativo. O gráfico demonstra a evolução temporal do número de requisições (representada pela linha sólida) e a previsão realizada pelo sistema (linha tracejada), já as linhas horizontais denotam os limiares estabelecidos pelas políticas de escalonamento. Os pontos destacados com marcações circulares nas previsões são coloridos em verde quando indicam detecção antecipada da necessidade de mudança de política para expansão dos recursos, e em vermelho quando o sistema não conseguiu prever a ultrapassagem do limiar antes de sua ocorrência.

A análise quantitativa dos dados revelou que, durante o período monitorado, o sistema demonstrou capacidade de antecipação em aproximadamente 62,5% dos eventos de pico de carga. Este percentual foi calculado considerando a razão entre o número de escalonamentos antecipados bem-sucedidos e o número total de eventos que demandaram mudança de política de recursos. O escalonamento antecipado bem-sucedido é aquele em que a predição identificou a necessidade de recursos adicionais antes que os limiares críticos fossem atingidos, permitindo a alocação proativa de recursos.

Nos casos em que o sistema não conseguiu prever antecipadamente a ultrapassagem do limiar (marcações em vermelho), abordagens reativas podem ser combinadas com a estratégia proativa para garantir uma resposta em um tempo hábil antes de aguardar a próxima previsão.

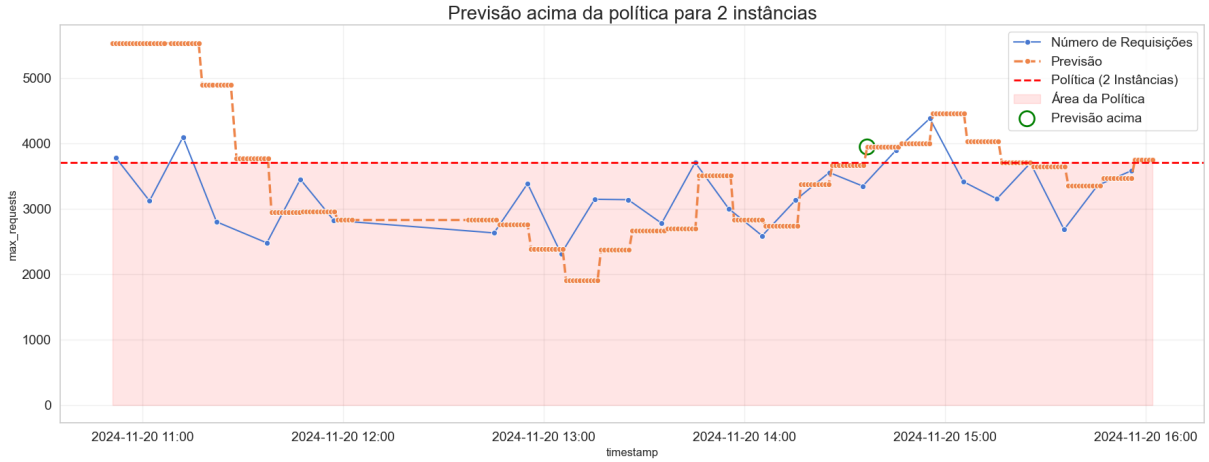


Figura 38 – Análise Dinâmica do Escalonamento de Recursos

Os resultados em produção validam a eficácia do sistema para o gerenciamento proativo de recursos computacionais. A taxa de antecipação de 62,5% dos eventos de alta demanda demonstra um avanço significativo em relação às abordagens puramente reativas. Os casos não antecipados (37,5%) sugerem oportunidades de aprimoramento do

