UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DO PICI

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PAULO RICARDO FERNANDES RODRIGUES

MAPEAMENTO SISTEMÁTICO SOBRE TÉCNICAS DE MONITORAMENTO DE FALTAS BASEADAS EM MINERAÇÃO DE DADOS

FORTALEZA

2025

PAULO RICARDO FERNANDES RODRIGUES

MAPEAMENTO SISTEMÁTICO SOBRE TÉCNICAS DE MONITORAMENTO DE FALTAS BASEADAS EM MINERAÇÃO DE DADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus do Pici da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientadora: Prof. Dra. Valéria Lelli Leitão Dantas

FORTALEZA

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R615m Rodrigues, Paulo Ricardo Fernandes.
Mapeamento sistemático sobre técnicas de monitoramento de faltas baseadas em mineração de dados / Paulo Ricardo Fernandes Rodrigues. – 2025.
65 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Ciências, Curso de Computação, Fortaleza, 2025.
Orientação: Profa. Dra. Valéria Lelli Leitão Dantas.

1. Detecção de faltas. 2. Mineração de dados. I. Título.

CDD 005

PAULO RICARDO FERNANDES RODRIGUES

MAPEAMENTO SISTEMÁTICO SOBRE TÉCNICAS DE MONITORAMENTO DE FALTAS BASEADAS EM MINERAÇÃO DE DADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus do Pici da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dra. Valéria Lelli Leitão Dantas (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Ismayle de Sousa Santos
Universidade Estadual do Ceará (UECE)

Profa. Dra. Fabiana Gomes Marinho
IFCE Maracanaú (IFCE)

AGRADECIMENTOS

À minha família por todo o apoio, paciência e compreensão em todos os momentos desta jornada.

À minha orientadora, Profa. Dra. Valéria Lelli Leitão Dantas, por todo o apoio nos últimos anos e as orientações em sala de aula, nos projeto acadêmicos e neste trabalho de conclusão de curso.

Aos meus amigos, tanto do âmbito acadêmico quanto fora dele, pela companhia, suporte e bons momentos compartilhados ao longo desta jornada.

À minha namorada, Charlotte, pelo sempre oferecer um forte apoio. Sua presença foi fundamental para me ajudar a manter a concentração e a calma nos momentos mais desafiadores.

A todos os professores da UFC Campus do Pici, por todo o aprendizado essencial para minha formação pessoal e profissional.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

“O que sabemos é uma gota, o que ignoramos é um oceano.”

(Isaac Newton)

RESUMO

A crescente complexidade e interconexão dos sistemas de software modernos tornam a detecção de faltas um desafio significativo, uma vez que técnicas tradicionais de testes mostram-se insuficientes, especialmente em sistemas dinâmicos e de grande escala. Para enfrentar esse desafio, diversas abordagens de monitoramento em tempo real baseadas em mineração de dados têm sido propostas. Essas abordagens, por meio da análise contínua do sistema e de uso de técnicas de detecção de anomalias, são capazes de identificar faltas em sistemas de software. Este trabalho apresenta um mapeamento sistemático da literatura sobre essas técnicas de monitoramento, visando identificar desafios de implementação e fornecer uma visão abrangente do estado atual da pesquisa. O mapeamento foi realizado utilizando a base de dados Scopus e foram levantados 20 estudos que ajudam a responder questões de pesquisas definidas neste trabalho. A análise permitiu identificar não apenas as principais técnicas de detecção, mas também os tipos de sistemas monitorados, as categorias de faltas e as métricas de avaliação utilizados nas abordagens propostas. Entre os principais desafios identificados, destaca-se a dificuldade na rotulação de dados para o treinamento de modelos supervisionados, a complexidade na interpretação dos modelos gerados e a necessidade de realizar o monitoramento de forma computacionalmente eficiente.

Palavras-chave: Detecção de faltas; Detecção de anomalias; Mineração de dados.

ABSTRACT

The increasing complexity and interconnection of modern software systems makes fault detection a significant challenge, as traditional testing techniques have proven insufficient, particularly in dynamic and large-scale systems. To address this issue, various real-time monitoring approaches based on data mining have been proposed. These approaches, through continuous system analysis and the use of anomaly detection techniques, are capable of identifying faults in software systems. This study presents a systematic mapping of the literature on these monitoring techniques, aiming to identify implementation challenges and provide a comprehensive overview of the current state of the research. The mapping was conducted using the Scopus database, and 20 studies were selected to help answer the research questions defined in this work. The analysis enabled the identification of not only the main detection techniques but also the types of monitored systems, fault categories, and evaluation metrics used in the proposed approaches. Among the main challenges identified are the difficulty in labeling data for training supervised models, the complexity of interpreting the generated models, and the need for computationally efficient monitoring.

Keywords: Fault detection; Anomaly detection; Data mining.

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A crescente complexidade dos sistemas de software modernos é impulsionada por múltiplos fatores convergentes: o aumento em escala e heterogeneidade dos sistemas, a expansão para novos domínios de aplicação, a natureza dinâmica das condições operacionais, as demandas competitivas do mercado e a evolução constante das capacidades de hardware. Esta confluência de fatores torna o gerenciamento desses sistemas um desafio cada vez maior, exigindo abordagens mais sofisticadas para garantir sua confiabilidade, como afirmam Galster *et al.* (2017). Salfner *et al.* (2010) ressaltam que, diante desse cenário de crescente complexidade, as técnicas tradicionais de Verificação e Validação apresentam limitações, visto que os métodos tradicionais raramente levam em consideração o estado dinâmico do sistema em execução, o que os impede de capturar sua real complexidade. Testes convencionais, geralmente baseados em casos de teste, não conseguem compreender plenamente o comportamento do sistema. Além disso, a imprevisibilidade das interações entre componentes e a rápida evolução dos estados do sistema exigem abordagens mais adaptativas e automatizadas para a detecção eficaz de faltas.

Dessa forma, diante das limitações das abordagens tradicionais, torna-se essencial adotar estratégias capazes de monitorar o sistema durante sua execução. Nesse contexto, técnicas de monitoramento em tempo real baseadas em mineração de dados mostram-se promissoras, pois possibilitam a análise contínua do sistema, identificando padrões anômalos que indicam a existência de faltas. A mineração de dados permite a extração de informações relevantes a partir de dados coletados do sistema em operação, identificando anomalias que podem indicar a presença de faltas. Essa estratégia possibilita que o sistema acione mecanismos de autorreparo ou forneça informações importantes aos mantenedores do software.

Portanto, torna-se altamente valioso que os profissionais responsáveis pelo desenvolvimento e manutenção de software adotem o monitoramento de faltas baseado em mineração de dados para aprimorar a confiabilidade de sistemas cada vez mais complexos, visto que a incorporação de técnicas de monitoramento de faltas que utilizem mineração de dados contribui para a resiliência dos sistemas e tomada de decisões, otimizando a manutenção e reduzindo falhas nos sistemas.

No entanto, apesar do potencial, ainda são poucos os estudos que oferecem um panorama abrangente das abordagens de monitoramento de faltas baseadas em mineração de dados, identificando de forma clara os desafios, as técnicas de detecção de anomalias, os artefatos utilizados, os tipos de faltas detectáveis e as estratégias de pré-processamento. A falta de

pesquisas que explorem essas questões de maneira integrada, especialmente em ambientes reais, dificulta o progresso e a adoção mais ampla dessas abordagens. Portanto, é crucial que futuras pesquisas se dediquem a explorar essas questões, fornecendo uma visão mais completa da área e identificando as barreiras que precisam ser superadas para desenvolver soluções robustas e eficientes.

Este trabalho apresenta um mapeamento sistemático da literatura sobre o monitoramento de faltas em sistemas de software usando técnicas de detecção de anomalias baseadas em mineração de dados. Conforme Kitchenham e Charters (2007), o mapeamento sistemático, como metodologia de pesquisa, visa reunir estudos sobre uma área específica para fornecer uma visão geral sobre a mesma. Este mapeamento busca responder cinco questões de pesquisa:

- **QP1**: Quais sistemas de software se beneficiam do monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados?
- **QP2**: Quais tipos de faltas podem ser detectadas por meio do monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados, e quais artefatos do sistema são empregados nesse processo?
- **QP3**: Quais técnicas e algoritmos de detecção de anomalias baseados em mineração de dados, bem como métodos de pré-processamento de dados, podem ser utilizadas para monitorar faltas?
- **QP4**: Como as abordagens de monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados são avaliadas?
- **QP5**: Quais são os desafios em implementar o monitoramento de faltas utilizando métodos de detecção de anomalias baseadas em mineração de dados?

Ao responder estas questões, o mapeamento pretende fornecer uma compreensão abrangente do estado atual da pesquisa nesta área, identificando tendências, desafios e oportunidades para trabalhos futuros.

O restante deste trabalho está estruturado da seguinte forma: a seção "Fundamentação Teórica"aborda os principais conceitos necessários para a compreensão do estudo. "Trabalhos Relacionados"apresenta estudos relevantes ao tema e que se assemelham com a proposta deste mapeamento. Em "Objetivos", são definidos o objetivo geral e os específicos. O "Processo Metodológico"descreve o processo de realização do mapeamento sistemático. Em "Resultados", são discutidos os dados obtidos e as respostas às questões de pesquisa. Em "Discussão", apresenta-se uma análise dos resultados obtidos, bem como uma avaliação das limitações identificadas. Por

fim, a seção "Conclusão e Trabalhos Futuros"apresenta as considerações finais e sugere direções para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta o referencial teórico que fundamenta este trabalho, estruturado em três seções principais. A seção 2.1 detalha o processo de realização de um mapeamento sistemático. Em seguida, na seção 2.2, são apresentados os fundamentos da mineração de dados. A seção 2.3 descreve conceitos relacionados à Verificação e Validação de Software, subdividindo-se em três subseções: a terminologia essencial da área, os métodos de detecção de faltas e as técnicas de detecção de anomalias.

2.1 Mapeamento Sistemático

Segundo Petersen *et al.* (2008), o mapeamento sistemático tem como principal objetivo proporcionar uma visão ampla e estruturada sobre uma área de pesquisa específica. Para isso, o mapeamento deve identificar estudos relevantes, sintetizar resultados já consolidados e reunir publicações realizadas ao longo do tempo. Esse processo permite uma compreensão detalhada do estado da arte do tema investigado, além de facilitar a identificação de desafios, tendências emergentes e direções futuras para investigação.

Conforme Kitchenham e Charters (2007), o principal objetivo do mapeamento sistemático é identificar, avaliar e interpretar todas as pesquisas relevantes para o tema em questão, resumindo as evidências existentes sobre tratamentos, tecnologias ou fenômenos de interesse. Os autores também destacam que a razão para realizar um mapeamento sistemático reside na necessidade de compilar informações de forma detalhada e imparcial, facilitando o acesso ao conhecimento existente e fornecendo uma base sólida para pesquisas futuras.

Kitchenham e Charters (2007) também desenvolvem as etapas da realização de um mapeamento sistemático, que deve ser estruturado em três fases principais: Planejamento, Condução e Resultados. Na fase de Planejamento, identifica-se a necessidade de realizar o mapeamento e define-se o protocolo, um artefato essencial que detalha os procedimentos metodológicos a serem seguidos. Ele inclui elementos como as questões de pesquisa que orientam o estudo, os critérios de inclusão e exclusão e a *string* de busca, aplicada em bases de dados para recuperar os estudos pertinentes. As bases de dados, por sua vez, são ferramentas de busca que possuem particularidades específicas, exigindo ajustes na *string* de busca para garantir a recuperação precisa dos trabalhos.

Na fase de Condução, o mapeamento sistemático é executado de fato. Nesta etapa,

os estudos são selecionados e avaliados para garantir sua qualidade e relevância. Após a seleção, procede-se à extração e síntese dos dados, onde as informações coletadas são organizadas e analisadas para responder às questões de pesquisa propostas. Essa fase é crucial para consolidar os dados e identificar padrões ou tendências na literatura.

Por fim, na fase de Resultados, são descritos os resultados obtidos e as conclusões do mapeamento sistemático. Essa etapa também inclui uma discussão crítica sobre os achados, abordando as ameaças à validade do estudo e sugerindo trabalhos futuros que possam expandir ou aprofundar a pesquisa.

Petersen *et al.* (2008) não segmentam o processo de realização do mapeamento sistemático em fases principais, como proposto por Kitchenham e Charters (2007). Em vez disso, Petersen *et al.* (2008) descrevem as atividades existentes nas fases de Kitchenham e Charters (2007) como fases individuais. Assim, devido à estrutura mais consolidada em três macrofases bem definidas (planejamento, condução e resultados), foi adotada a abordagem de Kitchenham e Charters (2007) neste trabalho para realizar um mapeamento sistemático sobre o tema monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados.

2.2 Mineração de Dados

Segundo Hand e Adams (2015), a mineração de dados é uma tecnologia cujo objetivo é extrair estruturas e padrões significativos de conjuntos de dados complexos. Essa disciplina possui uma sobreposição significativa com áreas como estatística, aprendizado de máquina e inteligência artificial, constituindo um campo multidisciplinar. A mineração de dados utiliza uma combinação de técnicas e métodos, incluindo algoritmos de aprendizado de máquina, modelos estatísticos e abordagens de reconhecimento de padrões. A partir da análise de dados históricos, essas técnicas permitem prever comportamentos futuros, identificar grupos de dados semelhantes (*clustering*) e detectar relações entre variáveis (associação).

Fayyad *et al.* (1996) enfatizam o uso da mineração de dados no processo de descoberta de conhecimento em bancos de dados (KDD - *Knowledge Discovery in Databases*). Este processo envolve etapas como limpeza de dados para remoção de ruídos e inconsistências, integração de dados provenientes de múltiplos sistemas, transformação dos dados, aplicação de algoritmos para extração de padrões, e culmina na interpretação e validação dos resultados obtidos.

No contexto da detecção de anomalias, a mineração de dados permite a identificação de observações que se desviam significativamente do comportamento normal esperado. Neste mapeamento, a mineração de dados será explorada como uma ferramenta para auxiliar na detecção de anomalias. Ao analisar grandes volumes de dados gerados durante a execução de sistemas, a mineração de dados permite identificar padrões anômalos que podem indicar faltas em sistemas de software.

2.3 Verificação e Validação de Software

Esta seção introduz conceitos fundamentais relacionados à área de Verificação e Validação de Software, fornecendo uma base teórica para a compreensão das abordagens discutidas ao longo deste trabalho. A subseção 2.3.1 estabelece a terminologia essencial da área. Em 2.3.2, são explorados os conceitos e métodos relacionados à detecção de faltas em sistemas de software. Por fim, a subseção 2.3.3 aborda as técnicas relacionadas à detecção de anomalias.

2.3.1 Terminologia

Esta subseção estabelece a terminologia fundamental relacionada à confiabilidade de software, definindo os conceitos essenciais que serão utilizados ao longo deste trabalho. Conforme IEEE (2010), os seguintes termos relacionados a problemas de software são definidos:

- **Defeito**: Uma imperfeição ou deficiência em um artefato onde ele não atende aos seus requisitos ou especificações e precisa ser reparado ou substituído.
- **Erro**: Um erro é definido como uma ação humana que resulta em um comportamento ou estado incorreto no software.
- **Falta**: Uma falta é a manifestação de um erro no software. Ela ocorre quando o erro se materializa em um comportamento anômalo ou defeito no sistema.
- **Falha**: Uma falha refere-se à incapacidade do software de desempenhar sua função esperada ou de operar dentro dos limites estabelecidos.

