UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

LIDIA PERSIDE GOMES NASCIMENTO

Aprendizagem de Máquina na Engenharia de Software: Uma Abordagem Técnica para Análise de Defeitos Escapados

Recife

2023

LIDIA PERSIDE GOMES NASCIMENTO

Aprendizagem de Máquina na Engenharia de Software: Uma Abordagem Técnica para Análise de Defeitos Escapados

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do titulo de Mestre em Ciência da Computação.

Área de Concentração: Inteligência Computacional

Orientador: Profº Drº Ricardo Bastos Cavalcante Prudêncio

Recife

2023

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

N244a Nascimento, Lidia Perside Gomes
Aprendizagem de máquina na engenharia de software: uma abordagem técnica para análise de defeitos escapados / Lidia Perside Gomes Nascimento – 2023.
57 f.: il., fig., tab.

Orientador: Ricardo Bastos Cavalcante Prudêncio.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.
Inclui referências.

1. Inteligência computacional. 2. Análise de defeitos escapados. 3. Change request. 4. Aprendizado de máquina. I. Prudêncio, Ricardo Bastos Cavalcante (orientador). II. Título

006.31 CDD (23. ed.) UFPE - CCEN 2024 – 005

Lidia Perside Gomes Nascimento

“Aprendizagem de Máquina na Engenharia de Software: Uma Abordagem Técnica para Análise de Defeitos Escapados”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional

Aprovado em: 03/10/2023.

BANCA EXAMINADORA

_____

Prof. Dr. Alexandre Cabral Mota
Centro de Informática / UFPE

_____

Profa. Dra. Regina Rosa Parente
Instituto Federal do Maranhão, Campus Bacabal

_____

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio
Centro de Informática / UFPE
(**Orientador**)

A Deus, aos meus pais, amigos queridos, professores e a todos que em meu coração habitam.

AGRADECIMENTOS

Agradeço, acima de tudo, a Deus. que em Sua Palavra é dito "Porque Eu, o SENHOR teu Deus, te tomo pela tua mão direita; e te digo: Não temas, Eu te ajudo"(Isaías 41:13).

Agradeço a minha mãe, Patricia Kao, por ser a mãe amorosa e dedicada, e minha melhor amiga. Ela é razão dos meus sorrisos, admiração, força e de todo meu amor. Ao meu pai, Jose Maria por todo o apoio e pela minha irmã, Ana Sophia, por quem tanto amo. A mãe Celeide (in memoriam), a melhor avó que eu poderia ter tido: "só enquanto eu respirar vou me lembrar de você".

Agradeço ao Profº Drº Ricardo Prudêncio, que me acolheu como sua orientanda, bem como pelos seus ensinamentos e trocas de conhecimento. Sem ele, juntamente com o Profº Drº Alexandre Mota não teria tido a oportunidade de atuar no time de Pesquisa do Projeto CIn/Motorola, do qual muito fará diferença em minha vida, em aspectos incontáveis, imensuráveis e imedíveis.

Agradeço ao meu amigo Augusto Garcia, pela jornada de aprendizado constante e autoconhecimento. A minha amiga, a Profª Drª Janaína Menezes, que por muitas vezes me tranquilizou nos momentos de angústia. Agradeço, ainda, a Ingrid Assis que tanto me incentivou e amparou.

Agradeço aos colaboradores do Projeto CIn/Motorola, que me acolheram com muito carinho e empenho. Em especial, ao Audir Paiva, que me acompanhou desde o momento de entrada. Ao Pedro Cruz, por se manter em prontidão.

Todos vocês, a quem tenho gratidão e orgulho por ter cruzado o caminho, me mostram constantemente que o que há de mais belo é fazer as coisas que se ama, com muito zelo e dedicação. A vocês, meu coração é repleto de gratidão!

Por fim, agradeço ao CIn/Motorola, através do projeto de pesquisa entre a Motorola Mobility (empresa Lenovo) e o CIn-UFPE, e ao CNPq que constantemente investem em educação e pesquisa. Sou muito grata por toda a trajetória percorrida no espaço que me dado em forma de oportunidade.

RESUMO

A realização do Testes de Software (TS) é de extrema importância no que diz respeito a garantir a qualidade adequada de um software que esteja em desenvolvimento. Esse processo, quando bem realizado, tem uma série de vantagens como uma boa reputação da empresa de software por parte dos usuários finais e a economia de custos com reparos em defeitos. Quando um defeito é encontrado, normalmente é aberto um relatório de falhas que poderá levar a uma correção no software por parte de desenvolvedores. Nessa dissertação, relatórios de falhas recebem a nomenclatura de Change Requests (CRs), contendo informações que descrevem o problema encontrado no software desde a abertura da CR até a solução da mesma. Grandes empresas, de maneira geral, costumam ter um grande número de CRs abertas semanalmente e que devem passar pela equipe de testes e/ou desenvolvimento para que possam ser inspecionadas e corrigidas de maneira adequada. Este trabalho tem como foco a utilização de técnicas de Aprendizado de Máquina (AM) para automatizar uma tarefa relevante da triagem de CRs, o processo de Escaped Defect Analysis (EDA), no contexto de uma aplicação real da indústria. Nessa aplicação, Defeitos Escapados (DE) são bugs ou problemas que deveriam ter sido detectados por uma equipe de teste específica, mas que, por alguma razão, foram acidentalmente encontradas por uma outra equipe. A ocorrência das DEs é considerada arriscada, tendo em vista que costumam estar relacionadas a falhas nas atividades de testes. O EDA geralmente é realizado de forma manual pela equipe de engenheiros de software, que precisam ler todo o conteúdo textual contido em cada CR para identificar se é um DE ou não, o que passa a ser desafiador e demorado. Na solução aqui abordada, o conteúdo de cada CR é pré-processada por operações textuais e são representadas em forma de atributos para que um classificador de AM venha a retornar a probabilidade dos rótulos de EDA. Os experimentos realizados nesta pesquisa contou com um conjunto de dados de 3767 CRs, que foram fornecidos pela empresa parceira (Motorola Mobility Comércio de Produtos Eletrônicos Ltda). Diferentes tipos de algoritmos de AM foram aplicados para a contrução de classificadores, onde alto valores de AUC puderam ser alcançados (costumeiramente maiores que 0,8), nos experimentos realizados com validação cruzada. Além disso, os experimentos indicam um bom compromisso entre o número de EDs corretamente identificados e o número de CRs que devem ser inspecionados na EDA.

