UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE

DEPARTAMENTO DE COMPUTAÇÃO

MAYARA DA SILVA OLIVEIRA

UMA REVISÃO SISTEMÁTICA DE LITERATURA DA APLICAÇÃO DE APRENDIZADO DE MÁQUINA EM TESTES DE SOFTWARE

ALEGRE

2023

MAYARA DA SILVA OLIVEIRA

UMA REVISÃO SISTEMÁTICA DE LITERATURA DA APLICAÇÃO DE APRENDIZADO DE MÁQUINA EM TESTES DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao curso de Ciência da Computação da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.
Orientador: Prof. Dr. Clayton Vieira Fraga Filho.

ALEGRE

2023

MAYARA DA SILVA OLIVEIRA

UMA REVISÃO SISTEMÁTICA DE LITERATURA DA APLICAÇÃO DE APRENDIZADO DE MÁQUINA EM TESTES DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 20 de julho de 2023.

COMISSÃO EXAMINADORA

Prof. Dr. Clayton Vieira Fraga Filho
Universidade Federal do Espírito Santo
Orientador

Prof. M.Sc. Giuliano Prado de Morais Giglio
Universidade Federal do Espírito Santo

Jacson R. C. Silva

Prof. Dr. Jacson Rodrigues Correia da Silva
Universidade Federal do Espírito Santo

RESUMO

Recentemente, os trabalhos que envolvem Aprendizado de Máquina (AM) e testes de software estão se intensificando. Nestes estudos, há propostas de utilizações de métodos e técnicas de AM buscando melhorias nas atividades de testes. Dessa forma, o objetivo deste estudo foi identificar na literatura trabalhos que propõem a utilização de métodos e técnicas de AM a testes de software. Assim, esta pesquisa foi desenvolvida como uma revisão sistemática de literatura com a seguinte sequência de etapas: contextualização, busca, identificação dos estudos, seleção, extração e síntese dos dados. Com estas etapas, 17 trabalhos foram selecionados, analisados e categorizados em relação às questões de pesquisa. Diante disso, foi possível apresentar quais atividades, níveis, tipos e técnicas de testes de software estão envolvidos e quais vantagens e desvantagens são apontadas pelos autores dos estudos analisados. Assim, foi possível concluir que as pesquisas sobre aprendizado de máquina aplicado a testes de software têm demonstrado resultados promissores, proporcionando benefícios significativos e contribuindo para a qualidade e eficiência dos processos de teste. No entanto, ainda há espaço para aprimoramentos e a realização de mais estudos detalhados na área.

Palavras-Chave: Testes de software; Aprendizado de Máquina; Engenharia de Software; Inteligência Artificial.

ABSTRACT

Recently, the number of research works involving Machine Learning (ML) and software testing has been increasing. In these studies, there are proposals for the use of ML methods and techniques to enhance testing activities. Thus, the aim of this study was to identify literature works that propose the utilization of ML methods and techniques for software testing. Accordingly, this research was conducted as a systematic literature review with the following steps: contextualization, search, study identification, selection, data extraction, and data synthesis. Through these steps, 17 works were selected, analyzed, and categorized in relation to the research questions. Consequently, it was possible to present the software testing activities, levels, types, and techniques involved, as well as the advantages and disadvantages pointed out by the authors of the analyzed studies. Thus, it was possible to conclude that research on machine learning applied to software testing has shown promising results, providing significant benefits, and contributing to the quality and efficiency of testing processes. However, there is still room for improvement and further detailed studies in the field.

Keywords: Software Testing; Machine Learning; Software Engineering; Artificial Intelligence.

LISTA DE FIGURAS

Figura 1: Principais atividades de testes de software.....14
Figura 2: Pirâmide de testes.....17
Figura 3: Diagrama de Venn Inteligência Artificial, Aprendizado de Máquina e Ciência de Dados.....20
Figura 4: Tipos de Aprendizado de Máquina.....22
Figura 5: Fluxograma de identificação dos estudos. 30
Figura 6: Tabela de caracterização dos estudos incluídos.....39

LISTA DE GRÁFICOS

Gráfico 1: Proporção de estudos retornados, selecionados e incluídos. 32
Gráfico 2: Relação de estudos excluídos por critérios de exclusão, base IEEE Xplore. 33
Gráfico 3: Relação de estudos excluídos por critérios de exclusão, base Taylor & Francis. 34
Gráfico 4: Relação de estudos excluídos por critérios de exclusão, base Wiley Online. 35
Gráfico 5: Relação de estudos excluídos por critérios de exclusão, base World Scientific. 36
Gráfico 6: Relação de estudos excluídos por critérios de exclusão, em todas as bases. 37
Gráfico 7: Publicações por ano, considerando os estudos incluídos. 40
Gráfico 8: Publicações por base de dados, considerando os estudos incluídos. 41
Gráfico 9: Publicações por delineamento, considerando os estudos incluídos. 43
Gráfico 10: Publicações por país, considerando os estudos incluídos. 44

LISTA DE TABELAS

Tabela 1: String de busca com operadores booleanos. 27
Tabela 2: Resultados das pesquisas com a expressão de busca em todas as bases. 28
Tabela 3: Resultado da busca em todas as bases com os termos “Inteligência Artificial” e “artificial intelligence”. 28
Tabela 4: Resultados de busca e seleção dos estudos. 29
Tabela 5: Relação entre motivos de exclusão, número de trabalhos e percentual representado da base World Scientific. 49
Tabela 6: Relação entre motivos de exclusão, número de trabalhos e percentual representado da base Wiley Online. 50
Tabela 7: Relação entre motivos de exclusão, número de trabalhos e percentual representado da base Taylor & Francis. 51
Tabela 8: Relação entre motivos de exclusão, número de trabalhos e percentual representado da base IEEE Xplore. 52

SUMÁRIO

RESUMO....0
ABSTRACT....1
LISTA DE FIGURAS....2
LISTA DE GRÁFICOS....3
LISTA DE TABELAS....4
1. INTRODUÇÃO....10
1.1. O Problema e Sua Importância....11
1.2. Objetivos....11
1.2.1. Objetivo Geral....11
1.2.2. Objetivos Específicos....11
2. REFERENCIAL TEÓRICO....13
2.1. Testes de Software....13
2.1.1. Atividades de Testes de Software....14
2.1.2. Níveis de Teste de Software....16
2.1.3. Tipos de Teste de Software....18
2.1.4. Técnicas de Teste de Software....19
2.2. Aprendizado de Máquina....20
2.2.1. Algoritmos de Aprendizado de Máquina: Aplicações e Desafios....22
2.2.2. Aprendizado de Máquina e Testes de Software....23
3. METODOLOGIA....24
3.1. Contextualização....24
3.2. Questões de Pesquisa....24
3.3. Estratégias de Busca e Identificação dos Trabalhos....24
3.4. Seleção dos Trabalhos....25
3.4.1. Critérios de Inclusão....25
3.4.2. Critérios de Exclusão....26
3.5. Extração e Síntese dos Dados....26
4. RESULTADOS E DISCUSSÃO....27
4.1. Pesquisa dos Trabalhos....27
4.2. Seleção dos Trabalhos....29
4.2.1. Processo de seleção dos Estudos....29
4.3. Questões de Pesquisa....38
4.4. Análise dos Trabalhos Excluídos....48

5. CONCLUSÃO 54
5.1. Trabalhos Futuros 55
APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa 61
APÊNDICE B – Estudos da base Taylor & Francis excluídos dessa pesquisa 73
APÊNDICE C – Estudos da base Wiley Online excluídos dessa pesquisa 76
APÊNDICE D – Estudos da base World Scientific excluídos dessa pesquisa 83

1. INTRODUÇÃO

O surgimento dos problemas relacionados à qualidade de software se deu com o desenvolvimento do primeiro grande sistema, assim, passaram a ser adotadas técnicas formais de gerenciamento de software (SOMMERVILLE, 2011). A partir daí, essas técnicas em conjunto a novas tecnologias e melhores testes de software, resultaram em melhorias na garantia da qualidade de software (SOMMERVILLE, 2011). Neste contexto, o teste consiste em um conjunto de atividades para avaliar a qualidade de um software, identificar erros cometidos em seu projeto e construção, e reduzir falhas durante sua execução (PRESSMAN; MAXIM, 2016). Ainda, a área de testes é delimitada como uma das áreas de conhecimento definidas no Software *Engineering Body of Knowledge* (SWEBOK), e essa, junto às demais áreas de conhecimento, estão diretamente relacionadas à qualidade (IEEE COMPUTER SOCIETY, 2014).

Há diversas razões para um software apresentar falhas, mas a principal causa é a falta de testes adequados (UDDIN; ANAND, 2019). Assim, testes são essenciais ao desenvolver um software, dado que a má execução ou a falta dessas atividades podem ser perigosas e causar prejuízos (UDDIN; ANAND, 2019). Porém, existem diversas maneiras de testar um software, podendo variar de acordo com os objetivos, contexto, técnicas e metodologia de desenvolvimento, tornando o processo de testar um software um desafio (BERTOLINO, 2007). Nesta perspectiva, Belay (2020) demonstrou que ferramentas podem ser projetadas para apoiar as atividades de teste e há a possibilidade de se utilizar outras habilidades e técnicas, como por exemplo o Aprendizado de Máquina.

Segundo Monard e Baranaukas (2003), o Aprendizado de Máquina é uma área de Inteligência Artificial que tem como objetivo o desenvolvimento de técnicas computacionais sobre o aprendizado e a construção de sistemas capazes de adquirir conhecimento automaticamente. Com isso, existem diversos algoritmos de Aprendizado de Máquina, porém, é necessário entender o potencial e as limitações destes, de acordo com os problemas nos quais são aplicados (MONARD; BARANAUSKAS, 2003).

Ainda, Belay (2020) apontou em seus resultados, que em sistemas de grande escala há o desafio de se ter artefatos desatualizados, assim, propôs o uso de Aprendizado de Máquina a fim de simplificar os testes e aumentar a produtividade.

1.1. O Problema e Sua Importância

Segundo estudo realizado por Durelli *et al.* (2019), apesar desta ser uma área de pesquisa consideravelmente nova, há publicações realizadas desde a década de 80. A partir daí, há ao menos um estudo relacionado à aplicação de Aprendizado de Máquina em testes de software ao ano, se tornando ainda mais relevante a partir de 2010 (DURELLI *et al.,* 2019). Sendo assim, o presente trabalho busca trazer um panorama atualizado sobre tal tema, acerca dos últimos 5 anos.

Além disso, de acordo com Kitchenham e Charters (2007), uma revisão sistemática traz diferentes observações e resultados devido aos critérios, estudos selecionados e metodologias diferentes. Com isso, profissionais e pesquisadores dessa abrangente área poderão ter acesso a um compilado de informações de diferentes estudos e contextos (KITCHENHAM; CHARTERS, 2007). Nesse sentido, este trabalho se apresenta potencialmente relevante, visto que a aplicação de Aprendizado de Máquina a testes de software tem se intensificado.

1.2. Objetivos

1.2.1. Objetivo Geral

Este trabalho de conclusão de curso teve como objetivo identificar na literatura estudos que propõem a utilização de métodos e técnicas de Aprendizado de Máquina a testes de software, analisar a frequência das pesquisas e os processos envolvidos, bem como fornecer uma visão geral das pesquisas realizadas, pontuando também vantagens e desafios apontados pelos autores dos estudos.

1.2.2. Objetivos Específicos

Foram elencados os seguintes objetivos específicos deste trabalho:

- Sintetizar um referencial teórico com os estudos realizados sobre a aplicação de Aprendizado de Máquina em testes de software.

- Identificar os resultados e as propostas de melhorias nas atividades de testes de software dos trabalhos selecionados.
- Apoiar e direcionar a geração de novos estudos.

2. REFERENCIAL TEÓRICO

2.1. Testes de Software

Testes de software consistem no conjunto de atividades a fim de avaliar um sistema ou um componente do mesmo, de forma manual ou automatizada, para verificar se este atende aos requisitos especificados e identificar diferenças entre resultados esperados e o resultado que foi obtido (IEEE COMPUTER SOCIETY, 1983). Com isso, há mais de 30 anos atrás, em um estudo feito por GELPERIN; HETZEL (1988), a importância dos testes de software já era ressaltada e notava-se o aumento dos esforços e ganhos significativos na implementação de métodos de teste para a melhoria dos softwares.

Assim, as atividades de teste estão entre as etapas mais críticas do ciclo de vida do desenvolvimento de software, são custosas e intensivas, mas cruciais, sendo necessárias técnicas aprimoradas e metodologias inovadoras em tais atividades (JAMIL *et al.* 2016). Além disso, estas atividades podem produzir um importante capital intelectual e, com isso, novos projetos podem se beneficiar da experiência adquirida com projetos anteriores (ANDRADE *et al.*, 2013; MACIEL; SOUZA; VIJAYKUMAR, 2018).

Com o objetivo de elucidar, a International Organization for Standardization (ISO) e o Institute of Electrical and Electronics Engineers (IEEE) estabelecem normas para definir e diferenciar termos usados na área de testes de software, incluindo erro, *bug*, defeito e falha IEEE 610 (1990).

Segundo o padrão IEEE 610 (1990), erro é definido como uma ação humana que produz um resultado incorreto. Trata-se da primeira etapa na geração de um defeito: um desenvolvedor comete um erro que acaba por introduzir um defeito no código.

Por sua vez, o termo *bug* é frequentemente usado de maneira informal para se referir a um defeito no software. No entanto, *bug* não é um termo formalmente definido pelo IEEE ou ISO em suas normas de testes de software.

O termo defeito, conforme definido pela norma ISO/IEC 24765 (2010), é um passo além do erro. Trata-se de um problema no sistema que pode levar a uma falha.

Um defeito é uma imperfeição ou deficiência no software que causa um desvio do comportamento esperado.

Por fim, o termo falha é definido pelo IEEE como a manifestação de um erro. Em outras palavras, uma falha é o comportamento observável e anormal do software devido a um ou mais defeitos (IEEE 610, 1990).

2.1.1. Atividades de Testes de Software

As atividades de teste de software há muito são categorizadas, e a categorização mais utilizada é baseada nas etapas tradicionais do processo de software (AMMANN; OFFUTT, 2016). Sendo assim, segundo Pfleeger (2004), o processo de teste envolve as quatro seguintes atividades principais: planejamento de testes, projeto de casos de testes, execução dos testes e avaliação dos resultados. Porém, o processo de teste de software pode ser adaptado de acordo com as necessidades do projeto ou do cliente a fim de aumentar a garantia da qualidade do produto (BIRAN; COTTON, 2017).

Ainda assim, é importante que independentemente da fase em que o projeto se encontra, algumas etapas devem ser bem definidas para a execução das atividades de teste (DELAMARO, 2016).

Figura 1: Principais atividades de testes de software.

Fonte: Própria autora.

Apesar de existirem testes que só podem ser realizados após a implantação de alguma parte do sistema, outros podem ser antecipados, projetados e construídos durante todas as etapas de desenvolvimento do software. Além disso, as partes mais demoradas do teste são, na verdade, o projeto e a construção do teste, assim, as atividades de teste podem e devem ser realizadas durante todo o desenvolvimento (AMMANN; OFFUTT, 2016).

O planejamento de teste geralmente é a primeira fase, onde é feita a organização de todas as atividades que devem ser realizadas em todo o processo. Na sequência, o projeto de teste, onde são desenvolvidos os casos de teste que serão utilizados (JAMIL *et al.* 2016; EVERETT; MCLEOD JR, 2007). A execução do teste é a fase do ciclo que engloba a execução dos casos, e as falhas serão identificadas e relatadas (JAMIL *et al.* 2016; EVERETT; MCLEOD JR, 2007).

A execução dos testes de software pode ser de dois tipos: manual e automatizado. No teste manual, resumidamente um testador executa o programa com alguns dados de teste e compara os resultados com o que é esperado. Já nos testes automatizados, os testes são codificados em um programa que é executado no sistema em desenvolvimento. Este segundo tipo é geralmente mais ágil que o teste manual, principalmente em testes de regressão onde ocorre a reexecução de testes anteriores para verificar se as alterações não introduziram novos defeitos (SOMMERVILLE, 2011).

Na análise dos testes, também conhecida como avaliação, é a última etapa do processo, na qual a análise do defeito é feita pelo desenvolvedor, podendo ser feita junto com o cliente (JAMIL *et al.* 2016; EVERETT; MCLEOD JR, 2007).

O software pode conter uma grande quantidade de casos de teste, e a execução de todos esses testes pode aumentar o custo do processo de teste, assim, existem ao menos três estratégias de teste de regressão que visam reduzir esse trabalho: seleção do teste de regressão, minimização do conjunto de testes e priorização dos casos de teste (NAHEED; NADEEM; ZAMAN, 2022). Assim, tantas atividades e *testwares* envolvidos tornam o teste de software um processo intensivo em conhecimento e se torna útil fornecer suporte automatizado para tarefas de aquisição, processamento, análise e disseminação de conhecimento para reutilização (ANDRADE *et al.*, 2013; SOUZA; FALBO; VIJAYKUMAR, 2015).

Ainda, sempre que ocorre uma modificação em um software, é necessário ao final do ciclo de desenvolvimento realizar uma verificação adicional para assegurar que nenhuma nova falha surgiu no código que já havia sido testado anteriormente, a fim de garantir que ele continua funcionando de forma eficaz. (NAHEED; NADEEM; ZAMAN, 2022).

Diante disso, o uso de testes automatizados tem aumentado consideravelmente nos últimos anos (SOMMERVILLE, 2011; DURELLI *et al.*, 2019). Porém, não podem ser realizados somente este tipo de teste, uma vez que testes automatizados costumam ser limitados e só podem verificar se um programa faz aquilo a que é proposto (SOMMERVILLE, 2011).

2.1.2. Níveis de Teste de Software

O enfoque inicial da realização de testes deve ser nas menores partes de um software, este é comumente chamado de teste de unidade. Porém, existem outros níveis de testes como o teste de integração, realizado conforme o sistema é construído, teste de validação ou de sistema, e testes de aceitação (PRESSMAN; MAXIM, 2016). Assim, cada nível diferente de teste pode acompanhar cada etapa de desenvolvimento e artefatos envolvidos, ou seja, testes podem ser elaborados a partir de requisitos e especificações, artefatos de design ou código-fonte. (AMMANN; OFFUTT, 2016).

Segundo a IEEE, (2010), testes de unidade são testes de rotinas e módulos individuais pelo desenvolvedor ou por um testador independente. Ou também, um teste de programas ou módulos individuais para garantir que não haja erros de análise ou programação. Os desenvolvedores realizam testes de unidade em partes individuais do código-fonte designadas e utilizam dados dos testes diferentes dos dados de testes da equipe de garantia de qualidade (SINGH; SINGH 2012). Os testes de unidade convencionais também contêm oráculos que são representados como asserções que comparam o comportamento observado com os resultados esperados após a execução do teste (FELBINGER; WOTAWA; NICA, 2018).

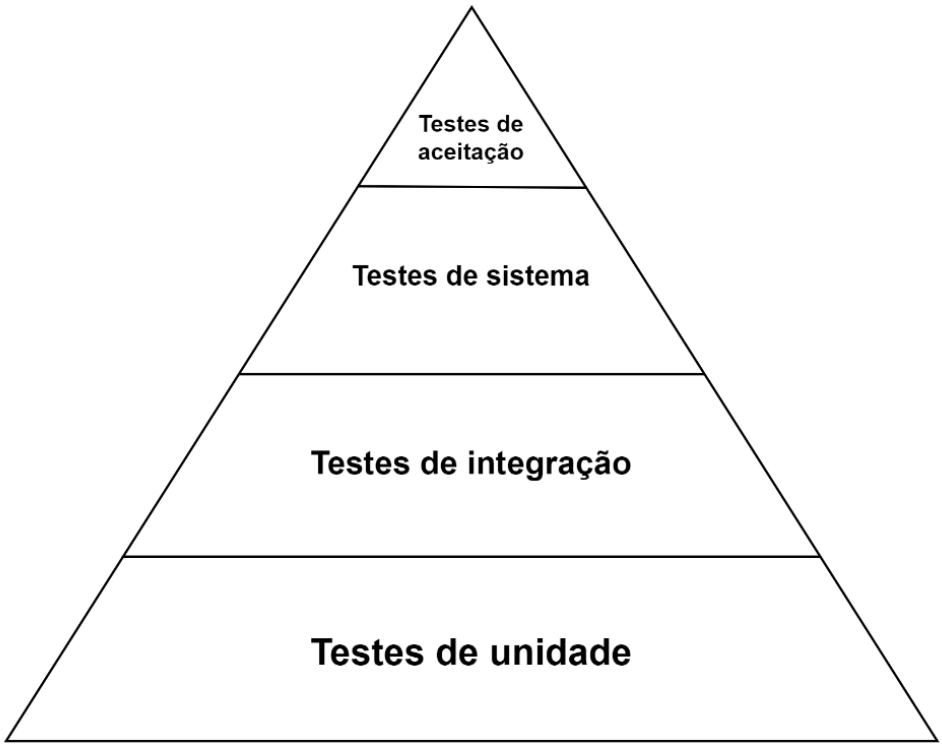
Uma vez que os módulos de um único sistema de software foram desenvolvidos de forma independente, eles são integrados e, muitas vezes, surgem erros na construção, uma vez que a integração foi feita (JAMIL *et al.* 2016). Assim, teste de

integração pode ser definido como a vinculação progressiva de teste de programas ou módulos para garantir o seu funcionamento adequado no sistema completo, ou seja, testes nos quais os componentes de software são combinados e testados para avaliar a interação entre eles (IEEE, 2010).