Dessa forma, é possível estabelecer as relações entre defeito, erro, falta e falha. O erro representa a causa raiz de um problema no software, podendo resultar em um defeito no sistema. Um defeito, quando manifestado durante a execução do software, é classificado como uma falta. No entanto, quando identificado em fases anteriores à execução, como durante análise estática do código, mantém-se apenas como defeito. Por fim, a falha é o efeito observável que

impacta a funcionalidade ou a operação do software, sendo uma consequência da falta.

Visando tornar o software mais confiável, é essencial mitigar os efeitos dos problemas citados. Shivakumar (2015) ressalta que a redução da probabilidade de falhas em uma aplicação de software pode ser alcançada por meio de quatro etapas principais:

1. **Predição de faltas**: envolve o desenvolvimento de modelos capazes de identificar, ainda antes da execução do sistema, componentes suscetíveis a se tornarem fontes de faltas. Com base nas definições de IEEE (2010), Portanto, baseando-se nas definições de IEEE (2010), essa etapa também pode ser considerada como detecção de defeitos. Geralmente, a predição de faltas é realizada por meio de técnicas de análise estática do código da aplicação.
2. **Prevenção de faltas**: consiste em identificar, durante a etapa de definição de requisitos, áreas do processo de desenvolvimento suscetíveis a gerarem faltas. Uma estratégia comum de prevenção é a adoção de revisões de código como prática padrão no desenvolvimento da aplicação.
3. **Detecção de faltas**: refere-se à identificação de faltas já existentes no sistema. Essa etapa pode ser realizada de diversas formas, incluindo a execução de casos de teste e o monitoramento do sistema com o uso de técnicas de detecção de anomalias. Conforme as definições, essa etapa também pode ser interpretada como predição de falhas, uma vez que a identificação de faltas pode implicar na ocorrência de falhas.
4. **Tolerância a faltas**: trata da estratégia de tornar o sistema resistente a faltas, garantindo que ele continue funcionando mesmo na ocorrência de problemas. Isso pode ser alcançado, por exemplo, por meio do desenvolvimento de sistemas com componentes independentes, de modo que uma falha em um componente não comprometa o funcionamento dos demais.

Outras etapas cruciais para evitar a ocorrência de futuras falhas são os processos de localização e diagnóstico da falta. A localização refere-se à identificação precisa do ponto no código ou no sistema onde a falta se originou, enquanto o diagnóstico busca compreender as causas subjacentes do problema, ou seja, o erro que a desencadeou. Embora esses processos sejam fundamentais para a correção e prevenção de faltas, eles não constituem o foco principal deste trabalho, que se concentra na etapa de detecção. No entanto, esses processos serão brevemente discutidos no contexto das abordagens de detecção, uma vez que o funcionamento da abordagem pode influenciar diretamente a capacidade de localizar e diagnosticar faltas. Por exemplo, algumas abordagens de detecção de faltas não apenas identificam a presença ou

ausência de uma falta, mas também são capazes de determinar seu tipo específico.

A estratégia de detecção de faltas que este trabalho propõe investigar consiste no uso de técnicas de detecção de anomalias. Segundo IEEE (1994), no contexto de Verificação e Validação de Software, o termo "anomalia"era utilizado como sinônimo para conceitos como erro, falta, falha, incidente, defeito ou bug. No entanto, essa definição foi revisada na versão IEEE (2010), que passou a caracterizar "anomalia"como um termo geral para descrever qualquer anormalidade, irregularidade, inconsistência ou desvio em relação ao comportamento esperado de um sistema. Essa atualização afastou-se da abordagem anterior a fim de promover o uso de termos mais específicos, como falha, falta e defeito, visando tornar as discussões e análises sobre problemas de software mais precisas.

Embora a definição apresentado por IEEE (2010) atualmente forneça uma base sólida para descrever anomalias no software, no contexto deste mapeamento sistemático, é adotada a perspectiva apresentada por Chandola *et al.* (2009). Assim como IEEE (2010), Chandola *et al.* (2009) definem anomalias como padrões de comportamento que se desviam do esperado, no entanto, enquanto o padrão IEEE foca na terminologia e categorização dos problemas dentro da área de Engenharia de Software, Chandola *et al.* (2009) abordam as anomalias como parte de um campo de pesquisa mais amplo, voltado para a detecção desses desvios em diferentes domínios. Portanto, o conceito de detecção de anomalias utilizado neste trabalho abrange uma gama de técnicas que podem ser aplicadas no contexto de detecção de faltas.

2.3.2 Detecção de Faltas

A detecção de faltas pode ser realizada por meio de duas abordagens principais: a execução de casos de teste e o monitoramento em tempo real do sistema para identificar ou prever a ocorrência de faltas. Ambas as estratégias são descritas a seguir:

- **Testes Convencionais**: Essa abordagem utiliza oráculos de teste, mecanismos capazes de distinguir comportamentos corretos e incorretos em um sistema. Como ressaltado por Shahamiri *et al.* (2009), um oráculo de teste é responsável por comparar o comportamento observado do sistema com um resultado esperado ou um conjunto de regras predefinidas. Essa abordagem é eficaz em cenários onde os comportamentos esperados são bem conhecidos e podem ser especificados antecipadamente. Dentre os tipos de testes comumente aplicados, destacam-se os testes unitários, que verificam o funcionamento isolado de unidades do código, como funções ou métodos; os testes de integração, que avaliam a

interação entre diferentes módulos ou componentes do sistema; e os testes de desempenho, que analisam o comportamento do sistema sob condições específicas, como carga alta ou uso intensivo de recursos.

- **Monitoramento em Tempo de Execução**: Diferentemente dos testes convencionais, o monitoramento contínuo não depende de oráculos de teste predefinidos. Em vez disso, ele detecta desvios em relação ao comportamento normal do sistema. Essa abordagem é especialmente relevante para sistemas complexos e dinâmicos, como destacado por Salfner *et al.* (2010), pois é capaz de identificar padrões anômalos em tempo real, mesmo em cenários onde é difícil de definir de forma precisa o comportamento esperado. No entanto, uma limitação dessa abordagem é que nem sempre é claro se uma anomalia detectada representa de fato uma falta. Por exemplo, um comportamento incomum, mas válido dentro do funcionamento esperado do sistema, pode ser erroneamente classificado como uma falta, resultando em um falso positivo.

Uma vez detectada uma falta, a próxima etapa é localizá-la e realizar seu diagnóstico. A localização refere-se à identificação do ponto exato no código ou no sistema onde a falta ocorreu. Já o diagnóstico busca compreender as causas subjacentes da falta, ou seja, o erro que a causou. Embora esses processos sejam etapas fundamentais para a correção e prevenção de faltas, eles não são o foco principal deste mapeamento sistemático, que se concentra na etapa de detecção. No entanto, esses processos serão brevemente discutidos no contexto das abordagens de detecção, uma vez que o funcionamento da abordagem pode influenciar diretamente a capacidade de localizar e diagnosticar faltas. Por exemplo, algumas abordagens de detecção de faltas não apenas identificam a presença ou ausência de uma falta, mas também são capazes de determinar seu tipo específico.

2.3.3 Detecção de Anomalias

Diversas abordagens para detecção de anomalias já foram propostas e podem ser classificadas em dois tipos principais: métodos estatísticos e métodos baseados em regras. Os métodos estatísticos examinam dados históricos para identificar padrões normais de comportamento e sinalizam desvios sempre que ocorrem variações inesperadas. Por outro lado, os métodos baseados em regras estabelecem limites ou condições específicas para o funcionamento do sistema, detectando anomalias sempre que essas regras são violadas. Essas regras podem ser baseadas em conhecimento especializado, políticas de segurança ou especificações técnicas.

Neste trabalho, serão explorados exclusivamente os métodos estatísticos.

Em sua taxonomia, Grohmann *et al.* (2020) classificam as técnicas de detecção de anomalias em duas: detecção baseada em previsão de séries temporais e detecção baseada em modelagem do estado normal. Ambas as abordagens buscam modelar o comportamento normal do sistema e classificar as anomalias baseado no desvio do comportamento esperado, mas diferem em sua implementação e aplicação, conforme detalhado a seguir:

- **Detecção baseada em previsão de séries temporais**: séries temporais são sequências de dados coletados em intervalos de tempo específicos, representando o comportamento de um sistema ao longo do tempo. Essa técnica utiliza dados históricos para prever o comportamento futuro do sistema. Quando o comportamento previsto diverge significativamente do comportamento real observado, uma anomalia é identificada e alertada, o que evidentemente permite a detecção de faltas.
- **Detecção baseada em modelagem do estado normal**: essa abordagem utiliza modelos mais explícitos que se concentram no desvio dos dados monitorados em relação ao comportamento esperado do sistema. Técnicas como *clustering* das condições normais e definição de *thresholds* adaptativos são comumente empregadas. Esses métodos são eficientes para identificar anomalias com base em padrões predefinidos de normalidade, permitindo a detecção de desvios que fogem ao comportamento padrão. É importante destacar que essa estratégia pode empregar séries temporais na criação dos modelos, porém, não utilizará a previsão de dados futuros para detectar anomalias.

Além dessas duas categorias, para o contexto deste mapeamento, também será considerada uma terceira abordagem: classificadores baseados em aprendizagem de máquina. Essa técnica é relevante porque a detecção de anomalias pode ser interpretada como uma tarefa de classificação, na qual os dados são categorizados como anômalos ou não anômalos.

As técnicas de detecção de anomalias também podem ser classificadas conforme a metodologia de treinamento dos modelos, organizadas em dois grupos principais, como descrito a seguir:

- **Técnicas Supervisionadas**: Essas técnicas dependem de dados rotulados para o treinamento do modelo. Isso significa que o conjunto de dados utilizado para treinar o algoritmo contém exemplos claramente identificados como "normais"ou "anômalos". Durante o treinamento, o modelo aprende a distinguir entre esses dois estados com base nos padrões presentes nos dados rotulados. Após o treinamento, o modelo é capaz de classificar novos

dados não vistos anteriormente, atribuindo-lhes uma categoria (normal ou anômalo).

- **Técnicas Não Supervisionadas**: Diferentemente das técnicas supervisionadas, as técnicas não supervisionadas não requerem dados rotulados para treinamento. Em vez disso, elas aprendem os padrões normais de comportamento a partir dos dados disponíveis, sem a necessidade de exemplos prévios de anomalias. Durante a execução, o modelo sinaliza como anômalos os comportamentos que se desviam significativamente desses padrões normais.

O entendimento das classificações das técnicas de detecção de anomalias é fundamental para compreender o funcionamento das abordagens de monitoramento de faltas, pois cada técnica oferece uma maneira distinta de identificar comportamentos fora do padrão esperado.

3 TRABALHOS RELACIONADOS

Esta seção apresenta uma análise de quatro estudos relacionados à área de monitoramento de faltas, destacando suas abordagens metodológicas, populações investigadas e estratégias de intervenção. A Tabela 1 resume os aspectos principais de cada trabalho.Salfner *et al.* (2010) e Grohmann *et al.* (2020) apresentam taxonomias de métodos de previsão de falhas, que, conforme a definição em IEEE (2010), equivale à detecção de faltas, pois prever falhas exige detectá-las previamente. Bhandari *et al.* (2022) mapeiam problemas em conjuntos de dados de previsão de faltas e propõem estratégias de mitigação, focando na análise do código-fonte antes da execução do sistema. Jieyang *et al.* (2022) revisam técnicas de detecção de faltas baseadas em mineração de dados para sistemas industriais *Big Data*.

O trabalho de Salfner *et al.* (2010) é um *survey* que fornece uma visão abrangente do tema de previsão de falhas em tempo real. Este *survey* oferece contribuições significativas para a compreensão do estado da arte até 2010. Os autores também desenvolvem uma taxonomia que classifica as abordagens de previsão de falhas em três tipos principais baseadas no tipo de dados utilizados: rastreamento de falhas, monitoramento de sintomas e relato de erros detectados. O *survey* mapeia as técnicas existentes e suas áreas de aplicação, além de analisar as métricas de avaliação comumente utilizadas para validar a eficácia dos métodos de previsão. No entanto, por ser um *survey*, não segue uma metodologia sistemática de busca e seleção de estudos. Além disso, não mapeia os diferentes tipos de falhas que podem ser previstas. Por fim, tendo sido publicado em 2010, não contempla os avanços significativos ocorridos na última década nas áreas de Verificação e Validação de Software e detecção de anomalias.

Grohmann *et al.* (2020) propõe uma taxonomia das técnicas de previsão de falhas em Objetivos a Nível de Serviço (SLO) em sistemas de software. Falhas de SLO se referem à inabilidade do sistema em cumprir os objetivos de qualidade de serviço definidos pelo usuário. A taxonomia foi desenvolvida a partir de um mapeamento sistemático que abrangeu 67 artigos e outros documentos relevantes, organizando as técnicas de previsão de falhas em três dimensões: (1) o alvo da previsão, (2) o horizonte temporal, e (3) o tipo de modelagem. O estudo também classifica as técnicas de detecção de anomalia em dois tipos: técnicas de predição de séries temporais e técnicas baseadas na modelagem do estado normal do sistema. Entretanto, o estudo deixa uma lacuna ao não explorar outras características, como áreas de aplicação, métricas de avaliação utilizadas para medir a eficácia das técnicas e, principalmente, métodos de pré-processamento dos dados.

Já no estudo de Bhandari *et al.* (2022), os autores realizam uma investigação de problemas em conjuntos de dados de previsão de faltas, identificando desafios como alta dimensionalidade, valores faltantes e desbalanceamento de classes. São analisadas várias abordagens mitigadoras, incluindo algoritmos de *aprendizado de máquina* e técnicas de pré-processamento. Diferentemente desse trabalho, que foca em previsão de faltas (análise preditiva pré-execução), o presente trabalho mapeia abordagens de detecção de faltas em tempo real durante a operação do sistema, estabelecendo uma distinção fundamental nos objetivos e metodologia.

O trabalho de Jieyang *et al.* (2022) apresenta uma revisão sistemática das abordagens de detecção de faltas baseadas em mineração de dados, com foco específico em máquinas e sistemas industriais de Big Data, incluindo sistemas industriais de Internet das Coisas (IoT). Os pesquisadores revelam duas metodologias fundamentais de processamento de dados: o processamento em lotes, onde conjuntos de dados separados são analisados de forma agrupada, e o processamento de *streams*, que permite a análise contínua e em tempo real dos dados à medida que são gerados. As abordagens identificadas por Jieyang *et al.* (2022) são intrinsecamente contextualizadas para sistemas industriais, concentrando-se primariamente no monitoramento e detecção de faltas em máquinas industriais, com ênfase em técnicas de análise de desempenho e identificação de anomalias em ambientes de produção. Em contraste, o presente mapeamento direciona seu foco para faltas relacionadas a software, estabelecendo uma diferença significativa no escopo da investigação.