Palavras-chave: análise de defeitos escapados; change request; aprendizado de máquina.

ABSTRACT

Software Testing (TS) phase is of utmost importance to ensure the adequate quality of a software under development. This process, when well-executed, offers several advantages, such as a good reputation of the software company among the end users and cost savings on bug repairs. When an issue is found, normally a bug report document is opened, which can result on fixing the software. In this dissertation, bug reports receive the nomenclature of Change Requests, which stores information describing the software since the CR is opened until it is closed. In general, large companies tend to have a significant number of CRs opened weekly, which must go through the testing and/or development teams for proper inspection and correction. Therefore, this work focuses on the use of Machine Learning (ML) techniques to automate the Escaped Defect Analysis (EDA) process, in the context of a real application. In this study, Escaped Defects (EDs) refer to bugs or issues that should have been detected by a specific testing team but were accidentally discovered by another team. The occurrence of EDs is considered risky as they are often related to testing activity failures. EDA is typically performed manually by software engineers who need to read all the textual content in each CR to determine whether it is an ED or not, making it a challenging and time-consuming task. In the solution presented here, the content of each CR is preprocessed through textual operations and represented as attributes, enabling an ML classifier to return the probability of EDA labels. The experiments in this research used a dataset of 3767 CRs provided by the partner company (Motorola Mobility Comércio de Produtos Eletrônicos Ltda). Various ML algorithms were applied to construct classifiers, where high values of Area Under the Curve (AUC), typically greater than 0.8, were achieved in cross-validation experiments. Additionally, the experiments indicate a good balance between the number of correctly identified EDs and the number of CRs that need to be inspected in EDA.

Keywords: escaped defect analysis; change request; machine learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizagem de Máquina
AUC	Area Under the Curve Receiver Operating Characteristic
CR	Change Request
CV	Cross Validation
DE	Defeito Escapado
DT	Decision Trees
EDA	Escaped Defect Analysis
ETC	Extra Trees Classifier
KNN	K-nearest Neighbors
MLP	Multi-Layer Perceptron
NB	Naive Bayes
PLN	Processamento de Linguagem Natural
RNA	Rede Neural Artificial
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TS	Teste de Software

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO DA PESQUISA

O processo de testes para verificação da existência de bugs[1] é essencial na implementação de qualquer software em desenvolvimento. O Teste de Software (TS) tem como objetivo melhorar a qualidade do software sendo desenvolvido, antes que ele seja lançado para o mercado. O TS é um processo normalmente custoso e meticuloso (MYERS et al., 2004), que gera vários artefatos durante a execução das atividades de teste, por exemplo, na forma de relatórios de erros. Esses relatórios contêm informações relevantes para descrever o erro encontrado no software, como sumário do erro, passos executados de teste, componentes, resultados esperados, dentre outras. No contexto deste trabalho, relatórios de erros são chamados de documentos de Change Request (CR).

Em grandes empresas, os TSs são realizados por diferentes equipes de teste que são designadas para cobrir áreas, recursos e aspectos específicos do software em teste. Uma falha em uma determinada área do software pode ser encontrada pela equipe de testes que deveria cobrir aquela área ou acidentalmente por outra equipe. Neste último cenário, o problema é considerado um Defeito Escapado (DE), cuja ocorrência deve ser reconhecida por se tratar de um defeito não identificado pela equipe de testes responsável por cobri-lo. A Análise de Defeitos Escapados, ou Escaped Defect Analysis (EDA), é um processo que envolve principalmente a inspeção dos CRs criados por diferentes equipes de teste para identificar a ocorrência de DEs (SAAPUNKI, 2023). Além disso, a EDA pretende compreender as causas profundas dos DEs e determinar as ações necessárias para evitar que tais situações voltem a acontecer.

Os DEs tendem a acontecer em maior escala nas grandes empresas. Em muitos casos, o processo de verificação da EDA é feito manualmente, o que requer tempo e recursos humanos significativos. Nesse contexto, o Aprendizagem de Máquina (AM) tem um grande potencial para auxiliar o processo de EDA ao priorizar as CRs que devem ser inspecionados. Com base em dados históricos de EDA, os modelos de AM podem ser treinados para encontrar padrões regulares de características de CRs que poderiam ser usados para prever DEs. Dado um conjunto de CRs a serem verificados pela equipe de EDA, o modelo de AM pode ser usado para classificar as CRs por uma probabilidade estimada de DE. Assim, a equipa pode priorizar a sua inspeção nas CRs com maior probabilidade de reportar DEs, acelerando assim o processo

[1] Erros, falhas ou defeitos no design, desenvolvimento ou operação do software.

de EDA e, com isso, minimizando os custos para a empresa.

1.2 MOTIVAÇÃO

A utilização de testes manuais no processo de garantia de qualidade pode ser onerosa, e a automação dos testes é amplamente desejada. No entanto, a automação dos testes é uma tarefa complexa, influenciada por fatores como a natureza mutável do sistema em desenvolvimento e a ausência de uma representação clara do sistema como um script executável (VICCARI, 2009). A equipe de testes, assim como a equipe de desenvolvimento, frequentemente opera sob pressão devido aos prazos restritos do ciclo de desenvolvimento, buscando lançar as versões de software no mercado. Essa situação aumenta ainda mais a atratividade das técnicas de automação de testes (VICCARI, 2009).

Uma das abordagens menos custosas para a automação dos testes é a execução automatizada dos testes, em que um conjunto de dados originado de testes manuais é utilizado para treinar um modelo de AM (DALAL et al., 1999). As técnicas baseadas em modelos prometem reduzir consideravelmente o custo da geração de casos de teste, aumentar a eficácia do processo e agilizar o ciclo de testes (VICCARI, 2009; DALAL et al., 1999).

O processo de EDA desempenha um papel importante dentro do TS, pois permite que se analise potenciais riscos nas diversas atividades de testes, que podem eventualmente levar a novos DEs. De fato, minimizar DEs é importante, tendo em vista que quanto mais tarde o bug for encontrado, maior será o custo para corrigi-lo. Portanto, a detecção precoce de falhas melhora a qualidade do software e reduz os custos para a empresa (DAWSON et al., 2010; EASTWOOD, 1993; MOAD, 1990).