Já os testes de sistemas são testes conduzidos em um sistema completo e integrado para avaliar a conformidade do sistema com seus requisitos especificados (IEEE, 2010). O teste de sistema garante que as unidades integradas não interfiram ou perturbem a programação de qualquer outro módulo (JAMIL *et al.* 2016). No entanto, o teste de sistemas grandes ou intensamente complexos pode ser um procedimento extremamente demorado, pois quanto mais componentes houver no aplicativo, mais difícil será testar cada combinação (JAMIL *et al.* 2016).

Por fim, de acordo com IEEE, 2010, o teste de aceitação é o teste de um sistema ou unidade funcional, geralmente realizado pelo cliente em suas instalações com a participação do fornecedor do software para garantir que os requisitos contratuais e critérios de aceitação estabelecidos sejam atendidos. Sendo assim, teste de aceitação normalmente se refere à realização de um teste para determinar se o requisito especificado do produto ou software foi atendido ou não (JAMIL *et al.* 2016).

Figura 2: Pirâmide de testes.



Fonte: Própria autora, adaptado de HAEUPTLE, 2021.

2.1.3. Tipos de Teste de Software

Os testes funcionais, também nomeados como testes de caixa-preta, têm como fundamento os requisitos do sistema. O teste de caixa preta é uma técnica de teste que testa essencialmente a funcionalidade do aplicativo sem entrar em detalhes no nível de implementação. Já os testes estruturais, de caixa branca, têm como foco a estrutura interna ou a implementação do sistema, isto é, os caminhos lógicos do sistema e as integrações entre os componentes. O teste de caixa branca é significativamente eficaz, pois é o método de teste que não apenas testa a funcionalidade do software, mas também testa a estrutura interna do aplicativo (JAMIL *et al.* 2016; JOVANOVIĆ, 2006).

Já o teste de caixa cinza é a combinação da técnica de teste de caixa branca e caixa preta, servindo as vantagens de ambas (JAMIL *et al.* 2016; JOVANOVIĆ, 2006). A necessidade desse tipo de teste surgiu porque nesse tipo de teste o testador tem conhecimento da estrutura interna da aplicação, testando assim a funcionalidade de uma forma melhor levando em consideração a estrutura interna da aplicação (JAMIL *et al.* 2016).

A garantia de qualidade de software é uma parte crítica e cara do ciclo de vida do software. A identificação de defeitos de software antes do estágio de teste pode reduzir os custos e o tempo de manutenção. Desta forma, quanto mais cedo um defeito for encontrado no processo de desenvolvimento, menor poderá ser o custo para corrigi-lo (PIERCE, 1996).

Um oráculo de teste, normalmente expresso como uma condição, documenta o comportamento pretendido de uma unidade sob um determinado prefixo de teste, sendo assim, sintetizar um oráculo de teste funcional é um problema desafiador, pois ele deve capturar a funcionalidade pretendida ao invés da funcionalidade implementada (DINELLA *et al.* 2022).

Os oráculos de teste são geralmente usados para determinar se os comportamentos observados dos sistemas sob teste estão corretos (TIMO; PETRENKO; RAMESH 2019). Projetar um oráculo de teste é frequentemente uma tarefa manual e custosa, exigindo o envolvimento de especialistas, especialmente na

indústria, onde produtos de software possuem especificações funcionais complexas a serem testadas (UKAI *et al.* 2019).

2.1.4. Técnicas de Teste de Software

Diferentes técnicas de teste são mais adequadas a diferentes abordagens de engenharia de software e em diferentes momentos. Estas costumam estar associadas ao tipo de teste a ser realizado. Assim, as técnicas de caixa preta e de caixa branca não são antagônicas, é inclusive recomendado que sejam utilizadas de forma complementar (PRESSMAN; MAXIM, 2016).

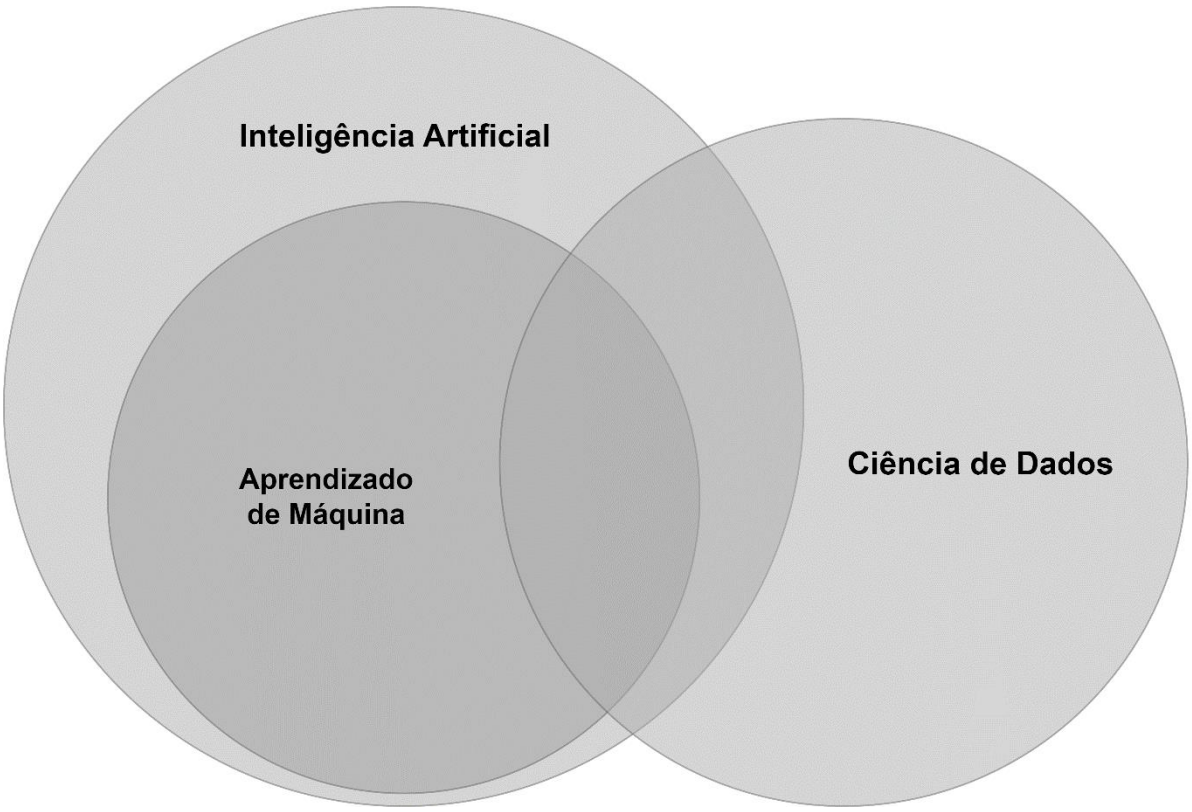
Sendo assim, as técnicas de teste fornecem orientações sistemáticas para a elaboração de testes que exercitarão a lógica interna e as interfaces dos componentes do software ou que exercitem os domínios de entrada e saída do programa, com o objetivo de descobrir erros no funcionamento, comportamento e desempenho do mesmo (PRESSMAN; MAXIM, 2016).

Em uma pesquisa realizada, foi notada a perda enfrentada pela indústria de software devido a problemas de qualidade e testes, incluindo a perda financeira por falhas e/ou um impacto negativo no crescimento das empresas (KUMAR, 2019). Assim, é notável a importância da descoberta de erros enquanto o sistema não está em uso, objetivando a entrega de um software confiável e reduzindo as chances da ocorrência de danos irreparáveis causados por falhas operacionais (SOMMERVILLE, 2011).

2.2. Aprendizado de Máquina

Inteligência Artificial, dentre diversas definições, é classificada como a ramo da computação cientifica que se preocupa com a automação de comportamento inteligente (HAENLEIN, 2019). Esta, segundo KULIN *et al.* (2021), envolve áreas como a robótica, processamento de linguagem natural, recuperação de informações, visão computacional e o Aprendizado de Máquina. Os trabalhos nesta área se iniciaram por volta dos anos 1950, tal assunto ganhou notoriedade mais de meio século depois com o crescimento de *Big Data* e melhorias no poder de computação, tomando espaço fora do ambiente acadêmico. (HAENLEIN, 2019). Ainda, conforme apresentado na Figura 3, na área de Ciência de Dados, as técnicas mais empregadas são fornecidas pela Inteligência Artificial com desenvolvimentos de Aprendizado de Máquina (FACELI *et al.,* 2021).

Figura 3: Diagrama de Venn Inteligência Artificial, Aprendizado de Máquina e Ciência de Dados.



Fonte: Própria autora.

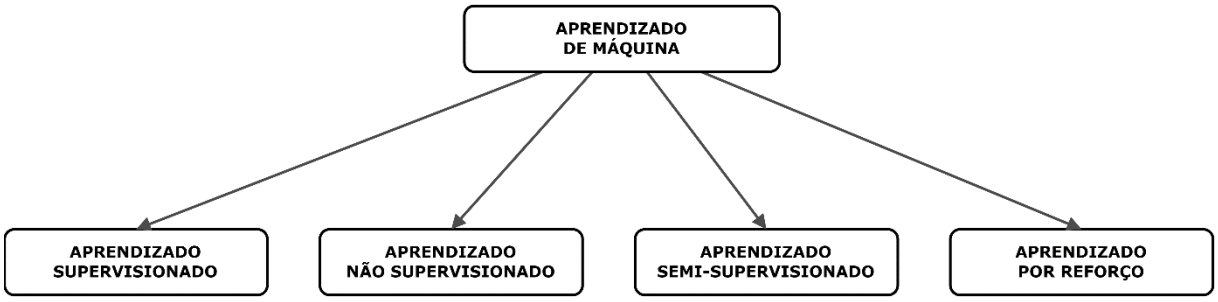
O sistema de aprendizado busca aprender os fatos baseado no que recebe como entrada em seu treinamento ou durante seu trabalho. Há vários locais e diferentes formas que pode ser usado, mas o que sempre fica é o aprendizado, melhor dizendo, habilidade de adquirir conhecimento internamente, identificando padrões nos

dados de entrada (GOODFELLOW, 2016).Diante disso, Aprendizado de Máquina se trata de uma área da Inteligência Artificial formulada com o objetivo de, diante de um conjunto de dados, buscar relações a fim de tomar decisões ou previsões a partir dos mesmos (CARUSO; CAVALHEIRO, 2021; ZHANG *et al.,* 2020). Neste sentido, de forma resumida, um sistema de aprendizado é um programa capaz de tomar decisões baseando-se no acúmulo de experiências de soluções de problemas anteriores (MONARD; BARANAUSKAS, 2003).

O primeiro tipo de Aprendizado de Máquina (AM) é o aprendizado supervisionado, que funciona sob supervisão e conjuntos de dados rotulados, estes dados que já conhecem a resposta de destino são chamados de conjuntos de dados de rótulo (AL-SAHAF *et al.* 2019; GUPTA *et al.* 2022). O aprendizado supervisionado funciona em duas categorias diferentes, Regressão e Classificação; a regressão funciona em dados contínuos que definem a relação entre variáveis dependentes e independentes enquanto a classificação é uma tarefa em que cada exemplo/instância é classificado em uma das categorias predefinidas (AL-SAHAF *et al.* 2019; GUPTA *et al.* 2022). Já o segundo tipo de Aprendizado de Máquina, o aprendizado não supervisionado, não possui dados rotulados.

O aprendizado não supervisionado conta com os algoritmos de *clustering* e visam aprender um modelo que pode agrupar instâncias em *clusters* separados com base nas características intrínsecas dos dados não rotulados (AL-SAHAF *et al.* 2019; GUPTA *et al.* 2022). O terceiro tipo de Aprendizado de Máquina é o Aprendizado por Reforço, que é a aprendizagem baseada em recompensas, geralmente utilizado quando é necessária a tomada de uma série de decisões (AL-SAHAF *et al.* 2019; GUPTA *et al.* 2022). Assim, são recebidas recompensas por tentativas bem-sucedidas e punições quando malsucedidas. A partir daí, a tentativa será minimizar as ações inapropriadas e maximizar as apropriadas aprendendo com suas experiências e ações (MAHADEVKAR *et al.* 2022). Outro tipo, por último, é o de Aprendizado semi-supervisionado, que possui algoritmos que são treinados em dados que são rotulados e sem rótulo (MAHADEVKAR *et al.* 2022).

Figura 4: Tipos de Aprendizado de Máquina.



Fonte: Própria autora, adaptado de SARKER, 2021.

2.2.1. Algoritmos de Aprendizado de Máquina: Aplicações e Desafios

Conforme Caruso e Cavalheiro (2021), ao citar Raschka (2019), boa parte das aplicações de Inteligência Artificial cotidianas são baseadas em Aprendizado de Máquina, como assistentes de voz, filtros de *spams* em *e-mails*, carros autônomos e sistemas de recomendação de filmes e outros. Além disso, a aplicação de técnicas de Inteligência Artificial e Aprendizado de Máquina estão sendo utilizadas em pesquisas científicas, em uma variedade de áreas que utilizam grandes volumes de dados (FACELI *et al.* 2021).

Assim, os algoritmos de Aprendizado de Máquina podem ser categorizados com base em sua semelhança em termos de sua funcionalidade e estrutura, como exemplos: algoritmos de regressão, algoritmos baseados em instâncias, de árvore de decisão, algoritmos bayesianos, de agrupamento, redes neurais artificiais, algoritmos de aprendizado profundo, algoritmos de redução de dimensionalidade, entre outros (JIANG *et al.* 2017).

De acordo com Murphy, Kaiser e Arias (2007), conforme Marijan e Gotlieb (2020), os sistemas de Aprendizado de Máquina tem o potencial de mudar constantemente de comportamento à medida em que mais dados se tornam disponíveis. Porém, algoritmos de aprendizado, em algumas situações, podem cair no problema de superadaptação, que costuma ocorrer quando há não conformidades nos dados ou estes não correspondem adequadamente ao espaço de dados possíveis (COPPIN, 2010).

2.2.2. Aprendizado de Máquina e Testes de Software

Apesar de existirem registros desde 1995 da aplicação de algoritmos de Aprendizado de Máquina às atividades de testes de software, somente há poucos anos tal relação tem sido de interesse de pesquisadores e profissionais, sendo verificado o aumento dos estudos (DURELLI *et al.,* 2019).

As pesquisas na área de engenharia de software relacionadas a Aprendizado de Máquina têm se intensificado. Hourani, Hammad e Lafi (2019) avaliaram o impacto da Inteligência Artificial no teste de software, e encontraram que a Inteligência Artificial pode obter resultados promissores no teste de software. Os testes realizados por Inteligência Artificial reduzirão o tempo de lançamento no mercado, aumentará a eficiência da organização para produzir software mais sofisticado e criará testes automatizados mais inteligentes. Bombiri, Poda e Ouedraogo (2023), em uma mini revisão sobre a aplicação de Aprendizado de Máquina na qualidade de software, encontraram que o gerenciamento da qualidade de software por meio de métodos de Aprendizado de Máquina é um campo de pesquisa ativo, onde a previsão de defeitos de software e a previsão de manutenibilidade são problemas que são tratados por métodos de Aprendizado de Máquina.

Lima, Cruz e Ribeiro (2019), realizaram uma revisão de literatura sobre Inteligência Artificial aplicada a Testes de Software, e os achados mostram que o teste de caixa preta é, de longe, o método preconizado de teste de software mais explorado quando a Inteligência Artificial é aplicada, e os três métodos mais tradicionais de Aprendizado de Máquina (supervisionado, não supervisionado e de reforço) são comumente usados em testes de caixa preta. Em seus estudos, Vedpal e Chauhan (2021) investigaram o papel do Aprendizado de Máquina no teste de software e concluíram que os resultados das técnicas baseadas em Aprendizado de Máquina possuem maior eficácia quando comparadas a outras técnicas.

Por fim, destaca-se a existência de diversos algoritmos de Aprendizado de Máquina, no entanto, as limitações e o potencial dos mesmos devem ser entendidos e avaliados de acordo com os problemas nos quais os algoritmos são aplicados (MONARD; BARANAUSKAS, 2003).

3. METODOLOGIA

Este estudo foi desenvolvido como uma revisão sistemática de literatura e suas etapas foram adaptadas de diretrizes para a realização de trabalhos neste formato na área de engenharia de software. (KITCHENHAM; CHARTERS, 2007; PETERSEN; VAKKALANKA; KUZNIARZ, 2015). Assim, a partir dos estudos de Kitchenham e Charters (2007) e Petersen, Vakkalanka e Kuzniarz (2015), foram definidas as etapas para a realização desta revisão de literatura.

3.1. Contextualização

Ao início da realização deste trabalho, foi realizada a leitura acerca do tema de testes, engenharia de software, Inteligência Artificial e Aprendizado de Máquina. Esta etapa foi fundamental para a contextualização da problemática e embasamento científico para a realização deste trabalho.

3.2. Questões de Pesquisa

Com base nas investigações feitas por Durelli *et al.* (2019) e com o intuito de atender ao contexto da área de pesquisa e aos objetivos propostos, foram elaboradas as seguintes questões de pesquisa:

- Q1. Qual a quantidade e frequência das pesquisas sobre Aprendizado de Máquina aplicado a testes de software?
- Q2. Quais atividades, artefatos, níveis, tipos e técnicas de testes de software estão envolvidos nas pesquisas realizadas?
- Q3. Quais as vantagens e desvantagens citadas da aplicação de Aprendizado de Máquina em testes de software?

3.3. Estratégias de Busca e Identificação dos Trabalhos

A fim de identificar estudos publicados entre os anos de 2018 e 2022 que abordassem a aplicação de Aprendizado de Máquina em testes de software, foi realizada uma busca extensiva nas seguintes bases de dados: ACM Digital Library, *IEEE Xplore, Google Scholar, Science Direct, Taylor & Francis, Wiley Online e World Scientific*. Tais bases foram selecionadas pelo fato de o acesso a seus artigos estarem

disponíveis por meio dos periódicos CAPES. Ainda, foram considerados trabalhos nos idiomas português e inglês.

Na estratégia de busca, foram utilizadas palavras-chaves relevantes na área da pesquisa, e tais termos foram escolhidos por meio da leitura de estudos sobre o tema e consenso entre o(a) aluno(a) e o(a) professor(a) orientador(a). Os termos de busca utilizados foram: “aprendizagem de máquina”, “Aprendizado de Máquina”, “*machine learning*”, “teste de software”, “testes de software”, “software *test*” e “software *testing*”. A busca foi realizada utilizando tais termos combinados por meio de operadores booleanos, sendo necessário a adaptação da *string* de consulta, de acordo com a formatação utilizada no campo de busca de cada base.

Ainda, de acordo com os campos de busca avançada apresentados em cada base de dados, era possível acrescentar outros parâmetros para refinar ainda mais a busca, como por exemplo: o ano de publicação, tipo do estudo e área de pesquisa.

3.4. Seleção dos Trabalhos

Os trabalhos foram buscados nas bases de dados pelos envolvidos no presente trabalho. Após a busca, foi realizada uma seleção inicial a partir da análise dos títulos e resumos dos trabalhos e os estudos repetidos foram excluídos.

3.4.1. Critérios de Inclusão

Um conjunto de critérios de elegibilidade foram definidos a fim de direcionar a seleção dos trabalhos, considerando as características esperadas e definidas na metodologia deste estudo. Porém, um trabalho atender a um ou todos os critérios não o torna obrigatoriamente parte dos estudos incluídos, só o torna potencialmente selecionado (KITCHENHAM; CHARTERS, 2007).

Sendo assim, os critérios de inclusão elencados foram:

i. O estudo apresenta a aplicação de Aprendizado de Máquina a testes de software;
ii. O estudo foi publicado entre 2018 e 2022.

3.4.2. Critérios de Exclusão

Durante a pesquisa, foram elencados critérios de exclusão, critérios estes que se tornam úteis para o refinamento da pesquisa. Dentre eles, podemos citar: tipo do estudo realizado, tipos de metodologia descrita no trabalho, tipo de software considerado, ano de publicação, idioma da publicação, base em que foi publicado o trabalho, entre outros. Assim, temos os seguintes principais critérios de exclusão:

i. Questionários(*surveys*), revisões sistemáticas, mapeamentos sistemáticos, artigos de replicação ou revisão, estudos empíricos, trabalhos preliminares, *work-in-progress*, capítulos de livro, palestras, artigos de relatos de *insights*, experiências e desafios, reflexões críticas e outros;
ii. Softwares embarcados, softwares de automação de maquinários industriais, sistemas ciberfísicos, softwares de análises de imagens e outros.
iii. Estudos sobre testes de Aprendizado de Máquina, análise de envelhecimento de código, previsão de defeitos e falhas, análise e cálculo de métricas como testabilidade, confiabilidade e manutenibilidade, testes de *hardware* e identificação de *malwares*.

3.5. Extração e Síntese dos Dados

Após a seleção dos estudos, os dados foram extraídos pelos pesquisadores. Uma planilha do *Microsoft Excel* foi preenchida com alguns critérios como: fonte, autores, contexto, delineamento do trabalho e data da publicação (KITCHENHAM; CHARTERS, 2007). Foi realizada uma revisão da extração dos dados e, por fim, estes foram reunidos para que fosse possível categorizar e confrontar os resultados de acordo com as questões de pesquisa propostas.

4. RESULTADOS E DISCUSSÃO

4.1. Pesquisa dos Trabalhos

Um total de 470 artigos foram identificados a partir das ferramentas de pesquisa citadas na seção 3.3, utilizando-se de uma expressão de busca com operadores booleanos, conforme abaixo.