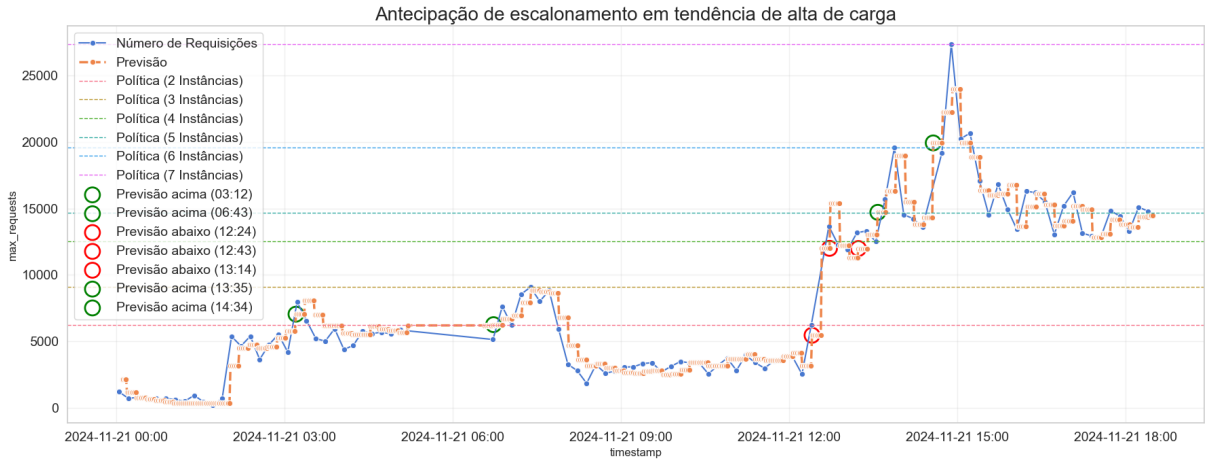


Figura 39 – Antecipação no escalonamento em tendência de alta de carga

modelo preditivo e do sistema de autoescalonamento autonômico proativo, especialmente para cenários com mudanças abruptas no padrão de tráfego.

## 4.5 Ameaças à validade

Para garantir a validade dos resultados do estudo, foram consideradas e abordadas diferentes categorias de ameaças potenciais:

**Validade Interna:** A variabilidade natural do ambiente de produção representa uma ameaça potencial à validade interna do estudo. Para mitigar este aspecto, testes em diferentes horários e dias, contemplando diversos cenários com variados padrões de carga, foram realizados. Complementarmente, foram feitas avaliações com dados históricos para fortalecer a análise. Outro fator que pode comprometer a validade interna é a inicialização do sistema, que pode impactar o desempenho inicial devido à ausência de dados como entrada para o módulo de análise que realiza a previsão. Como estratégia de mitigação, além de desconsiderarmos as três primeiras previsões no cálculo das métricas, analisamos o comportamento do sistema em diferentes padrões de variância e amplitude de carga.

**Validade Externa:** Embora os resultados possam não ser diretamente generalizáveis para todos os tipos de aplicações, eles são aplicáveis a contextos que apresentem padrões de carga similares aos analisados neste estudo, conforme demonstrado na análise da série temporal.

**Validade de Construto:** A definição de "antecipação bem-sucedida" pode apresentar aspectos subjetivos. Para este estudo, consideramos como bem-sucedida toda previsão que identificou a ultrapassagem do limiar superior da política de escalonamento antes da sua efetiva ocorrência.

# 5 Trabalhos Futuros

Com base nos resultados obtidos neste trabalho, algumas direções para pesquisas futuras incluem:

1. **Investigar possíveis aprimoramentos do modelo LSTM**: Explorar o uso de camadas adicionais explorando vantagens de outras abordagens de inteligência artificial, como camadas convolucionais (CNN) ou mecanismos de atenção, a fim de otimizar a detecção de picos de carga de trabalho avaliando também a eficiência em termos operacionais aplicados neste contexto.

2. **Ampliar a base de dados**: Coletar e processar uma quantidade maior de dados de carga de trabalho, incluindo séries temporais de diferentes aplicações e ambientes, a fim de avaliar o desempenho do modelo LSTM em um escopo mais amplo.

3. **Avaliar modelos de previsão ponta a ponta**: Avaliar modelos de previsão nos quais a saída seja diretamente a quantidade de instâncias da aplicação necessárias através da coleta de dados de políticas geradas pelo módulo de planejamento (Plan) por um longo período de operação do sistema de autoescalonamento autonômico proativo com o objetivo de simplificar ainda mais a solução.

4. **Avaliar técnicas de aprendizado por reforço** para otimização contínua das políticas de autoescalonamento.