Adotar técnicas de AM para auxiliar o processo de EDA pode trazer benefícios como agilizar a análise em si, reduzir custos e detectar de forma mais precisa os padrões presentes nas bases de dados contidas nas CRs. Diante disso, esta pesquisa aborda duas questões fundamentais relacionadas ao uso de AM para EDA:

1. Como aplicar a AM para automatizar o processo de EDA?

2. Quais são os pontos em comum de cada classe do EDA?

Para esta pesquisa, a estratégia consiste em utilizar dados quantitativos coletados a partir do repositório das CRs da empresa parceira, provenientes de suas fontes internas de EDA.

Essas CRs serão usadas para desenvolver um classificador de AM e, assim, automatizar o processo de EDA. A integração do classificador na ferramenta é fundamental para o sucesso dessa automação.

1.3 OBJETIVOS

Nesta seção, será apresentado o objetivo geral desta pesquisa, bem como seus objetivos específicos.

1.3.1 Objetivo Geral

O objetivo geral desta pesquisa foi investigar a aplicação de técnicas de AM na construção de classificadores para identificar automaticamente as CRs como DEs ou não. Estes classificadores serão usados para identificar termos e padrões textuais nos sumários das CRs, de forma a predizer a classe de EDA a qual as CRs pertencem (e.g., DE ou não).

1.3.2 Objetivos Específicos

Os objetivos específicos deste trabalho foram:

- Realizar uma revisão bibliográfica sobre EDA e de algoritmos de classificação e técnicas de AM para classificação de CRs;
- Identificar as informações relevantes para a classificação de dados do EDA, aplicando os conceitos de AM;
- Desenvolver os classificadores de AM que melhor se adéquem para o problema proposto de identificação das classes de CRs;
- Avaliar o desempenho dos classificadores, a partir de métricas de desempenho preditivas clássicas de AM.

1.4 TRABALHO DESENVOLVIDO

Neste trabalho, foi investigado o uso de processamento de texto e técnicas de AM para classificação de EDA em um cenário de aplicação real. Na solução desenvolvida, dado uma CR, é realizada uma etapa de extração de características para produzir uma representação vetorial do conteúdo textual da CR analisada. A extração de características é realizada por diferentes operadores de processamento de texto aplicados no resumo de cada CR, incluindo tokenização, normalização, remoção de stopwords, lematização, stemming e filtragem por frequência de termo. A representação vetorial é fornecida como entrada para um modelo de AM treinado a partir de dados históricos fornecidos pela empresa parceira.

Uma série de experimentos foi conduzida usando diferentes algoritmos de classificação de AM em um conjunto de 3.767 CRs. O desempenho da AM nos experimentos foi medido usando a área sob a curva ROC[2], ou Area Under the Curve Receiver Operating Characteristic (AUC), que agrega o número de verdadeiros positivos ao número de falsos positivos produzidos pelo modelo de AM em avaliação. Nos experimentos, foram geralmente observados valores de AUC superiores a 0,8, o que representa um bom compromisso entre taxas de verdadeiros positivos e falsos positivos. No nosso contexto de aplicação, este resultado significa que uma parte significativa das CRs que não são relevantes no processo de EDA podem ser automaticamente identificadas e descartadas para análise, ao mesmo tempo que atingem baixos níveis de falsos positivos.

1.5 PROPOSTA

É importante observar que a proposta deste trabalho envolve a aplicação de técnicas de AM para abordar o problema de detecção de rótulos relacionados as CRs, sendo realizado através da implementação de um classificador de AM.

Diante do problema descrito, foi desenvolvido uma estratégia que utiliza algoritmos de AM para realizar a classificação com base nos padrões de palavras encontrados nos sumários das CRs. Essa estratégia visa identificar automaticamente o rótulo (categoria) do defeito usando técnicas de Processamento de Linguagem Natural (PLN) e algoritmos de AM.

Para que a estratégia baseada em AM se torne eficaz, foram coletadas informações do conjunto de dados da empresa parceira, que inclui descrições de testes textuais. Nesse contexto,

[2] Característica de Operação do Receptor, do inglês Receiver Operating Characteristic (ROC)

os casos de teste foram recuperados do conjunto de dados a partir de consultas com palavras-chave relevantes extraídas das CRs. Os dados dos casos de teste foram então combinados e classificados (MAGALHÃES et al., 2016). Para que a estratégia funcionasse adequadamente, os dados precisaram ser processados minuciosamente, incluindo a extração das características que seriam utilizadas pelo classificador, conforme detalhado na seção 3.3.

1.6 TRABALHOS RELACIONADOS

A utilização de algoritmos de AM para a Classificação de texto e a atribuição automática de categorias a problemas, como bugs, tornou-se uma abordagem fundamental em projetos de desenvolvimento de software de grande porte. Essa abordagem é de extrema importância para a solução de problemas relacionados ao reconhecimento de padrões (KITTLER et al., 1998).

Um dos primeiros casos documentados dessa abordagem foi apresentado por Čubranić e Murphy em 2004 (MURPHY; CUBRANIC, 2004). Nesse trabalho, foi realizada uma triagem de CRs utilizando algoritmos de AM. Esse marco abriu caminho para uma série de pesquisas e aplicações subsequentes na área.

Com o tempo, tornou-se comum a utilização de sistemas de rastreamento de bugs para identificar a evolução de produtos em desenvolvimento. Esses sistemas são ricos em informações relevantes para o monitoramento de bugs e têm se tornado uma fonte valiosa de dados para aplicação de algoritmos de AM (GOYAL; SARDANA, 2019). Portanto, a aplicação de técnicas de classificação de texto e AM em sistemas de rastreamento de bugs é uma abordagem cada vez mais difundida e crucial para a qualidade e eficiência no desenvolvimento de software.

A sugestão de Anvik (ANVIK, 2006) para a correção de bugs envolve a semi-automação do processo de análise de CRs. Ele propõe a criação de um sistema de recomendação de bugs, que seria responsável por sugerir um conjunto de possíveis desenvolvedores adequados para analisar uma determinada CR. Essa sugestão seria baseada em métricas como a experiência prévia do desenvolvedor com as CRs similares e a similaridade entre a CR em questão e as CRs anteriores em que o desenvolvedor trabalhou.

A ideia por trás desse sistema é reconhecer que nem todos os desenvolvedores possuem o mesmo conjunto de conhecimentos e habilidades. Portanto, a alocação de CRs a desenvolvedores com experiência relevante e conhecimento sobre o tipo de problema apresentado na CR pode aumentar significativamente a taxa de sucesso na resolução dos bugs. Além disso, essa abordagem tem o potencial de acelerar o processo de correção, uma vez que as CRs podem

ser alocadas com mais eficiência para os desenvolvedores mais qualificados.