Tabela 1: *String* de busca com operadores booleanos.

***String* de busca**
("Aprendizado de Máquina" ***OR*** "aprendizagem de máquina" ***OR*** "*machine learning*")
AND
("teste de software" ***OR*** "testes de software" ***OR*** "software *test*" ***OR*** "software *testing*")

Fonte: Própria autora.

Apesar da expressão de busca contar com termos em português, indo ao encontro da proposta metodológica de incluir trabalhos neste idioma, nas bases utilizadas na pesquisa não foram encontrados estudos em português. Assim, o total de 470 trabalhos refere-se apenas a artigos em inglês, sendo retornados 270 artigos na base *IEEE Xplore*, 39 trabalhos na Taylor & Francis, 123 na base de dados *Wiley Online* e 38 na *World Scientific,* conforme Tabela 4.

Dentre as bases utilizadas neste trabalho, além das citadas possivelmente anteriormente, estariam inclusas as bases ACM *Digital Library*, *Google Scholar* e *Science Direct*, devido a relevância dos trabalhos contidos nelas. Porém, devido ao alto número de estudos retornados, conforme apresentado na Tabela 2, elas foram desconsideradas para manter a viabilidade do estudo, considerando a quantidade de pesquisadores envolvidos e cronograma previsto.

O *Google Scholar*, como exemplo, tem como proposta que, de apenas uma base, seja possível encontrar diversos tipos de estudos de múltiplas fontes, funcionando como um indexador para outras bases. Assim, imagina-se que teriam estudos duplicados, visto que são retornados estudos de outras bases já utilizadas.

Tabela 2: Resultados das pesquisas com a expressão de busca em todas as bases.

Base de dados	Quantidade de estudos
ACM	1.124
Google Scholar	14.300
IEEE Xplore	270
Science Direct	499
Taylor & Francis	39
Wiley Online	123
World Scientific	38
Total	**16.393**

Fonte: Própria autora.

Ainda sobre a pesquisa inicial, em relação aos termos utilizados na *string* de busca, inicialmente o termo “Artificial inteligência“ e sua tradução “*artificial intelligence*” haviam sido utilizados na estratégia.

Porém, devido a um número maior de trabalhos encontrados, conforme a Tabela 3, esse termo foi removido da pesquisa, pois consideramos que em muitos casos é utilizado de forma geral e/ou genérica em títulos e resumos, retornando trabalhos que não correspondem totalmente aos critérios de inclusão e o tema da pesquisa.

Tabela 3: Resultado da busca em todas as bases com os termos “Inteligência Artificial” e “*artificial intelligence*”.

Base de dados	Quantidade de estudos
ACM	1.425
Google Scholar	15.400
IEEE Xplore	539
Science Direct	607
Taylor & Francis	63
Wiley Online	135
World Scientific	47
Total	**18.216**

Fonte: Própria autora.

4.2. Seleção dos Trabalhos

4.2.1. Processo de seleção dos Estudos

Diante dos resultados das buscas realizadas nas bases de dados, foram lidos os títulos e resumos dos 470 artigos retornados. Nesta etapa, os critérios de inclusão e exclusão foram utilizados para realizar uma seleção inicial dos trabalhos que, até este ponto, estavam de acordo com os parâmetros de elegibilidade da pesquisa. Diante disso, foram selecionados 108 trabalhos, sendo eles: 58 na base IEEE *Xplore*, 7 na base Taylor & Francis, 32 na Wiley *Online* e 11 na World *Scientific*.

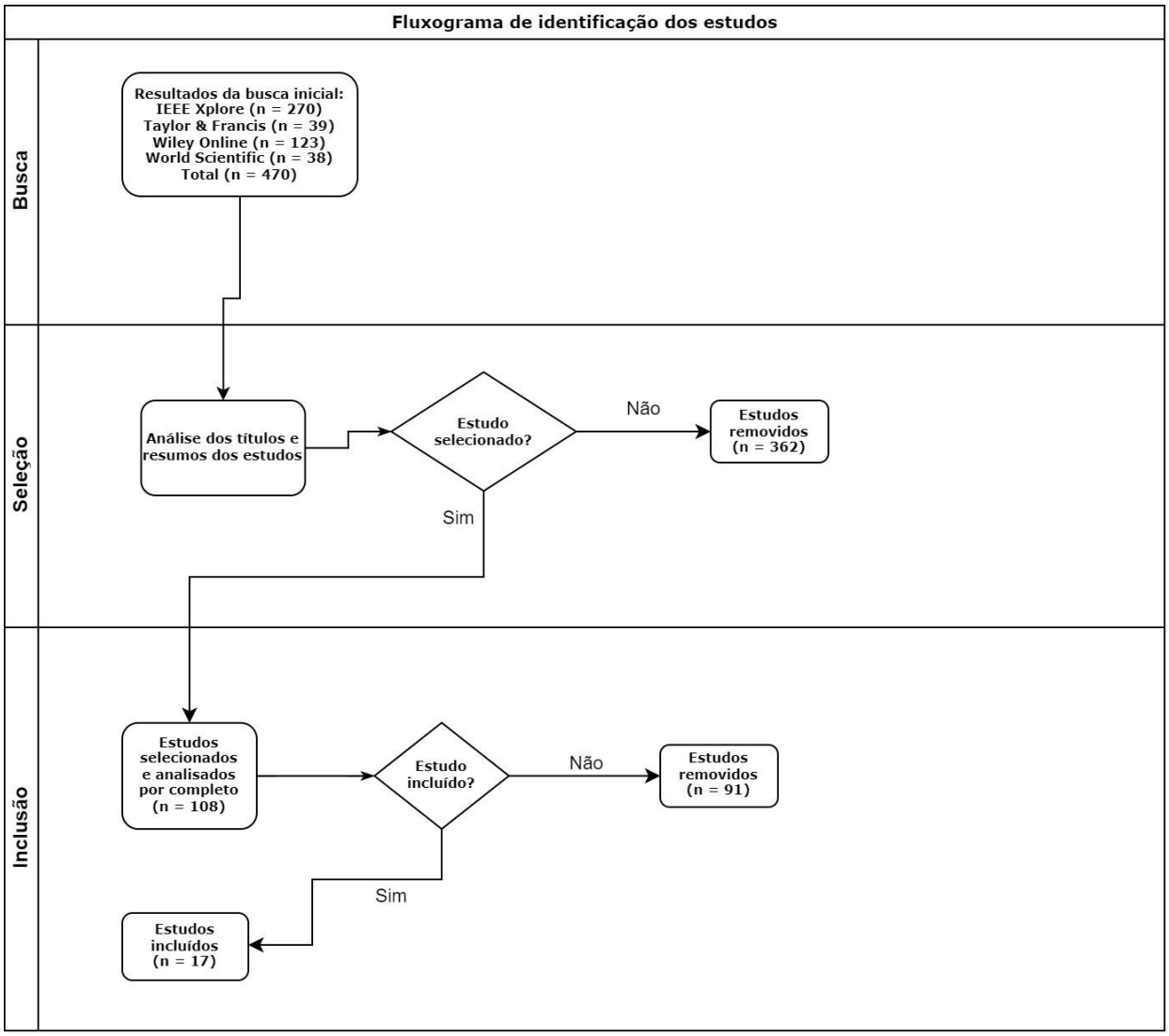
Depois de selecionar os trabalhos com base em seus títulos e resumos, estes 108 artigos foram elencados para leitura completa. Por fim, após leitura de todos os textos por completo e remoção dos estudos duplicados entre bases, 17 trabalhos atenderam aos critérios de elegibilidade e foram incluídos nesta revisão. A figura 5 ilustra o processo de seleção do estudo e a tabela 4 os quantifica.

Tabela 4: Resultados de busca e seleção dos estudos.

Base de dados	Estudos retornados	Estudos selecionados	Estudos incluídos
IEEE Xplore	270	58	8
Taylor & Francis	39	7	0
Wiley online	123	32	6
World Scientific	38	11	3
Total	**470**	**108**	**17**

Fonte: Própria autora.

Figura 5: Fluxograma de identificação dos estudos.



Fonte: Própria autora.

Os dados apresentados no Gráfico 1 mostram as proporções de estudos retornados, selecionados e incluídos de diferentes bases de dados: IEEE Xplore, Taylor & Francis, Wiley Online e World Scientific. A análise desses dados nos fornece uma visão valiosa das tendências e padrões na pesquisa sobre a aplicação do Aprendizado de Máquina em testes de software.

No primeiro ponto, nota-se que a base IEEE Xplore teve o maior número de estudos retornados (270), bem como o maior número de estudos selecionados (58) e incluídos (8). Isso pode sugerir que a IEEE Xplore, que é uma fonte amplamente respeitada para pesquisa em engenharia e tecnologia, contém uma quantidade significativa de literatura relevante sobre a aplicação de Aprendizado de Máquina em

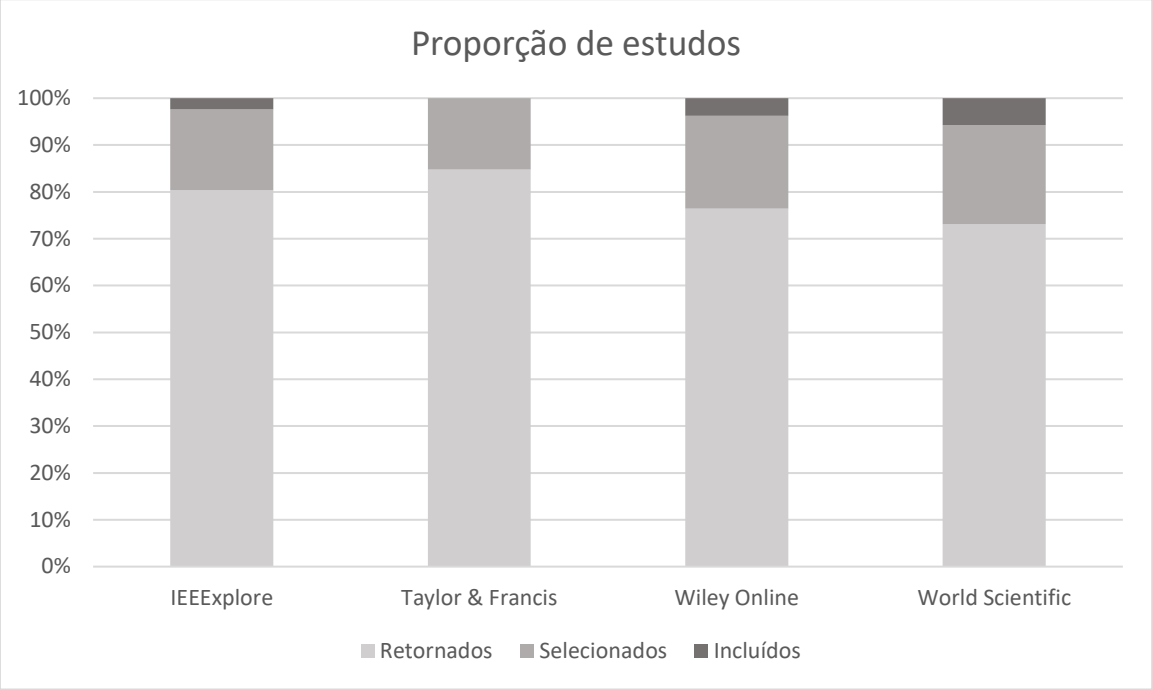
testes de software. E, também, pode indicar que os pesquisadores nesta área tendem a publicar seus trabalhos nesta base de dados.

No segundo ponto, é interessante notar que, apesar da Taylor & Francis ter retornado um total de 39 estudos, nenhum deles foi incluído na revisão final. Isso pode indicar que, embora essa base de dados possa conter trabalhos relacionados ao Aprendizado de Máquina e testes de software, os critérios de seleção e inclusão utilizados neste estudo podem não ter considerado esses trabalhos relevantes ou de alta qualidade o suficiente para inclusão. Isso sugere a importância de avaliar cuidadosamente a qualidade e relevância dos estudos durante o processo de revisão da literatura.

Por fim, observando as bases de dados Wiley Online e World Scientific, nota-se que ambas retornaram uma quantidade razoável de estudos (123 e 38, respectivamente), com uma proporção menor sendo selecionada e ainda menos sendo incluída. Isso indica que essas bases também contêm literatura relevante, embora em uma extensão menor do que a IEEE Xplore, e que os estudos de ambas foram considerados de alta qualidade e relevância suficiente para inclusão, ainda que em menor quantidade.

Portanto, esses dados nos dão uma visão interessante do campo da aplicação do Aprendizado de Máquina em testes de software, bem como das tendências de publicação e da qualidade dos estudos disponíveis nas diferentes bases de dados.

Gráfico 1: Proporção de estudos retornados, selecionados e incluídos.



Fonte: Própria autora.

O Gráfico 2 apresenta a relação de estudos excluídos por diferentes critérios a partir da base de dados IEEE Xplore. Ele oferece uma visão importante da quantidade e da natureza dos estudos que foram considerados não relevantes para este trabalho de conclusão de curso.

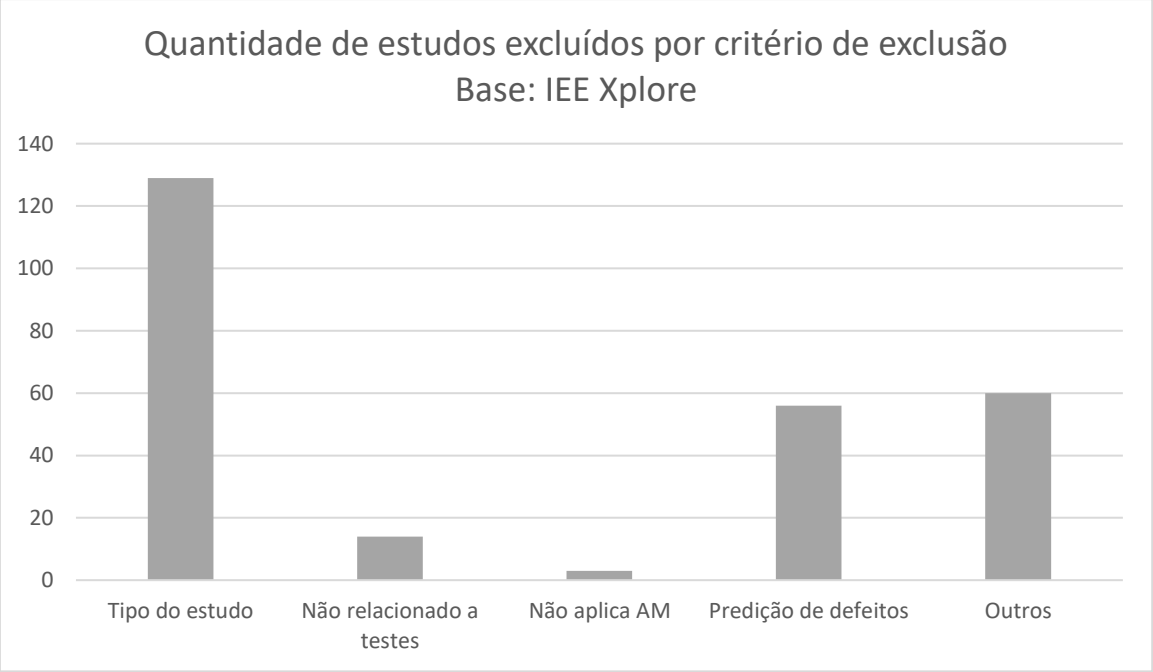
A categoria "Tipo do estudo" teve o maior número de exclusões, com 129 estudos sendo removidos. Isso indica que uma quantidade significativa de pesquisas retornadas da base IEEE Xplore não atendia aos critérios de inclusão do tipo de estudo definidos para esta revisão sistemática. Esses estudos excluídos possuíam naturezas variadas como estudos teóricos, ensaios, opiniões, entre outros, que não se encaixavam no escopo do estudo atual que buscava trabalhos empíricos, revisões sistemáticas, estudos de caso, entre outros.

A categoria "Predição de defeitos" contou com 56 estudos excluídos. Embora a predição de defeitos possa envolver elementos de testes de software e Aprendizado de Máquina, isso sugere que esses estudos foram considerados não diretamente relevantes para o objetivo principal desta revisão, que é a aplicação de Aprendizado de Máquina em testes de software, não considerando a predição de defeitos.

Os critérios "Não relacionado a testes" e "Não aplica AM" são por si só bastante autoexplicativos. Se um estudo não estava relacionado ao teste de software (14 estudos) ou não aplicava Aprendizado de Máquina (3 estudos), ele não seria relevante para a revisão que tem como foco a intersecção dessas duas áreas.

Por fim, a categoria "Outros", com 60 exclusões, abrange uma variedade de motivos, incluindo questões como qualidade do estudo, falta de informações suficientes, impossibilidade de acesso ao estudo entre outros motivos.

Gráfico 2: Relação de estudos excluídos por critérios de exclusão, base IEEE Xplore.



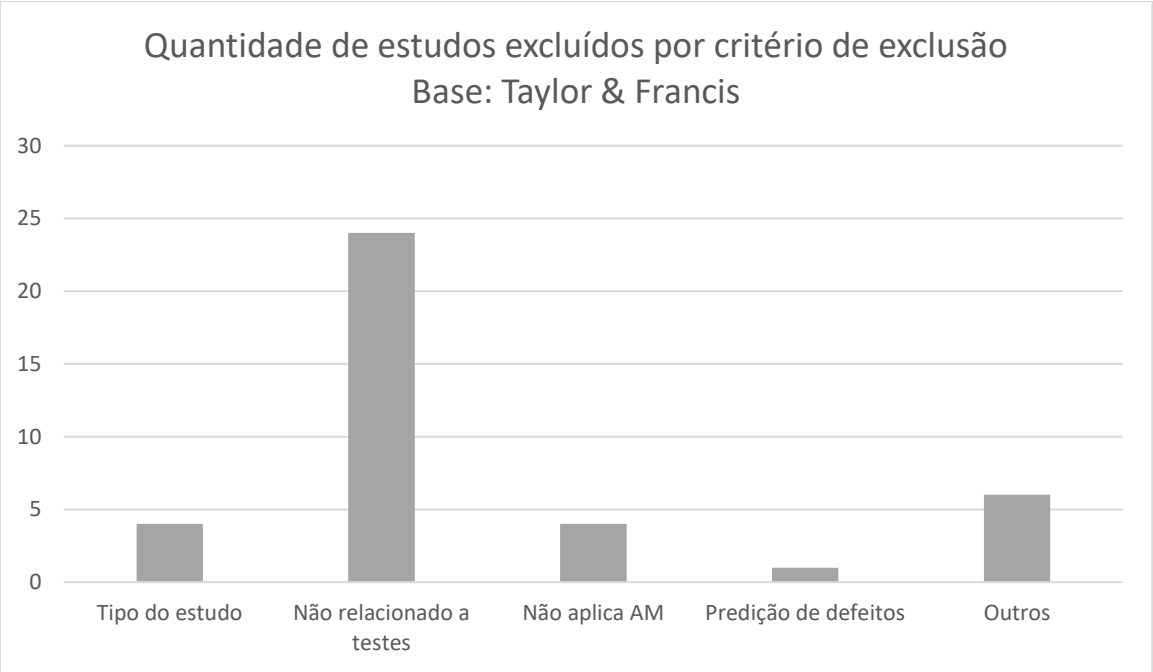
Fonte: Própria autora.

O Gráfico 3 representa os estudos excluídos da base de dados Taylor & Francis por diferentes critérios de exclusão. É notável que o critério de exclusão com maior representatividade é "Não relacionado a testes", com 24 estudos excluídos. Isso sugere que uma quantidade significativa de trabalhos identificados nesta base de dados não está diretamente ligada ao contexto de testes de software, embora possa envolver elementos de Aprendizado de Máquina. A aplicação de critérios de exclusão rigorosos é crucial para garantir a relevância e a qualidade dos estudos selecionados para uma revisão sistemática.

Comparativamente, as razões para exclusão na base de dados IEEE Xplore são bastante diferentes. Por exemplo, "Tipo do estudo" e "Predição de defeitos" foram

os critérios de exclusão mais representativos, indicando uma possível diferença no foco temático entre as duas bases de dados. Na Taylor & Francis, as exclusões por "Tipo do estudo" e "Predição de defeitos" foram consideravelmente menores. Além disso, é importante notar que o total de estudos excluídos na Taylor & Francis (39 estudos) é muito menor do que na IEEE Xplore (262 estudos), refletindo possivelmente uma menor disponibilidade de estudos sobre o tema nas bases de dados da Taylor & Francis.

Gráfico 3: Relação de estudos excluídos por critérios de exclusão, base Taylor & Francis.



Fonte: Própria autora.