Essas direções de pesquisa futura visam explorar melhorias na precisão das previsões, na generalização do modelo e na integração com sistemas de autoescalonamento em tempo real, ampliando ainda mais os benefícios da adoção de algoritmos de aprendizado profundo para a previsão de cargas de trabalho em aplicações legadas.

# 6 Conclusão

Este trabalho demonstrou que a previsão da carga de trabalho de aplicações legadas utilizando algoritmos de aprendizado profundo, como o LSTM, pode superar modelos estatísticos tradicionais como ARIMA e ETS. Na análise dos dados benchmark Clarknet Traces, o modelo LSTM apresentou desempenho superior em todas as métricas avaliadas, com melhorias de 8,77% em RMSE e 18,02% em MAPE na série temporal com resolução de 1 minuto, mantendo sua eficácia em diferentes escalas temporais.

A investigação das diferentes granularidades temporais revelou que a resolução de 10 minutos oferece o melhor equilíbrio entre precisão e eficiência computacional para aplicações práticas em sistemas de autoescalonamento proativo. Em contrapartida, a resolução de 1 minuto apresentou custos computacionais significativamente maiores, enquanto a resolução de 1 hora mostrou-se insuficiente para capturar variações de curto prazo.

Quando aplicado à série temporal do sistema real, o LSTM manteve sua competitividade, superando os modelos ARIMA e ETS nas métricas absolutas (MAE e MAPE), embora os modelos ETS tenham se destacado nas métricas quadráticas (RMSE e MSE). Os resultados evidenciaram uma excelente capacidade de generalização do LSTM, não exigindo re-treinamento a cada previsão, enquanto os modelos ARIMA e ETS demandaram ajustes contínuos. Esta característica representa uma vantagem significativa para o LSTM em ambientes de produção, onde as previsões são utilizadas para apoiar políticas de escalonamento automático em tempo real.

A capacidade de realizar previsões multi-horizonte precisas, viabilizando políticas proativas mais sofisticadas através de análise de tendências, também garante ao LSTM vantagem competitiva em relação aos modelos estatísticos tradicionais. O LSTM provou-se eficaz para a previsão de cargas de trabalho baseadas em quantidade de requisições, destacando-se em ambientes que demandam previsões precisas e em tempo real, com menor necessidade de retreinamentos ou ajustes frequentes.

Os benefícios teóricos da implementação do sistema de autoescalonamento proativo autonômico no contexto de um sistema legado real on-premises demonstraram resultados práticos significativos: O sistema demonstrou uma expressiva otimização na utilização de recursos computacionais, alcançando uma redução de 37,9% em comparação com o escalonamento manual previamente utilizado, enquanto manteve a capacidade efetiva de resposta a picos de demanda. Esta otimização foi possível através da identificação precisa tanto de períodos que requeriam menos recursos quanto de momentos críticos que demandavam expansão da infraestrutura.

Quando aplicado a dados inéditos em produção, o modelo de previsão desenvol-

vido WFLS-LSTM manteve um desempenho robusto, apresentando métricas aceitáveis (RMSE: 1041, MSE: 1252544, MAE: 728, MAPE: 19,84%) mesmo em condições mais desafiadoras do que as observadas durante o treinamento. Embora estas métricas indiquem um aumento nos erros em comparação com a fase de treinamento/teste, elas demonstram a capacidade do modelo de se adaptar a cenários reais com maior variabilidade e volatilidade.

Outro aspecto relevante foi a capacidade do sistema em antecipar aproximadamente 62,5% dos eventos de pico de carga, representando um avanço significativo em relação a abordagens puramente reativas. Os 37,5% de casos não antecipados abrem oportunidades de pesquisa para futuras melhorias do sistema.

O processo de autoajuste das políticas de escalonamento demonstrou a capacidade do sistema em otimizar continuamente seus parâmetros, adaptando-se às condições reais de operação e maximizando a eficiência na alocação de recursos. Esta característica, combinada com a capacidade preditiva do modelo WFLS-LSTM, estabelece uma base para o desenvolvimento de sistemas de autoescalonamento proativos autonômicos no contexto de sistemas legados implantados em data centers locais.