Em resumo, a proposta de Anvik (ANVIK, 2006) visa melhorar a eficiência e a qualidade do processo de correção de bugs, aproveitando o conhecimento prévio dos desenvolvedores e a similaridade entre as CRs para fazer alocações mais inteligentes.

Magalhães(MAGALHÃES et al., 2020) e Magalhães et al. (MAGALHÃES et al., 2016) propõem a utilização da Descrição de uma CR para recuperar Casos de Teste e avaliar a cobertura de código, através de casos de teste textuais, sendo este último também abordado neste trabalho, para que ocorra a exploração de técnicas de Regressão para priorizar CRs específicas. Contudo, o estudo realizado neste trabalho adota uma abordagem de Classificação para determinar a categoria apropriada de cada CR, com base nas palavras presentes no Sumário da CR.

Lins et al. (LINS et al., 2022) adotaram a estrutura TF-IDF para a vetorização da relevância das palavras-chave que se encaixavam no escopo do pré-processamento das palavras presentes no Sumário. Esta abordagem foi semelhante à adotada neste estudo. No entanto, neste trabalho, optou-se por utilizar a técnica baseada em Frequência Binária. É importante notar que a técnica TF-IDF também foi implementada na fase de experimentos deste estudo, e seus resultados na fase de treinamento de dados foram similares aos da Técnica Binária.

Lins et al. (LINS et al., 2022) exploram, ainda, técnicas de Classificação, embora seu foco tenha sido na detecção da criticidade das CRs. Neste estudo, a abordagem de Classificação se concentra em determinar a classe apropriada para cada CR.

1.7 ETAPAS DA DISSERTAÇÃO

Esta pesquisa é dividida em três etapas principais:

- Etapa 1: Pesquisa teórica que teve como objetivo a sondagem de todo o embasamento teórico necessário para o desenvolvimento deste trabalho, juntamente com a revisão sistemática da literatura, que consiste na utilização de Strings de Busca para encontrar conteúdos que possam se enquadrar com a pesquisa e, após a inclusão, excluir tudo que não é visto como útil para o desenvolvimento deste projeto. Foi enquadrado, ainda, na primeira etapa, a coleta de todo material necessário para o desenvolvimento textual que foi realizado por meio de bibliotecas digitiais, sendo necessário a exação de assuntos voltados à Inteligência Artificial, tendo como o foco maior o EDA, Classificadores de AM e CRs;

- Etapa 2: Fundamentado no Design Science Research[3] para sua utilização como guia no desenvolvimento e avaliação dos Classificadores, já que é nesta etapa onde toda a programação é realizada. Houve, também, a definição dos critérios de sucesso para os Classificadores (o que incluí as métricas de desempenho, taxas de acerto). O desenvolvimento dos artefatos aqui gerados, juntamente com a projeção de acordo com os critérios estabelecidos na etapa da revisão bibliográfica também é realizado nesta etapa, bem como ao que se refere a criação dos Classificadores, Estruturação dos Dados e Componentes necessários para que a rotulação dos bugs venham ocorrer. A análise dos resultados também ocorreu nesta fase, onde a identificação da eficácia dos Classificadores desenvolvidos foi realizada, bem como as limitações e melhorias. Para o desenvolvimento dos algoritmos, foi escolhida a Linguagem de Programação Python, por sua simplicidade e rapidez;

- Etapa 3: Processo de experimentação, composto pela verificação do desempenho dos Classificadores para a previsão das CRs, análise comparativa dos Classificadores em relação aos resultados obtidos e ajustes no código para um melhor desempenho dos classificadores trabalhados.

Todo o trabalho e estudo aqui apresentado foi desenvolvido tendo como base diferentes aspectos teóricos da Mineração de Dados em conjunto com um estudo aprofundado dos algoritmos que foram utilizados. Em termos de comparativos, ocorreu a análise dos Classificadores de forma individualizada, tendo os resultados sempre comparados de acordo com o acervo gerado e obtido.

1.8 ORGANIZAÇÃO DA DISSERTAÇÃO

O restante dessa dissertação está dividida nos seguintes capítulos:

- Conceitos Fundamentais: no capítulo 2, o embasamento conceitual necessário para compreender o contexto deste trabalho é apresentado, incluindo conceitos relacionados ao campo de estudo;

[3] Problemas práticos que são encontrados no decorrer do trabalho e direcionam para mudanças para que o objetivo desejado seja alcançado com maior praticidade

- AM para Automação de EDA: no capítulo 3, é apresentada uma descrição detalhada do uso de técnicas de AM para automatizar o EDA, abordando todo o processo envolvido na implementação desenvolvida;
- Experimentos: no capítulo 4, são apresentados os experimentos realizados para avaliação dos classificadores construídos;
- Conclusões: finalmente, no capítulo 5, são apresentados a contribuições e trabalhos futuros.

2 CONCEITOS FUNDAMENTAIS

Durante o processo de desenvolvimento de um software, não há garantia de que ele funcionará sem apresentar erros, sendo assim necessário um monitoramento constante do software desde as primeiras fases do seu desenvolvimento (ABRAN et al., 2004; PIGOSKI, 1996). Além disso, quanto maior o escopo do projeto, maior a probabilidade de surgirem erros. De igual forma, quanto maior a equipe envolvida, maior a probabilidade de ocorrerem falhas (VICCARI, 2009). Por esses motivos, a realização de TS é de extrema necessidade para identificar qualquer defeito de software ou bug (MYERS et al., 2004). Os bugs, contudo, são considerados uma categoria específica de defeitos, pois não causam falhas de maneira constante e previsível. Detectar todas as possíveis falhas que um software pode apresentar é praticamente inviável, a menos que o sistema seja pequeno (PAN, 1999). Falhas existentes em um software que não são encontradas pelo time de teste são definidas como Defeitos Escapados (DE).

Por mais que o processo de EDA não faça parte do gerenciamento de falhas da empresa parceira, é um processo com responsabilidades e importância definidas, tendo em vista que o EDA é acionado após a causa do defeito ser identificada e a ação necessária para o problema ser tomada e devidamente registrada. As informações, tais como, causa raiz, ação tomada, rótulos de bug, dentre outros; são armazenadas dentro do data set da empresa e ficam disponíveis para que os colaboradores da empresa, desde que tenham acesso, possam visualizar.