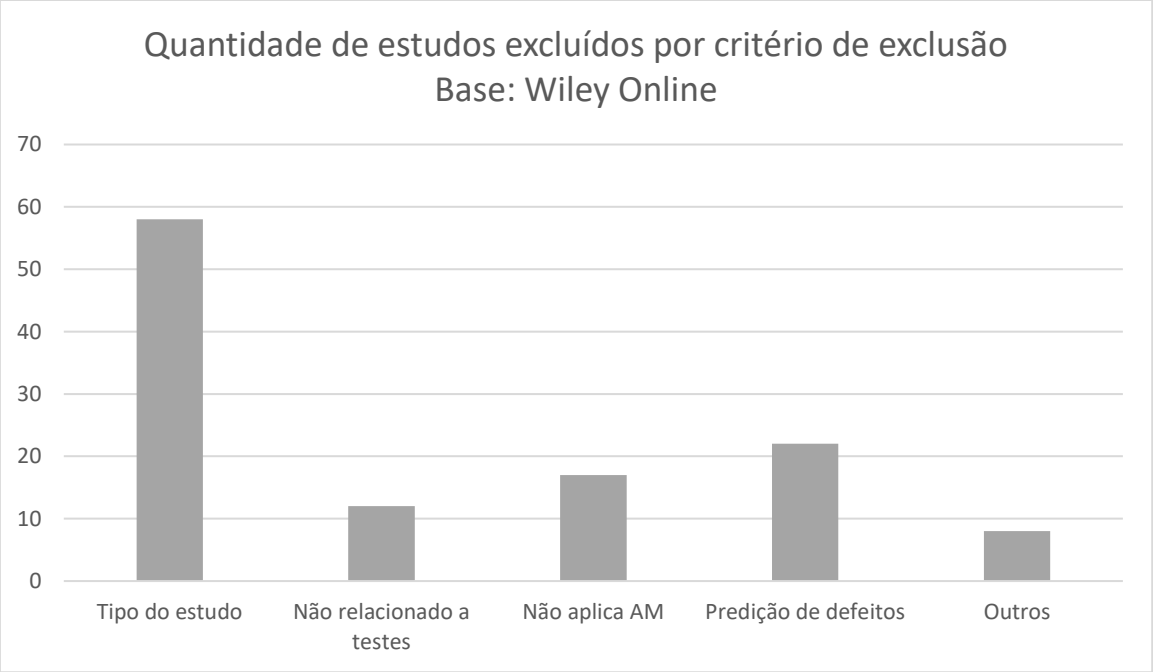
O Gráfico 4 mostra a relação de estudos excluídos pela base de dados Wiley Online de acordo com diferentes critérios de exclusão. O maior grupo de exclusões se deve ao critério "Tipo do estudo", que contabilizou 58 exclusões. Isso representa aproximadamente 47% dos estudos excluídos, indicando que quase metade dos estudos retornados dessa base de dados não correspondia ao tipo de estudo desejado para esta revisão sistemática.

A categoria "Não aplica AM" foi responsável por 17 exclusões. Isso sugere que uma parcela significativa dos estudos que inicialmente pareciam relevantes, por fim, não se relacionavam diretamente com a aplicação do Aprendizado de Máquina, o que é uma exigência central desta revisão.

A "Predição de defeitos" resultou na exclusão de 22 estudos, cerca de 18% do total. Como discutido anteriormente, embora a predição de defeitos possa envolver elementos de testes de software e Aprendizado de Máquina, estes estudos não são diretamente relevantes para a finalidade desta revisão. Em "Não relacionado a testes", houve 12 exclusões, representando quase 10% dos estudos excluídos. Esses estudos não possuíam a conexão necessária com o contexto de testes de software.

Por último, a categoria "Outros" registrou 8 exclusões, aproximadamente 6.5% do total. Essas exclusões mostram que a escolha da base de dados pode ter um impacto significativo sobre os estudos retornados, e que é importante aplicar rigorosamente os critérios de exclusão para garantir a relevância e qualidade da revisão sistemática.

Gráfico 4: Relação de estudos excluídos por critérios de exclusão, base Wiley Online.



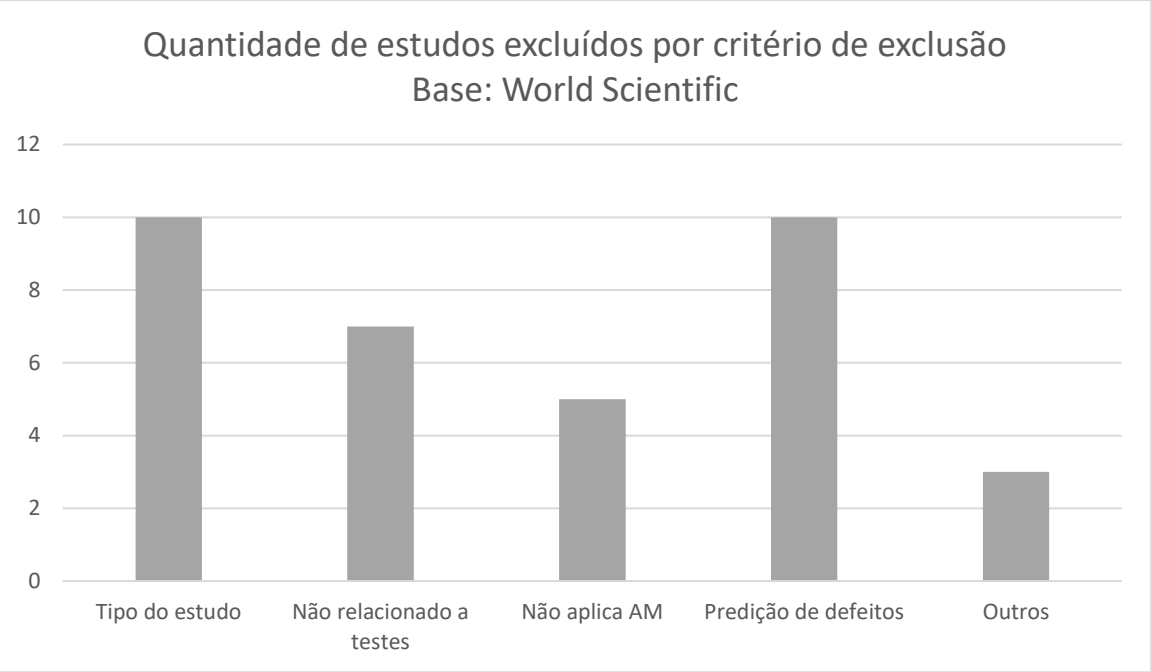
Fonte: Própria autora.

O Gráfico 5 representa a relação de estudos excluídos da base de dados World Scientific por diferentes critérios de exclusão. No total, foram 35 estudos excluídos, com "Tipo do estudo" e "Predição de defeitos" liderando a contagem de exclusões com 10 estudos cada. Isso sugere que quase um terço dos estudos encontrados nesta base não correspondiam ao tipo de estudo necessário para a revisão ou estavam focados em previsão de defeitos, não sendo diretamente relevantes para a intersecção de Aprendizado de Máquina e testes de software.

O critério "Não relacionado a testes" resultou na exclusão de 7 estudos, representando cerca de 20% do total de estudos excluídos. Isto indica que uma parcela relevante dos estudos não estava diretamente relacionada ao contexto de testes de software. O critério "Não aplica AM" teve 5 estudos excluídos. E, por fim, a categoria "Outros" teve 3 estudos excluídos. Estes provavelmente foram excluídos por uma variedade de outros motivos mencionados anteriormente que coincidem com as outras bases.

Em comparação com as outras bases de dados discutidas anteriormente, o World Scientific parece ter uma distribuição mais uniforme de exclusões entre os critérios. Entretanto, é importante notar que o total de estudos excluídos nessa base (35 estudos) é menor do que em IEEE Xplore, Taylor & Francis e Wiley Online.

Gráfico 5: Relação de estudos excluídos por critérios de exclusão, base World Scientific.



Fonte: Própria autora.

O Gráfico 6 mostra a relação de estudos excluídos em todas as bases de dados de acordo com os critérios de exclusão definidos e, com um total de 453 estudos excluídos, podemos avaliar a distribuição e proporções dessas exclusões de forma geral.

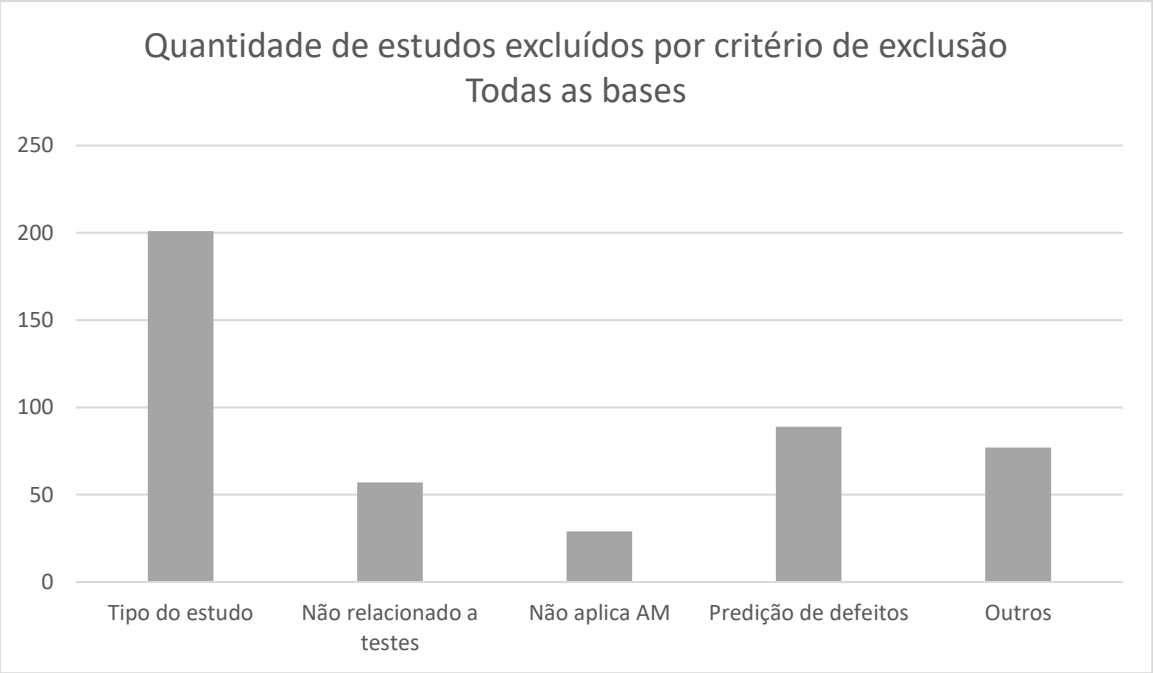
O critério de exclusão mais frequente foi "Tipo do estudo", responsável pela exclusão de 201 estudos, o que representa aproximadamente 44% do total de estudos excluídos. Foi observada uma grande quantidade de estudos empíricos, estudos

preliminares, questionários, revisões e mapeamentos sistemáticos, artigos de replicação e revisão, entre outros, não relevantes para este trabalho.

O segundo critério mais frequente foi "Predição de defeitos", que levou à exclusão de 89 estudos, representando cerca de 20% do total de estudos excluídos. Assim, há uma grande quantidade de estudos, porém, os resultados de alguns estudos analisados indicam módulos de um sistema que potencialmente apresentariam defeitos, não sendo possível correlacionar a aplicação de Aprendizado de Máquina a uma atividade de teste específica.

A categoria "Outros" resultou na exclusão de 77 estudos, representando aproximadamente 17% do total de estudos excluídos. Como exemplo, temos: trabalhos que tratavam métricas de software, sistemas embarcados, artigos de acesso pago, testes de software como tema secundário e assim por diante. Por fim, "Não relacionado a testes" e "Não aplica AM" levaram à exclusão de 57 (aproximadamente 13%) e 29 (aproximadamente 6%) estudos respectivamente.

Gráfico 6: Relação de estudos excluídos por critérios de exclusão, em todas as bases.



Fonte: Própria autora.

4.3. Questões de Pesquisa

Q1. Qual a quantidade e frequência das pesquisas sobre Aprendizado de Máquina aplicado a testes de software?

Em relação às características dos trabalhos, todos os trabalhos selecionados foram publicados em inglês entre 2018 e 2022. Porém, os estudos foram realizados em 9 países diferentes, sendo eles: Brasil, China, Estados Unidos, Finlândia, Índia, Luxemburgo, Portugal, Reino Unido e Turquia. Dentre eles, o país com maior número de estudos selecionados foi a China (5). Em relação ao ano de publicação dos estudos, o maior número de publicações ocorreu em 2021 (5), sendo um número bem próximo ao dos demais anos.

Além disso, os autores utilizaram como metodologia de trabalho, os delineamentos estudo de caso e estudo experimental. Já se tratando do meio de publicação, nove estudos foram publicados em revistas, sendo cinco internacionais e quatro nacionais.

Ainda, foi identificado o fator de impacto de cada meio de publicação. Quanto às revistas, o *ranking Journal Citation Reports* da Clarivate foi consultado. Já em relação às conferências, simpósios e *workshops*, foi verificado o *Impact Score* no *Best Conferences Ranking*, da *Research.com*.

Todas as características dos estudos selecionados estão dispostas na Figura 6.

Figura 6: Tabela de caracterização dos estudos incluídos.

	País	Meio de publicação	Fator de Impacto	Delineamento
CABRAL et al. (2022)	Brasil	International Journal	1.5	Estudo de Caso
GUO et al. (2019)	China	International Journal	0.9	Estudo Experimental
HARDIN et al. (2018)	EUA	International Workshop	-	Estudo Experimental
KAHLES et al. (2019)	Finlândia	International Conference	3.6	Estudo Experimental
KOO et al. (2019)	EUA	International Conference	3.6	Estudo de Caso
KOROGLU et al. (2018)	Turquia	International Conference	3.6	Estudo Experimental
KOROGLU; SEN (2020)	Turquia	International Journal	1.5	Estudo Experimental
LI et al. (2022)	China	International Journal	0.9	Estudo Experimental
LIU et al. (2022)	China	National Journal	1.6	-
MA et al. (2021)	Luxemburgo	International Conference	11.8	Estudo Experimental
MARTINS et al. (2021)	Portugal	International Conference	1.8	Estudo de Caso
NAEEM et al. (2019)	China	National Journal	2.0	Estudo Experimental
OZ et al. (2021)	Turquia	International Journal	0.9	Estudo Experimental
PORRES et al. (2020)	Finlândia	International Conference	1.8	Estudo Experimental
RAJ et al. (2018)	Índia	International Symposium	2.8	Estudo Experimental
TSIMPOURLAS et al. (2021)	Reino Unido	National Journal	1.6	-
WANG et al. (2021)	China	National Journal	2.0	Estudo de Caso

Fonte: Própria autora.

Ao verificar a quantidade de publicações por ano, dentre os estudos selecionados, é possível identificar uma variação. Tal resultado vai ao encontro dos estudos de Durelli et al. (2019), onde é identificado que há pelo menos um estudo relacionado à aplicação de Aprendizado de Máquina em testes de software ao ano.

O Gráfico 7 apresenta a distribuição de publicações incluídas na revisão sistemática por ano, de 2018 a 2022. A partir dos dados apresentados, é possível identificar algumas tendências.

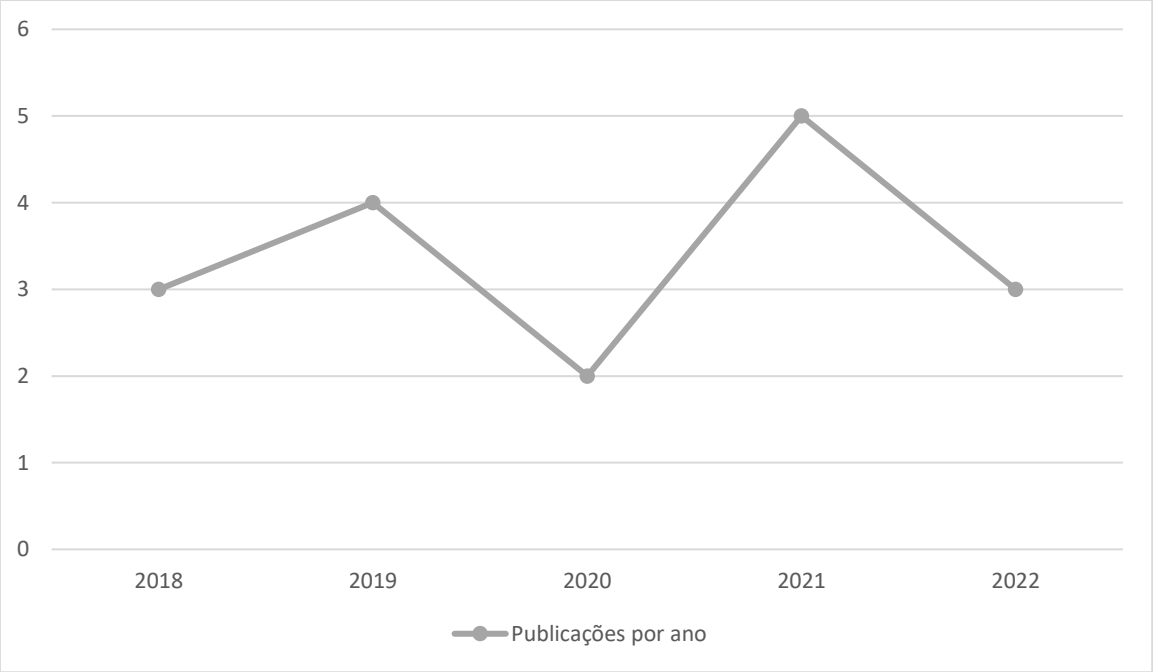
Em 2018, houve 3 publicações, o que representa cerca de 18% dos estudos incluídos. Em 2019, houve um pequeno aumento na produção de estudos, com 4 publicações, o que representa aproximadamente 24% do total. Percebe-se que, em 2020, houve uma pequena queda no número de publicações, com apenas 2 estudos incluídos, representando cerca de 12% do total. Isto poderia ser resultado de uma variedade de fatores, incluindo possivelmente o impacto da pandemia da COVID-19 na produção de pesquisa. No entanto, em 2021, houve um aumento significativo no número de publicações, com 5 estudos incluídos, representando cerca de 30% dos

estudos. Isto pode indicar um renovado interesse e avanços na aplicação do Aprendizado de Máquina em testes de software.

Em 2022, houve 3 publicações, representando cerca de 18% dos estudos. Embora isso represente uma pequena queda em relação ao pico de 2021, ainda é um número comparável aos anos anteriores, sugerindo uma tendência contínua de interesse na área.

Assim, a quantidade de publicações por ano indica que o uso de Aprendizado de Máquina em testes de software é uma área de pesquisa em crescimento, que tem atraído uma atenção considerável nos últimos anos. Esta tendência está alinhada com o crescimento geral do campo de Aprendizado de Máquina e Inteligência Artificial na indústria e na academia.

Gráfico 7: Publicações por ano, considerando os estudos incluídos.



Fonte: Própria autora.

O Gráfico 8 apresenta a quantidade de estudos incluídos por base de dados. A partir desses dados, podemos ver a predominância de algumas bases sobre as outras no que se refere aos estudos selecionados para a revisão.

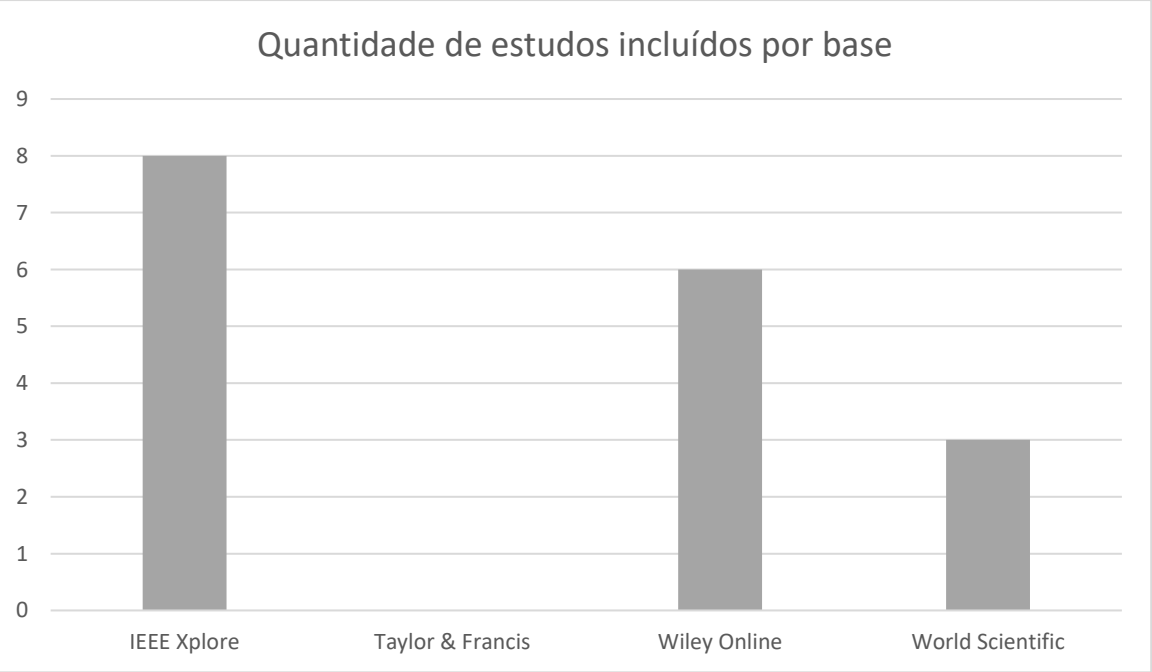
A base IEEE Xplore teve o maior número de estudos incluídos, com 8 estudos, que representam 47% dos estudos selecionados. Isso sugere que a base de dados IEEE Xplore é uma fonte significativa de pesquisa relevante para a interseção de

Aprendizado de Máquina e testes de software. Já a base Wiley Online também contribuiu com uma quantidade significativa de estudos, com 6 estudos incluídos, representando cerca de 35% dos estudos selecionados. Isso reforça a ideia de que essas duas bases, IEEE Xplore e Wiley Online, são importantes recursos para pesquisas nesta área.

Por outro lado, a base World Scientific contribuiu com 3 estudos, o que representa cerca de 18% dos estudos selecionados. Embora menor em número, a contribuição desta base ainda é relevante, considerando a especificidade do tópico em análise. E a base Taylor & Francis não contribuiu com nenhum estudo para esta revisão. Isso pode sugerir que esta base tem menos foco ou relevância no tópico específico de Aprendizado de Máquina aplicado a testes de software, pelo menos no período em questão.