Tabela 1 – Comparação Trabalhos Relacionados

Título	Referência	Proposta	População	Intervenção
A Survey of Online Failure Prediction Methods	Salfner *et al.* (2010)	Survey e taxonomia	Sistemas sujeito a falhas	Métodos de previsão de falhas
A Taxonomy of Techniques for SLO Failure Prediction in Software Systems	Grohmann *et al.* (2020)	Mapeamento sistemático e taxonomia	Sistemas sujeito a falhas de SLO	Métodos de previsão de falhas
Data quality issues in software fault prediction: a systematic literature review	Bhandari *et al.* (2022)	Revisão sistemática	Conjuntos de dados de previsão de falhas	Identificação dos problemas de qualidade de dados e técnicas de mitigação
A systematic review of data-driven approaches to fault diagnosis and early warning	Jieyang *et al.* (2022)	Revisão sistemática	Sistemas industriais Big Data sujeito a falhas	Técnicas de detecção de falta baseadas em mineração de dados

Fonte: Elaborada pelo autor

4 OBJETIVOS

Este trabalho pretende investigar como as abordagens de detecção de anomalias baseadas em mineração de dados são aplicadas ao monitoramento de faltas. Para alcançar uma compreensão mais abrangente e detalhada do tema, são propostos os seguintes objetivos específicos:

- Identificar algoritmos e métodos de pré-processamento baseados em mineração de dados aplicados no monitoramento de faltas;
- Identificar e analisar tipos de sistemas de software que se beneficiam do monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados;
- Identificar e analisar os tipos de faltas que podem ser detectados com o monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados e os artefatos utilizados no processo; e
- Identificar e analisar como as técnicas de monitoramento de faltas baseadas em mineração de dados podem ser avaliadas.

5 PROCEDIMENTO METODOLÓGICO

A abordagem definida para a condução da pesquisa é o mapeamento sistemático. O mapeamento sistemático permite estruturar o estado da arte de um campo de estudo, identificando tendências, lacunas e oportunidades de pesquisa.

A metodologia de execução do mapeamento é baseada nas diretrizes propostas por Kitchenham e Charters (2007). A metodologia proposta é organizada em três fases principais: Planejamento, Condução e Apresentação dos Resultados. Nas subseções seguintes é detalhado o processo de realização de cada fase.

5.1 Planejamento

A fase de planejamento estabelece a fundação para a condução do mapeamento sistemático. Essa fase exige determinar a necessidade da revisão sistemática e, então, desenvolver um protocolo detalhado da revisão, que estabelece os métodos a serem utilizados e também ajuda a reduzir o viés do pesquisador.

Após uma avaliação da relevância deste mapeamento, foi desenvolvido um protocolo (disponível no apêndice A) que orienta a condução do mapeamento sistemático, descrevendo cada fase do processo. Uma das atividades previstas no protocolo é a definição das questões de pesquisa, as quais foram elaboradas com base no tema central do estudo e nos objetivos da pesquisa, tendo como função direcionar a investigação. As questões definidas são as seguintes:

- **QP1**: Quais sistemas de software se beneficiam do monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados?
- **QP2**: Quais tipos de faltas podem ser detectadas por meio do monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados, e quais artefatos do sistema são empregados nesse processo?
- **QP3**: Quais técnicas e algoritmos de detecção de anomalias baseados em mineração de dados, bem como métodos de pré-processamento de dados, podem ser utilizadas para monitorar faltas?
- **QP4**: Quais técnicas de pré-processamento de dados podem ser utilizadas para auxiliar modelos de monitoramento de faltas baseados em mineração de dados?
- **QP5**: Como as abordagens de monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados são avaliadas?

- **QP6**: Quais são os desafios em implementar o monitoramento de faltas utilizando métodos de detecção de anomalias baseadas em mineração de dados?

O passo seguinte é a criação de uma estratégia de busca, dividida em 5 subtarefas: (1) método de busca; fontes de pesquisa; (3) *string* de busca; (4) critérios de seleção; e (5) processo de seleção dos estudos.

Para a seleção das fontes de pesquisa, optou-se pela utilização da base de dados Scopus. Essa escolha foi motivada pelo seu extenso acervo de publicações científicas na área de Engenharia de Software. A Scopus também oferece publicações suficientes que propõe soluções de monitoramento de faltas com mineração de dados para embasar o mapeamento sistemático.

Para realizar a busca na base de dados Scopus, utilizou-se uma *string* de busca, uma combinação estruturada de palavras-chave e operadores booleanos feita para localizar publicações relevantes para o mapeamento. A *string* foi projetada para capturar estudos que propõem abordagens de monitoramento de faltas em sistemas de software, utilizando técnicas de detecção de anomalias que baseadas em mineração de dados, e que operem em tempo real (durante a execução do sistema). Além disso, a busca foi restrita a estudos aplicáveis ao domínio da Engenharia de Software. A *string* de busca definida é a seguinte:

((("fault detection"OR "fault verification"OR "fault diagnosis"OR "fault identification"OR "fault localization"OR "fault monitoring")

OR

("anomaly detection"OR "anomaly verification"OR "anomaly diagnosis"OR "anomaly identification"OR "anomaly localization"OR "anomaly monitoring"))

AND

("runtime"OR "run-time"OR "realtime"OR "real time"OR "real-time"OR "dynamic"OR "online"OR "on-line"OR "execution time"OR "operational phase")

AND

("software engineering"OR "software system"OR "software application"OR "software development"OR "software process")

AND

("data mining"OR "machine learning"OR "pattern recognition"OR "statistical learning"OR "statistical analysis"OR "data analysis"))

Para garantir a relevância e a qualidade dos estudos selecionados, foram estabelecidos

critérios de inclusão e exclusão, que orientaram o processo de triagem. Esses critérios são descritos a seguir:

Critérios de Inclusão

1. Estudo deve apresentar abordagem de detecção de faltas utilizando mineração de dados.
2. Estudo deve apresentar abordagem de detecção de faltas que funcione em tempo de execução.
3. Estudo deve estar acessível em texto completo.
4. Estudo deve estar escrito em português ou inglês.
5. Estudo deve estar dentro do contexto de engenharia de software.

Critérios de Exclusão

1. Estudo apresenta abordagem focada no monitoramento de faltas de hardware
2. Estudo apresenta abordagem que utiliza dados de sensores ou outro tipo de hardware para monitoramento
3. Estudo apresenta abordagem que utiliza oráculo de testes

O processo de seleção dos estudos ocorreu em duas etapas. Primeiro, foi realizada uma triagem preliminar por meio da leitura dos títulos e resumos dos estudos obtidos pela *string* de busca, avaliando sua conformidade com os critérios de inclusão e exclusão. Em seguida, na segunda etapa, verificou-se a acessibilidade ao texto completo e realizou-se a leitura integral dos artigos que passaram pela triagem inicial, selecionando aqueles considerados relevantes.

Os critérios de inclusão e exclusão foram definidos com base no objetivo central do mapeamento sistemático: identificar estudos primários que propõem soluções viáveis para a detecção de faltas ou anomalias em sistemas de software, utilizando técnicas de mineração de dados ou aprendizado de máquina, e operem em tempo real. A exclusão de estudos relacionados a hardware ou que dependam de dados de hardware foi essencial para garantir que as abordagens identificadas sejam aplicáveis ao contexto de sistemas de software. Além disso, para assegurar a qualidade e a confiabilidade da análise, apenas estudos acessíveis na íntegra e escritos em português ou inglês foram considerados, pois mesmo que a string de busca esteja em inglês, é possível que apareçam artigos em português que sejam encontrados por terem resumo em inglês.

Também foram excluídos estudos que utilizam oráculos de testes, uma vez que

essas abordagens estão associadas testes que não empregam técnicas de detecção de anomalia e métodos estatísticos para a detecção de faltas. Essa exclusão foi necessária para manter o foco em abordagens dinâmicas e em tempo real, alinhadas com os objetivos do mapeamento sistemático.

Para organizar e mapear as informações relevantes de cada estudo, foi elaborado um formulário de extração. Esse formulário foi projetado para capturar dados de cada estudo selecionado, permitindo uma análise do panorama do tema. O formulário preenchido pode ser acessado online[1] e as informações coletadas foram:

- Uma descrição geral do estudo
- O funcionamento da solução proposta
- O tipo de aplicação em que a solução é aplicada, como sistemas distribuídos, IoT ou microsserviços
- O tipo de falta monitorada, como problemas relacionados a desempenho ou segurança
- Os artefatos de sistema monitorados, como *logs* ou uso de CPU
- Tipo de abordagem de detecção de anomalias utilizada (classificação com aprendizado de máquina, modeladagem do estado normal ou previsão de séries temporais)
- Métodos de pré-processamento de dados aplicados
- Tipo de técnica de treino utilizada (supervisionada ou não supervisionada).
- Algoritmos de mineração de dados utilizados
- As métricas de avaliação utilizadas
- Notas sobre o estudo

5.2 Condução

Nesta fase, é conduzido o mapeamento sistemático, seguindo a estratégia de busca definida na fase de Planejamento. Esse processo envolve a execução das consultas na base Scopus, utilizando a *string* de busca estabelecida. A busca inicial resultou em 90 estudos potencialmente relevantes.

Na primeira de seleção, foi realizada uma análise dos resumos dos 90 estudos inicialmente identificados. Com base nessa análise, 52 estudos foram excluídos por não atenderem aos critérios de inclusão e exclusão, por exemplo, por apresentarem abordagens relacionadas a hardware, dependência de dados de hardware, ou por não estarem alinhados ao escopo de

[1] https://github.com/PauloFRC/FormularioExtracaoMonitoramentoFaltas

detecção dinâmica e em tempo real. Após essa etapa, 38 estudos permaneceram para avaliação mais detalhada.

Na segunda etapa, dos 38 estudos restantes, não foi possível acessar o texto completo de 14. Os 24 estudos restantes foram submetidos à leitura completa. Dessa análise, quatro estudos foram excluídos por não cumprirem os critérios de inclusão e exclusão. Ao final da condução, 20 estudos foram selecionados para compor o mapeamento sistemático, sendo importante destacar que nenhum estudo em português foi encontrado. A Tabela 2 apresenta a quantidade de estudos selecionados em cada etapa do processo de seleção. Os estudos selecionados ao final estão listados nas Tabelas 3 e 4 com ID, título, referência, autores e ano.

Tabela 2 – Etapas de Seleção dos Estudos

Artigos Retornados pela Scopus	Seleção pelo Resumo (1° Etapa)	Seleção pela Leitura Completa (2° Etapa)
90	38	20

Fonte: Elaborada pelo autor

Tabela 3 – Estudos Finais

ID	Título	Referência	Autores	Ano
P1	2D2N: A Dynamic Degenerative Neural Network for Classification of Images of Live Network Data	Flanagan *et al.* ()	Kieran Flanagan, Enda Fallon, Paul Jacob, Abir Awad e Paul Connolly	2019
P2	A Dynamic Anomaly Detection Approach Based on Permutation Entropy for Predicting Aging-Related Failures	Wang *et al.* ()	Shuguang Wang, Minyan Lu, Shiyi Kong e Jun Ai	2020
P3	An RBM Anomaly Detector for the Cloud	Monni *et al.* ()	Cristina Monni, Mauro Pezze e Gaetano Prisco	2019
P4	Anomaly-based Fault Detection System in Distributed System	Kim e Hariri (2007)	Byoung uk Kim e Salim Hariri	2007
P5	Automatically repairing tensor shape faults in deep learning programs	Wu *et al.* (2022)	Dangwei Wu, Beijun Shen, Yuting Chen, He Jiang e Lei Qiao	2022
P6	DB-Outlier Detection Algorithm using Divide and Conquer approach over Dynamic DataStream	Elahi *et al.* (2008)	Manzoor Elahi, Xinjie Lv, Wasif Nisar, Imran Ali Khan, Ying Qiao e Hongan Wang	2008
P7	iFeedback: Exploiting User Feedback for Real-time Issue Detection in Large-Scale Online Service Systems	Zheng *et al.* ()	Wujie Zheng, Haochuan Lu, Yangfan Zhou, Jianming Liang, Haibing Zheng e Yuetang Deng	2019
P8	Improving log anomaly detection via spatial pooling: Combining SPClassifier with ensemble method	Uchida *et al.* (2024)	Hironori Uchida, Keitaro Tominaga, Hideki Itai, Yujie Li e Yoshihisa Nakatoh	2024

Fonte: Elaborada pelo autor

Tabela 4 – Estudos Finais (continuação)

ID	Título	Referência	Autores	Ano
P9	Linnaeus: A highly reusable and adaptable ML based log classification pipeline	Catovic *et al.* ()	Armin Catovic, Carolyn Cartwright, Yasmin Tesfaldet Gebreyesus e Simone Ferlin	2021
P10	Log-based Anomaly Detection Without Log Parsing	Le e Zhang (2021)	Van-Hoang Le e Hongyu Zhang	2021
P11	Machine learning based network intrusion detection for data streaming IoT applications	Yahyaoui *et al.* ()	Aymen Yahyaoui, Haithem Lakhdhar, Takoua Abdellatif, Rabah Attia	2021
P12	Machine Learning-Based Run-Time Anomaly Detection in Software Systems: An Industrial Evaluation	Huch *et al.* ()	Fabian Huch, Mojdeh Golagh, Ana Petrovska e Alexander Krauss	2018
P13	MapReduce based Classification for Fault Detection in Big Data Applications	Shafiq *et al.* ()	M. Omair Shafiq, Maryam Fekri e Rami Ibrahim	2017
P14	Online Diagnosis of Performance Variation in HPC Systems Using Machine Learning	Tuncer *et al.* ()	Ozan Tuncer, Emre Ates, Yijia Zhang, Ata Turk, Jim Brandt, Vitus J. Leung, Manuel Egele e Ayse K. Coskun	2019
P15	Practical Anomaly Detection over Multivariate Monitoring Metrics for Online Services (PER)	Liu *et al.* (2023a)	Jinyang Liu, Tianyi Yang, Zhuangbin Chen, Yuxin Su, Cong Feng, Zengyin Yang e Michael R. Lyu	2023
P16	Proactive Detection of Software Aging Mechanisms in Performance Critical Computers	Gross *et al.* ()	Kenny C. Gross, Vatsal Bhardwaj e Randy Bickford	2002
P17	Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection	Brown *et al.* ()	Andy Brown, Brian Hutchinson, Aaron Tuor e Nicole Nichols	2018
P18	Self-Supervised Log Parsing	Nedelkoski *et al.* ()	Sasho Nedelkoski1, Jasmin Bogatinovski1, Alexander Acker, Jorge Cardoso e Odej Kao	2021
P19	TRACK-Plus: Optimizing Artificial Neural Networks for Hybrid Anomaly Detection in Data Streaming Systems	Alnafessah e Casale ()	Ahmad Alnafessah e Giuliano Casale	2020
P20	Unlocking Deeper Understanding: Leveraging Explainable AI for API Anomaly Detection InsightsUnlocking Deeper Understanding: Leveraging Explainable AI for API Anomaly Detection Insights	Jones *et al.* ()	Mike Jones, Masrufa Bayesh e Sharmin Jahan	2024

Fonte: Elaborada pelo autor

6 RESULTADOS

Nesta seção, são apresentados os resultados do mapeamento sistemático, visando responder às questões de pesquisa propostas. A discussão dos resultados fornece uma visão abrangente dos achados, contribuindo para o avanço do conhecimento na área.

6.1 QP1 - Quais sistemas de software se beneficiam do monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados?

Entender quais sistemas de software se beneficiam do monitoramento de faltas baseado em mineração de dados é essencial para orientar estudos e implementações futuras. Ao identificar os contextos em que essas técnicas são mais eficazes, pesquisadores e desenvolvedores podem direcionar esforços para aprimorar a detecção e a mitigação de faltas. Na tabela 5 são apresentados os tipos de aplicações encontrados e os estudos que abordaram cada uma.