Nesse capítulo, são revisados os conceitos importantes para a contextualização dessa dissertação, incluindo conceitos de TS em si (Seção2.1), conceitos sobre análise de DEs (Seção 2.2) e finalmente conceitos sobre a tarefa de classificação de relatórios de erros (Seção 2.4).

2.1 TESTES DE SOFTWARE E RELATÓRIOS DE ERROS

A crescente demanda dos usuários por qualidade em softwares impulsiona o uso cada vez mais frequente de TS (VICCARI, 2009). De fato, esse processo é de extrema importância, uma vez que busca identificar erros de forma sistemática (MYERS et al., 2004), fornecendo uma maior garantia sobre a qualidade do produto sendo testado. O TS pode ser definido como o processo de validação e verificação de um produto ou serviço que garanta o cumprimento dos requisitos de negócios, bem como que sua funcionalidade seja conforme o esperado, para que as características desejadas sejam devidamente implementadas (VICCARI, 2009), tornando

assim a qualidade do teste diretamente relacionada à qualidade dos profissionais envolvidos, capazes de discernir alterações relevantes no software (MYERS et al., 2004).

É bem compreendido que sistemas são inerentemente complexos e, à medida que crescem, há uma maior tendência a introdução de erros (MYERS et al., 2004). Essa complexidade pode ultrapassar os limites de gerenciamento (VICCARI, 2009). Cada alteração no código pode potencialmente gerar novos defeitos, o que significa que casos de teste que foram executados inicialmente não garantem mais o funcionamento adequado, exigindo novas rodadas de testes. Este processo de verificação naturalmente aumenta os custos, que podem ser significativos (PAN, 1999). No entanto, a não realização de testes pode acarretar custos ainda maiores (MYERS et al., 2004; EASTWOOD, 1993; MOAD, 1990).

Existem vários tipos de TS, com diferentes propósitos, realizados tanto de forma manual como de forma automática (VICCARI, 2009). Quando um comportamento inesperado é identificado em um software sendo testado, um relatório é aberto pelo testador, descrevendo a potencial falha, ou CR, encontrada.

Softwares específicos, como o JIRA[1], são utilizados para armazenar e gerenciar CRs, que terão um ciclo de vida até uma eventual correção de uma falha por parte da equipe de desenvolvimento. O ciclo de vida de uma CR será apresentado na seção 2.1.1.

2.1.1 Ciclo de Vida de CRs

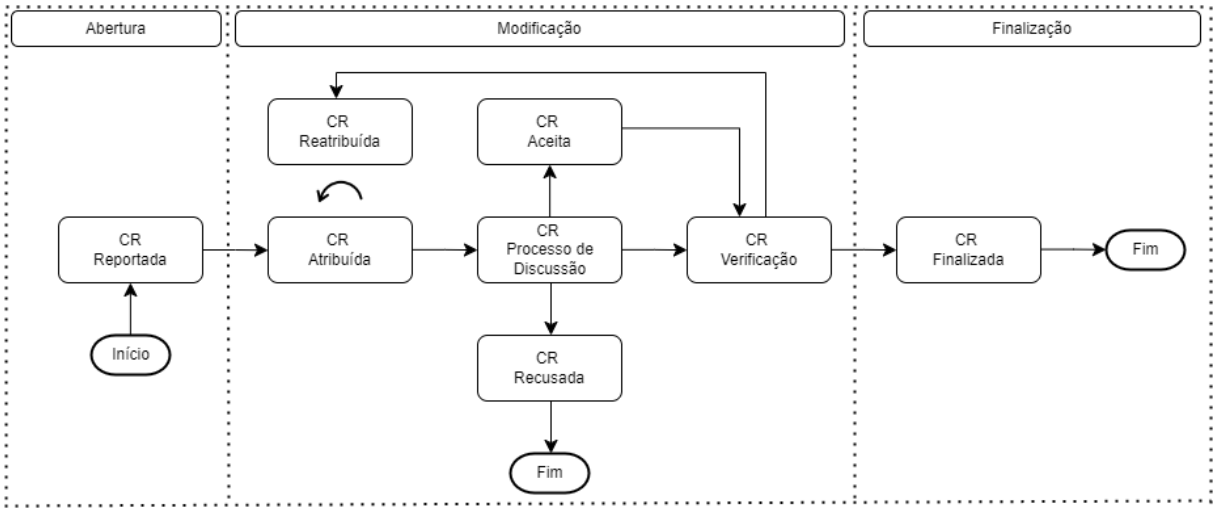
A solicitação que envolve uma CR é fundamental para a implementação de alterações no software. Inicia-se com a concepção de uma ideia de alteração, a partir da qual detalhes são agregados ao sumário e descrição da CR. Posteriormente, após a realização da mudança, o software passará a operar com uma nova versão. Todo esse trajeto é percorrido por meio de um ciclo que denominamos de Ciclo de Vida de uma CR.

O ciclo de vida da CR, adaptado à realidade deste trabalho concordante a proposta de Ihara et al. (IHARA; OHIRA; MATSUMOTO, 2009), é composto por três ações principais, as quais perpassam por diversas etapas, conforme visualizado na Figura 1.

Na etapa de Abertura da CR, uma nova CR é criada no repositório que centraliza todas as CRs. Cada CR é acompanhada por uma série de informações cruciais para a compreensão e efetiva implementação da solicitação, incluindo ID, resumo, descrição, tipo de alteração

[1] Software comercial desenvolvido pela empresa Australiana Atlassian

Figura 1 – Ciclo de Vida da CR



Fonte: autora, 2023 **Adaptação:** Adaptado de IHARA; OHIRA; MATSUMOTO

proposta, componente afetado e versão do software. Nessa fase, também podem ser incluídas informações adicionais relevantes.

A etapa de Atribuição da CR, ocorrendo durante a fase de modificação, engloba a designação da equipe responsável por tratar da correção ou aprimoramento vinculado à CR. Essa alocação é de suma importância para garantir uma resolução eficaz (LUCCA; PENTA; GRADARA, 2002). Entretanto, trata-se de um processo inteiramente manual e que demanda considerável tempo, especialmente devido ao potencial de um alto número de CRs relatadas diariamente, variando entre dezenas e até centenas (CAVALCANTI et al., 2013). Uma alocação inadequada pode resultar em impactos negativos em todo o projeto (CAVALCANTI et al., 2014). Em algumas situações, é necessário reatribuir todas as informações da CR logo após a primeira atribuição.