No entanto, é importante notar que a quantidade de estudos selecionados de uma base de dados não necessariamente reflete a qualidade ou a relevância desses estudos. A qualidade e a relevância são determinadas pelos critérios de seleção e inclusão, e não simplesmente pelo número de estudos produzidos.

Gráfico 8: Publicações por base de dados, considerando os estudos incluídos.



Fonte: Própria autora.

O Gráfico 9 apresenta a distribuição dos estudos incluídos na revisão sistemática com base em seu delineamento. De acordo com os dados, a maioria dos estudos foi categorizada como "Estudo Experimental", seguida por "Estudo de Caso", enquanto dois estudos não indicaram seu delineamento.

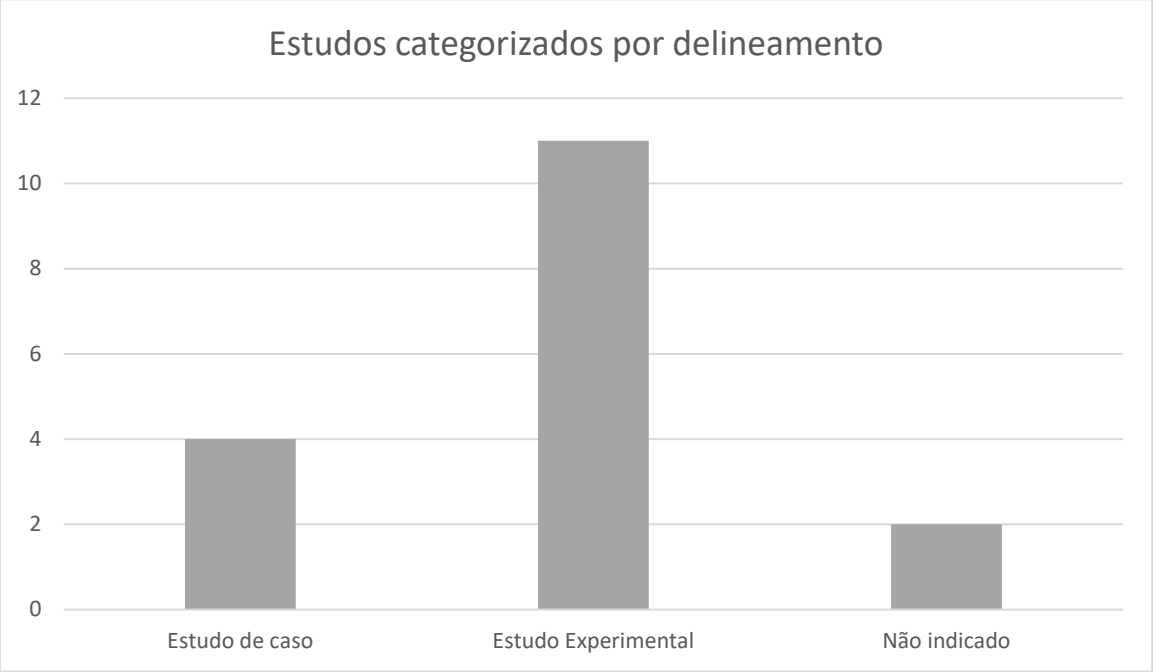
Destes, 11 dos estudos, foram classificados como "Estudo Experimental". Isso sugere que a maioria dos estudos nessa área tende a usar métodos experimentais para avaliar a aplicação do Aprendizado de Máquina em testes de software. Estes estudos, muitas vezes, envolvem a criação de experimentos controlados para testar hipóteses específicas ou para avaliar a eficácia de diferentes técnicas de Aprendizado de Máquina.

Por outro lado, 4 dos estudos, foram categorizados como "Estudo de Caso". Esses estudos, normalmente, envolvem o exame aprofundado de uma situação específica ou um exemplo particular de uso do Aprendizado de Máquina em testes de software.

Houve ainda dois estudos que não indicaram seu delineamento. Isso sugere que nem todos os estudos são claros ou explícitos sobre sua metodologia, o que pode ter implicações para a interpretação e replicabilidade desses estudos.

A distribuição dos delineamentos sugere que a pesquisa nesta área é amplamente baseada em métodos experimentais, embora os estudos de caso também sejam um método importante. Entretanto, é relevante que os pesquisadores sejam claros sobre seus delineamentos para auxiliar na interpretação e na avaliação dos resultados.

Gráfico 9: Publicações por delineamento, considerando os estudos incluídos.



Fonte: Própria autora.

O Gráfico 10 mostra a distribuição geográfica dos estudos incluídos na revisão, que é bastante diversificada, com pesquisas provenientes de várias partes do mundo.

A China lidera em número de publicações, com 5 estudos incluídos, representando aproximadamente 29% do total. Isso é congruente com o papel proeminente da China na pesquisa e desenvolvimento tecnológico atual.

A Turquia tem o segundo maior número de estudos, com 3 trabalhos, o que representa cerca de 18% do total. Os Estados Unidos e a Finlândia contribuíram com 2 estudos cada, que representam aproximadamente 12% do total para cada um. Esses países são conhecidos por suas fortes tradições em pesquisa e desenvolvimento em ciências da computação e tecnologia.

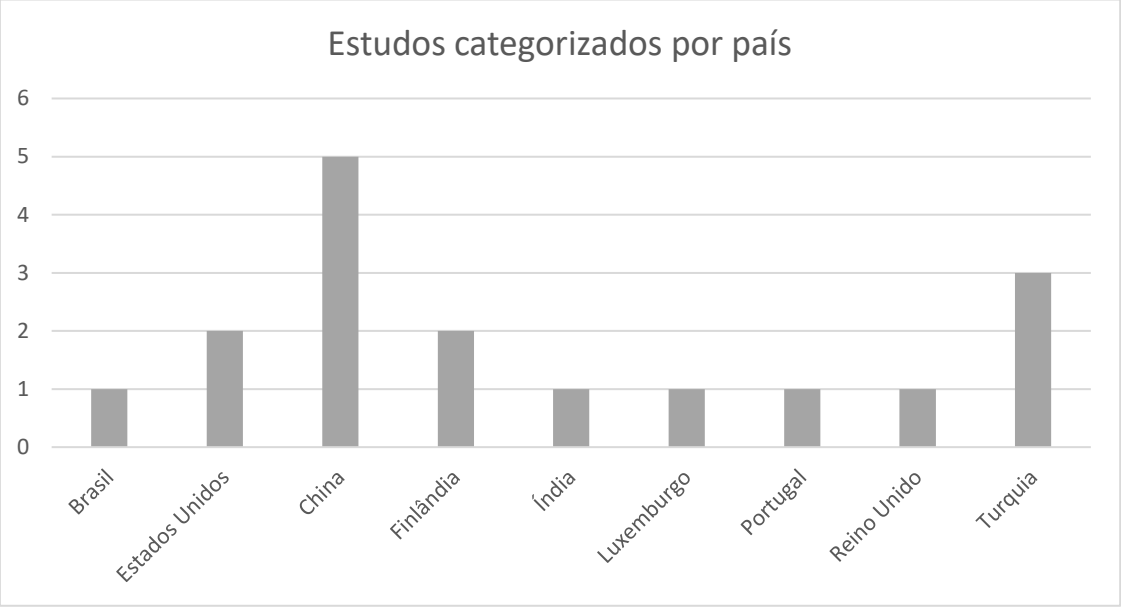
Brasil, Índia, Luxemburgo, Portugal e Reino Unido contribuíram cada um com um estudo, representando cerca de 6% do total de estudos selecionados para cada um desses países.

Esta distribuição geográfica diversificada sugere que o tópico de aplicação de Aprendizado de Máquina em testes de software é de interesse global, com pesquisas sendo realizadas em diferentes contextos e realidades culturais e tecnológicas. No entanto, é importante notar que essa distribuição pode ser influenciada por fatores

como o acesso à base de dados pesquisada, a língua de publicação, entre outros fatores que podem afetar a visibilidade e a acessibilidade da pesquisa realizada em diferentes países.

Por fim, embora a China apareça como o país com mais estudos publicados nesta área, isso não significa necessariamente que a qualidade ou a relevância da pesquisa seja maior do que em outros países. É crucial levar em consideração outros fatores, como exemplo o tamanho populacional do país, e, em relação ao estudo, a qualidade da pesquisa e a contribuição para o campo, ao avaliar o impacto da pesquisa de um país.

Gráfico 10: Publicações por país, considerando os estudos incluídos.



Fonte: Própria autora.

Q2. Quais atividades, artefatos, níveis, tipos e técnicas de testes de software estão envolvidos nas pesquisas realizadas?

Kahles *et al.* (2019) e Liu *et al.* (2022) exploraram em seus trabalhos a análise de causa raiz, abordagem utilizada para identificar causas fundamentais a partir de falhas encontradas. Além disso, a análise de causa raiz ajuda a determinar ações corretivas e melhorar a qualidade do software.

Em relação a interfaces gráficas, exceto trabalhos que faziam análise de imagem, Koroglu *et al.* (2018) e Oz *et al.* (2021) trataram em seus estudos a interação do usuário com a interface por meio da criação da ferramenta *QLearning-Based Exploration* (QBE) e pelo uso de aprendizado profundo generativo, respectivamente.

Assim, foi possível mapear fluxos principais de uso (geração de scripts de teste), aumentar cobertura de testes e, consequentemente, a identificação de defeitos.

Dentre dois dos estudos selecionados foram identificados pesquisadores, meio de publicação e país de origem em comum, o que aponta um provável grupo de pesquisa com interesse e publicações na área da aplicação de Aprendizado de Máquina a testes de software. Estes consideraram em seus estudos experimentais o registro de *bugs*, sendo possível a priorização da gravidade de defeitos (GUO *et al.*, 2019; LI *et al.*, 2022).

Uma abordagem proposta para o tema de estudos experimentais e de caso foi a geração de suítes de teste automática (RAJ; CHANDRASEKARAN, 2018; KOROGLU; SEM, 2020; Martins *et al.*, 2021). Assim, o Aprendizado de Máquina possibilita a produção de um conjunto de testes com um menor número de casos de teste, mas que mantém uma considerável cobertura do sistema.

Se tratando da execução dos testes, dois estudos possuem destaque: Tsimpourlas *et al.* (2021) ao tratar a classificação de execuções de teste (problema do oráculo) e Cabral *et al.* (2022) ao fazer um estudo de caso quanto a verificação em tempo de execução.

Além disso, foram identificados trabalhos que tratam a identificação de mutantes para realização de testes de mutação, inclusive com o objetivo de melhoria em testes regressivos, como exemplo a MuDelta, uma abordagem que identifica mutantes relevantes para *commits* (NAEEM *et al.* 2019; MA *et al.* 2021).

Ainda, a aplicação do Aprendizado de Máquina a fim de contribuir com testes de desempenho, metamórfico, exploratório e na geração de massa de dados de entrada foi investigada, resultando na criação de ferramentas como a PYSE, nome derivado de *Python and Symbolic Execution* (KOO *et al.* 2019; PORRES *et al.*, 2020).

Q3. Quais as vantagens e desvantagens citadas da aplicação de Aprendizado de Máquina em testes de software?

Em relação à utilização de algoritmos de Aprendizado de Máquina para apoiar a análise de causa raiz, Kahles *et al.* (2019) e Liu *et al.* (2022) relatam que o uso apresenta melhor desempenho e precisão (88,8% para o primeiro estudo), beneficiando engenheiros de teste. Pois, desta forma, eles podem verificar as causas

principais retornadas do algoritmo em vez de analisar manualmente todas as causas possíveis de todas as falhas encontradas. Além disso, nos estudos de Liu *et al.* (2022), a abordagem foi capaz de aproveitar o conhecimento de forma *cross-project.* Assim, a informação das falhas e da análise de causa raiz de um projeto pode contribuir para outro. Como desafio, Kahles *et al.* (2019) cita que a granularidade das categorias das falhas que foram estabelecidas pode ter sido alta, sendo necessários estudos que refine mais as falhas e causas raízes.

Koroglu *et al.* (2018) e Oz *et al.* (2022), em seus estudos experimentais encontraram resultados satisfatórios em relação a geração de scripts de teste e mapeamento de cenários de teste. Ainda, Koroglu *et al.* (2018) utilizou de validação com aleatoriedade em sua pesquisa, tornando seu estudo potencialmente generalizável por contar com muitos dados. No entanto, os fluxos alternativos não foram tão bem mapeados e considerados no estudo, sendo um direcionamento para próximas pesquisas.

No que diz respeito a análise de registros de defeitos de projetos de código aberto, foi verificado que classificadores obtém uma acurácia confiável ao prever a gravidade dos registros de defeitos em repositórios, sendo uma motivação para a expansão do escopo de tais estudos (GUO *et al.*, 2019; LI *et al.*, 2022). Ainda, Wang *et al.* (2021) em seu estudo de caso traz a contribuição de recomendação de defeitos para determinados desenvolvedores adequados para o reparo dos mesmos.

Como foi entendido, a redução do esforço nas atividades de teste de forma otimizada, reflete diretamente em um menor custo e na entrega de softwares melhores. Sendo assim, Raj e Chandrasekaran (2018) obtiveram como resultado dos seus estudos uma cobertura de testes satisfatório (cerca de 96%) por meio de melhorias na seleção de testes e na geração de suítes de teste. Já Martins *et al.* (2021), obteve resultados promissores que indicam benefícios ao integrar a ferramenta aos *pipelines* de integração contínua. Dessa forma, ao reduzir este conjunto, se obteve uma contribuição em *pipelines* de integração contínua, foram selecionados testes mais relevantes e o tempo médio de execução das suítes de teste foi reduzido em cerca de 10 minutos para um processo que demora cerca de 1 hora. Ainda, o método proposto por Raj e Chandrasekaran (2018), que utiliza o algoritmo *Neuroevolution of Augmenting Topologies* (NEAT), conseguiu performar em menos

tempo que outros métodos encontrados na literatura, gerando automaticamente suítes de teste para testes de caixa branca.

Seguindo esta linha de estudo, mas tratando de aplicações Android, Koroglu e Sen (2020) obtiveram como resultado de seu estudo experimental que o método proposto é eficaz e obteve maior desempenho que outras abordagens conhecidas ao tratar o problema de oráculo de teste, sendo notável uma melhora na eficácia e desempenho quando o desenvolvedor tem certa atuação.

Sobre a realização de verificação em tempo de execução, Cabral *et al.* (2022) propôs uma ferramenta personalizável que, dentre a análise para priorização de 1.170 violações em 110 projetos, obteve bons resultados. Assim, por meio da priorização de violações de verificação em tempo de execução é possível reduzir significativamente o esforço manual de inspeções. No entanto, como limitação se tem o número de violações analisadas e rotuladas disponíveis para o treinamento, algo intrínseco a abordagens baseadas em Aprendizado de Máquina.

Naeem *et al.* (2019) e Ma *et al.* (2021) realizaram estudos experimentais aplicando Aprendizado de Máquina a testes de mutação. Diante disso, no estudo de Ma *et al.* (2021) foram encontrados resultados com alta precisão e, matar tais mutantes encontrados levava a testes mais fortes ainda, que matam outros mutantes mais relevantes que quando identificados de forma aleatória. Assim, a capacidade calculada foi de revelação de falhas 27% maior que com estes aleatórios e foi possível incluir regressões nesta abordagem. Já Naeem et al (2019) analisou 10 programas e demonstrou que as técnicas preditivas de classificação de mutantes equivalentes podem prever resultados com boa previsão, mas, como desafio, quando se tem uma baixa qualidade nos conjuntos de teste se terá um baixo desempenho nos resultados.

Finalmente, a respeito da qualidade metodológica, observou-se muitos trabalhos empíricos, muitos estudos em que não se define o delineamento e/ou não se relata o percurso metodológico corretamente, assim como não se observou descrição de limitações dos trabalhos. Além disto, a literatura preconiza o uso de instrumentos e/ou ferramentas para auxiliar os pesquisadores na pesquisa inicial dos trabalhos, definição de critérios de exclusão e descarte de trabalhos duplicados (KITCHENHAM; CHARTERS, 2007).

Na realização de uma revisão sistemática de literatura, é possível reunir e sintetizar conhecimentos existentes referentes à uma área de pesquisa, neste caso, sobre a aplicação de Aprendizado de Máquina a testes de software. Porém, é notável que existem desafios que podem afetar os resultados obtidos. Assim, este trabalho apresenta forças e limitações.

Uma das limitações que podem ser citadas seria o viés dos pesquisadores, que pode afetar a validade e imparcialidade dos estudos. Outras questões a serem consideradas são: recursos disponibilizados, número de pesquisadores envolvidos, quantidade de estudos encontrados e o escopo reduzido desta pesquisa.

Dentre as forças podemos citar que este é um trabalho de revisão de um assunto em evidência na área da computação, que seguindo uma metodologia sólida conseguiu elencar bases relevantes, critérios de inclusão e exclusão pertinentes, realizando assim uma pesquisa abrangente e criteriosa. Desta forma, os resultados serão de grande proveito para pesquisadores e profissionais da área que procuram atualizações sobre o assunto.

Em relação a limitações relacionadas ao estudo no que tange os artigos analisados, notou-se que a falta de uma terminologia padronizada para se referir a técnicas de Aprendizado de Máquina e a testes de software nos estudos pode ter influenciado diretamente na seleção dos artigos para a amostra final.

E por fim, apesar dos desafios, uma revisão sistemática que aborda um tema em evidência na área da ciência da computação possui grande relevância (DURELI *et al.* 2019). Sendo assim, se torna parte da literatura como fonte de busca para pesquisadores e alunos da área.

4.4. Análise dos Trabalhos Excluídos

A análise dos estudos obtidos da base de dados World Scientific revelou que 35 estudos não atendiam aos critérios estabelecidos para a pesquisa em questão. As razões para a exclusão desses estudos são variadas e destacam a natureza diversificada das pesquisas conduzidas na área de engenharia de software. Para uma compreensão mais clara dos padrões emergentes e áreas que necessitam de mais atenção na pesquisa futura, as razões para a exclusão foram tabuladas e são apresentadas a seguir:

Tabela 5: Relação entre motivos de exclusão, número de trabalhos e percentual representado da base World Scientific.

Motivo	Número de trabalhos	Percentual
Tipo do estudo	10	28,57%
Não relacionado a testes	7	20%
Não utiliza AM	5	14,29%
Predição de defeitos	10	28,57%
Outros	3	8,57%

Fonte: Própria autora.

A razão para a exclusão dos estudos “Predição de Defeitos", sendo a mais frequente, abrangeu 28,57% do número total. Isso pode indicar um foco predominante nessa área, que embora seja relevante, não se alinha totalmente com o objetivo da pesquisa em análise. Igualmente, o percentual de 28,57% também representa estudos que foram removidos devido ao delineamento de estudo realizado.

Em seguida, com 20% das exclusões, estudos categorizados como "Não relacionado a testes" indicam que muitas pesquisas, mesmo dentro da engenharia de software, não se concentram especificamente nos testes. Já a "Não utiliza AM" representou 14,29% das exclusões, destacando que nem todos os trabalhos na área de engenharia de software utilizam essas técnicas em suas metodologias.

Finalmente, dentre a categoria “Outros” são representados trabalhos que trataram questões como, por exemplo, a previsão de manutenibilidade, estimativa de confiabilidade e testes de softwares de AM, que constituíram o restante das exclusões e representou 8,57%. Esses temas, embora relevantes para a área de engenharia de software, são tangenciais ao objetivo principal da pesquisa em análise.

Na revisão dos estudos obtidos da base de dados Wiley Online, foram identificados 117 estudos que não satisfaziam os critérios estabelecidos para esta investigação. Uma gama de motivos levou à exclusão desses estudos, que reflete a diversidade de abordagens e tópicos dentro da disciplina da engenharia de software. Assim, para facilitar a compreensão das áreas predominantes e aquelas que poderiam se beneficiar de mais investigações, os motivos para a exclusão foram tabulados. A tabela a seguir apresenta esses dados:

Tabela 6: Relação entre motivos de exclusão, número de trabalhos e percentual representado da base Wiley Online.

Motivo	Número de trabalhos	Percentual
Tipo do estudo	58	49,57%
Não relacionado a testes	12	10,26%
Não utiliza AM	17	14,53%
Predição de defeitos	22	18,80%
Outros	8	6,84%

Fonte: Própria autora.

A tabela evidencia que a maior categoria de exclusão é relacionada ao “Tipo do estudo, compreendendo 49,57% do total, e, dentre eles, destaca-se os estudos empíricos. Embora estudos empíricos sejam uma pedra angular na pesquisa de engenharia de software, muitos desses trabalhos não se alinham com o foco da nossa investigação. A segunda maior categoria é “Predição de defeitos”, com 18,80% dos estudos. Isso destaca que uma proporção substancial de trabalhos na área se concentra na previsão de defeitos, uma área de pesquisa essencial, mas que não está diretamente relacionada ao nosso objetivo.

A categoria "Não utiliza AM" representou 14,53% das exclusões, indicando que ainda existe uma grande quantidade de pesquisa na engenharia de software que não adota essas técnicas. Os estudos classificados como "Não relacionado a testes" foram a quarta maior categoria, com 10,26% das exclusões, mostrando que uma quantidade significativa de pesquisa na engenharia de software não se concentra diretamente nos testes, mesmo que sejam um componente essencial do desenvolvimento de software.

Outras razões para exclusão, que representam 6,84%, incluíram a natureza do trabalho como testes não ser o primordial do trabalho, e tópicos como "Método de Otimização" e "Software Embarcado".