# Referências

MOHALIK, S. K.; NARENDRA, N. C.; BADRINATH, R.; LE, D.-H. Adaptive service-oriented architectures for cyber physical systems. In: *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. [S.l.: s.n.], 2017. p. 57–62. Citado 2 vezes nas páginas 9 e 27.

TUKEY, J. W. *Exploratory Data Analysis*. [S.l.]: Addison-Wesley, 1977. Citado 2 vezes nas páginas 9 e 28.

MESSIAS, V. R.; ESTRELLA, J. C.; EHLERS, R.; SANTANA, M. J.; SANTANA, R. C.; REIFF-MARGANIEC, S. Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure. *Neural Computing and Applications*, v. 27, n. 8, p. 2383–2406, Nov 2016. ISSN 1433-3058. Disponível em: <https://doi.org/10.1007/s00521-015-2133->. Citado 8 vezes nas páginas 9, 20, 22, 34, 40, 41, 47 e 60.

KUMAR, J.; SINGH, A. K. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems*, v. 81, p. 41–52, 2018. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com-/science/article/pii/S0167739X1730044>. Citado 6 vezes nas páginas 9, 22, 41, 42, 47 e 59.

SINGH, P.; GUPTA, P.; JYOTI, K. Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, v. 22, n. 2, p. 619–633, Jun 2019. ISSN 1573-7543. Disponível em: <https://doi.org/10.1007/s10586-018-2868->. Citado 4 vezes nas páginas 9, 42, 43 e 44.

RADHIKA, E.; Sudha Sadasivam, G. A review on prediction based autoscaling techniques for heterogeneous applications in cloud environment. *Materials Today: Proceedings*, v. 45, p. 2793–2800, 2021. ISSN 2214-7853. International Conference on Advances in Materials Research - 2019. Disponível em: <https://doi.org/10.1016/j.matpr.2020.11.78>. Citado na página 18.

CHANG, H. The differences and advantages between cloud services and traditional services. In: *2022 8th Annual International Conference on Network and Information Systems for Computers (ICNISC)*. [S.l.: s.n.], 2022. p. 497–499. Citado na página 18.

NIC.BR, N. de Informação e Coordenação do P. B. (Ed.). *Pesquisa sobre o uso das tecnologias de informação e comunicação no setor público brasileiro : TIC Governo Eletrônico 2021 = Survey on the use of information and communication technologies in the Brazilian public sector : ICT Electronic Government 2021*. São Paulo, SP: Comitê Gestor da Internet no Brasil, 2022. Edição bilíngue : português / inglês, Vários colaboradores, Vários tradutores, Bibliografia, ISBN 978-65-86949-68-1. Citado na página 18.

MURUGESAN, G. K. Cloud services — boon or bane: A comprehensive review. In: *SoutheastCon 2024*. [S.l.: s.n.], 2024. p. 108–112. Citado na página 18.

LINTHICUM, D. Why companies are leaving the cloud. *InfoWorld*, 2 2024. Disponível em: <https://www.infoworld.com/article/3712861/why-companies-are-leaving-the-cloud.htm>. Citado 2 vezes nas páginas 18 e 19.

PATIBANDLA, R. S. M. L.; KURRA, S. S.; MUNDUKUR, N. B. A study on scalability of services and privacy issues in cloud computing. In: RAMANUJAM, R.; RAMASWAMY, S. (Ed.). *Distributed Computing and Internet Technology*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 212–230. ISBN 978-3-642-28073-3. Disponível em: <https://doi.org/10.1007/978-3-642-28073-3_1>. Citado na página 18.

REIS, J.; HOUSLEY, M. *Fundamentos de Engenharia de Dados: Projete e Construa Sistemas de Dados Robustos*. Novatec Editora, 2023. ISBN 9788575228777. Disponível em: <https://books.google.com.br/books?id=Q1fiEAAAQBA>. Citado na página 19.

POZDNIAKOVA, O.; MAžEIKA, D.; CHOLOMSKIS, A. Adaptive resource provisioning and auto-scaling for cloud native software. In: DAMAšEVIčIUS, R.; VASILJEVIENė, G. (Ed.). *Information and Software Technologies. ICIST 2018*. Springer, Cham, 2018. (Communications in Computer and Information Science, v. 920). Disponível em: <https://doi.org/10.1007/978-3-319-99972-2_>. Citado na página 19.