A análise revelou que os sistemas de software de grande escala e alta complexidade são os que mais se beneficiam com o monitoramento de faltas baseado em mineração de dados para detecção de faltas. Os estudo P2 e P3 destacam que a crescente complexidade dos softwares modernos amplia os desafios relacionados à manutenibilidade, citando sistemas de nuvem como um exemplo. P8 argumenta que os sistemas de software modernos, devido à sua complexidade, geram enormes volumes de dados, o que torna as análises desafiadoras. P11 comenta que os sistemas de *Internet of Things* (IoT) estão sendo cada vez mais utilizados, o que aumenta a complexidade na análise e no gerenciamento dos dados gerados. Esses exemplos ilustram como a complexidade e a escala dos sistemas modernos exigem abordagens avançadas de monitoramento para garantir sua confiabilidade e eficiência.

A inviabilidade dos testes tradicionais torna-se ainda mais evidente em sistemas de *Big Data* que processam *streaming* de dados, como os descritos nos estudos P6, P13 e P19. Esses estudos destacam que a alta complexidade e o volume massivo de dados tornam a detecção manual ou estática de faltas impraticável, exigindo abordagens automatizadas. Da mesma forma, os estudos P14 e P15 reforçam a importância da detecção automatizada de anomalias em sistemas críticos e de alto desempenho, nos quais falhas podem resultar em impactos econômicos e operacionais significativos. No entanto, esses dados gerados em grande escala podem ser úteis para treinar modelos que auxiliem no monitoramento de faltas, permitindo a prevenção de falhas.

A detecção de anomalias também desempenha um papel crucial em *self-healing*

systems, sistemas projetados para reparar faltas e autorrecuperar de problemas durante a execução. Um exemplo é o gerenciamento do fenômeno de *software aging* (envelhecimento de software), abordado nos estudos P2 e P16. Esse fenômeno, caracterizado pela degradação progressiva do desempenho devido ao acúmulo de erros ou vazamentos de recursos, afeta diretamente a confiabilidade e a disponibilidade de sistemas de longa execução. Para mitigar esses efeitos, adota-se o *software rejuvenation* (rejuvenescimento de software), um processo que, segundo Cotroneo *et al.* (2014), inclui ações como liberação de recursos e recarga de recursos do sistema. Considerando que o melhor momento para essas ações serem executadas, conforme Dohi *et al.* (2000), é quando o sistema está próximo da falha, usar técnicas de monitoramento de faltas é essencial para a otimização do *software rejuvenation*.

Tabela 5 – Tipos de aplicações utilizadas no monitoramento de faltas

Tipo de Aplicação	Estudos	Quantidade	Porcentagem
Sistemas online	P1, P2, P3, P7, P15	5	25%
Sem tipo específica	P8, P10, P16, P17, P18	5	25%
Sistemas Big Data	P6, P13, P19	3	15%
Sistemas industriais de larga escala	P9, P12	2	10%
Sistemas na nuvem	P3	1	5%
Sistemas distribuídos	P4	1	5%
Sistemas de *deep learning*	P5	1	5%
Sistemas IoT	P11	1	5%
Sistemas de alta performance	P14	1	5%
APIs	P20	1	5%

Fonte: Elaborada pelo autor

6.2 QP2 - Quais artefatos de sistema são utilizados para monitorar diferentes tipos de falta?

A compreensão dos tipos de faltas detectáveis e seus respectivos artefatos de monitoramento é crucial para direcionar estratégias de pesquisa e implementação no campo de monitoramento de faltas. O mapeamento dos tipos de faltas, dos artefatos utilizados em cada estudo e da relação entre eles evidencia quais recursos são mais eficazes para identificar diferentes categorias de faltas em sistemas de software.

Na Tabela 6 são apresentados os tipos de faltas utilizados para testar a abordagem de cada estudo. Estudos que empregam um ou mais conjuntos de dados diversificados, bem como aqueles que utilizam dados de faltas sintéticos (gerados artificialmente), são classificados como detecção de tipos de faltas "geral", enquanto os estudos que não esclarecem o tipo de falta

monitorada no experimento são classificados como "não definido". Dos 20 estudos analisados, a distribuição dos tipos de faltas abordados é a seguinte: 4 estudos tratam de faltas relacionadas ao desempenho, 4 focam em faltas associadas à Segurança da Informação, e 5 abordam faltas relacionadas ao tráfego de rede (incluindo as relacionadas à Segurança da Informação). Além disso, 5 estudos lidam com faltas gerais, enquanto os demais se concentram em tipos específicos, como faltas de módulos do sistema operacional, formato de tensores, faltas sintéticas, interface do usuário, componentes do sistema e um estudo que não define explicitamente o tipo de falta. É importante ressaltar que muitas das técnicas encontradas são genéricas o suficiente para funcionar com diferentes tipos de falta, mesmo que só tenham sido realizados experimentos para um tipo de falta específico. Por exemplo, P17 testa sua abordagem com dados de faltas relacionadas à Segurança da Informação, mas destaca que poderia ser utilizada para outros tipos de faltas.

A Tabela 7 apresenta os artefatos monitorados em cada estudo. Estudos que utilizam conjuntos de dados com múltiplos artefatos são classificados como "geral". Dos 20 estudos analisados, 7 utilizam logs, 6 monitoram recursos do sistema (como uso de memória, CPU e operações de I/O), 6 focam no tráfego de rede, 1 emprega dados do sistema, 1 utiliza feedback dos usuários e 2 são classificados como "geral".

Tabela 6 – Tipos de falta monitoradas

Tipo de Falta	Estudos	Quantidade	Porcentagem
Geral	P6, P8, P9, P10, P15, P18	6	30%
Tráfego de rede	P1, P3, P4, P11, P17, P20	6	25%
Segurança da informação	P1, P11, P17, P20	4	20%
Desempenho	P2, P14, P16, P19	4	20%
Módulos do sistema operacional	P4	1	5%
Formato de tensores	P5	1	5%
Interface do usuário	P7	1	5%
Componentes do sistema	P13	1	5%
Não definido	P12	1	5%

Fonte: Elaborada pelo autor

Tabela 7 – Artefatos monitorados

Artefato	Estudos	Quantidade	Porcentagem
Logs	P8, P9, P10, P13, P17, P18, P19	7	35%
Recursos do sistema	P2, P3, P4, P12, P14, P16	6	30%
Tráfedo de rede	P1, P3, P4, P11, P14, P20	6	25%
Geral	P6, P15	2	10%
Dados do sistema	P5	1	5%
Feedback do usuário	P7	1	5%

Fonte: Elaborada pelo autor

A análise dos estudos revela que 35% abordagens de monitoramento de faltas baseadas em mineração de dados (P8, P9, P10, P13, P17, P18, P19) utilizam *logs* como artefato para detecção de anomalias. Jia *et al.* (2018) evidenciam como *logs* são uma das informações mais importantes para detectar faltas em um sistema. Isso se deve ao fato de que os *logs* são altamente eficientes em capturar o comportamento do sistema e seu estado geral, pois podem registrar eventos, operações e possíveis problemas. Portanto, por seu poder de expressão, *logs* permitem detectar quase todo tipo de falta. Essa combinação entre a riqueza de informações contidas nos *logs* e as técnicas avançadas de automação da análise torna as abordagens baseadas em *logs* umas das mais promissoras e aplicáveis para a detecção de faltas em sistemas de software. O funcionamento das técnicas de análise dos *logs* são discutidas na resposta na questão de pesquisa QP4.

No contexto de segurança, o monitoramento de faltas baseado em mineração de dados mostra-se particularmente vantajoso, especialmente com a aplicação de técnicas não supervisionadas. Como destacado em P1, abordagens de detecção de anomalia não supervisionadas permitem a detecção de vulnerabilidades "zero-day"(falhas de segurança ainda não documentadas e para as quais não existem correções disponíveis) e eventos "Cisne Negro"(ocorrências raras e imprevisíveis com alto impacto no sistema), superando as limitações das técnicas tradicionais, que tipicamente dependem de bases de conhecimento de ameaças conhecidas. A eficácia dessas técnicas deriva de sua capacidade de identificar desvios significativos dos padrões normais do sistema, sem a necessidade de assinaturas pré-definidas. Neste cenário, os estudos P1, P11, P17 e P20 propuseram abordagens para detectar faltas relacionadas à Segurança da Informação. Entre eles, apenas P17 não monitora o tráfego de rede, optando por analisar logs como artefato principal. Como ressaltado por P1, a atividade da rede serve como um indicador robusto para identificar invasões e comportamentos maliciosos, destacando sua utilidade na detecção de ameaças.

Da mesma forma, o monitoramento de faltas baseado em mineração de dados também se destaca na detecção de problemas de desempenho, permitindo a identificação de degradação sistêmica, gargalos de recursos e faltas associadas ao *software aging*. Os estudos P2 e P16 monitoram recursos do sistema como uso de CPU e de memória para detectar faltas relacionadas ao *software aging*. De maneira semelhante, P14 realiza experimentos para monitorar faltas relacionadas ao desempenho utilizando métricas de CPU e memória, além de monitorar a atividade de rede. O único estudo que apresenta uma abordagem de detecção de faltas de

desempenho sem monitorar recursos do sistema é P19, que utiliza *logs* como artefato. Portanto, recursos do sistema, como uso de CPU e memória, são os principais artefatos utilizados para monitorar faltas associadas ao desempenho.

6.3 QP3 - Quais técnicas e algoritmos de detecção de anomalias baseados em mineração de dados, bem como métodos de pré-processamento de dados, podem ser utilizadas para monitorar faltas?

Como destacado na Fundamentação Teórica, as técnicas de detecção de anomalias podem ser classificadas de diferentes maneiras. Na Tabela 8, os estudos selecionados foram organizados com base no tipo de abordagem utilizada para a solução. Observa-se que as técnicas de classificação com aprendizado de máquina e as técnicas baseadas na modelagem do estado normal do sistema estão bem representadas, com 13 abordagens utilizando classificação e 7 baseadas em modelagem do estado normal, predominando as técnicas de classificação entre os estudos. No entanto, apenas um dos estudos selecionados emprega a previsão de séries temporais como método para detecção de anomalias. Além disso, um dos estudos, P18, apresenta duas abordagens, uma de classificação usando aprendizado de máquina e uma que usa modelagem do estado normal.

Por outro lado, na Tabela 9, os estudos são categorizados conforme o tipo de algoritmo de treinamento da solução, ou seja, supervisionado ou não supervisionado. A análise revela que ambas as abordagens possuem uma representação equilibrada entre os estudos selecionados, com 12 abordagens supervisionadas e 9 não supervisionadas, demonstrando a viabilidade e a aplicabilidade de ambas as técnicas para a detecção de faltas em tempo real.

Tabela 8 – Classificação de técnicas por tipo de abordagem

Tipo de Treinamento	Estudos	Quantidade	Porcentagem
Classificação usando aprendizado de máquina	P4, P5, P7, P8, P9, P10, P11, P12, P13, P14, P18, P19, P20	13	65%
Detecção por modelagem do estado normal	P1, P2, P3, P6, P16, P17, P18	7	35%
Detecção por previsão de séries temporais	P15	1	5%

Fonte: Elaborada pelo autor

Para responder a esta questão de pesquisa, as técnicas foram organizadas em três subseções principais. A subseção 6.3.1 abordará as técnicas classificadoras, que incluem métodos

Tabela 9 – Classificação de técnicas por tipo de treinamento

Tipo de Abordagem	Estudos	Quantidade	Porcentagem
Supervisionada	P4, P5, P7, P8, P9, P10, P11, P12, P13, P14, P18, P20	12	60%
Não supervisionada	P1, P2, P3, P6, P15, P16, P17, P18, P19	9	45%

Fonte: Elaborada pelo autor

supervisionados baseados em algoritmos de aprendizado de máquina para classificação. A subseção 6.3.2 focará nas técnicas baseadas na previsão de séries temporais, que utilizam dados históricos para prever comportamentos futuros e identificar desvios. A subseção 6.3.3 tratará das técnicas baseadas na modelagem do estado normal do sistema, que buscam detectar anomalias a partir do desvio em relação ao comportamento esperado. Por fim, serão exploradas as diferentes estratégias de pré-processamento de dados empregadas nos estudos selecionados, uma etapa crucial para garantir a eficácia na detecção de faltas, pois a qualidade e a preparação dos dados impactam diretamente o desempenho dos modelos.

6.3.1 Técnicas Classificadoras

Nos estudos selecionados, diferentes classificadores de aprendizado de máquina foram citados e aplicados em experimentos. A Tabela 10 organiza esses algoritmos em quatro categorias principais: (1) baseados em árvores de decisão, presentes em 4 estudos; (2) baseados em *Naive Bayes*, utilizados em 3 estudos; (3) baseados em redes neurais, aplicados em 4 estudos; e (4) outros algoritmos, como SVM[1] e regressão logística[2], que apresenta 5 estudos. Os primeiros três tipos de algoritmos são detalhados nas próximas seções: os algoritmos baseados em árvores de decisão são discutidos na Seção 6.3.1.1, os baseados em *Naive Bayes* na Seção 6.3.1.2, e os baseados em redes neurais na Seção 6.3.1.3.

Tabela 10 – Algoritmos de aprendizado de máquina

Algoritmo	Estudos	Quantidade	Porcentagem aproximada
Outros	P4, P5, P7, P9, P18	5	38%
Baseados em árvores de decisão	P9, P11, P14, P20	4	30%
Baseados em *Naive Bayes*	P4, P9, P11, P13	4	30%
Redes neurais	P8, P10, P12, P19	4	30%

Fonte: Elaborada pelo autor

[1] Máquinas de Vetores de Suporte (SVM) é um modelo de aprendizado de máquina supervisionado utilizado para classificação e regressão, que busca encontrar um hiperplano ótimo para separar os dados em diferentes classes.

[2] A regressão logística é um modelo estatístico usado para prever a probabilidade de uma determinada classe ou evento.

6.3.1.1 Classificadores Baseados em Árvores de Decisão

Conforme Mienye e Jere (2024), os algoritmos de árvores de decisão operam através da partição dos dados em subgrupos, utilizando diferentes atributos até atingir um critério de parada estabelecido por meio do uso de métricas como taxa de ganho[3] ou o guia de Gini[4]. A estrutura resultante é uma árvore hierárquica, onde cada nó interno representa uma decisão com base em um atributo. Esta estratégia serviu como base para o desenvolvimento de diversos algoritmos. Além do algoritmo de árvore de decisão clássico, Mienye e Jere (2024) citam as variações *Random Forest*, que combina múltiplas árvores de decisão para melhorar a precisão; *Rotation Forest*, que aplica transformações nos dados para aumentar a diversidade das árvores; e árvores de decisão com *boosting* de gradiente, que iterativamente ajustam as árvores para corrigir erros residuais.

Nesse contexto, quatro estudos demonstraram a aplicação efetiva de algoritmos baseados em árvores de decisão. P9 introduz a tecnologia *Linnaeus*, um *pipeline* de processamento de mensagens de log e classificação de faltas que integra múltiplos algoritmos de aprendizagem de máquina, incluindo *Random Forest*. Em P11, foram conduzidos experimentos comparativos entre cinco algoritmos de classificação diferentes: *Naive Bayes* e quatro variantes baseadas em árvores de decisão (*J48*, *Random Forest*, *Random Tree* e *REPTree*). P14 realizou experimentos de detecção de faltas em sistemas de alto desempenho, avaliando três modelos baseados em árvores: árvore de decisão clássica, *Random Forest* e *Adaptive Boosting* (*AdaBoost*). Por fim, P20 também realizou experimentos de detecção de faltas usando o algoritmo *Random Forest*.