Ao avançar para a análise dos estudos oriundos da base de dados Taylor & Francis, foram identificados 39 estudos que não atendiam aos critérios estabelecidos para a presente pesquisa, correspondendo a todos os estudos retornados na busca. Os motivos para a exclusão desses estudos são bastante variados, indicando a vasta gama de tópicos e abordagens na área da engenharia de software. A seguinte tabela resume esses dados, agrupando os estudos por motivo de exclusão, quantificando cada grupo e calculando a respectiva percentagem:

Tabela 7: Relação entre motivos de exclusão, número de trabalhos e percentual representado da base Taylor & Francis.

Motivo	Número de trabalhos	Percentual
Tipo do estudo	4	10,26%
Não relacionado a testes	24	61,54%
Não utiliza AM	4	10,26%
Predição de defeitos	1	2,56%
Outros	6	15,38%

Fonte: Própria autora.

A maioria esmagadora dos estudos, equivalente a 61,54%, foi excluída por não estar relacionada ao campo dos testes de software. Isso reflete a vastidão do domínio da engenharia de software, que abrange uma gama muito maior de tópicos além dos testes. Além disso, 10,26% dos estudos foram excluídos porque não aplicavam técnicas de Aprendizado de Máquina, o que denota uma área em que ainda há espaço para expansão e aplicação dessas técnicas.

A categoria "Tipo do estudo" causou a exclusão de 10.26% dos estudos. Isso se dá devido a nem todos os formatos ou abordagens de pesquisa serem relevantes para o objetivo específico deste trabalho, que é entender a aplicação de técnicas de AM em testes de software. Cerca de 15,38% dos estudos foram excluídos por vários outros motivos que não se encaixam nas categorias acima, reforçando a complexidade e diversidade da literatura na engenharia de software, e 2,56% por tratar de predição de defeitos.

A base de dados IEEE Xplore foi outra fonte de pesquisa considerada neste estudo. Após a avaliação, 262 estudos foram excluídos, tendo os motivos de exclusão classificados em diversas categorias. Estes também se dividem principalmente entre "Tipo do estudo", "Outros", "Predição de defeitos", "Não utiliza AM" e "Não relacionado a testes". A tabela a seguir proporciona um resumo desses dados, agrupando os estudos por motivo de exclusão e, em seguida, apresentando a quantidade e a porcentagem correspondente de cada grupo.

Tabela 8: Relação entre motivos de exclusão, número de trabalhos e percentual representado da base IEEE Xplore.

Motivo	Número de trabalhos	Percentual
Tipo do estudo	129	49,24%
Não relacionado a testes	14	5,34%
Não utiliza AM	3	1,15%
Predição de defeitos	56	21,37%
Outros	60	22,90%

Fonte: Própria autora.

O motivo mais comum para a exclusão foi o "Tipo do estudo", que representa quase metade (49,24%) de todos os estudos excluídos. Isso sugere que muitos estudos na base de dados IEEE Xplore adotam formatos ou abordagens que não são pertinentes ao objetivo específico deste trabalho. A categoria "Outros" e "Predição de defeitos" também representam uma parcela significativa das exclusões, com 22,90% e 21,37% respectivamente. O primeiro destes destaca a diversidade de tópicos na literatura existente, enquanto o segundo demonstra uma quantidade considerável de trabalhos que, apesar de se voltarem para um tópico próximo do nosso objeto de estudo, não se dedicam exatamente ao uso de técnicas de Aprendizado de Máquina em testes de software.

Por fim, 1,15% dos estudos não tratavam a aplicação de Aprendizado de Máquina e 5,34% não estavam relacionados a testes de software. Esses percentuais, embora baixos, ressaltam a importância de uma seleção de estudos bem fundamentada para garantir que os resultados da pesquisa sejam relevantes e precisos.

Diante disso, essas descobertas indicam a necessidade de estudos mais específicos sobre a aplicação de técnicas de Aprendizado de Máquina em testes de software, considerando a grande quantidade de estudos que tiveram que ser excluídos por não se enquadrarem nesse escopo. Além disso, a prevalência de estudos excluídos devido ao tipo de estudo sugere que mais pesquisas podem ser necessárias para entender melhor como diferentes formatos e abordagens de pesquisa podem contribuir para o campo.

A análise e discussão dos dados apresentados nesta pesquisa evidenciam que o campo de aplicação do Aprendizado de Máquina aos testes de software é um

domínio de pesquisa emergente e crescente, tanto em termos de interesse quanto de contribuições científicas. A tendência de crescimento observada nas publicações por ano e a diversidade geográfica das contribuições enfatizam a relevância e o alcance global deste campo de estudo. Além disso, a análise das bases de dados revela a importância de um processo de revisão criterioso na identificação e seleção de trabalhos relevantes.

Apesar de a base IEEE Xplore ter retornado o maior número de estudos, não necessariamente proporcionou a maior quantidade de estudos relevantes para a revisão. Assim, a escolha e avaliação das bases de dados representam etapas fundamentais no processo de revisão de literatura.

A categorização dos estudos por delineamento e a análise dos motivos de exclusão também oferecem insights valiosos sobre a natureza da pesquisa realizada neste campo. A predominância de estudos experimentais sugere uma ênfase na aplicação prática das técnicas de Aprendizado de Máquina nos testes de software, enquanto a exclusão de muitos estudos devido à previsão de defeitos indica a necessidade de delimitar claramente o escopo da revisão.

Em suma, esta pesquisa oferece uma visão abrangente do campo de aplicação do Aprendizado de Máquina aos testes de software, destacando as tendências emergentes, os desafios enfrentados e as oportunidades para futuras pesquisas. Ao identificar e sintetizar os principais estudos realizados, este trabalho contribui para a compreensão e avanço do conhecimento neste campo promissor da Engenharia de Software.

5. CONCLUSÃO

A crescente atenção dada às pesquisas sobre a aplicação de aprendizado de máquina aos testes de software, conforme evidenciado nos últimos anos, demonstra a importância e a relevância do campo. Entre 2018 e 2022, a produção de estudos em revistas e conferências de renome em diferentes países reflete a natureza global e diversificada da pesquisa, englobando uma gama de atividades, artefatos, níveis, tipos e técnicas de testes de software.

Nossa análise sistemática da literatura revelou avanços importantes na aplicação de técnicas de aprendizado de máquina para aprimorar a eficiência e precisão de várias atividades de teste de software. Da análise de causa raiz à geração automática de suítes de teste, a utilização de aprendizado de máquina tem se mostrado promissora, contribuindo para a entrega de software de melhor qualidade, reduzindo custos e aumentando a eficiência dos processos de teste.

No entanto, a pesquisa também destacou desafios significativos que precisam ser enfrentados para expandir ainda mais o uso de aprendizado de máquina em testes de software. A necessidade de refinar as categorias de falhas e causas raízes, considerar os fluxos alternativos nos estudos de interação com interfaces gráficas, a disponibilidade de conjuntos de dados adequados para o treinamento de algoritmos e garantir a qualidade metodológica nas pesquisas são questões críticas que necessitam de mais investigação.

A análise dos estudos incluídos e excluídos e o exame do número de publicações ao longo do tempo e por base de dados permitiram uma visão aprofundada da evolução e do estado atual do campo. Com base nesta análise, fica evidente que, apesar dos avanços significativos, ainda há espaço para estudos mais robustos e detalhados na área.

A aplicação do aprendizado de máquina aos testes de software é um campo de pesquisa promissor que tem demonstrado resultados significativos, mas ainda apresenta desafios. Esta revisão da literatura fornece um panorama das tendências atuais, das vantagens e dos desafios, e pode ser utilizada como referência para outros trabalhos na exploração e aprimoramento desta área de pesquisa.

5.1. Trabalhos Futuros

Durante a leitura dos estudos, identificou-se que a falta de padronização da terminologia dos trabalhos das bases dificultava o entendimento das metodologias utilizadas e algoritmos aplicados. Com isso, podem ser obtidos benefícios com a elaboração de novos estudos, de preferência se estes dispuserem de uma descrição metodológica mais bem definida, contribuindo no entendimento do trabalho, já que serão sempre necessárias outras atualizações do estado da arte sobre este e outros assuntos.

Além disso, novos trabalhos com metodologia e validação bem descritas auxiliam em um direcionamento para a padronização de termos nas áreas de Inteligência Artificial e Testes de Software, além da identificação de lacunas.

Ademais, propõe-se que a metodologia descrita neste trabalho, incluindo questões de busca, critérios de inclusão e exclusão, possam ser replicados por grupos maiores de pesquisadores, utilizando maior número de bases e termos, alcançando assim dados mais robustos e passíveis de generalização.

Assim, a fim de complementar este estudo, podem ser realizadas pesquisas que analisem os métodos de Aprendizado de Máquina utilizados, com enfoque nas técnicas de inteligência artificial.

REFERÊNCIAS

AL-SAHAF, H. *et al.* A Survey on Evolutionary Machine Learning. **Journal of the Royal Society of New Zealand**, v. 49, n. 2, p. 205–228, 3 abr. 2019.

AMMANN, P.; OFFUTT, J. **Introduction to Software Testing**. Edition 2 ed. Cambridge, United Kingdom; New York, NY, USA: Cambridge University Press, 2017.

ANDRADE, J. *et al.* An Architectural Model for Software Testing Lesson Learned Systems. **Information and Software Technology**, v. 55, n. 1, p. 18–34, 1 jan. 2013.

BELAY, E. **Challenges of Large-Scale Software Testing and the Role of Quality Characteristics - An Empirical Study of Software Testing**. [S.l: s.n.], 2020. Disponível em: https://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-93124.

BERTOLINO, A. Software Testing Research: Achievements, challenges, dreams. **FoSE 2007: Future of Software Engineering**, p. 85–103, 2007.

BIRAN, O.; COTTON, C. Explanation and justification in machine learning: A survey. In: **IJCAI-17 Workshop on Explainable AI (XAI).** p. 8-13. 2017.

BOMBIRI, O.; PODA, P.; OUEDRAOGO, T. F. Application of Machine Learning in Software Quality: A Mini-review. **2023 IEEE Multi-conference on Natural and Engineering Sciences for Sahel's Sustainable Development (MNE3SD)**. IEEE, 2023. p. 1-7.

CABRAL, L. *et al.* RVprio: A Tool for Prioritizing Runtime Verification Violations. **Software Testing, Verification and Reliability**, v. 32, n. 5, p. e1813, 2022.

CARUSO, A. L. M.; CAVALHEIRO, S. A.C.; Integração entre Pensamento Computacional e Inteligência Artificial: uma Revisão Sistemática de Literatura. **Anais do XXXII Simpósio Brasileiro de Informática na Educação (SBIE 2021)**, p. 1051–1062, 2021.

CHAUHAN, P.; VEDPAL, P. G. Prediction of Tumors in Brain MRI Images using Grasshopper Optimization Algorithm with Image Segmentation and Enhancement Approach. **Design Engineering**, p. 12369-12381, 2021.

COPPIN, B. **Inteligência Artificial.** Grupo GEN, 2010. E-book. ISBN 978-85-216-2936-8.

DINELLA, E. *et al.* Toga: A neural method for test oracle generation. In: **Proceedings of the 44th International Conference on Software Engineering**. 2022. p. 2130-2141.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. Introdução ao teste de software (2 edição ed.). 2016.

DURELLI, V. H.S. *et al.* Machine learning applied to software testing: A systematic mapping study. **IEEE Transactions on Reliability**, v. 68, n. 3, p. 1189–1212, 2019.

EVERETT, G. D.; MCLEOD JR, R. **Software testing: testing across the entire** software **development life cycle**. John Wiley & Sons, 2007.

FACELI, K.; LORENA, A. C.; GAMA, J *et al.*; Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina. Grupo GEN, 2021. E-book. ISBN 9788521637509.

FELBINGER, H.; WOTAWA, F.; NICA, M. Adapting unit tests by generating combinatorial test data. **2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)**. IEEE, 2018. p. 352-355.

GELPERIN, D.; HETZEL, B. The growth of software testing. **Communications of the ACM**, v. 31, n. 6, p. 687-695, 1988.

GUO, S. *et al.* Identify severity bug report with distribution imbalance by CR-SMOTE and ELM. **International Journal of Software Engineering and Knowledge Engineering**, v. 29, n. 02, p. 139-175, 2019.

GUPTA, V. *et al.* An Overview of Supervised Machine Learning Algorithm. In: **2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)**. IEEE, 2022. p. 87-92.

HAENLEIN, M.; Uma Breve História da Inteligência artificial : Sobre o Passado , Presente e Futuro do Artificial Inteligência. 2019.

HAEUPTLE, K.; **Shared Language for Talking About Test Strategy**. 6 de dezembro de 2021. Disponível em: https://blogs.sap.com/2021/12/06/shared-language-for-talking-about-test-strategy/.

HARDIN, B.; KANEWALA, U. Using semi-supervised learning for predicting metamorphic relations. In: **Proceedings of the 3rd International Workshop on Metamorphic Testing**. 2018. p. 14-17.

HOURANI, H.; HAMMAD, A.; LAFI, M. The impact of artificial intelligence on software testing. In: **2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)**. IEEE, 2019. p. 565-570.

IEEE COMPUTER SOCIETY. **IEEE Standard Glossary of Software Engineering Terminology**. New York: [s.n.], 1983.

IEEE COMPUTER SOCIETY. **Guide to the Software Engineering Body of Knowledge SWEBOK. A Project of the IEEE Computer Society**. [S.l: s.n.], 2014. Disponível em: <https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOKv3.pdf%0Ahttps://www.computer.org/ieeecs-swebokdelivery-portlet/swebok/SWEBOKv3.pdf?token=0xoxEZFQ48HsyMDHDaTAd2EmcJV32I22>.

ISO/IEC. **Systems and Software Engineering - Vocabulary**. ISO/IEC Standard 24765:2010(E). Geneva, Switzerland: International Organization for Standardization, 2010.

JAMIL *et al.* Software Testing Techniques: A Literature Review. In: **2016 6th International Conference on Information and Communication Technology for The Muslim World** (ICT4M), Jakarta, Indonesia, 2016, pp. 177-182, doi: 10.1109/ICT4M.2016.045.

JIANG, Y. G. *et al.* Exploiting feature and class relationships in video categorization with regularized deep neural networks. **IEEE transactions on pattern analysis and machine intelligence**, v. 40, n. 2, p. 352-364, 2017.

JOVANOVIĆ, I. Software testing methods and techniques. **The IPSI BgD Transactions on Internet Research**, v. 30, 2006.

KAHLES, J. *et al.* Automating root cause analysis via machine learning in agile software testing environments. In: **2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)**. IEEE, 2019. p. 379-390.

KITCHENHAM, B. A.; CHARTERS, S. M. Guidelines for performing systematic literature reviews in software engineering. **Technical report, Ver. 2.3 EBSE Technical Report. EBSE**, v. 1, p. 1–54, 2007.

KOO, J. *et al.* Pyse: Automatic worst-case test generation by reinforcement learning. In: **2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)**. IEEE, 2019. p. 136-147.

KOROGLU, Y. *et al.* QBE: QLearning-based exploration of android applications. In: **2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)**. IEEE, 2018. p. 105-115.

KOROGLU, Y.; SEN, A. Functional test generation from UI test scenarios using reinforcement learning for android applications. **Software Testing, Verification and Reliability**, v. 31, n. 3, p. e1752, 2021.

KULIN, M. *et al.* A survey on machine learning-based performance improvement of wireless networks: PHY, MAC and network layer. **Electronics**, v. 10, n. 3, p. 318, 2021.

KUMAR, G A. **A review on challenges in** software **testing. Journal of Information and Computational Science** v. 9, n. 6, p. 166–174, 2019.

LIMA, R.; CRUZ, A. M. R.; RIBEIRO, J. Artificial intelligence applied to software testing: A literature review. In: **2020 15th Iberian Conference on Information Systems and Technologies (CISTI)**. IEEE, 2020. p. 1-6.

LI, H. *et al.* Identifying High-impact Bug Reports with Imbalance Distribution by Instance Fuzzy Entropy. **International Journal of Software Engineering and Knowledge Engineering**, v. 32, n. 09, p. 1389-1417, 2022.

LIU, X. *et al.* An unsupervised cross project model for crashing fault residence identification. **IET Software**, v. 16, n. 6, p. 630-646, 2022.

MA, W. *et al.* Mudelta: Delta-oriented mutation testing at commit time. In: **2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)**. IEEE, 2021. p. 897-909.

MACIEL, C. P. C; SOUZA, É. F. De; V., Nandamudi L.; An Empirical Study on the Knowledge Management Practice in Software Testing. **XXI Ibero-American Conference on Software Engineering (CIBSE) - Experimental Software Engineering (ESELAW) Track**, p. 1–14, 2018.

MAHADEVKAR, S. V. *et al.* A review on machine learning styles in computer vision-techniques and future directions. **IEEE Access**, 2022.

MARIJAN, D.; GOTLIEB, A.; Software Testing for Machine Learning. **Association for the Advancement of Artificial Intelligence**, p. 7, 2020.

MARTINS, R. *et al.* Supervised learning for test suit selection in continuous integration. In: **2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)**. IEEE, 2021. p. 239-246.

MONARD, M. C.; BARANAUSKAS, José Augusto. Conceitos sobre Aprendizado de Máquina. **Sistemas inteligentes: fundamentos e aplicações**, p. 89–114, 2003.

MURPHY, C.; KAISER, G. E.; ARIAS, M. An approach to software testing of machine learning applications. 2007.

NAEEM, M. R.*et al.* A machine learning approach for classification of equivalent mutants. **Journal of Software: Evolution and Process**, v. 32, n. 5, p. e2238, 2020.

NAHEED, M.; NADEEM, A.; ZAMAN, Q. U. A Requirement Based Approach to Test Case Prioritization for Regression Testing. In: **2022 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST)**. IEEE, 2022. p. 358-363.

OZ, M. *et al.* On the use of generative deep learning approaches for generating hidden test scripts. **International Journal of Software Engineering and Knowledge Engineering**, v. 31, n. 10, p. 1447-1468, 2021.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L.; Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, v. 64, p. 1–18, 2015. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2015.03.007>.

PFLEEGER, S. L.; Engenharia de Software: Teoria e Prática, 2ª Edição, São Paulo: Prentice Hall, 2004.

PIERCE, P. Software verification and validation. In: **IEEE Technical Applications Conference. Northcon/96. Conference Record**. IEEE, 1996. p. 265-268.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software. Uma Abordagem Profissional - 8ª Edição**. 8ª Ed. ed. [S.l: s.n.], 2016.

PORRES, I. *et al.* Automatic exploratory performance testing using a discriminator neural network. In: **2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (icstw).** IEEE, 2020. p. 105-113.

RAJ, H. P.; CHANDRASEKARAN, K.; NEAT Algorithm for Testsuite generation in Automated Software Testing. In: **2018 IEEE Symposium Series on Computational Intelligence (SSCI)**. IEEE, 2018. p. 2361-2368.

SINGH, S. K.; SINGH, A. Software testing. **Vandana Publications**, 2012.

SOMMERVILLE, I.; **Engenharia de** software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, É. F; FALBO, R. A.; VIJAYKUMAR, N. L. Using the findings of a mapping study to conduct a research project : a case in knowledge management in software testing. In: **2015 41st Euromicro Conference on Software Engineering and Advanced Applications.** IEEE, 2015. p. 208-215.

TIMO, O. N.; PETRENKO, A.; RAMESH, S. Using imprecise test oracles modelled by FSM. In: **2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW).** IEEE, 2019. p. 32-39.

TSIMPOURLAS, Foivos *et al.* Embedding and classifying test execution traces using neural networks. **IET Software**, v. 16, n. 3, p. 301-316, 2022.

UDDIN, A.; ANAND, A.; Importance of Software Testing in the Process of Software Development. **IJSRD - International Journal for Scientific Research & Development**, v. 6, n. 12, p. 141–145, 2019. Disponível em: <https://www.researchgate.net/publication/331223692_Importance_of_Software_Testing_in_the_Process_of_Software_Development>.

UKAI, H. *et al.* Reduce test cost by reusing test oracles through combinatorial join. In: **2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)**. IEEE, 2019. p. 260-263.

WANG, S. *et al.* Toward more accurate developer recommendation via inference of development activities from interaction with bug repair process. **Journal of Software: Evolution and Process**, v. 33, n. 5, p. e2341, 2021.

ZHANG, J. M. *et al.* Machine Learning Testing: Survey, Landscapes and Horizons. **IEEE Transactions on Software Engineering**, v. X, n. X, p. 1–1, 2020.