SILVA, T. P. da; NETO, A. F. R.; BATISTA, T. V.; LOPES, F. A. S.; DELICATO, F. C.; PIRES, P. F. Horizontal auto-scaling in edge computing environment using online machine learning. In: *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. AB, Canada: [s.n.], 2021. p. 161–168. Citado na página 19.

NUNES, J. P. K. S.; BIANCHI, T.; IWASAKI, A. Y.; NAKAGAWA, E. Y. State of the art on microservices autoscaling: An overview. In: *SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE (SEMISH), 48*. Porto Alegre: Sociedade Brasileira de Computação, 2021. p. 30–38. ISSN 2595-6205. Disponível em: <https://doi.org/10.5753-/semish.2021.1580>. Citado 3 vezes nas páginas 19, 24 e 25.

HUSIN, H. S.; CUI, L.; HAMID, H. R. H.; ABDULLAH, N. Y. Time series analysis of web server logs for an online newspaper. In: *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*. New York, NY, USA: Association for Computing Machinery, 2013. (ICUIMC '13). ISBN 9781450319584. Disponível em: <https://doi.org/10.1145/2448556.244855>. Citado na página 19.

CHATFIELD, C. *The Analysis of Time Series: An Introduction*. 6th. ed. [S.l.]: Chapman and Hall/CRC, 2016. 2834–2839 p. Citado 2 vezes nas páginas 19 e 27.

LORIDO-BOTRáN, T.; MIGUEL-ALONSO, J.; LOZANO, J. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, v. 12, 12 2014. Citado 2 vezes nas páginas 20 e 25.

TSAY, R.; CHEN, R. *Nonlinear Time Series Analysis*. Wiley, 2018. (Wiley Series in Probability and Statistics). ISBN 9781119264057. Disponível em: <https://books.google-.com.br/books?id=LeU0jwEACAA>. Citado na página 21.

SINGH, P.; GUPTA, P.; JYOTI, K. Tasm: Technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, v. 22, n. 2, p. 619–633, 2018. Citado 3 vezes nas páginas 22, 47 e 59.

KUMAR K. GANGADHARA RAO, S. B. D. V. K. Forecasting of cloud computing services workload using machine learning. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, v. 12, n. 11, p. 4841–4846, 2021. Accessed on: 1 Feb. 2024. Disponível em: <https://turcomat.org/index.php/turkbilmat/article/view/666>. Citado 3 vezes nas páginas 22, 44 e 59.

SINGH, P.; KAUR, A.; GUPTA, P.; GILL, S. S.; JYOTI, K. Rhas: robust hybrid auto-scaling for web applications in cloud computing. *Cluster Computing*, v. 24, n. 2, p. 717–737, Jun 2021. ISSN 1573-7543. Disponível em: <https://doi.org/10.1007/s10586-020-03148->. Citado 3 vezes nas páginas 22, 44 e 47.

MORETTIN, P.; TOLOI, C. *Análise de séries temporais: modelos lineares univariados*. BLUCHER., 2018. ISBN 9788521213529. Disponível em: <https://books.google.com.br-/books?id=UwC5DwAAQBA>. Citado na página 22.

JANARDHANAN, D.; BARRETT, E. Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models. In: *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. [S.l.: s.n.], 2017. p. 55–60. Citado 2 vezes nas páginas 22 e 58.

ERL, T.; PUTTINI, R.; MAHMOOD, Z. *Cloud Computing: Concepts, Technology & Architecture*. Prentice Hall, 2013. (The Prentice Hall service technology series from Thomas Erl). ISBN 9780133387520. Disponível em: <https://books.google.com.br-/books?id=zqhpAgAAQBA>. Citado na página 24.

IMDOUKH, M.; AHMAD, I.; ALFAILAKAWI, M. G. Machine learning-based auto-scaling for containerized applications. *Neural Computing and Applications*, v. 32, p. 9745–9760, 2020. Received: 30 December 2018, Accepted: 21 September 2019, Published: 08 October 2019, Issue Date: July 2020. Disponível em: <https://doi.org/10.1007-/s00521-019-04507->. Citado na página 24.