Os resultados obtidos destacam a eficácia das abordagens baseadas em árvores de decisão. P11 demonstrou o poder das abordagens baseadas em árvores de decisão ao mostrar os ótimos resultados atingidos em termos de acurácia e desempenho desses algoritmos nos experimentos, superando o *Naive Bayes*. P14 cita Tuncer *et al.* (2017), que evidenciou o melhor desempenho desses modelos na detecção de anomalias em sistemas de alto desempenho quando comparados a *SVM* e *KNN*.

As seguintes vantagens dos algoritmos baseados em árvores de decisão são destacados em P14:

- Capacidade natural de classificação multiclasse

[3] A taxa de ganho mede a redução da entropia ao dividir os dados com base em um atributo específico, ajudando a escolher o melhor atributo para a divisão.

[4] O índice de Gini avalia a impureza de um conjunto de dados, indicando a probabilidade de uma amostra ser classificada incorretamente se for rotulada aleatoriamente. Quanto menor o índice, mais homogêneo é o grupo.

- Adaptabilidade ao comportamento das combinações entre *features*
- Alta explicabilidade dos modelos gerados e suas predições
- Auxílio na redução da dimensionalidade dos dados através da seleção de features relevantes

Uma vantagem adicional, não mencionada nos estudos selecionados, mas destacada por Vázquez-Novoa *et al.* (2023), é o potencial de paralelização desses algoritmos, especialmente o *Random Forest*. Esta característica é particularmente relevante para sistemas que necessitam processar grandes volumes de dados.

6.3.1.2 Classificadores Baseados em Naive Bayes

Segundo Kaviani e Dhotre (2017), *Naive Bayes* é um método de classificação popular e subconjunto de teoria de decisão Bayesiana. O classificador assume que os atributos são independentes entre si e realiza a predição a partir do conhecimento da probabilidade a priori das classes (com que frequência cada classe aparece) e as probabilidades condicionais (quão provável é cada valor de atributo, dado uma classe). Embora simples, é uma técnica poderosa e que serviu como base para outros algoritmos, como *Naive Bayes* gaussiano e *Naive Bayes* multinomial.

O algoritmo *Naive Bayes* foi empregado em quatro estudos selecionados. Em P4, os autores propuseram uma abordagem supervisionada para detecção de anomalias baseada em regras, utilizando o *Naive Bayes* como *benchmark* para avaliação comparativa. A tecnologia *Linnaeus*, apresentada em P9, incorpora o *Naive Bayes* em seu pipeline. P11, conforme mencionado anteriormente, incluiu o *Naive Bayes* em sua análise comparativa de diferentes classificadores. De particular interesse é o estudo P13, que implementou uma versão paralelizada do *Naive Bayes* em um ambiente *Big Data*, permitindo não apenas a detecção de faltas, mas também sua classificação em múltiplas categorias.

O *Naive Bayes* destaca-se por suas características vantajosas para sistemas de detecção de faltas em larga escala. Sua simplicidade matemática e eficiência computacional o tornam especialmente adequado para processamento de grandes volumes de dados, como demonstrado em P13, que implementou com sucesso uma versão paralelizada em um ambiente *Big Data*. Uma característica fundamental do algoritmo é a suposição de independência entre *features*, que, embora possa parecer uma limitação, na prática possibilita uma implementação eficiente de paralelização. Interessantemente, mesmo quando essa suposição de independência não é completamente satisfeita, o algoritmo mantém sua eficácia. Como citado em P4, Domingos e

Pazzani (1996) demonstraram que o *Naive Bayes* apresenta bom desempenho em uma ampla variedade de condições, mesmo quando a premissa de independência não é estritamente atendida.

6.3.1.3 Classificadores Baseados em redes neurais

Uhrig (1995) define uma rede neural como vários elementos de processamento conectados para formar uma rede com funções de ponderação ajustáveis para cada entrada. Esses elementos são geralmente organizados em uma sequência de camadas, com conexões completas ou aleatórias entre elas. Normalmente, há três ou mais camadas: uma camada de entrada, onde os dados são apresentados à rede por meio de um buffer de entrada; uma camada de saída, com um buffer que armazena a resposta de saída para uma determinada entrada; e uma ou mais camadas intermediárias ou "ocultas". O uso de redes neurais é amplamente utilizado em várias aplicações devido a sua flexibilidade, incluindo detecção de anomalias, e muitas variações já foram propostas. Uma prova da flexibilidade das redes neurais é seu uso amplo em métodos supervisionados, não supervisionados e até na etapa de pré-processamento, mas nesta subseção será focado seu uso em métodos supervisionados.

Diante disso, quatro estudos aplicaram redes neurais como classificação. Em P8, mensagens de *log* são transformadas em atributos bidimensionais e tratadas como imagens, sendo então processadas por uma rede neural de uma única camada para realizar a classificação. Já em P10, os *logs* também são utilizados para detectar anomalias, mas com uma abordagem mais abrangente: redes neurais são aplicadas tanto na etapa de pré-processamento quanto na classificação, utilizando um modelo baseado em transformadores (*Transformer*) para realizar as predições. P12 realiza experimentos em um conjunto de dados industrial e usa um modelo de redes neural recorrente com memória de longo e curto prazo (LSTM) para fazer as classificações. Por fim, P19 usa uma rede neural com apenas três camadas para detectar faltas de desempenho em sistemas de *Big Data*.

É interessante observar como o uso de redes neurais varia desde as mais simples, com poucas camadas, até arquiteturas mais complexas, como as redes neurais recorrentes (RNNs) e as baseadas em transformadores. No entanto, vale ressaltar que tanto P8 quanto P19, que utilizam redes neurais mais simples, trabalham com logs pré-processados, o que contribui significativamente para a eficácia dos modelos. P10 faz referência ao trabalho clássico de Vaswani *et al.* (2017), que propôs o *Transformer*, um modelo poderoso que usa mecanismos de atenção para modelar eficientemente as relações entre os atributos. P12 menciona as redes

neurais recorrentes com memória de curto e longo prazo (LSTM), citando o estudo de Hochreiter e Schmidhuber (1997), que propuseram esse modelo como uma evolução das RNNs, visando melhorar o aprendizado de dependências temporais em dados sequenciais. Os modelos LSTM são especialmente eficazes em cenários como os do experimento descrito em P12, que utiliza séries temporais para treinamento.

6.3.2 Técnicas de Previsão de Séries Temporais

Grohmann *et al.* (2020) descreveram o funcionamento de técnicas de detecção de anomalia baseadas na previsão de séries temporais. A ideia utilizar um modelo que entende o comportamento normal do sistema a partir de análise de séries temporais para prever o estado futuro. A diferença entre o estado previsto e o estado real pode implicar na existência de uma falta.

Embora seja uma técnica poderosa, apenas um dos estudos selecionados a utilizou, P15, que realizou experimentos em dados públicos e industriais. Nele, os dados pré-processados são alimentados em uma máquina colaborativa, proposta pelos autores, que captura relações entre os artefatos monitorados tanto na dimensão dos atributos quanto na dimensão temporal. Esse mecanismo gera dois tipos de vetores: vetores de artefatos, que representam as características dos componentes monitorados, e vetores temporais, que capturam a evolução desses artefatos ao longo do tempo. A máquina colaborativa é inspirado na máquina de fatorização proposta por Rendle (2010), que é amplamente utilizada para modelar relações complexas em dados estruturados. Após a geração dos vetores, um modelo de detecção de anomalias baseado em predição, proposto por Hundman *et al.* (2018), é aplicado para realizar as predições. Esse modelo utiliza redes neurais recorrentes de memória de curto e longo prazo (LSTM), uma arquitetura de aprendizado profundo especialmente eficaz para lidar com dados sequenciais e temporais, permitindo a identificação de padrões e anomalias ao longo do tempo.

É evidente o poder e a flexibilidade das redes neurais LSTM quando aplicadas a contextos de séries temporais para detecção de anomalias. No estudo P12, as LSTMs também foram utilizadas, mas de forma supervisionada, classificando dados com base em padrões previamente conhecidos. Já em P15, elas foram empregadas de maneira não supervisionada, antecipando o estado futuro do sistema.

6.3.3 Técnicas de Modelagem do Estado Normal

Grohmann *et al.* (2020) destacam o funcionamento das técnicas de detecção de anomalias baseadas na modelagem do estado normal do sistema. A abordagem consiste em criar um modelo que represente o comportamento esperado do sistema em condições normais de operação. Em seguida, calcula-se o desvio entre o estado real observado e o estado esperado modelado, permitindo a identificação de anomalias quando esse desvio ultrapassa um determinado limite.

Dentre os estudos selecionados, sete utilizam abordagem baseada em modelagem do estado normal (P1, P2, P3, P6, P16, P17 e P18) e de maneiras bem variadas. P1 realiza *clustering* em dados de tráfego de rede representados como séries temporais, convertendo-os em uma representação bidimensional. A detecção de mudanças nesses dados, tratados como imagens, é feita por meio de uma rede neural convolucional (CNN), que classifica possíveis faltas. Em P2, os autores propõem um novo algoritmo intitulado Entropia de Permutação Multidimensional Multiescala (MMPE), que aplica uma medida de complexidade (entropia de permutação) de forma multidimensional e multiescalar em séries temporais. Em seguida, testa a abordagem com dois algoritmos de detecção de anomalias não supervisionados: *Isolated Forest* e Máquina de Vetores de Suporte de uma classe (OC-SVM). P3 propõe usar o cálculo de energia livre, conceito originado na física, para detectar faltas em sistemas nuvem. Esse cálculo é realizado por meio da rede neural estocástica Máquina de Boltzmann Restrita (RBM), que modela o estado normal do sistema. A diferença entre a energia livre modelada e a real é então utilizada para detectar anomalias. P6 propõe uma maneira de utilizar o algoritmo *KNN* de maneira não supervisionada e eficiente combinando *clustering* e divisão dos dados em blocos (*chunks*). P16 aplica a Técnica de Estimativa de Estado Multivariada (MSET), um método de detecção de anomalias, para identificar sintomas de software aging. A diferença entre o estado modelado e o estado real é calculada usando o Teste de Razão de Probabilidade Sequencial (SPRT), que determina a presença de anomalias. P17 alimenta mensagens de *logs* tokenizadas em uma LSTM para modelar o comportamento esperado do sistema. Com base nisso, calcula um *score* de anomalia para cada entrada, permitindo a detecção de desvios. P18 modela o estado normal com uma estratégia autossupervisionada chamada *Masked Language Modeling* (MLM), que oculta *tokens* dos *logs* e os transforma em alvos de predição para uma rede neural, permitindo que fossem realizados experimentos tanto de forma supervisionada quanto não supervisionada.

Diante dessa análise, fica evidente a grande diversidade de técnicas e métodos aplicáveis na modelagem do estado normal do sistema. Destaca-se, mais uma vez, o poder das

LSTMs, que, nesse contexto, foram utilizadas para detectar anomalias por meio da modelagem do comportamento esperado do sistema. Outra abordagem interessante é o uso da entropia de permutação, como proposto por P2, que se mostrou eficaz na detecção de variações dinâmicas, tanto qualitativas quanto quantitativas, em séries temporais, conforme demonstrado por Cao *et al.* (2004). Além disso, o algoritmo MSET, utilizado por P16, merece destaque por sua ampla aplicação em áreas como monitoramento de equipamentos e predição da vida útil de produtos eletrônicos. Segundo Song *et al.* (2017), o MSET opera por meio da geração de uma matriz de memória que representa o estado normal do sistema e é ferramenta pensada para a detecção de anomalias. Portanto, apesar da alta variabilidade de abordagens, é notável que técnicas consolidadas na detecção de anomalias têm sido adaptadas e aplicadas com sucesso no contexto de detecção de faltas, reforçando a versatilidade e a eficácia desses métodos.

6.4 QP4 - Quais técnicas de pré-processamento de dados podem ser utilizadas para auxiliar modelos de monitoramento de faltas baseados em mineração de dados?

Uma etapa essencial para garantir a eficácia das abordagens de detecção de anomalias citadas na questão de pesquisa QP3 é o pré-processamento dos dados. Esta fase de preparação dos dados é crucial, pois transforma os artefatos coletados em formatos mais adequados para análise, potencializando significativamente o desempenho das técnicas de detecção de anomalias. Tomar e Agarwal (2014) ressaltam a importância dessa etapa para a mineração de dados e classificam as técnicas de pré-processamento em quatro categorias principais:

- **Limpeza dos dados**. Refere-se ao tratamento de dados faltantes, inconsistentes, incompletos ou duplicados.
- **Gerenciamento de dados desbalanceados**. Consiste em balancear as classes de um conjunto de dados, uma vez que distribuições desbalanceadas podem prejudicar o desempenho de modelos ou técnicas de mineração de dados.
- **Transformação dos dados**. Envolve a aplicação de técnicas para alterar a representação dos dados, como normalização ou discretização, tornando-os mais adequados para análise.
- **Redução de dimensionalidade**. trata da redução do número de atributos (dimensões) de um conjunto de dados, geralmente por meio da remoção ou combinação de atributos, evitando perder informações relevantes.

Dentre os 20 estudos selecionados, sete utilizam *logs* como base para o monitoramento de faltas. No entanto, os *logs* geralmente não são dados estruturados de forma que possam

ser diretamente utilizados por modelos de análise sem um pré-processamento adequado. Por isso, serão exploradas técnicas de pré-processamento essenciais para transformar esse poderoso artefato em uma fonte de informação útil e eficaz para o monitoramento de faltas.

Na literatura, muitos estudos propuseram métodos para tornar os *logs* mais úteis para a realização de diversas tarefas. Liu *et al.* (2023b) realizaram uma revisão de literatura sobre métodos de pré-processamento de *logs* e os classificaram em seis tipos:

- **Filtragem**: etapa relacionada à limpeza dos dados, que consiste em remover informações problemáticas ou irrelevantes dos *logs*.
- **Transformação**: processo de alterar a representação dos *logs*, como a conversão de texto em vetores numéricos, relacionado à etapa de transformação dos dados
- **Enriquecimento**: Refere-se à adição de informações aos *logs* que não estavam originalmente presentes, como IDs, métricas ou rótulos.
- **Redução**: consiste na remoção de mensagens de log para reduzir o volume de dados, procurando manter a representatividade do conjunto
- **Integração**: consiste na combinação de conjuntos de *logs* distintos em um único conjunto unificado.
- **Abstração**: trata da transformação dos *logs* de modo que se tornem mais simples, removendo detalhes desnecessários.

Esses métodos são empregados nos sete estudos selecionados, com destaque para o uso de *log parsers*, ferramentas especializadas em transformar *logs* em formatos estruturados e adequados para análise.

O estudo P8 propõe uma abordagem sofisticada em três etapas. Primeiro, um *parser* segmenta os *logs* em *templates* e *arguments*. Os *templates* correspondem às partes fixas dos *logs*, que permanecem constantes em múltiplas instâncias de um mesmo evento, enquanto os arguments representam as partes variáveis, que podem incluir identificadores, valores numéricos, *timestamps* ou outros dados específicos da execução do sistema. Em seguida, os *logs* são convertidos em sequências de séries temporais baseadas em valores inteiros. Por fim, os dados são transformados em uma representação bidimensional semelhante a imagens, onde os atributos de cada evento são organizados temporalmente. Essa representação permite a aplicação de *spatial pooling*, uma técnica de redução de dimensionalidade que facilita a detecção de anomalias por redes neurais convolucionais.