Durante essa fase, a CR pode ser aceita ou não para implementação. Discussões sobre os aspectos da CR são conduzidas por meio de comentários, que podem ser inseridos diretamente nas informações da CR, permitindo que as discussões se iniciem mesmo antes da aceitação formal da CR.

Na etapa de Verificação da CR, que se inicia após a atribuição da CR a uma equipe e sua aceitação, ocorre a revisão da solução implementada para garantir que a CR tenha sido tratada de forma adequada e esteja em conformidade com os requisitos de qualidade (CAVALCANTI et al., 2014). Caso ajustes adicionais se façam necessários, a CR retorna à equipe responsável pela implementação. Se essa equipe não for capaz de resolver a CR, ela pode ser redirecionada para outra equipe. Somente quando uma solução satisfatória é atingida, a CR é marcada como

"Finalizada"e encerrada, completando assim o ciclo de vida da CR. A partir desse ponto, a alteração pode ser liberada para a produção.

A representação visual do ciclo de vida da CR, com as ações mencionadas, pode ser observada de forma mais precisa na Figura 1. Cada uma dessas ações desempenha um papel crucial no processo de gerenciamento de falhas e manutenção de software, garantindo que as solicitações de mudanças sejam tratadas com eficiência e eficácia.

É importante salientar que todo o processo relacionado a uma CR é transparente para as partes interessadas por meio da ID da CR, bem como pelo repositório que armazena essas informações. As informações englobam todos os detalhes pertinentes à CR, incluindo seu estado de mudança, linha de código afetada (no caso de software) e a versão correspondente. O impacto da CR no projeto também é documentado. Adicionalmente, recursos suplementares podem ser anexados, se necessário, para facilitar a compreensão da equipe em relação ao problema em questão.

2.2 DEFEITOS ESCAPADOS

Um DE pode ser entendido como uma falha no software que deveria ter sido identificada pela equipe de testes em uma etapa específica, mas que, por alguma razão, passou despercebida durante o processo de teste (MAFRA, 2008). Normalmente, essas falhas são descobertas pelo usuário final ou após as etapas de validação planejadas (CRUZ, 2018). Devido a essa possibilidade, torna-se crucial realizar uma análise e avaliação dos casos de DE com o objetivo de guiar soluções adequadas (TOUATI et al., 2018). A análise utilizada neste trabalho é EDA, que consiste na investigação das falhas que passaram despercebidas, visando compreender as razões subjacentes e identificar medidas preventivas para evitar falhas similares no futuro (VANDERMARK, 2003; CRUZ, 2018). A realização cuidadosa dessa análise é de extrema importância para aprimorar a qualidade do software e reduzir os custos, uma vez que a correção de falhas resultantes de DE após a exposição aos usuários finais pode exigir investimentos significativos (VANDERMARK, 2003; MYERS et al., 2004; MAFRA, 2008). Detectar o problema durante as fases de teste, por outro lado, contribui para a contenção dos custos. Além disso, as falhas relacionadas a DE podem abalar a confiança e prejudicar o relacionamento entre os fornecedores de software e seus clientes (CRUZ, 2018), já que a credibilidade do software fornecido pode ser questionada.

Vandermark (VANDERMARK, 2003) sugere uma abordagem para reduzir a ocorrência de

DE, que envolve uma análise minuciosa dos Casos de Teste. Essa abordagem visa examinar todos os casos de teste com base em critérios bem definidos, a fim de identificar quais têm maior probabilidade de resultar em um menor ou maior número de DE. Para realizar essa análise, várias estratégias podem ser adotadas, como mencionado por Cruz (CRUZ, 2018), que destaca a inclusão de testes unitários, a implementação de testes integrados ou a adoção de automação de testes. Essas estratégias buscam mitigar a ocorrência de DE, melhorar a detecção de falhas em estágios iniciais e, consequentemente, aprimorar a qualidade geral do software.

2.2.1 Análise de Defeito Escapado

O processo de EDA é uma prática que busca aprimorar a qualidade de um produto ou software (SAAPUNKI, 2023). Seu objetivo é identificar as razões pelas quais determinados bugs passaram pela fase de garantia de qualidade interna, executada pelo time de testadores responsáveis por identificar possíveis discrepâncias no sistema desenvolvido. Por meio do EDA, ações adequadas são definidas para evitar que o problema ocorra novamente no futuro. Esse processo resulta em economia de custos, pois quanto mais cedo uma falha for detectada, mais econômica será a solução da CR (SAAPUNKI, 2023; MYERS et al., 2004; VANDERMARK, 2003).

O EDA começa após a correção de uma falha identificada, com o objetivo de investigar e analisar a causa raiz. Após a correção, todas as CRs recebem o status de "FINALIZADA". Isso permite iniciar o EDA para identificar a causa raiz do problema, bem como determinar o momento da detecção da falha e avaliar quando ela deveria ter sido originalmente identificada (SAAPUNKI, 2023).

Para aprimorar o processo de EDA, Vandermark (VANDERMARK, 2003) sugere a colaboração entre equipes de desenvolvimento e testes, além de enfatizar os seguintes pontos:

- Separar os DEs em categorias úteis para uma análise aprofundada;
- Executar estatísticas nos dados categorizados;
- Identificar e implementar mudanças gerais de processo necessárias, tendo como base os resultados estatísticos;
- Identificar e implementar mudanças de baixo nível que seja necessária com base das análises aprofundadas de DEs específicos;

- Usar métricas para demonstrar a eficácia das mudanças do processo.

A qualidade do produto, seja no hardware ou software, é um fator de competitividade de grande importância em empresas de grande porte (CRUZ, 2018). Conforme mencionado por (MARTINS; LAUGENI, 2005), "é possível afirmar que em todas as visões de qualidade, indicam que o foco será direcionado principalmente à satisfação dos clientes e mercados e, consecutivamente, à melhora dos resultados empresariais". Esse gerenciamento visa aumentar a competitividade de uma empresa por meio da melhoria contínua do produto e de seus processos (INDEZEICHAK et al., 2005). Portanto, o processo de EDA é um método de resolução de problemas que se concentra em identificar as causas primárias e secundárias que levaram a um determinado problema, incidente ou evento (SAAPUNKI, 2023), o que torna o EDA capaz de responder a três perguntas fundamentais:

1. O que ocasionou a causa raiz?
2. Como obter uma compreensão ampla sobre o problema para se poder encontrar uma solução relacionada ao problema raiz?
3. Como coletar as informações necessárias para o aprendizado da análise feita e definir soluções para evitar que o mesmo problema, ou ter a mesma causa raiz, torne a acontecer?