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continua)

ESTUDO	S/N	MOTIVO
An Empirical Study on Software Failure Classification with Multi-label and Problem-Transformation Techniques	N	Tipo do estudo
Localizing Faults in Cloud Systems	N	Tipo do estudo
Virtualized-Fault Injection Testing: A Machine Learning Approach	N	Outros
How high will it be? Using machine learning models to predict branch coverage in automated testing	N	Tipo do estudo
A classifier-based test oracle for embedded software	N	Outros
Assessing Technical Debt in Automated Tests with CodeScene	N	Outros
A Test Architecture for Machine Learning Product	N	Outros
A Survey of Software Quality for Machine Learning Applications	N	Tipo do estudo
An AI Software Test Method Based on Scene Deductive Approach	N	Outros
Towards a Bayesian Network Model for Predicting Flaky Automated Tests	N	Tipo do estudo
Metamorphic Testing 20 Years Later: A Hands-on Introduction	N	Tipo do estudo
A Replication Study: Just-in-Time Defect Prediction with Ensemble Learning	N	Predição de defeitos
Testing Vision-Based Control Systems Using Learnable Evolutionary Algorithms	N	Não aplica AM
Predictive Analytics for Software Testing	N	Predição de defeitos
Generating Test Input with Deep Reinforcement Learning	N	Tipo do estudo
Enhancing Supervised Classifications with Metamorphic Relations	N	Não relacionado a testes
Mining User's Browsing History to Personalize Web Search	N	Não relacionado a testes
Revisiting AI and Testing Methods to Infer FSM Models of Black-Box Systems	N	Tipo do estudo
Software Testing as a Problem of Machine Learning: Towards a Foundation on Computational Learning Theory	N	Tipo do estudo
Testing Service Oriented Architectures Using Stateful Service Virtualization via Machine Learning	N	Predição de defeitos
Machine Learning Augmented Fuzzing	N	Não relacionado a testes
Risk-Based Testing of Self-Adaptive Systems Using Run-Time Predictions	N	Predição de defeitos
A Dynamic ReLU on Neural Network	N	Não relacionado a testes

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

An Improved Approach to Software Defect Prediction using a Hybrid Machine Learning Model	N	Predição de defeitos
DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems	N	Outros
Predictive Mutation Testing	N	Predição de defeitos
Studying Aging-Related Bug Prediction Using Cross-Project Models	N	Predição de defeitos
Machine Learning Applied to Software Testing: A Systematic Mapping Study	N	Tipo do estudo
An Artificial Intelligence Approach to EDA Software Testing: Application to Net Delay Algorithms in FPGAs	N	Tipo do estudo
Defect Prediction based on Machine Learning using System Test Parameters	N	Predição de defeitos
Using Machine Learning to Prioritize Automated Testing in an Agile Environment	N	Predição de defeitos
Invited Paper: What is AI Software Testing? and Why	N	Tipo do estudo
The Impact of Artificial Intelligence on Software Testing	N	Tipo do estudo
Metamorphic Testing for Quality Assurance of Protein Function Prediction Tools	N	Outros
AI for Testing Today and Tomorrow: Industry Perspectives	N	Tipo do estudo
Using Machine Learning to Classify Test Outcomes	N	Tipo do estudo
Challenges of Testing Machine Learning Based Systems	N	Tipo do estudo
Learning to Optimize the Alloy Analyzer	N	Tipo do estudo
An Extensive Study on Cross-Project Predictive Mutation Testing	N	Tipo do estudo
SmokeOut: An Approach for Testing Clustering Implementations	N	Tipo do estudo
An Empirical Assessment of Machine Learning Approaches for Triaging Reports of a Java Static Analysis Tool	N	Tipo do estudo
An RBM Anomaly Detector for the Cloud	N	Tipo do estudo
Classifying False Positive Static Checker Alarms in Continuous Integration Using Convolutional Neural Networks	N	Tipo do estudo
A Model-Based Approach to Generate Dynamic Synthetic Test Data	N	Tipo do estudo
Directing a Search Towards Execution Properties with a Learned Fitness Function	N	Tipo do estudo
Testing Machine Learning Algorithms for Balanced Data Usage	N	Tipo do estudo
Learning to Restrict Test Range for Compiler Test	N	Tipo do estudo

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

Machine Learning to Guide Performance Testing: An Autonomous Test Framework	N	Tipo do estudo
Coverage-Guided Learning-Assisted Grammar-Based Fuzzing	N	Não relacionado a testes
IRRT: An Automated Software Requirements Traceability Tool based on Information Retrieval Model	N	Não relacionado a testes
Improving Performance in Software Defect Prediction Using Variational Autoencoder	N	Predição de defeitos
Think Your Artificial Intelligence Software Is Fair? Think Again	N	Tipo do estudo
Experimental Validation of Inheritance Metrics' Impact on Software Fault Prediction	N	Predição de defeitos
A Comprehensive Investigation of Modern Test Suite Optimization Trends, Tools and Techniques	N	Tipo do estudo
An Extended Abstract of "Metamorphic Testing: Testing the Untestable"	N	Tipo do estudo
Effectiveness of Weighted Neural Network on Accuracy of Software Fault Localization	N	Outros
Implementation of Machine Learning Techniques in Software Reliability: A framework	N	Tipo do estudo
Testing and Quality Validation for AI Software–Perspectives, Issues, and Practices	N	Tipo do estudo
SBST in the Age of Machine Learning Systems - Challenges Ahead	N	Tipo do estudo
One-Size-Fits-None? Improving Test Generation Using Context-Optimized Fitness Functions	N	Tipo do estudo
Testing Cyber-Physical Systems via Evolutionary Algorithms and Machine Learning	N	Outros
Software Metrics Artifacts Making Web Quality Measurable	N	Tipo do estudo
Testing Scientific Software with Invariant Relations: A Case Study	N	Tipo do estudo
Raising the Quality of Bug Reports by Predicting Software Defect Indicators	N	Predição de defeitos
Evaluation of Judicial Imprisonment Term Prediction Model Based on Text Mutation	N	Outros
ACT Testbot and 4S Quality Metrics in XAAS Framework	N	Outros
Control strategy of a Multiple Hearth Furnace enhanced by machine learning algorithms	N	Outros
An Empirical Framework for Code Smell Prediction using Extreme Learning Machine	N	Predição de defeitos
Trade-off analysis using synthetic training data for neural networks in the automotive development process	N	Outros
Weight and Cluster Based Test Case Prioritization Technique	N	Outros
Machine Learning Model for Smart Contracts Security Analysis	N	Não relacionado a testes

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

Data generators: a short survey of techniques and use cases with focus on testing	N	Tipo do estudo
Quality Model for Testing Augmented Reality Applications	N	Tipo do estudo
Software Framework for Data Fault Injection to Test Machine Learning Systems	N	Outros
An Empirical Comparison of Two Different Strategies to Automated Fault Detection: Machine Learning Versus Dynamic Analysis	N	Tipo do estudo
Predicting Cascading Failures in Power Grids using Machine Learning Algorithms	N	Predição de defeitos
A Cost-Sensitive Approach to Enhance the use of ML Classifiers in Software Testing Efforts	N	Predição de defeitos
A Novel Software Defect Prediction Method Based on Isolation Forest	N	Predição de defeitos
Hyper-Parameter Optimization of Classifiers, Using an Artificial Immune Network and Its Application to Software Bug Prediction	N	Predição de defeitos
Empirical Study of Software Test Suite Evolution	N	Tipo do estudo
Reinforcement Learning Guided Symbolic Execution	N	Tipo do estudo
Software Engineering for Data Analytics	N	Outros
Use of Silicon-based Sensors for System Reliability Prediction	N	Outros
Artificial Intelligence Applied to Software Testing: A Literature Review	N	Tipo do estudo
Software and System Reliability Engineering for Autonomous Systems Incorporating Machine Learning	N	Tipo do estudo
Predicting Survived and Killed Mutants	N	Predição de defeitos
Towards a Deep Learning Model for Vulnerability Detection on Web Application Variants	N	Tipo do estudo
Test Data Generation for False Data Injection Attack Testing in Air Traffic Surveillance	N	Outros
SiMut: Exploring Program Similarity to Support the Cost Reduction of Mutation Testing	N	Outros
A Preliminary Investigation into Using Machine Learning Algorithms to Identify Minimal and Equivalent Mutants	N	Tipo do estudo
On Using k-means Clustering for Test Suite Reduction	N	Tipo do estudo
Selective Regression Testing based on Big Data: Comparing Feature Extraction Techniques	N	Tipo do estudo
Combinatorial Methods for Explainable AI	N	Tipo do estudo
Test Automation Improvement Model - TAIM 2.0	N	Tipo do estudo

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

Flexible FPGA Based Digital IC Test Development Education Laboratory Design and Application	N	Outros
Poster: Performance Testing Driven by Reinforcement Learning	N	Tipo do estudo
Can We Predict the Quality of Spectrum-based Fault Localization?	N	Predição de defeitos
Prioritizing Runtime Verification Violations	N	Predição de defeitos
Callisto: Entropy-based Test Generation and Data Quality Assessment for Machine Learning Systems	N	Tipo do estudo
Tool Support for Refactoring Manual Tests	N	Tipo do estudo
An Empirical Evaluation of Mutation Operators for Deep Learning Systems	N	Tipo do estudo
Attribute-based Granular Evaluation for Performance of Machine Learning Models	N	Tipo do estudo
Testing acoustic scene classifiers using Metamorphic Relations	N	Outros
Automated identification of metamorphic test scenarios for an ocean-modeling application	N	Outros
Neural Network Classification for Improving Continuous Regression Testing	N	Predição de defeitos
Effectiveness of dataset reduction in testing machine learning algorithms	N	Outros
Automatic Bug Triage in Software Systems Using Graph Neighborhood Relations for Feature Augmentation	N	Predição de defeitos
Attribute Rule performance in Data Mining for Software Deformity Prophecy Datasets Models	N	Outros
Feature Extraction to Heterogeneous Cross Project Defect Prediction	N	Predição de defeitos
A Comprehensive Decomposition towards the Facets of Quality in IoT	N	Outros
Trends in Software Engineering Processes using Deep Learning: A Systematic Literature Review	N	Tipo do estudo
SafeOps: A Concept of Continuous Safety	N	Outros
Machine Learning based Learning Disability Detection using LMS	N	Outros
Software Defect Prediction Using Neural Network Based SMOTE	N	Predição de defeitos
Using Machine Learning To Secure IOT Systems	N	Outros
Avionics Testing with Artificial Intelligence Support	N	Outros
Empirical evaluation of the active learning strategies on software defects prediction	N	Predição de defeitos
A Lightweight Fault Localization Approach based on XGBoost	N	Predição de defeitos

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

A Survey of the Use of Test Report in Crowdsourced Testing	N	Tipo do estudo
Code Coverage Similarity Measurement Using Machine Learning for Test Cases Minimization	N	Outros
OpenCGRA: An Open-Source Unified Framework for Modeling, Testing, and Evaluating CGRAs	N	Tipo do estudo
Software Visualization and Deep Transfer Learning for Effective Software Defect Prediction	N	Predição de defeitos
MockSniffer: Characterizing and Recommending Mocking Decisions for Unit Tests	N	Tipo do estudo
XCSF with Experience Replay for Automatic Test Case Prioritization	N	Tipo do estudo
Cognitive Computing in Software evaluation	N	Tipo do estudo
Data-Agnostic Local Neighborhood Generation	N	Tipo do estudo
Tackling Feature Selection Problems with Genetic Algorithms in Software Defect Prediction for Optimization	N	Predição de defeitos
The Application of Big Data Analytics in Online Foreign Language Learning among College Students : Empirical Research on Monitoring the Learning Outcomes and Predicting Final Grades	N	Tipo do estudo
Software Defect Prediction and Localization with Attention-Based Models and Ensemble Learning	N	Predição de defeitos
Automatically Predicting Bug Severity Early in the Development Process	N	Predição de defeitos
Manifold for Machine Learning Assurance	N	Tipo do estudo
A comparative recapitulating study of recent sentiment analysis approaches	N	Tipo do estudo
Collective Risk Minimization via a Bayesian Model for Statistical Software Testing	N	Tipo do estudo
Lightweight Assessment of Test-Case Effectiveness Using Source-Code-Quality Indicators	N	Outros
Grammar Based Directed Testing of Machine Learning Systems	N	Tipo do estudo
Defect Prediction With Semantics and Context Features of Codes Based on Graph Representation Learning	N	Predição de defeitos
Generative Adversarial Networks-Based Imbalance Learning in Software Aging-Related Bug Prediction	N	Predição de defeitos
Blockchain and Federated Learning for Collaborative Intrusion Detection in Vehicular Edge Computing	N	Outros
Learning to Predict Software Testability	N	Outros

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

Automatically Authoring Regression Tests for Machine-Learning Based Systems	N	Outros
Evolving Software to be ML-Driven Utilizing Real-World A/B Testing: Experiences, Insights, Challenges	N	Tipo do estudo
What Are We Really Testing in Mutation Testing for Machine Learning? A Critical Reflection	N	Tipo do estudo
More Effective Test Case Generation with Multiple Tribes of AI	N	Tipo do estudo
An Empirical Study of Flaky Tests in Python	N	Tipo do estudo
Keynote Talk - Testing Machine Learning-Enabled Systems	N	Tipo do estudo
Boosting Exploratory Testing of Industrial Automation Systems with AI	N	Outros
Digital Twin-based Anomaly Detection in Cyber-physical Systems	N	Outros
Summary of: Adaptive Metamorphic Testing with Contextual Bandits	N	Tipo do estudo
Poster: Repair Cross Browser Layout Issues by Combining Learning and Search-based technique	N	Tipo do estudo
A Large-scale Study on API Misuses in the Wild	N	Tipo do estudo
Large Scale Evaluation of Natural Language Processing Based Test-to-Code Traceability Approaches	N	Tipo do estudo
Test Automation with Grad-CAM Heatmaps - A Future Pipe Segment in MLOps for Vision AI?	N	Tipo do estudo
Prioritized Test Generation Guided by Software Fault Prediction	N	Predição de defeitos
Boosted Exploratory Test Architecture: Coaching Test Engineers with Word Similarity	N	Tipo do estudo
A Combinatorial Approach to Explaining Image Classifiers	N	Tipo do estudo
Active Machine Learning to Test Autonomous Driving	N	Outros
Automatic Equivalent Mutants Classification Using Abstract Syntax Tree Neural Networks	N	Tipo do estudo
Combinatorial Testing Metrics for Machine Learning	N	Tipo do estudo
SYSMODIS: A Systematic Model Discovery Approach	N	Tipo do estudo
Empirical Analysis of Artificial Immune System Algorithms for Aging Related Bug Prediction	N	Predição de defeitos
A Novel approach of GUI Mapping with image based widget detection and classification	N	Outros
Automated User Experience Testing through Multi-Dimensional Performance Impact Analysis	N	Outros
Automated Performance Testing Based on Active Deep Learning	N	Predição de defeitos

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

Test Suites as a Source of Training Data for Static Analysis Alert Classifiers	N	Predição de defeitos
Using Machine Learning to Build Test Oracles: an Industrial Case Study on Elevators Dispatching Algorithms	N	Outros
Learning Off-By-One Mistakes: An Empirical Study	N	Tipo do estudo
Engineering an Intelligent Essay Scoring and Feedback System: An Experience Report	N	Tipo do estudo
Towards Automated Metamorphic Test Identification for Ocean System Models	N	Outros
Artificial Intelligence For Software Testing-Perspectives And Practices	N	Tipo do estudo
Metamorphosis Relationship Generation Based on Fixed Memory Step Gradient Descent Method with Noise	N	Outros
A Literature Review of Using Machine Learning in Software Development Life Cycle Stages	N	Tipo do estudo
Efficient and Effective Generation of Test Cases for Pedestrian Detection - Search-based Software Testing of Baidu Apollo in SVL	N	Outros
Fast Screen Content Coding in HEVC Using Machine Learning	N	Tipo do estudo
Research and practice of remote monitoring system for road test of intelligent connected vehicle	N	Outros
An Improved Adaptive Sliding Mode Control of a Robotic Manipulator with Hysteresis Nonlinearity Based on Adaptive Speed Factor	N	Outros
Trend Application of Machine Learning in Test Case Prioritization: A Review on Techniques	N	Tipo do estudo
Research on vehicle network intrusion detection technology based on dynamic data set	N	Outros
Initial Data Corruption Impact on Machine Learning Models' Performance in Energy Consumption Forecast	N	Outros
Averaged Errors as a Risk Factor for Intelligent Forecasting Systems Operation in the Power Industry	N	Outros
Advancement of automatic generation control in power systems with large share of variable energy resources	N	Tipo do estudo
Metamorphic Testing on Multi-module UAV Systems	N	Tipo do estudo
Software Defect Prediction Approaches: A Review	N	Predição de defeitos
Robot Testing from a machine learning perspective	N	Tipo do estudo
Role of Machine Learning in Software Testing	N	Tipo do estudo

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

Qunomon: A FAIR testbed of quality evaluation for machine learning models	N	Outros
An Effective Crowdsourced Test Report Clustering Model Based on Sentence Embedding	N	Tipo do estudo
Reports Aggregation of Crowdsourcing Test Based on Feature Fusion	N	Outros
Load Balancing Optimization for Transformer in Distributed Environment	N	Outros
Machine Learning Testing: Survey, Landscapes and Horizons	N	Tipo do estudo
Enhancement of Mutation Testing via Fuzzy Clustering and Multi-Population Genetic Algorithm	N	Outros
Combining Genetic Programming and Model Checking to Generate Environment Assumptions	N	Outros
Astraea: Grammar-Based Fairness Testing	N	Tipo do estudo
Graph Neural Network for Source Code Defect Prediction	N	Predição de defeitos
Adaptive Random Testing for Multiagent Path Finding Systems	N	Tipo do estudo
Bridging Fuzz Testing and Metamorphic Testing for Classification of Machine Learning	N	Outros
Software Defect Data Collection Framework for Github	N	Predição de defeitos
Fuzzy Cognitive Map Adaptive Algorithm	N	Outros
Online Estimation of Plant Participation Factors for Automatic Generation Control in Power Systems With Variable Energy Resources	N	Outros
Systematic Mapping: Artificial Intelligence Techniques in Software Engineering	N	Tipo do estudo
Using Search-Based Software Testing to Guide the Strive for Robust Machine Learning Components: Lessons Learned Across Systems and Simulators in the Mobility Domain	N	Tipo do estudo
Evaluating Features for Machine Learning Detection of Order- and Non-Order-Dependent Flaky Tests	N	Tipo do estudo
Systematic Training and Testing for Machine Learning Using Combinatorial Interaction Testing	N	Tipo do estudo
To Seed or Not to Seed? An Empirical Analysis of Usage of Seeds for Testing in Machine Learning Projects	N	Tipo do estudo
An Ensemble Model for Software Defect Prediction	N	Predição de defeitos
A Combinatorial Approach to Fairness Testing of Machine Learning Models	N	Outros
Machine Learning Based Invariant Generation: A Framework and Reproducibility Study	N	Tipo do estudo
Machine Learning Based Invariant Generation: A Framework and Reproducibility Study	N	Predição de defeitos

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

Towards improving explainability, resilience and performance of cybersecurity analysis of 5G/IoT networks (work-in-progress paper)	N	Tipo do estudo
Testing Deep Learning Models: A First Comparative Study of Multiple Testing Techniques	N	Tipo do estudo
Combination Frequency Differencing for Identifying Design Weaknesses in Physical Unclonable Functions	N	Outros
Win GUI Crawler: A tool prototype for desktop GUI image and metadata collection	N	Tipo do estudo
Automatic Error Classification and Root Cause Determination while Replaying Recorded Workload Data at SAP HANA	N	Outros
Augmenting Equivalent Mutant Dataset Using Symbolic Execution	N	Tipo do estudo
Repairing Fragile GUI Test Cases Using Word and Layout Embedding	N	Tipo do estudo
A Preliminary Study on Generating Well-Formed Q# Quantum Programs for Fuzz Testing	N	Tipo do estudo
TCP-Net: Test Case Prioritization using End-to-End Deep Neural Networks	N	Tipo do estudo
Monitoring Approaches for Security and Safety Analysis: Application to a Load Position System	N	Não relacionado a testes
TauLiM: Test Data Augmentation of LiDAR Point Cloud by Metamorphic Relation	N	Não relacionado a testes
Fairkit-learn: A Fairness Evaluation and Comparison Toolkit	N	Predição de defeitos
Repairing Fragile GUI Test Cases Using Word and Layout Embedding	N	Tipo do estudo
Risk Assessment and Management using Machine Learning Approaches	N	Predição de defeitos
Testing Machine Learning Systems in Industry: An Empirical Study	N	Tipo do estudo
Comparing and Combining Analysis-Based and Learning-Based Regression Test Selection	N	Tipo do estudo
Robust Active Learning: Sample-Efficient Training of Robust Deep Learning Models	N	Tipo do estudo
Quality Assurance of Generative Dialog Models in an Evolving Conversational Agent Used for Swedish Language Practice	N	Tipo do estudo
Fairness-aware Configuration of Machine Learning Libraries	N	Predição de defeitos
TOGA: A Neural Method for Test Oracle Generation	N	Tipo do estudo
METHODS2TEST: A dataset of focal methods mapped to test cases	N	Tipo do estudo
Application of Machine Learning for GUI Test Automation	N	Tipo do estudo

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(continuação)

Predicting and Analyzing College Students' Performance Based on Multifaceted Data Using Machine Learning	N	Não relacionado a testes
Physics-Guided Adversarial Machine Learning for Aircraft Systems Simulation	N	Tipo do estudo
Systematic Mapping of Machine Learning–Based Malware Detection Studies	N	Tipo do estudo
Analysis of Tree-Family Machine Learning Techniques for Risk Prediction in Software Requirements	N	Tipo do estudo
A Machine Learning Model for Predicting Performance of Gamified Software Test Specialist	N	Tipo do estudo
An Approach For Verifying And Validating Clustering Based Anomaly Detection Systems Using Metamorphic Testing	N	Outros
A Review of Quality Assurance Research of Dialogue Systems	N	Não aplica AM
Business Decision making using Data Science	N	Tipo do estudo
Software Defect Prediction based on Bayesian Optimization Random Forest	N	Predição de defeitos
Metamorphic Testing of Classification Program for the COVID-19 Intelligent Diagnosis	N	Não relacionado a testes
Efficient Feature Enhancement for Few-Shot Object Detection	N	Não aplica AM
The Inadequacy of Discrete Scenarios in Assessing Deep Neural Networks	N	Tipo do estudo
Towards Energy Consumption and Carbon Footprint Testing for AI-driven IoT Services	N	Tipo do estudo
Implementation of LSSVM in Classification of Software Defect Prediction Data with Feature Selection	N	Predição de defeitos
Data Type Bugs Taxonomy: Integer Overflow, Juggling, and Pointer Arithmetics in Spotlight	N	Tipo do estudo
Regression Testing Prioritization Technique Based on Historical Execution Information	N	Tipo do estudo
Unbalanced data processing for software defect prediction	N	Predição de defeitos
Online Craftsman Ordering Application Development using Waterfall Methodology	N	Não relacionado a testes
Intelligent Islanding Classification with Optimal k-Nearest Neighbors Technique for Single Phase Grid Integrated PV System	N	Tipo do estudo
Improving Defect Prediction Using Combination of Software Metrics	N	Predição de defeitos
Test and Evaluation Harnesses for Learning Systems	N	Tipo do estudo

APÊNDICE A – Estudos da base IEEE Xplore excluídos dessa pesquisa

(conclusão)

An Effective Undersampling Approach to Deal with Class Imbalance Problem in Software Defect Prediction	N	Predição de defeitos
Using Developer Factors and Horizontal Partitioning to Recommend Bug Severity in Open-Source Software Projects	N	Predição de defeitos
Automated Web Testing using Machine Learning and Containerization	N	Tipo do estudo
Application of Convolution Neural Networks and Randomforest for Software Test	N	Predição de defeitos
Software Defect Prediction Using a Combination of Oversampling and Undersampling Methods	N	Predição de defeitos
Telemetry-Based Software Failure Prediction by Concept-Space Model Creation	N	Predição de defeitos
Predicted of Software Fault Based on Random Forest and K-Nearest Neighbor	N	Predição de defeitos
IRRT: An Automated Software Requirements Traceability Tool based on Information Retrieval Model	N	Não relacionado a testes
An Adaptive Approach for Fault Localization using R-CNN	N	Predição de defeitos
Performance Analysis of Feature Selection Techniques in Software Defect Prediction using Machine Learning	N	Predição de defeitos
Message from the Program Co-Chairs	N	Não relacionado a testes
A Model-Based Approach to Generate Dynamic Synthetic Test Data: A Conceptual Model	N	Tipo do estudo

Fonte: Própria autora.