A tecnologia *Linnaeus* proposta por P9, apresenta uma arquitetura flexível de pré-

processamento de *logs* com múltiplas etapas configuráveis. O processo inicia-se com a tokenização, que consiste na segmentação dos *logs* em unidades menores e mais manejáveis. Subsequentemente, o sistema oferece diversas opções de pré-processamento que podem ser configuradas conforme as necessidades específicas: remoção de *stop words* (palavras comuns que agregam pouco valor semântico), filtragem de palavras baseada em limites de frequência pré-estabelecidos, além da aplicação de técnicas de normalização e regularização dos dados. A normalização ajusta os dados para uma escala comum, reduzindo variações indesejadas e garantindo comparabilidade entre atributos. A normalização ajusta os dados para uma escala comum, garantindo comparabilidade entre atributos, enquanto a regularização previne o sobreajuste aos dados.

P10 apresenta uma abordagem diferenciada, dispensando o uso de *log parsing*. Em vez disso, após realizar tokenização, conversão das palavras para minúsculas e remoção de caracteres não alfanuméricos, o modelo de aprendizagem profunda BERT, um codificador pré-treinado que consegue extrair semântica dos *logs*, transforma os *logs* em representações vetoriais para serem usados na detecção de anomalias.

Os *logs* usados nos experimentos de P17 passam por tokenização e um limiar de frequência é usado para substituir *tokens* pouco frequentes com o *token* especial "fora do vocabulário". O uso desse *token* evita que os modelos treinados encontrem palavras não vistas anteriormente.

P18 implementa uma abordagem autossupervisionada inovadora no pré-processamento de *logs*. Após a tokenização inicial, aplica-se uma técnica de mascaramento, onde tokens aleatórios são ocultados e substituídos por um token especial denominado MÁSCARA. Estas sequências mascaradas são utilizadas como entrada para um modelo de rede neural de classificação, enquanto os *tokens* ocultados servem como alvos de precisão. O processo inclui ainda etapas adicionais de pré-processamento, como vetorização e codificação posicional das sequências de *tokens*. A classificação entre *templates* e variáveis é determinada pelo nível de confiança do modelo na predição dos tokens mascarados: alta confiabilidade indica que o token é parte do template, enquanto baixa confiabilidade sugere que se trata de uma variável. Esta abordagem permite que o modelo aprenda uma representação semântica geral dos dados através da análise contextual de cada token, possibilitando a diferenciação eficiente entre partes constantes e variáveis dos logs para posterior detecção de anomalias.

Portanto, fica evidente a importância crucial do pré-processamento de dados para a

eficácia das abordagens de detecção de anomalias. Com base na taxonomia proposta por Liu *et al.* (2023b), observa-se que a filtragem é uma técnica amplamente utilizada, como em P9, que remove *stop words* e filtra palavras com base em sua frequência; em P10, que elimina caracteres não alfanuméricos; e em P17, que substitui *tokens* pouco frequentes por um *token* especial. Já a transformação é a etapa mais comum, presente em praticamente todos os estudos analisados. A tokenização é uma etapa inicial essencial, utilizada de forma consistente. Além disso, destacam-se outras técnicas de transformação, como a conversão de *logs* em representações bidimensionais em P8, a vetorização semântica com o uso de codificadores baseados em redes neurais em P10, e a aplicação de mascaramento, vetorização e codificação posicional em P18. Por fim, a redução de dimensionalidade também é aplicada, como em P8, que utiliza *spatial pooling* para simplificar os dados matriciais. Essas técnicas demonstram a diversidade e a eficácia das abordagens de pré-processamento, adaptando os *logs* para serem utilizados de forma eficiente em modelos de detecção de anomalias.

Além dos métodos de pré-processamento aplicados a *logs*, diversas técnicas são utilizadas para tratar outros tipos de artefatos. P1 e P6 empregam *clustering*, um conjunto de técnicas que agrupam dados semelhantes, para segmentar o conjunto de dados antes de aplicar técnicas de detecção de anomalias. Embora o *clustering* possa ser usado diretamente para detectar anomalias, nesses estudos ele atua como uma etapa preliminar para auxiliar outros modelos. P2 utiliza *Principal Component Analysis* (PCA), uma técnica de redução de dimensionalidade que preserva a maior parte da informação original com um número menor de componentes. P3 aplica normalização em séries temporais, garantindo que os dados estejam em uma escala consistente. P4 propõe duas formas avançadas de vetorização dos dados, LVR e GVR, capazes de avaliar tanto o comportamento geral do sistema quanto faltas específicas. P7 enfrenta um desafio semelhante ao dos estudos que utilizam *logs*, pois trabalha com textos de *feedback* não estruturados. Para extrair semântica desses textos, são aplicadas diversas técnicas de processamento de linguagem natural (PLN), gerando vetores que podem ser utilizados por modelos de detecção de anomalias. P11 emprega o algoritmo *OneRAttributeEval* para selecionar de forma inteligente os atributos mais relevantes para o treinamento do modelo. P12 aplica múltiplas técnicas de pré-processamento, incluindo a substituição de dados faltantes por interpolação linear, o enriquecimento dos dados com a criação de novos atributos, a rotulação das classes e o balanceamento dessas classes usando subamostragem aleatória, técnica que remove aleatoriamente dados da classe predominante, para evitar viés no modelo. P14 descarta

os primeiros e últimos 30 segundos da inicialização e terminação do monitoramento, evitando a inclusão de dados que representem estados transitórios. Além disso, normaliza as séries temporais e utiliza o algoritmo *Independent Component Analysis* (ICA), que separa sinais misturados em componentes independentes, para extrair os atributos mais relevantes. P15 também aplica normalização aos dados. Por fim, P20 utiliza o coeficiente de correlação de Pearson, uma métrica que mede a força e a direção de uma relação linear entre duas variáveis, para selecionar os atributos mais significativos.

Assim, é possível observar a aplicação de diversas técnicas de pré-processamento, alinhadas à classificação proposta por Tomar e Agarwal (2014), nos estudos selecionados. P12 realiza limpeza dos dados ao aplicar interpolação linear para substituir valores faltantes, enquanto P14 descarta dados problemáticos coletados durante a inicialização ou terminação do sistema de monitoramento. Além disso, P12 gerencia dados desbalanceados por meio de subamostragem aleatória, garantindo uma distribuição mais equilibrada das classes. No que diz respeito à transformação dos dados, P3, P14 e P15 aplicam normalização, um método clássico que ajusta os dados para seguir escalas consistentes. Já P3 e P7 utilizam vetorização, transformando os dados em representações numéricas adequadas para análise. Quanto à redução de dimensionalidade, P2 e P14 empregam as técnicas baseadas em análise de componentes PCA e ICA, respectivamente. P11 e P20 também adotam estratégias de redução de dimensionalidade, com P11 utilizando o algoritmo *OneRAttributeEval* para seleção inteligente de atributos e P20 aplicando o coeficiente de correlação de Pearson para identificar os atributos mais relevantes. Essas técnicas evidenciam a variedade de abordagens de pré-processamento aplicadas nos estudos, adaptando-se às necessidades específicas de cada contexto e garantindo que os dados estejam preparados para a detecção eficiente de anomalias.

6.5 QP5 - Como as abordagens de monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados são avaliadas?

A avaliação das técnicas de monitoramento de faltas baseadas em mineração de dados geralmente se divide em duas categorias principais: métricas de classificação e métricas de desempenho. Ambas são essenciais para compreender a eficácia e a eficiência dos modelos, mas são aplicadas em contextos diferentes e com objetivos distintos. Na subseção 6.5.1, são caracterizadas as métricas de classificação, e na subseção 6.5.2, são discutidas as métricas de desempenho.

6.5.1 Métricas de Classificação

Como problemas de detecção de anomalias são essencialmente problemas de classificação binária (anômalo ou normal), as métricas de classificação mais comuns estão relacionadas à quantidade de acertos e erros do modelo. A Tabela 11 introduz siglas importantes para o entendimento dessas métricas e, em seguida, as métricas de classificação utilizadas nos estudos selecionados são definidas.

Tabela 11 – Classificação

Sigla	Significado
TP (True Positive)	Número de positivos corretamente classificados.
TN (True Negative)	Número de negativos corretamente classificados.
FP (False Positive)	Número de negativos incorretamente classificados como positivos.
FN (False Negative)	Número de positivos incorretamente classificados como negativos.

Fonte: Elaborada pelo autor

- **Acurácia** mede a proporção de previsões corretas em relação ao total de previsões feitas pelo modelo. É calculada da seguinte forma:

$$\text{Acurácia} = \frac{TP+TN}{TP+TN+FP+FN} \tag{6.1}$$

- **Precisão** representa a proporção de verdadeiros positivos entre todas as previsões positivas feitas pelo modelo. É calculada da seguinte forma:

$$\text{Precisão} = \frac{TP}{TP+FP} \tag{6.2}$$

- **Revocação** mede a proporção de verdadeiros positivos entre todos os casos positivos reais. É calculada da seguinte forma:

$$\text{Recall} = \frac{TP}{TP+FN} \tag{6.3}$$

- **F1-score** é a média harmônica entre precisão e recall, equilibrando essas métricas. É calculada da seguinte forma:

$$F1 = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Revocação}} \tag{6.4}$$

- **Coeficiente de Correlação de Matthews (MCC)** considera todas as categorias da matriz de confusão, sendo útil para conjuntos de dados desbalanceados. É calculada da seguinte forma:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{6.5}$$

- **Taxas de Verdadeiros Positivos (TPR), Verdadeiros Negativos (TNR), Falsos Positivos (FPR) e Falsos Negativos (FNR)** são métricas que detalham o desempenho do modelo em termos de erros e acertos.

Na Tabela 12, são apresentadas as métricas de classificação utilizadas por cada um dos estudos. Dos 20 estudos analisados, 6 empregam acurácia, 12 utilizam precisão, 12 adotam revocação, 10 fazem uso do F1-score, 1 emprega MCC, e 6 incluem métricas como FPR, TPR, FNR ou TNR. As métricas mais comuns são precisão, revocação e *F1-score*, com todos os estudos que utilizaram precisão também adotando revocação.

Tabela 12 – Métricas de classificação utilizadas

Métrica	Estudos	Quantidade	Porcentagem
Precisão	P2, P3, P4, P5, P7, P8, P10, P12, P14, P15, P18, P19	12	60%
Revocação	P2, P3, P4, P5, P7, P8, P10, P12, P14, P15, P18, P19	12	60%
F1-score	P2, P3, P4, P5, P8, P10, P14, P15, P18, P19	10	50%
Acurácia	P2, P3, P4, P5, P11, P18	6	30%
FPR, TPR, FNR ou TNR	P3, P4, P5, P8, P14, P17	6	35%
MCC	P8	1	5%

Fonte: Elaborada pelo autor

Conforme Salfner *et al.* (2010), como falhas são eventos raros, bem como faltas, a acurácia não parece ser uma métrica apropriada, visto que um modelo que classifica todos os casos como normais pode atingir uma alta acurácia devido ao desbalanceamento natural dos dados, mascarando sua ineficácia na detecção de anomalias.

Assim, nesse cenário, métricas como precisão, revocação e *F1-score* são mais informativas que acurácia. A precisão indica a confiabilidade das detecções, mostrando quantas das anomalias identificadas são realmente faltas. Já a revocação mede a capacidade do modelo de detectar as anomalias presentes, evitando que faltas passem despercebidas. O *F1-score*, por sua vez, combina precisão e revocação em uma única métrica, sendo útil para equilibrar a importância de minimizar falsos positivos (alertas incorretos) e falsos negativos (faltas não detectadas).

O Coeficiente de Correlação de Matthews (MCC) também destaca-se por sua robustez matemática em cenários com classes desbalanceadas. Diferentemente de outras métricas, o MCC considera todas as categorias da matriz de confusão (verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos) de forma balanceada.

Por fim, as taxas específicas (TPR, TNR, FPR, FNR) fornecem uma decomposição detalhada do comportamento do modelo. O TPR indica a capacidade do modelo de identificar

corretamente as anomalias presentes, enquanto o TNR reflete sua eficácia em reconhecer padrões normais. Já o FPR mede a frequência com que padrões normais são incorretamente classificados como anomalias, e o FNR revela a proporção de anomalias que passaram despercebidas.

Para calcular essas métricas, é necessário dispor de um conjunto de dados específico para o tipo de falta que será monitorada. Normalmente, os dados são divididos em dois subconjuntos principais: treino e teste. O conjunto de treino é utilizado para ajustar os parâmetros do modelo e otimizá-lo para a tarefa específica, enquanto o conjunto de teste é empregado para avaliar seu desempenho em dados não vistos.

Em modelos supervisionados, a presença de rótulos é indispensável tanto no treinamento quanto na avaliação. Já em modelos não supervisionados, embora os rótulos não sejam necessários para o treinamento, eles ainda são importantes para a validação dos resultados durante a fase de teste. Para facilitar esse processo, muitos estudos utilizam conjuntos de dados públicos de repositórios, nos quais a rotulagem já foi realizada. No entanto, em outros cenários, onde a rotulagem não está disponível, são adotadas diferentes estratégias para a geração dos rótulos. As estratégias utilizadas para permitir o uso das métricas em conjuntos de dados não rotulados nos estudos foram as seguintes:

- Os estudos P4, P14 e P19 constroem seus conjuntos de dados de teste executando o sistema em estado normal e injetando anomalias sintéticas para simular anomalias.
- Os estudo P9 e P12 coletam dados durante a execução do sistema e definem como anômalos os registros coletados no período anterior a uma falha ocorrer.
- O estudo P7 utiliza feedback dos usuários para identificar anomalias.

6.5.2 Métricas de Desempenho

Embora as métricas de classificação sejam amplamente empregadas na avaliação dos modelos nos estudos selecionados, aspectos relacionados ao desempenho computacional não foram explorados com a mesma profundidade. No entanto, métricas que avaliam o desempenho nas fases de treinamento e predição são essenciais, especialmente em aplicações cada vez mais complexas e de grande escala, para garantir que os modelos sejam viáveis em ambientes reais.

Nesse contexto, Carvalho *et al.* (2022) classificam, no contexto de testes para IoT, três subcaracterísticas de desempenho: comportamento temporal, uso de recursos e capacidade.

1. **Comportamento temporal**: refere-se ao nível em que o tempo de resposta e processamento e as taxas de transferência de um produto ou sistema, no desempenho de suas

funções, atendem aos requisitos.

2. **Utilização de recursos**: trata do grau em que as quantidades e os tipos de recursos usados por um produto ou sistema, ao desempenhar suas funções, atendem aos requisitos.
3. **Capacidade**: refere-se ao grau em que os limites máximos de um produto ou parâmetro do sistema atendem aos requisitos.

Embora essas classificações tenham sido originalmente desenvolvidas para IoT, elas também se aplicam ao contexto de monitoramento de faltas, visto que, nos estudos selecionados, foram aplicadas métricas definidas por Carvalho *et al.* (2022) relacionadas a comportamento temporal e utilização de recursos, entretanto a subcaracterística de capacidade não foi explorada. A seguir, detalham-se essas métricas e suas aplicações: na subseção 6.4.2.1, são abordadas as métricas relacionadas ao comportamento temporal, enquanto na subseção 6.4.2.2, discute-se a utilização de recursos.