Tais questões podem, ainda, serem reformuladas para aplicar a utilização da técnica dos 5 porquês, que visa encontrar a causa raiz do problema e questionar o "por quê" 5 vezes (ou a quantidade necessária para a questão a ser respondida) para se poder entender o que aconteceu, que podem ser traduzidas conforme a simplificação realizada por (WEISS, 2012), que segue os seguintes passos:

- Iniciar a análise com a afirmação da situação que se deseja entender ou, em outras palavras, iniciar com o problema;
- Por que a afirmação anterior é verdadeira?
- Por que a razão que explica o porquê da afirmação verdadeira?
- Deve se continuar perguntando o por quê até que não se possa mais perguntar os por quês;

- Ao cessar as respostas dos por quês significa que a causa raiz foi identificada

Para que essas perguntas sejam respondidas, se faz necessário separar os escapes em categorias úteis e com um teor significativo, de forma que se torna importante estabelecer quais pré-requisitos devem ser considerados para decidir quais categorias serão inclusas no processo de EDA (VANDERMARK, 2003). Esses passos podem ser:

- Etapa do Processo onde o DE deveria ter sido encontrado:
  - Revisão do conceito, design, código;
  - Testes (Grupo de Testes de Função, Sistema, Tradução, Drivers, Instalação, Desempenho e Injeção de Erro):
  - Desenvolvimento de informações contidas nas etapas anteriores.
- Componentes: local onde o DE foi detectado;

Esse processo tem um ponto muito importante no EDA para poder encontrar informações sobre o DE, inclusive em qual momento deveria ter ocorrido a detecção, que se enquadra na Etapa de Desenvolvimento acima listada. Na Etapa da Análise dos Componentes, que corresponde ao código que foi responsável pelo DE. Já na Etapa do Impacto dos DE envolve a análise de qual tipo de problema ocasionou o espape. Nessa etapa, há algumas subetapas onde será descoberto a versão do código em que o DE se originou, o local onde o defeito causa impacto, bem como a gravidade do DE. Tais pontos serão abordados com mais clareza na seção 2.4.

2.3 APRENDIZAGEM DE MÁQUINA

A AM se destaca pela capacidade de desenvolver técnicas que aprimoram o desempenho, tarefa ou experiência do software (MITCHELL, 1997; BRUNIALTI et al., 2015). Essa versatilidade possibilita a aplicação de diversas técnicas de AM na análise de dados textuais em diversos contextos (CAI; HE, 2011; YU; SHEN; XIE, 2013; BRUNIALTI et al., 2015).

Assim, as técnicas de AM têm como objetivo realizar o aprendizado com base nos dados presentes em seus conjuntos de treino e teste, visando a tomada de decisões apropriadas (RIBEIRO, 2021; DUARTE; STÅHL, 2019). Os modelos de AM podem ser categorizados principalmente em dois grupos: supervisionado e não supervisionado (CORCOVIA; ALVES, 2019).

O tipo de AM utilizado no desenvolvimento deste trabalho é o supervisionado, que implica na aplicação de regras generalizadas que relacionam os dados de entrada aos de saída (RIBEIRO, 2021). Nesse contexto, para cada algoritmo de AM utilizado, é essencial fornecer uma resposta associada, ou seja, um rótulo de classe, para cada Vetor de Valores[2] apresentado (LUDERMIR, 2021). Isso possibilita a construção de um classificador de AM capaz de determinar com precisão a classe das CRs que ainda não foram rotuladas.

Para viabilizar a AM, é necessário utilizar algoritmos projetados para esse fim, aplicados em dados coletados previamente. A escolha do algoritmo mais adequado, com melhor desempenho de previsão, é crucial para garantir a eficácia da solução. Para isso, foi indispensável a utilização de dados fornecidos pela empresa parceira, já que os dados têm o poder de otimizar e agilizar operações organizacionais, possibilitando a tomada de decisões baseadas em fatos e padrões extraídos dos dados coletados (SAAPUNKI, 2023).

A tomada de decisão é um processo que extrai padrões e fatos dos dados coletados, influenciando diretamente as decisões (MILLER, 2019). Nesse contexto, as decisões são embasadas em dados reais e fatos coletados, não em intuição ou observação (SAAPUNKI, 2023). O Classificador, portanto, informa o provável rótulo de bug, mas a decisão final cabe ao testador.

2.4 CLASSIFICAÇÃO AUTOMÁTICA DE CRS

Embora o processo de EDA possa não ser uma prática comum no gerenciamento de falhas da organização colaboradora, sua importância é indiscutível. Através do EDA, é possível conduzir análises aprofundadas e implementar melhorias ou ações corretivas conforme necessário, visando aprimorar a solução em questão e evitar a recorrência do problema. Portanto, modelos de AM são empregados para categorizar as CRs, abrangendo diferentes estágios de vida dessas ocorrências, contemplando uma variedade de objetivos e áreas específicas de interesse.

A análise de dados é uma etapa de extrema importância, já que os dados que são analisados são capazes de fornecer uma otimização, melhoria, eliminação de desperdício e satisfação do cliente im prova (SAAPUNKI, 2023). A partir desta etapa, os dados podem ser utilizados para aplicação de AM para se poder aplicar Classificador para previsão e ocorrer a tomada de decisão, sendo esta última validada por um testador da empresa parceira.

Para se poder ocorrer o processo da aplicação de Classificador de AM juntamente com a tomada de decisão através dos padrões das CR coletadas se faz necessário a tomada de

[2] Atributos

decisão baseada em dados de uso, ou seja, a tomada de decisão tem como base os dados reais e fatos coletados (SAAPUNKI, 2023) com a empresa parceira.

As solicitações de mudanças, ou CRs, são geralmente expressas por meio de descrições textuais em linguagem natural. Essa é uma tarefa comum no processo de manutenção de software, visando garantir que o produto funcione corretamente durante o seu uso (KAGDI; POSHYVANYK, 2009; CAVALCANTI et al., 2014). No entanto, esse processo de triagem das CRs requer um amplo conhecimento organizacional e equilíbrio, sendo manual e, muitas vezes, tedioso (HIEW; MURPHY; ANVIK, 2006; SAAPUNKI, 2023). Portanto, o gerenciamento de CRs desempenha um papel crucial no ciclo de manutenção de software, permitindo que os desenvolvedores lidem com correções de defeitos, melhorias e solicitações de alterações. Dado que uma CR frequentemente contém informações sobre alterações, novas funcionalidades, etapas de implementação e até mesmo novos softwares, ela também serve como uma forma de documentação, rastreando a evolução do software ao longo do tempo (CAVALCANTI et al., 2014).