KWAN, A.; WONG, J.; JACOBSEN, H.-A.; MUTHUSAMY, V. Hyscale: Hybrid and network scaling of dockerized microservices in cloud data centres. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.: s.n.], 2019. p. 80–90. Citado na página 25.

KEPHART, J.; CHESS, D. The vision of autonomic computing. *Computer*, v. 36, n. 1, p. 41–50, 2003. Citado na página 26.

IBM (Ed.). *An Architectural Blueprint for Autonomic Computing*. [S.l.], jun. 2005. Citado na página 26.

NGUYEN, T. A.; AIELLO, M.; YONEZAWA, T.; TEI, K. A self-healing framework for online sensor data. In: *2015 IEEE International Conference on Autonomic Computing*. [S.l.: s.n.], 2015. p. 295–300. Citado na página 26.

HABEN, S.; VOSS, M.; HOLDERBAUM, W. Previsão de séries temporais: Conceitos e definições essenciais. In: *Conceitos Básicos e Métodos em Previsão de Carga*. [S.l.]: Springer, 2023. Citado 3 vezes nas páginas 28, 29 e 32.

NIELSEN, A. *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*. O'Reilly Media, 2019. ISBN 9781492041627. Disponível em: <https://books.google.com.br/books?id=odCwDwAAQBA>. Citado na página 28.

DAMA, F.; SINOQUET, C. Analysis and modeling to forecast in time series: a systematic review. *CoRR*, abs/2104.00164, 2021. Disponível em: <https://arxiv.org/abs-/2104.0016>. Citado 3 vezes nas páginas 28, 31 e 54.

HYNDMAN, R. J.; ATHANASOPOULOS, G. *Forecasting: Principles and Practice*. 3rd. ed. Melbourne, Australia: OTexts, 2021. Disponível em: <http://otexts.com-/fpp3>http://otexts.com/fpp3. Citado 4 vezes nas páginas 29, 30, 32 e 33.

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*, v. 54, n. 1, p. 159–178, 1992. ISSN 0304-4076. Disponível em: <https://www.sciencedirect.com/science/article-/pii/030440769290104>. Citado na página 31.

DICKEY, D.; FULLER, W. Distribution of the estimators for autoregressive time series with a unit root. *JASA. Journal of the American Statistical Association*, v. 74, 06 1979. Citado na página 31.

WANG, F.; ZHAO, L. Complexity analysis of air traffic flow based on sample entropy. In: *2019 Chinese Control And Decision Conference (CCDC)*. [S.l.: s.n.], 2019. p. 5368–5371. Citado na página 34.

XUAN, A.; YIN, M.; LI, Y.; CHEN, X.; MA, Z. A comprehensive evaluation of statistical, machine learning and deep learning models for time series prediction. In: *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*. [S.l.: s.n.], 2022. p. 55–60. Citado na página 34.

BOX, G.; JENKINS, G.; REINSEL, G.; LJUNG, G. *Time Series Analysis: Forecasting and Control*. [S.l.]: Wiley, 2015. (Wiley Series in Probability and Statistics). ISBN 9781118674925. Citado na página 35.

HYNDMAN, R. J.; KOEHLER, A. B.; ORD, J. K.; SNYDER, R. D. *Forecasting with Exponential Smoothing*: The state space approach. 1. ed. Berlin, Heidelberg: Springer Berlin, Heidelberg, 2008. XIII, 362 p. (Springer Series in Statistics). ISSN 0172-7397. ISBN 978-3-540-71916-8. Disponível em: <https://doi.org/10.1007/978-3-540-71918->. Citado na página 35.

SIMS, C. A. Macroeconomics and reality. *Econometrica*, The Econometric Society, v. 48, n. 1, p. 1–48, 1980. Disponível em: <https://doi.org/10.2307/191201>. Citado na página 35.

DRAPER, N.; SMITH, H. *Applied Regression Analysis*. [S.l.]: Wiley, 1998. (Wiley Series in Probability and Statistics). ISBN 9780471170822. Citado na página 35.