6.5.2.1 Métricas de análise do comportamento temporal

Três estudos (P11, P14 e P15) avaliaram suas soluções utilizando três métricas relacionadas ao comportamento temporal: tempo de resposta, tempo de execução e *throughput*. Segundo Carvalho *et al.* (2022), o tempo de resposta refere-se ao tempo decorrido entre o envio de uma requisição e o momento em que a resposta é obtida, enquanto o tempo de execução mede o tempo necessário para completar uma única tarefa. Já o *throughput*, neste contexto, consiste na quantidade de tarefas concluídas em um determinado período, portanto é diretamente relacionado ao tempo de execução.

Diante disso, esses estudos utilizam essas métricas de maneiras variadas. P11 utiliza o *throughput* para mensurar a eficiência da detecção dos modelos de detecção de anomalias em um contexto de *Big Data*. Já P14 e P15 avaliam não apenas a eficiência de predição, mas também o desempenho do treinamento, calculando os tempos de execução dessas etapas. Além disso, P14 também calcula o tempo de resposta médio de detecção da seguinte forma: anomalias são injetadas no sistema em momentos aleatórios, e, para cada classificador, a diferença de tempo entre o surgimento da anomalia e sua detecção é registrada como latência. A partir desses valores, é calculada a média de tempo para todas as anomalias detectadas.

6.5.2.2 Métricas de análise da utilização de recursos

Somente duas métricas associadas à análise de utilização de recursos foram utilizadas nos estudos selecionados: consumo de CPU e consumo de memória. Segundo Carvalho *et al.* (2022), o uso de CPU mede a porcentagem de tempo em que a CPU está disponível para uso. Já o uso de memória refere-se a quantidade média de memória utilizada para o funcionamento de uma aplicação.

Dois estudos, P3 e P6, aplicam métricas relacionadas à utilização de recursos. P3 avalia o desempenho do detector de anomalias proposto avaliando a diferença do consumo de CPU e memória em sistemas nuvem. Já P6 utiliza somente o uso de CPU para avaliar sua abordagem em sistemas *Big Data*.

6.6 QP6 - Quais são os desafios em implementar o monitoramento de faltas utilizando métodos de detecção de anomalias baseadas em mineração de dados?

Os desafios na implementação do monitoramento de faltas permeiam diferentes etapas do processo, desde a coleta de dados até a avaliação dos modelos. Esta seção analisa quatro desafios identificados e as estratégias propostas para superá-los.

Coleta e Rotulação dos Dados

Na etapa de coleta de dados, duas abordagens principais são identificadas: a utilização de conjuntos de dados existentes em repositórios, como em P3 e P6, ou o monitoramento direto do sistema-alvo. A segunda abordagem é geralmente preferível, pois a eficácia das técnicas de detecção quando aplicadas a sistemas diferentes dos usados no treinamento ainda carece de validação adequada. Um desafio significativo é que a maioria das abordagens identificadas nos estudos é supervisionada, necessitando de dados rotulados para treinamento. Para contornar o alto custo da rotulação manual, diferentes estratégias têm sido propostas: P4 utiliza injeção de anomalias sintéticas, enquanto P9 define como anômalos os registros que precedem falhas. Embora estas alternativas reduzam custos, podem ser menos eficientes que a rotulação manual precisa.

Pré-processamento dos Dados

O pré-processamento dos dados coletados apresenta desafios específicos dependendo do modelo de detecção escolhido. Enquanto algoritmos baseados em árvores de decisão são relativamente robustos a diferentes escalas de atributos, outros, como SVMs, requerem normalização para garantir um desempenho adequado. Além disso, técnicas de redução de dimensionalidade, como PCA e seleção de atributos, podem ser necessárias para melhorar a eficiência e reduzir o ruído nos dados.

A predominância dos *logs* como fonte de dados exige técnicas específicas de pré-processamento para converter texto não estruturado em formatos adequados para análise. Isso inclui tokenização para segmentação de palavras ou símbolos, *parsing* para extração de padrões relevantes e filtragem de informações irrelevantes. Dependendo do modelo, também pode ser necessário aplicar técnicas de vetorização, como TF-IDF, para representar os *logs* de maneira eficiente. O tratamento de registros incompletos, remoção de duplicatas e agrupamento de eventos similares são etapas adicionais que podem impactar significativamente a qualidade dos resultados obtidos na detecção de falhas.

Processamento de Grandes Volumes de Dados

O desempenho em sistemas que geram grandes volumes de dados representa outro desafio significativo, como demonstrado nos estudos P6 e P13 com sistemas *Big Data*. O alto fluxo de informações exige soluções eficientes para armazenamento, processamento e análise.

Estratégias para lidar com esse desafio incluem técnicas de pré-processamento, como divisão dos dados em blocos ou *clusters* para distribuição eficiente do processamento, bem como a redução de dimensionalidade por meio de algoritmos como PCA e técnicas de seleção de atributos. Além disso, o uso de algoritmos paralelizáveis é fundamental para melhorar a escalabilidade e o tempo de resposta. Métodos como KNN e *Naive Bayes* podem ser otimizados para execução distribuída.

Seleção do Modelo de Detecção

Na seleção do modelo de detecção, é crucial considerar múltiplos requisitos além do desempenho, como interpretabilidade e controle de falsos positivos e negativos. Modelos baseados em redes neurais, embora poderosos e versáteis, geralmente oferecem menor interpre-

tabilidade que modelos baseados em árvores.

Na seleção do modelo de detecção, é crucial considerar múltiplos requisitos além do desempenho, como interpretabilidade, escalabilidade, custo computacional e controle de falsos positivos e negativos. Modelos baseados em redes neurais, embora poderosos e versáteis, geralmente oferecem menor interpretabilidade do que modelos baseados em árvores. Redes neurais profundas podem identificar padrões complexos e realizar previsões altamente precisas, mas seu funcionamento interno é frequentemente considerado uma "caixa-preta", dificultando a explicação dos resultados. Isso pode ser um problema em domínios onde a transparência é essencial, como segurança cibernética, sistemas financeiros e diagnósticos médicos. Por outro lado, modelos baseados em árvores de decisão oferecem maior interpretabilidade, permitindo visualizar a lógica por trás das decisões e identificar quais atributos mais influenciam os resultados.

Além disso, é crucial equilibrar a sensibilidade e a especificidade do modelo para reduzir falsos positivos e falsos negativos, considerando a gravidade do tipo de falta a ser detectada. Nesse contexto, técnicas de balanceamento de dados e otimização de métricas específicas, como precisão, revocação e *F1-score*, são frequentemente utilizadas para aprimorar o desempenho do modelo conforme os requisitos do sistema.

7 DISCUSSÃO

Este mapeamento sistemático evidenciou as vantagens significativas que as técnicas de monitoramento em tempo real apresentam em comparação com os métodos de testes tradicionais. A análise dos estudos selecionados permitiu identificar as etapas necessárias para implementação de técnicas de detecção de anomalias e os desafios inerentes a esse processo.

Este trabalho complementa contribuições anteriores, como as de Salfner *et al.* (2010) e Grohmann *et al.* (2020), citados na seção Trabalhos Relacionados, ao fornecer um panorama atualizado da área, considerando que muitos dos trabalhos analisados são recentes. Além disso, aborda questões essenciais para a implementação prática de soluções de monitoramento de faltas baseadas em mineração de dados, incluindo a análise dos sistemas que se beneficiam dessa estratégia, os tipos de faltas que podem ser monitoradas e as técnicas de pré-processamento fundamentais para a eficácia dessas soluções.

Entre os principais achados, que respondem diretamente às questões de pesquisa propostas, destacam-se: (1) a identificação de sistemas de grande escala e alta complexidade, como *Big Data*, nuvem e IoT, como os que mais se beneficiam dessas abordagens; (2) a categorização dos tipos de faltas, como problemas de desempenho e de Segurança da Informação; (3) a identificação dos artefatos monitorados, como logs e tráfego de rede; (4) a descrição das técnicas e algoritmos de detecção de anomalias baseadas em mineração de dados, como árvores de decisão e redes neurais; (5) a análise das técnicas de pré-processamento mais utilizadas, como tokenização e normalização; (6) a identificação de métricas de avaliação de modelos de detecção de anomalias, como precisão e tempo de execução; e (7) entendimento dos desafios na implementação de sistemas de monitoramento de faltas baseados em mineração de dados.

Entretanto, este mapeamento apresenta limitações significativas. Embora os estudos selecionados tenham fornecido informações valiosas, a pesquisa restringiu-se à base de dados Scopus e não incorporou estratégias complementares como *snowballing*, que consiste em analisar as referências dos estudos inicialmente selecionados. Além disso, um desafio adicional foi a inacessibilidade dos textos completos de 14 dos 64 estudos filtrados na primeira etapa de seleção. Outra dificuldade foi a variabilidade de definições e terminologias na literatura. Como destacado por Grohmann *et al.* (2020), a condução de uma revisão exaustiva utilizando strings de busca, conforme proposto por Kitchenham e Charters (2007), mostrou-se desafiadora devido ao volume expressivo de resultados e à diversidade de definições utilizadas.

Um exemplo dessa complexidade terminológica é o conceito de anomalias que,

segundo IEEE (2010), refere-se a irregularidades ou divergências das expectativas, mas neste trabalho foi aplicado no contexto específico de técnicas de classificação de anomalias em um âmbito além da Engenharia de Software. Para viabilizar a pesquisa com *strings* de busca, foram necessárias adaptações, como a inclusão de intervenções de sinônimos para "tempo real", mesmo que IEEE (2010) defina falta como inerentemente em tempo de execução. Essa adaptação foi necessária, pois a ausência desses termos resultava em buscas que retornavam principalmente muitos trabalhos relacionados à prevenção ou predição de faltas. Uma consequência clara dessa limitação na pesquisa foi a sub-representação de técnicas de detecção de anomalias baseadas na previsão de séries temporais.

Para avanços futuros na área, seria valioso conduzir mapeamentos ou revisões mais abrangentes, incorporando múltiplas bases de estudos e estratégias como *snowballing*, além de um refinamento mais preciso das strings de busca. Adicionalmente, outros temas merecem exploração mais aprofundada, como a viabilidade de *transfer learning*, que trata da capacidade de modelos treinados em determinados conjuntos de dados serem eficazes em outros cenários. O desenvolvimento dessa abordagem mostra-se particularmente útil por simplificar as etapas de coleta de dados e treinamento dos modelos.

8 CONCLUSÃO E TRABALHOS FUTUROS

Este mapeamento sistemático proporcionou uma visão abrangente da área de monitoramento de faltas baseado em mineração de dados. A partir de um conjunto inicial de 90 estudos identificados na base Scopus, 20 trabalhos foram selecionados por meio da leitura dos resumos e textos completos, utilizando critérios de inclusão e exclusão. O mapeamento foi conduzido seguindo o modelo proposto por Kitchenham e Charters (2007), estruturado em três fases: Planejamento, Condução e Resultados. No Planejamento, foram definidas as questões de pesquisa, a *string* de busca, os critérios de inclusão e exclusão e a base de estudos a ser feita a busca. Na fase de Condução, a *string* de busca foi executada na Scopus, resultando em 90 estudos, dos quais 20 foram selecionados após a triagem final. Na fase de Resultados, as questões de pesquisa foram respondidas.

Assim, foram identificados sistemas que se beneficiam de técnicas de monitoramento de faltas baseadas em mineração de dados, especialmente sistemas de grande escala e alta complexidade, como *Big Data*, computação distribuída, nuvem, IoT e sistemas industriais. Esses sistemas apresentam interações dinâmicas entre componentes heterogêneos, o que torna os métodos de teste tradicionais menos eficazes. Os tipos de faltas detectáveis incluem problemas de desempenho, segurança da informação e tráfego de rede, utilizando artefatos como logs, métricas de desempenho e tráfego de rede.

As técnicas de detecção de anomalias identificadas foram classificadas em três categorias principais: classificadores baseados em aprendizado de máquina, detecção por modelagem do estado normal e detecção por previsão de séries temporais. Dentre os algoritmos de aprendizado de máquina, destacam-se os baseados em árvores de decisão, *Naive Bayes* e redes neurais, amplamente utilizados na primeira categoria. Já nas técnicas de modelagem do estado normal e previsão de séries temporais, as redes neurais também se mostraram predominantes.

Quanto às métricas de avaliação, as mais comuns foram as métricas de classificação, como acurácia, precisão, *recall* e *F1-score*. No entanto, seria interessante que mais estudos incorporassem métricas de desempenho operacional, como *throughput*, uso de CPU e consumo de memória, para avaliar a eficiência computacional das técnicas propostas.

Entre os principais desafios identificados estão a dificuldade na rotulação de dados para treinamento de modelos supervisionados, a complexidade na interpretação dos modelos e a necessidade de eficiência computacional no monitoramento em tempo real.

Desse modo, a principal contribuição deste mapeamento sistemático reside no deta-

lhamento abrangente de como são implementados sistemas de monitoramento de faltas baseados em mineração de dados na prática. O estudo categorizou de forma estruturada os diferentes aspectos dessas implementações, identificando padrões entre os tipos de sistemas monitorados, as categorias de faltas detectadas, os artefatos utilizados como fonte de dados, os algoritmos de mineração empregados e os desafios relacionados. Esta categorização sistemática proporciona uma base sólida para pesquisadores e profissionais compreenderem o estado atual da arte, além de fornecer diretrizes práticas para a implementação de novos sistemas de monitoramento de faltas em diferentes domínios de aplicação.

Para trabalhos futuros, sugere-se explorar a viabilidade de *transfer learning* para simplificar a coleta de dados e o treinamento de modelos. Além disso, mapeamentos mais abrangentes, com múltiplas bases de dados e estratégias como *snowballing*, podem fornecer uma visão mais completa da área.

Assim, este trabalho contribui para uma compreensão atualizada da área de monitoramento de faltas baseadas em mineração de dados, fornecendo uma base para pesquisas futuras e o desenvolvimento de soluções mais eficazes e robustas. Ao identificar abordagens, desafios e oportunidades, este mapeamento direciona esforços futuros para aprimorar a confiabilidade e a manutenção de sistemas de software cada vez mais complexos.

REFERÊNCIAS

ALNAFESSAH, A.; CASALE, G. TRACK-Plus: Optimizing Artificial Neural Networks for Hybrid Anomaly Detection in Data Streaming Systems. v. 8, p. 146613–146626.

BHANDARI, K.; KUMAR, K.; SANGAL, A. L. Data quality issues in software fault prediction: a systematic literature review. **Artif. Intell. Rev.**, Kluwer Academic Publishers, USA, v. 56, n. 8, p. 7839–7908, dez. 2022. ISSN 0269-2821. Disponível em: <https://doi.org/10.1007/s10462-022-10371-6>.

BROWN, A.; TUOR, A.; HUTCHINSON, B.; NICHOLS, N. Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection. In: **Proceedings of the First Workshop on Machine Learning for Computing Systems**. Association for Computing Machinery. (MLCS'18), p. 1–8. ISBN 978-1-4503-5865-1. Disponível em: <https://doi.org/10.1145/3217871.3217872>.

CAO, Y.; TUNG, W.-W.; GAO, J. B.; PROTOPOPESCU, V. A.; HIVELY, L. M. Detecting dynamical changes in time series using the permutation entropy. **Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics**, v. 70, n. 4 Pt 2, p. 046217, out. 2004. ISSN 1539-3755.