APÊNDICE B – Estudos da base Taylor & Francis excluídos dessa pesquisa

(continua)

ESTUDO	S/N	MOTIVO
A phase wise approach for fault identification	N	Não aplica AM
Computational systems for music improvisation	N	Não relacionado a testes
Predicting field reliability based on two-dimensional warranty data with learning effects	N	Outros
Digital twin-driven rapid individualised designing of automated flow-shop manufacturing system	N	Não relacionado a testes
A fuzzy rule-based generation algorithm in interval type-2 fuzzy logic system for fault prediction in the early phase of software development	N	Predição de defeitos
TEM ExosomeAnalyzer: a computer-assisted software tool for quantitative evaluation of extracellular vesicles in transmission electron microscopy images	N	Não relacionado a testes
LsSQLIDP: Literature survey on SQL injection detection and prevention techniques	N	Não relacionado a testes
A Machine Learning Approach for Quantifying the Design Error Propagation in Safety Critical Software System	N	Não relacionado a testes
Seven HCI Grand Challenges	N	Não relacionado a testes
Enhanced Optimizer Algorithm and its Application to Software Testing	N	Não aplica AM
Artificial intelligence and offensive cyber weapons	N	Não relacionado a testes
A taxonomy of validation strategies to ensure the safe operation of highly automated vehicles	N	Não relacionado a testes
Agent-based Testing: An Automated Approach toward Artificial Reactions to Human Behavior	N	Não relacionado a testes
(AIAM2019) Artificial Intelligence in Software Engineering and inverse: Review	N	Tipo do estudo
Synthetic Aperture Radar (SAR) image processing for operational space-based agriculture mapping	N	Não relacionado a testes
Data-driven software reliability evaluation under incomplete knowledge on fault count distribution	N	Outros
Comparative study on web-based and cloud-based application	N	Não relacionado a testes

APÊNDICE B – Estudos da base Taylor & Francis excluídos dessa pesquisa

(continuação)

Technical debt as an indicator of software security risk: a machine learning approach for software development enterprises	N	Não relacionado a testes
Russian sawmill modernization (a case study). Part 2: improving the efficiency of wood chipping operations	N	Não relacionado a testes
Combinatorial testing: using blocking to assign test cases for validating complex software systems	N	Não aplica AM
Toward regulatory acceptance and improving the prediction confidence of in silico approaches: a case study of genotoxicity	N	Não relacionado a testes
Neural network-based multi-view enhanced multi-learner active learning: theory and experiments	N	Não relacionado a testes
Getting Tactical Communications for Land Forces Right	N	Não relacionado a testes
Improvement of QoS of outdoor prediction models using tuning and machine learning approach in fringe areas	N	Não relacionado a testes
Forecast of agricultural water resources demand based on particle swarm algorithm	N	Não relacionado a testes
Robotic process automation - a systematic mapping study and classification framework	N	Tipo do estudo
Performance comparison of Putnam model using new technology trends for software maintenance cost estimation	N	Outros
Transitioning to Human Interaction with AI Systems: New Challenges and Opportunities for HCI Professionals to Enable Human-Centered AI	N	Tipo do estudo
Enhancing Software Reliability and Fault Detection Using Hybrid Brainstorm Optimization-Based LSTM Model	N	Não relacionado a testes
Relationship between Social Exchanges and Financial Performance: Examining a Virtual Community Sponsored by a Cloud Provider	N	Não aplica AM
Statistical perspectives on reliability of artificial intelligence systems	N	Não relacionado a testes
Do we need to pay technical debt in blockchain software systems?	N	Não relacionado a testes
Digitising legislation: connecting regulatory mind-sets and constitutional values	N	Não relacionado a testes
Software reliability reckoning by applying neural network algorithm	N	Outros

APÊNDICE B – Estudos da base Taylor & Francis excluídos dessa pesquisa

(conclusão)

Application of voice recognition interaction and big data internet of things in urban fire fighting	N	Não relacionado a testes
A comprehensive survey on the ant lion optimiser, variants and applications	N	Tipo do estudo
Creating Formal Models from Informal Design Artefacts	N	Outros
Software security evaluation using multilevel vulnerability discovery modeling	N	Outros
Programming cobots by voice: a pragmatic, web-based approach	N	Não relacionado a testes

Fonte: Própria autora.

APÊNDICE C – Estudos da base Wiley Online excluídos dessa pesquisa

(continua)

ESTUDO	S/N	MOTIVO
Software assistants in software engineering: A systematic mapping study	N	Tipo do estudo
Hybrid support vector machine and K-nearest neighbor-based software testing for educational assistant	N	Não relacionado a testes
Detecting functional and security-related issues in smart contracts: A systematic literature review	N	Tipo do estudo
A fuzzing-based test-creation approach for evaluating digital TV receivers via transport streams	N	Não relacionado a testes
A computer-assisted instructional method based on machine learning in software testing class	N	Outros
Feature transfer learning by reinforcement learning for detecting software defect	N	Predição de defeitos
Ex2: Monte Carlo Tree Search-based test inputs prioritization for fuzzing deep neural networks	N	Outros
Metamorphic relation automation: Rationale, challenges, and solution directions	N	Tipo do estudo
Using goal–question–metric to compare research and practice perspectives on regression testing	N	Tipo do estudo
Guest editors' introduction to the special issue “Automatic Software Testing from the Trenches”	N	Tipo do estudo
Modelling and control of fuzzy-based systems using intelligent water drop algorithm	N	Não relacionado a testes
Advanced security testing using a cyber-attack forecasting model: A case study of financial institutions	N	Outros
Análise de mutações e suas aplicações industriais	N	Tipo do estudo
Improved mayfly optimization deep stacked sparse auto encoder feature selection scorched gradient descent driven dropout XLM learning framework for software defect prediction	N	Predição de defeitos
WIFLF: An approach independent of the target project for cross-project defect prediction	N	Predição de defeitos

APÊNDICE C – Estudos da base Wiley Online excluídos dessa pesquisa

(continuação)

IEEE International Conference on Software Testing, Verification and Validation (ICST 2020)	N	Tipo do estudo
Systematic analysis of software development in cloud computing perceptions	N	Tipo do estudo
Predictive maintenance of infrastructure code using “fluid” datasets: An exploratory study on Ansible defect proneness	N	Predição de defeitos
MLPNN-RF: Software fault prediction based on robust weight based optimization and Jacobian adaptive neural network	N	Predição de defeitos
ASATM: Automated security assistant of threat models in intelligent transportation systems	N	Outros
Machine learning-based test oracles for performance testing of cyber-physical systems: An industrial case study on elevators dispatching algorithms	N	Tipo do estudo
Improving learning experiences in software engineering capstone courses using artificial intelligence virtual assistants	N	Não relacionado a testes
Introduction to the special issue on automation of software test and test code quality	N	Tipo do estudo
Metamorphic testing of named entity recognition systems: A case study	N	Não aplica AM
Digital-first assessments: A security framework	N	Tipo do estudo
Deriving and evaluating a fault model for testing data science applications	N	Tipo do estudo
Selection of DevOps best test practices: A hybrid approach using ISM and fuzzy TOPSIS analysis	N	Tipo do estudo
Gravitational search algorithm-driven missing links prediction in social networks	N	Não relacionado a testes
Improved pairwise test suites for non-prime-power orders	N	Tipo do estudo
Eliminating the high false-positive rate in defect prediction through BayesNet with adjustable weight	N	Predição de defeitos
Prioritising test scripts for the testing of memory bloat in web applications	N	Não relacionado a testes

APÊNDICE C – Estudos da base Wiley Online excluídos dessa pesquisa

(continuação)

Supporting maintenance and testing for AI functions of mobile apps based on user reviews: An empirical study on plant identification apps	N	Tipo do estudo
A synergic quantum particle swarm optimisation for constrained combinatorial test generation	N	Não aplica AM
Metamorphic relation prioritization for effective regression testing	N	Não aplica AM
EESSI: A cross-platform ready-to-use optimised scientific software stack	N	Não relacionado a testes
Improving test automation maturity: A multivocal literature review	N	Tipo do estudo
The operation and maintenance governance of microservices architecture systems: A systematic literature review	N	Tipo do estudo
TRANSMUT-Spark: Transformation mutation for Apache Spark	N	Não aplica AM
Can laboratory parameters be an alternative to CT and RT-PCR in the diagnosis of COVID-19? A machine learning approach	N	Não relacionado a testes
Are duplicates really harmful? An empirical study on bug report summarization techniques	N	Tipo do estudo
Dissimilarity-based test case prioritization through data fusion	N	Não aplica AM
Quantum software testing: State of the art	N	Tipo do estudo
Syntax-based metamorphic relation prediction via the bagging framework	N	Predição de defeitos
A systematic mapping study on the employment of neural networks on software engineering projects: Where to go next?	N	Tipo do estudo
Learning to predict test effectiveness	N	Predição de defeitos
A Developer Relations (DevRel) model to govern developers in Software Ecosystems	N	Tipo do estudo
Quantum computing: A taxonomy, systematic review and future directions	N	Tipo do estudo
Confidence-driven weighted retraining for predicting safety-critical failures in autonomous driving systems	N	Predição de defeitos
Equivalent mutant identification using hybrid wavelet convolutional rain optimization	N	Predição de defeitos

APÊNDICE C – Estudos da base Wiley Online excluídos dessa pesquisa

(continuação)

Survey on test case generation, selection and prioritization for cyber-physical systems	N	Tipo do estudo
The impact of traceability on software maintenance and evolution: A mapping study	N	Tipo do estudo
Empirical analysis of practitioners' perceptions of test flakiness factors	N	Tipo do estudo
Neural network for software reliability analysis of dynamically weighted NHPP growth models with imperfect debugging	N	Predição de defeitos
Learning-based mutant reduction using fine-grained mutation operators	N	Tipo do estudo
Software defect prediction with imbalanced distribution by radius-synthetic minority over-sampling technique	N	Predição de defeitos
An ensemble-based predictive mutation testing approach that considers impact of unreached mutants	N	Predição de defeitos
From zero to fog: Efficient engineering of fog-based Internet of Things applications	N	Outros
Software defect prediction based on stacked sparse denoising autoencoders and enhanced extreme learning machine	N	Predição de defeitos
Optimizing demand-driven null dereference verification via merging branches	N	Tipo do estudo
The IEEE 12th International Conference on Software Testing, Verification & Validation	N	Tipo do estudo
testar – scriptless testing through graphical user interface	N	Tipo do estudo
An overview of ARAS method: Theory development, application extension, and future challenge	N	Tipo do estudo
Continuous improvement and adaptation of predictive models in smart manufacturing and model management	N	Predição de defeitos
Mobolic: An automated approach to exercising mobile application GUIs using symbiosis of online testing technique and customated input generation	N	Tipo do estudo

APÊNDICE C – Estudos da base Wiley Online excluídos dessa pesquisa

(continuação)

Long short-term memory on abstract syntax tree for SQL injection detection	N	Não relacionado a testes
Remote embedded devices test framework on the cloud	N	Outros
Software defect prediction using K-PCA and various kernel-based extreme learning machine: an empirical study	N	Tipo do estudo
CSSG: A cost-sensitive stacked generalization approach for software defect prediction	N	Predição de defeitos
Predicting just-in-time software defects to reduce post-release quality costs in the maritime industry	N	Predição de defeitos
Root causing, detecting, and fixing flaky tests: State of the art and future roadmap	N	Tipo do estudo
Comparison of threshold identification techniques for object-oriented software metrics	N	Tipo do estudo
BUGSJS: a benchmark and taxonomy of JavaScript bugs	N	Tipo do estudo
Heterogeneous fault prediction with cost-sensitive domain adaptation	N	Tipo do estudo
Classifying design-level requirements using machine learning for a recommender of interaction design patterns	N	Tipo do estudo
Improving deep-learning-based fault localization with resampling	N	Tipo do estudo
Automatically identifying potential regressions in the layout of responsive web pages	N	Tipo do estudo
Software defect prediction via LSTM	N	Predição de defeitos
Automating test oracles from restricted natural language agile requirements	N	Não aplica AM
DevOpRET: Continuous reliability testing in DevOps	N	Tipo do estudo
IMNRFixer: A hybrid approach to alleviate class-imbalance problem for predicting the fixability of Non-Reproducible bugs	N	Tipo do estudo
Empirical studies on software traceability: A mapping study	N	Tipo do estudo
Design science research for learning software engineering and computational thinking: Four cases	N	Não relacionado a testes

APÊNDICE C – Estudos da base Wiley Online excluídos dessa pesquisa

(continuação)

Fw-fuzz: A code coverage-guided fuzzing framework for network protocols on firmware	N	Não aplica AM
Tabular-expression-based method for constructing metamorphic relations	N	Não aplica AM
Automated testing in robotic process automation projects	N	Outros
End-user modeling of quality for web components	N	Tipo do estudo
Web service composition on IoT reliability test based on cross entropy	N	Não aplica AM
Exploring the industry's challenges in software testing: An empirical study	N	Tipo do estudo
Complexity vulnerability analysis using symbolic execution	N	Não aplica AM
Model-based hypothesis testing of uncertain software systems	N	Tipo do estudo
Automated tests for cross-platform mobile apps in multiple configurations	N	Não aplica AM
Software fault prediction using particle swarm algorithm with genetic algorithm and support vector machine classifier	N	Predição de defeitos
K-PSO: An improved PSO-based container scheduling algorithm for big data application	N	Outros
Memory models in symbolic execution: key ideas and new thoughts	N	Tipo do estudo
Model-based testing of Apache ZooKeeper: Fundamental API usage and watchers	N	Não aplica AM
An exploration of effective fuzzing for side-channel cache leakage	N	Tipo do estudo
Do different cross-project defect prediction methods identify the same defective modules?	N	Tipo do estudo
DEMINER: test generation for high test coverage through mutant exploration	N	Não aplica AM
An end-user-centric test generation methodology for performance evaluation of mobile networked applications	N	Tipo do estudo
UTCPredictor: An uncertainty-aware novel teaching cases predictor	N	Predição de defeitos
Using blog-like documents to investigate software practice: Benefits, challenges, and research directions	N	Tipo do estudo

APÊNDICE C – Estudos da base Wiley Online excluídos dessa pesquisa

(conclusão)

Application of rule-based expert systems in hardware-in-the-loop simulation case study: Software and performance validation of an engine electronic control unit	N	Tipo do estudo
Straddling the crevasse: A review of microservice software architecture foundations and recent advancements	N	Tipo do estudo
Parallel debugging: An investigative study	N	Tipo do estudo
Discovering common bug-fix patterns: A large-scale observational study	N	Tipo do estudo
Choosing the fitness function for the job: Automated generation of test suites that detect real faults	N	Predição de defeitos
A formal approach to automatically analyse extra-functional properties in mobile applications	N	Não aplica AM
Opcode sequence analysis of Android malware by a convolutional neural network	N	Não relacionado a testes
The impact of feature types, classifiers, and data balancing techniques on software vulnerability prediction models	N	Tipo do estudo
Branch coverage prediction in automated testing	N	Tipo do estudo
Automatic cloud service testing and bottleneck detection system with scaling recommendation	N	Não aplica AM
Software fault localisation: a systematic mapping study	N	Tipo do estudo
Low-rank representation for semi-supervised software defect prediction	N	Predição de defeitos
Testing MapReduce programs: A systematic mapping study	N	Tipo do estudo
A taxonomy of various attacks on smart card–based applications and countermeasures	N	Não relacionado a testes
Test cases generation for multiple paths based on PSO algorithm with metamorphic relations	N	Não aplica AM
Distributed Whale Optimization Algorithm based on MapReduce	N	Não aplica AM

Fonte: Própria autora.

APÊNDICE D – Estudos da base World Scientific excluídos dessa pesquisa

(continua)

ESTUDO	S/N	MOTIVO
A Unified Framework for Bug Report Assignment	N	Tipo do estudo
Nonlinear Dynamics Tools for Offline Signature Verification Using One-class Gaussian Process	N	Não relacionado a testes
Application of the Shuffled Frog Leaping Algorithm (SFLA) in Constructing Fuzzy Classification Systems	N	Não aplica AM
Software Defect Prediction Based on Cost-Sensitive Dictionary Learning	N	Predição de defeitos
An Investigation of Imbalanced Ensemble Learning Methods for Cross-Project Defect Prediction	N	Predição de defeitos
Using Ensembles for Class-Imbalance Problem to Predict Maintainability of Open Source Software	N	Outros
Approaches for the identification of driver mutations in cancer: A tutorial from a computational perspective	N	Não relacionado a testes
Semantic Service Search in IT Crowdsourcing Platform: A Knowledge Graph-Based Approach	N	Não relacionado a testes
An Effective Approach for Context Driven Testing in Practice — A Case Study	N	Tipo do estudo
Multi-dimensional Prediction Model for Cell Traffic in City Scale	N	Predição de defeitos
Automated Visual Testing of Application User Interfaces Using Static Analysis of Screenshots	N	Não aplica AM
Investigating Different Metrics for Evaluation and Selection of Mutation Operators for Java	N	Não aplica AM
DEJIT: A Differential Evolution Algorithm for Effort-Aware Just-in-Time Software Defect Prediction	N	Predição de defeitos
MuSim: Mutation-based Fault Localization Using Test Case Proximity	N	Não aplica AM
Efficient Deep Learning Models for Categorizing Chenopodiaceae in the Wild	N	Não relacionado a testes
A Logistic Growth Model for Software Reliability Estimation Considering Uncertain Factors	N	Outros
Metaheuristics and Software Engineering: Past, Present, and Future	N	Tipo do estudo
MTKeras: An Automated Metamorphic Testing Platform	N	Outros

APÊNDICE D – Estudos da base World Scientific excluídos dessa pesquisa

(conclusão)

Test Oracle Generation Based on BPNN by Using the Values of Variables at Different Breakpoints for Programs	N	Tipo do estudo
ITL-CNN: Integrated Transfer Learning-Based Convolution Neural Network for Ultrasound PCOS Image Classification	N	Não relacionado a testes
Heterogeneous Fault Prediction Using Feature Selection and Supervised Learning Algorithms	N	Predição de defeitos
Protein Structure Prediction Using Quantile Dragonfly and Structural Class-Based Deep Learning	N	Predição de defeitos
Automatic Identification of High-Impact Bug Report by Product and Test Code Quality	N	Tipo do estudo
Equivalent Mutants Detection Based on Weighted Software Behavior Graph	N	Não aplica AM
Improving Large-Gap Clone Detection Recall Using Multiple Features	N	Não relacionado a testes
Toward Improving the Quality of Mutation Operator and Test Case Effectiveness in Higher-Order Mutation Testing	N	Tipo do estudo
A Semantic Web-Enabled Approach for Dependency Management	N	Não relacionado a testes
Hybrid model with optimization tactics for software defect prediction	N	Predição de defeitos
Recommendation of Regression Techniques for Software Maintainability Prediction With Multi-Criteria Decision-Making	N	Predição de defeitos
Software Reliability Analysis Using Nonlinear Transfer Function	N	Predição de defeitos
The Effect of Environmental Metrics on Software Fault Prediction	N	Predição de defeitos
A Systematic Investigation of State-of-the-Art SystemC Verification	N	Tipo do estudo
A Survey on Industrial Information Integration 2016–2019	N	Tipo do estudo
Author Index Volume 30 (2020)	N	Tipo do estudo
Author Index Volume 28 (2018)	N	Tipo do estudo

Fonte: Própria autora.