BREIMAN, L. Random forests. *Machine Learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Disponível em: <https://doi.org/10.1023/A:101093340432>. Citado na página 35.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. p. 785–794. Disponível em: <https://doi.org/10.1145/2939672-.293978>. Citado na página 35.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297, 1995. Citado na página 35.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. Citado 2 vezes nas páginas 35 e 38.

CHO, K.; MERRIENBOER, B. van; BENGIO, Y.; SCHWENK, H. On the properties of neural machine translation: Encoder-decoder approaches. In: *Proceedings of EMNLP 2014*. [s.n.], 2014. Disponível em: <https://arxiv.org/abs/1406.107>. Citado na página 35.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, ; POLOSUKHIN, I. Attention is all you need. In: *Proceedings of NeurIPS 2017*. [s.n.], 2017. Disponível em: <https://arxiv.org/abs/1706.0376>. Citado na página 35.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, v. 1, n. 4, p. 541–551, 1989. Citado na página 35.

LIM, B.; ZOHREN, S. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, v. 379, n. 2194, p. 20200209, 2021. Disponível em: <https://royalsocietypublishing.org-/doi/abs/10.1098/rsta.2020.020>. Citado na página 35.

MEISENBACHER, S.; TUROWSKI, M.; PHIPPS, K.; RäTZ, M.; MüLLER, D.; HAGENMEYER, V.; MIKUT, R. Review of automated time series forecasting pipelines. *WIREs Data Mining and Knowledge Discovery*, v. 12, n. 6, p. e1475, 2022. Disponível em: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.147>. Citado 2 vezes nas páginas 35 e 67.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/323533a>. Citado na página 35.

GRAVES, A. *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin: Springer, 2012. (Studies in computational intelligence). Disponível em: <https://cds.cern.ch/record/150387>. Citado 2 vezes nas páginas 36 e 37.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org. Citado na página 36.

GERS, F.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: Continual prediction with lstm. *Neural computation*, v. 12, p. 2451–71, 10 2000. Citado na página 38.

CLARKNET. *ClarkNet HTTP Dataset*. 1995. https://ita.ee.lbl.gov/. Citado na página 46.

PANDAS, D. team. *pandas-dev/pandas: Pandas*. Zenodo, fev. 2020. Disponível em: <https://doi.org/10.5281/zenodo.350913>. Citado na página 47.

LANCIANO, G.; GALLI, F.; CUCINOTTA, T.; BACCIU, D.; PASSARELLA, A. Predictive auto-scaling with openstack monasca. In: *Predictive Auto-scaling with OpenStack Monasca*. [S.l.: s.n.], 2021. Citado na página 48.

CHICCO, D.; WARRENS, M. J.; JURMAN, G. The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, v. 7, p. e623, 2021. Citado na página 48.

ARIFF, N. A. M.; ISMAIL, A. R. Study of adam and adamax optimizers on alexnet architecture for voice biometric authentication system. In: *2023 17th International Conference on Ubiquitous Information Management and Communication (IMCOM)*. [S.l.: s.n.], 2023. p. 1–4. Citado na página 49.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Disponível em: <https://arxiv.org/abs/1412.698>. Citado na página 49.

TIAN, Y.; ZHANG, Y. A comprehensive survey on regularization strategies in machine learning. *Information Fusion*, v. 80, p. 146–166, 2022. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S156625352100230>. Citado na página 49.

Bai, Yuhan. Relu-function and derived function review. *SHS Web Conf.*, v. 144, p. 02006, 2022. Disponível em: <https://doi.org/10.1051/shsconf/20221440200>. Citado na página 49.

OGUNSANYA, M.; ISICHEI, J.; DESAI, S. Grid search hyperparameter tuning in additive manufacturing processes. *Manufacturing Letters*, v. 35, p. 1031–1042, 2023. ISSN 2213-8463. 51st SME North American Manufacturing Research Conference (NAMRC 51). Disponível em: <https://www.sciencedirect.com/science/article/pii-/S221384632300113>. Citado na página 50.

RABENSTEIN, B.; VOLZ, J. Prometheus: A next-generation monitoring system (talk). In: . Dublin: USENIX Association, 2015. Citado na página 76.