CARVALHO, L.; LELLI, V.; ANDRADE, R. Performance testing guide for iot applications. In: INSTICC. **Proceedings of the 24th International Conference on Enterprise Information Systems - Volume 2: ICEIS,**. [S.l.]: SciTePress, 2022. p. 667–678. ISBN 978-989-758-569-2.

CATOVIC, A.; CARTWRIGHT, C.; GEBREYESUS, Y. T.; FERLIN, S. Linnaeus: A highly reusable and adaptable ML based log classification pipeline. In: **2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)**. [s.n.]. p. 11–18. Disponível em: <https://ieeexplore.ieee.org/document/9474393>.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 41, n. 3, jul. 2009. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/1541880.1541882>.

COTRONEO, D.; NATELLA, R.; PIETRANTUONO, R.; RUSSO, S. A survey of software aging and rejuvenation studies. **J. Emerg. Technol. Comput. Syst.**, Association for Computing Machinery, New York, NY, USA, v. 10, n. 1, jan. 2014. ISSN 1550-4832. Disponível em: <https://doi.org/10.1145/2539117>.

DOHI, T.; GOSEVA-POPSTOJANOVA, K.; TRIVEDI, K. Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule. In: **Proceedings. 2000 Pacific Rim International Symposium on Dependable Computing**. [S.l.: s.n.], 2000. p. 77–84.

DOMINGOS, P.; PAZZANI, M. Beyond independence: Conditions for the optimality of the simple bayesian classi er. In: **Proc. 13th Intl. Conf. Machine Learning**. [S.l.: s.n.], 1996. p. 105–112.

ELAHI, M.; LV, X.; NISAR, W.; KHAN, I. A.; QIAO, Y.; WANG, H. DB-Outlier Detection Algorithm Using Divide and Conquer Approach over Dynamic DataStream. In: **2008 International Conference on Computer Science and Software Engineering**. [S.l.: s.n.], 2008. v. 4, p. 438–443.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI Magazine**, v. 17, n. 3, p. 37, Mar. 1996. Disponível em: <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1230>.

FLANAGAN, K.; FALLON, E.; JACOB, P.; AWAD, A.; CONNOLLY, P. 2D2N: A Dynamic Degenerative Neural Network for Classification of Images of Live Network Data. In: **2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)**. [s.n.]. p. 1–7. ISSN 2331-9860. Disponível em: <https://ieeexplore.ieee.org/document/8651695>.

GALSTER, M.; ZDUN, U.; WEYNS, D.; RABISER, R.; ZHANG, B.; GOEDICKE, M.; PERROUIN, G. Variability and complexity in software design: Towards a research agenda. **SIGSOFT Softw. Eng. Notes**, Association for Computing Machinery, New York, NY, USA, v. 41, n. 6, p. 27–30, jan. 2017. ISSN 0163-5948. Disponível em: <https://doi.org/10.1145/3011286.3011291>.

GROHMANN, J.; HERBST, N.; CHALBANI, A.; ARIAN, Y.; PERETZ, N.; KOUNEV, S. A taxonomy of techniques for slo failure prediction in software systems. **Computers**, v. 9, n. 1, 2020. ISSN 2073-431X. Disponível em: <https://www.mdpi.com/2073-431X/9/1/10>.

GROSS, K.; BHARDWAJ, V.; BICKFORD, R. Proactive detection of software aging mechanisms in performance critical computers. In: **27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002. Proceedings.** [s.n.]. p. 17–23. Disponível em: <https://ieeexplore.ieee.org/document/1199445>.

HAND, D. J.; ADAMS, N. M. Data mining. In: ____. **Wiley StatsRef: Statistics Reference Online**. John Wiley Sons, Ltd, 2015. p. 1–7. ISBN 9781118445112. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118445112.stat06466.pub2>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, p. 1735–1780, 11 1997.

HUCH, F.; GOLAGHA, M.; PETROVSKA, A.; KRAUSS, A. Machine learning-based run-time anomaly detection in software systems: An industrial evaluation. In: **2018 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE)**. [s.n.]. p. 13–18. Disponível em: <https://ieeexplore.ieee.org/document/8368453>.

HUNDMAN, K.; CONSTANTINOU, V.; LAPORTE, C.; COLWELL, I.; SODERSTROM, T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. 02 2018.

IEEE. Ieee standard classification for software anomalies. **IEEE Std 1044-1993**, p. i–, 1994.

IEEE. Ieee standard classification for software anomalies. **IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)**, p. 1–23, 2010.

JIA, T.; LI, Y.; ZHANG, C.; XIA, W.; JIANG, J.; LIU, Y. Machine deserves better logging: A log enhancement approach for automatic fault diagnosis. In: **2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)**. [S.l.: s.n.], 2018. p. 106–111.

JIEYANG, P.; KIMMIG, A.; DONGKUN, W.; NIU, Z.; ZHI, F.; JIAHAI, W.; LIU, X.; OVTCHAROVA, J. A systematic review of data-driven approaches to fault diagnosis and early warning. **J. Intell. Manuf.**, Springer-Verlag, Berlin, Heidelberg, v. 34, n. 8, p. 3277–3304, set. 2022. ISSN 0956-5515. Disponível em: <https://doi.org/10.1007/s10845-022-02020-0>.

JONES, M.; BAYESH, M.; JAHAN, S. Unlocking Deeper Understanding: Leveraging Explainable AI for API Anomaly Detection Insights. In: **Proceedings of the 2024 16th International Conference on Machine Learning and Computing**. Association for Computing Machinery. (ICMLC '24), p. 211–217. ISBN 979-8-4007-0923-4. Disponível em: <https://dl.acm.org/doi/10.1145/3651671.3651738>.

KAVIANI, P.; DHOTRE, S. Short survey on naive bayes algorithm. **International Journal of Advance Research in Computer Science and Management**, v. 04, 11 2017.

KIM, B. uk; HARIRI, S. Anomaly-based Fault Detection System in Distributed System. In: **5th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2007)**. [S.l.: s.n.], 2007. p. 782–789.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. v. 2, 01 2007.

LE, V.-H.; ZHANG, H. Log-based Anomaly Detection Without Log Parsing. In: **2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)**. [S.l.: s.n.], 2021. p. 492–504. ISSN 2643-1572.

LIU, J.; YANG, T.; CHEN, Z.; SU, Y.; FENG, C.; YANG, Z.; LYU, M. R. Practical Anomaly Detection over Multivariate Monitoring Metrics for Online Services. In: **2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)**. [S.l.: s.n.], 2023. p. 36–45. ISSN 2332-6549.

LIU, Y.; DANI, V. S.; BEEREPOOT, I.; LU, X. Turning logs into lumber: Preprocessing tasks in process mining. **ArXiv**, abs/2309.17100, 2023. Disponível em: <https://api.semanticscholar.org/CorpusID:263310442>.

MIENYE, I. D.; JERE, N. R. A survey of decision trees: Concepts, algorithms, and applications. **IEEE Access**, v. 12, p. 86716–86727, 2024. Disponível em: <https://api.semanticscholar.org/CorpusID:270649029>.

MONNI, C.; PEZZè, M.; PRISCO, G. An RBM Anomaly Detector for the Cloud. In: **2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)**. [s.n.]. p. 148–159. ISSN 2159-4848. Disponível em: <https://ieeexplore.ieee.org/document/8730157>.

NEDELKOSKI, S.; BOGATINOVSKI, J.; ACKER, A.; CARDOSO, J.; KAO, O. Self-supervised Log Parsing. In: DONG, Y.; MLADENIć, D.; SAUNDERS, C. (Ed.). **Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track**. [S.l.]: Springer International Publishing. p. 122–138. ISBN 978-3-030-67667-4.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. **Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering**, v. 17, 06 2008.

RENDLE, S. Factorization machines. In: **2010 IEEE International Conference on Data Mining**. [S.l.: s.n.], 2010. p. 995–1000.

SALFNER, F.; LENK, M.; MALEK, M. A survey of online failure prediction methods. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 42, n. 3, mar. 2010. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/1670679.1670680>.

SHAFIQ, M. O.; FEKRI, M.; IBRAHIM, R. MapReduce Based Classification for Fault Detection in Big Data Applications. In: **2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)**. [s.n.]. p. 637–642. Disponível em: <https://ieeexplore.ieee.org/document/8260703>.

SHAHAMIRI, S. R.; KADIR, W. M. N. W.; MOHD-HASHIM, S. Z. A comparative study on automated software test oracle methods. In: **2009 Fourth International Conference on Software Engineering Advances**. [S.l.: s.n.], 2009. p. 140–145.

SHIVAKUMAR, S. K. 2 - ensuring high availability for your enterprise web applications. In: SHIVAKUMAR, S. K. (Ed.). **Architecting High Performing, Scalable and Available Enterprise Web Applications**. Morgan Kaufmann, 2015. p. 59–99. ISBN 978-0-12-802258-0. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128022580000020>.

SONG, L.; GUO, L.; WANG, H.; YANG, S.; JIN, S.; DUAN, J.; XU, L. Anomaly detection method of space payload using multivariate state estimation technique and self-organizing feature map. In: **2017 International Conference on Dependable Systems and Their Applications (DSA)**. [S.l.: s.n.], 2017. p. 103–109.

TOMAR, D.; AGARWAL, S. A survey on pre-processing and post-processing techniques in data mining. **International journal of database theory and application**, v. 7, p. 99–128, 2014. Disponível em: <https://api.semanticscholar.org/CorpusID:1099754>.

TUNCER, O.; ATES, E.; ZHANG, Y.; TURK, A.; BRANDT, J.; LEUNG, V. J.; EGELE, M.; COSKUN, A. K. Online Diagnosis of Performance Variation in HPC Systems Using Machine Learning. v. 30, n. 4, p. 883–896. ISSN 1558-2183. Disponível em: <https://ieeexplore.ieee.org/document/8466019>.

TUNCER, O.; ATES, E.; ZHANG, Y.; TURK, A.; BRANDT, J.; LEUNG, V.; EGELE, M.; COşKUN, K. Diagnosing performance variations in hpc applications using machine learning. In: . [S.l.: s.n.], 2017. p. 355–373. ISBN 978-3-319-58666-3.

UCHIDA, H.; TOMINAGA, K.; ITAI, H.; LI, Y.; NAKATOH, Y. Improving log anomaly detection via spatial pooling: Combining SPClassifier with ensemble method. **Cognitive Robotics**, v. 4, p. 217–227, jan. 2024. ISSN 2667-2413.

UHRIG, R. Introduction to artificial neural networks. In: **Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics**. [S.l.: s.n.], 1995. v. 1, p. 33–37 vol.1.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L. u.; POLOSUKHIN, I. Attention is all you need. In: GUYON, I.; LUXBURG, U. V.; BENGIO, S.; WALLACH, H.; FERGUS, R.; VISHWANATHAN, S.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2017. v. 30. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.

VÁZQUEZ-NOVOA, F.; CONEJERO, J.; TATU, C.; BADIA, R. M. Scalable random forest with data-parallel computing. In: **European Conference on Parallel Processing**. [s.n.], 2023. Disponível em: <https://api.semanticscholar.org/CorpusID:261174907>.

WANG, S.; LU, M.; KONG, S.; AI, J. A dynamic anomaly detection approach based on permutation entropy for predicting aging-related failures. v. 22, n. 11, p. 1–18.

WU, D.; SHEN, B.; CHEN, Y.; JIANG, H.; QIAO, L. Automatically repairing tensor shape faults in deep learning programs. **Information and Software Technology**, v. 151, p. 107027, nov. 2022. ISSN 09505849.

YAHYAOUI, A.; LAKHDHAR, H.; ABDELLATIF, T.; ATTIA, R. Machine learning based network intrusion detection for data streaming IoT applications. In: **2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)**. [s.n.]. p. 51–56. Disponível em: <https://ieeexplore.ieee.org/document/9403457>.

ZHENG, W.; LU, H.; ZHOU, Y.; LIANG, J.; ZHENG, H.; DENG, Y. iFeedback: Exploiting User Feedback for Real-Time Issue Detection in Large-Scale Online Service Systems. In: **2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)**. [s.n.]. p. 352–363. ISSN 2643-1572. Disponível em: <https://ieeexplore.ieee.org/document/8952229>.

APÊNDICE A – UM MAPEAMENTO SISTEMÁTICO SOBRE MONITORAMENTO DE FALTAS UTILIZANDO TÉCNICAS DE DETECÇÃO DE ANOMALIA BASEADAS EM MINERAÇÃO DE DADOS

Paulo Ricardo Fernandes Rodrigues; Valéria L.L Dantas Universidade Federal do Ceará, Fortaleza, Brasil *valerialelli@ufc.br; paulorfrc@gmail.com*

Protocolo de Mapeamento Sistemático

Objetivo

Investigar e analisar abordagens de monitoramento para detecção de anomalias baseadas em técnicas de mineração de dados, mapeando e categorizando os métodos utilizados, domínios de aplicação, vantagens, limitações e lacunas presentes.

Questões de Pesquisa

- **QP1**: Quais sistemas de software se beneficiam do monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados?
- **QP2**: Quais tipos de faltas podem ser detectadas por meio do monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados, e quais artefatos do sistema são empregados nesse processo?
- **QP3**: Quais técnicas e algoritmos de detecção de anomalias baseados em mineração de dados, bem como métodos de pré-processamento de dados, podem ser utilizadas para monitorar faltas?
- **QP4**: Quais técnicas de pré-processamento de dados podem ser utilizadas para auxiliar modelos de monitoramento de faltas baseados em mineração de dados?
- **QP5**: Como as abordagens de monitoramento de faltas utilizando técnicas de detecção de anomalias baseadas em mineração de dados são avaliadas?
- **QP6**: Quais são os desafios em implementar o monitoramento de faltas utilizando métodos de detecção de anomalias baseadas em mineração de dados?

Palavras-chaves

Detecção de faltas; Monitoramento de faltas; Detecção de anomalias; Mineração de dados; Aprendizado de máquina; Software aging; Big Data.

Critérios PICOC

- População: Técnicas de monitoramento de faltas em tempo real
- Intervenção: Mineração de dados
- Comparação: Não definido
- Resultados: Métricas, artefatos e abordagens
- Contexto: De software

Seleção das Fontes

Base de artigos Scopus.

String de Busca

((("fault detection"OR "fault verification"OR "fault diagnosis"OR "fault identification"OR "fault localization"OR "fault monitoring")

OR

("anomaly detection"OR "anomaly verification"OR "anomaly diagnosis"OR "anomaly identification"OR "anomaly localization"OR "anomaly monitoring"))

AND

("runtime"OR "run-time"OR "realtime"OR "real time"OR "real-time"OR "dynamic"OR "online"OR "on-line"OR "execution time"OR "operational phase")

AND

("software engineering"OR "software system"OR "software application"OR "software development"OR "software process")

AND

("data mining"OR "machine learning"OR "pattern recognition"OR "statistical learning"OR "statistical analysis"OR "data analysis"))

Seleção dos Estudos

Critérios de Inclusão

1. Estudo deve apresentar abordagem de detecção de faltas utilizando mineração de dados.
2. Estudo deve apresentar abordagem de detecção de faltas que funcione em tempo de execução.
3. Estudo deve estar acessível em texto completo.
4. Estudo deve estar escrito em português ou inglês.
5. Estudo deve estar dentro do contexto de engenharia de software.

Critérios de Exclusão

1. Estudo apresenta abordagem focada no monitoramento de faltas de hardware.
2. Estudo apresenta abordagem que utiliza dados de sensores ou outro tipo de hardware para monitoramento.
3. Estudo apresenta abordagem que utiliza oráculo de testes.