No entanto, as CRs normalmente precisam passar por um processo de pré-processamento para limpar os dados, devido à possibilidade de duplicações, descrições inadequadas e atribuições incorretas que podem ocorrer devido à natureza manual desses dados (CAVALCANTI et al., 2014), como explorado com mais detalhes na seção 3.3.

Quando uma CR é criada, ela é armazenada e rastreada em sistemas apropriados para esse fim. Ferramentas analíticas podem ser utilizadas para explorar o conteúdo das CRs, o que pode levar à melhoria de diferentes processos relacionados ao TS. A aplicação de algoritmos de AM na classificação de CRs pode ter vários propósitos e pode ocorrer em diferentes estágios do ciclo de vida de uma CR. Essa abordagem também é realizável em diferentes partes das CRs, incluindo a gestão da equipe, os times de teste e desenvolvimento (KÖKSAL; ÖZTÜRK, 2022; ZHANG et al., 2015).

Os modelos de AM podem ser aplicados imediatamente após a criação de uma CR por um testador, permitindo a análise do problema para determinar se é uma falha que precisa ser corrigida ou uma situação que exige otimização e refatoração (HERZIG; JUST; ZELLER, 2013; PEREZ et al., 2021). Isso ocorre logo após a criação da CR, seguindo um processo que é geralmente composto por várias etapas, conforme proposto por (IHARA; OHIRA; MATSUMOTO, 2009), mas adaptado para se adequar à realidade deste trabalho. Essas etapas formam o ciclo de vida das CRs, que será discutido em detalhes na seção 2.1.1.

A aplicação dessas tarefas tem uma importância significativa na economia de tempo dos

desenvolvedores na etapa de correção, além de ajudar a reduzir custos ao evitar esforços redundantes. Os modelos de AM podem ser usados para priorizar CRs, prevendo a gravidade do problema relatado (LINS et al., 2022; SARAWAN; POLPINIJ; LUAPHOL, 2023). A definição da gravidade geralmente é uma tarefa atribuída a testadores ou gerentes, que precisam examinar visualmente o conteúdo das CRs, um processo demorado e suscetível a erros humanos.

Além disso, a utilização de AM pode prever qual desenvolvedor é o mais adequado para resolver um determinado problema, com base em alocações anteriores para problemas semelhantes. Isso já tem sido explorado na literatura com esse propósito (GOYAL; SARDANA, 2019; UDDIN; KOWSHER; SAKIB, 2022).

3 APRENDIZADO DE MÁQUINA PARA AUTOMAÇÃO DE EDA

O principal objetivo para a realização desta pesquisa foi a utilização de técnicas de AM para classificar se CRs são DEs ou não. Por padrão, o processo de EDA se concentra no defeito individual, a causa raiz e o motivo pelo qual o DE não foi detectado. Identificar DEs é importante para entender como evitar que este tipo de problema, bem como as suas causas, torne a acontecer em algum momento futuro (SAAPUNKI, 2023). Desta forma, métodos automáticos de identificação de DEs são de extrema importância, tendo em vista que ações a serem tomadas são disparadas exatamente a partir da identificação da CR como DE ou não.

No nosso contexto, modelos de classificação baseados em AM são capazes de retornar automaticamente a probabilidade de um CR ser ou não um DE, o que proporciona uma economia de esforço da equipe de testes, redução de custos, já que quanto mais cedo uma CR é identificada como um DE menos recursos são desempenhados durante o processo de EDA.

O presente trabalho foi desenvolvido com uma empresa parceira, com o objetivo de otimizar o processo de triagem de CRs após o seu fechamento. Nesse momento, informações da CR mantidas pela empresa parceira, serão usada pelo modelo de AM desenvolvido durante o processo de pesquisa deste trabalho. Para empresa parceira, essa pesquisa deverá propiciar a melhoria na eficácia das suas equipes de testes.

A classificação realizada pelo modelo de AM utiliza termos presentes no sumário das CRs, e que eram visualizadas em forma de resumo textual. A partir deste ponto, foi desenvolvido um estudo de caso, com a construção de diferentes classificadores, a partir dos dados contidos no histórico do EDA, sendo disponibilizado a partir do conjunto de dados da empresa parceira.

Para que o trabalho fosse realizado, foi necessário realizar três principais etapas:

1. Coleta de dados, onde um conjunto de dados das CRs foi coletado juntamente com os rótulos EDA correspondente a cada CR;

2. Extração das características, onde o texto contido no sumário das CRs foi pré-processado para que o treinamento fosse realizado com termos relevantes;

3. Treinamento, do qual os modelos de classificadores foram construídos a partir dos dados coletados e pré-processados nas etapas anteriores.

Após a construção dos classificadores, bem como a etapa de teste de qualidade das previsões ter sido concluída, os classificadores passaram a ser utilizados na implementação para

que o processo de EDA ocorresse de forma automática.

As etapas do trabalho serão descritas com um maior grau de detalhamento e, consequentemente, poderão ser melhor compreendidas nas próximas subseções.

3.1 EMPRESA PARCEIRA

A empresa parceira deste estudo opera como fabricante de dispositivos móveis, mantendo uma estreita conexão com o setor de telecomunicações. Como resultado, o produto que ela comercializa engloba tanto o desenvolvimento de hardware quanto de software. Com o intuito de aprimorar a experiência do usuário final, a empresa parceira procura utilizar os dados contidos nas CRs para identificar padrões relacionados a falhas (bugs), visando agilizar o processo de detecção e resolução. Esse procedimento não apenas reduz o tempo necessário para identificar e classificar problemas, mas também otimiza o uso de recursos.

É crucial salientar que os dados fornecidos pela empresa parceira e utilizados neste estudo não abrangem informações sensíveis nem detalhes que revelem os procedimentos internos referentes à detecção e correção de bugs. Adicionalmente, destaca-se que todas as terminologias empregadas nesta pesquisa são fundamentadas na literatura existente ou foram ajustadas conforme necessário para a elaboração desta dissertação.

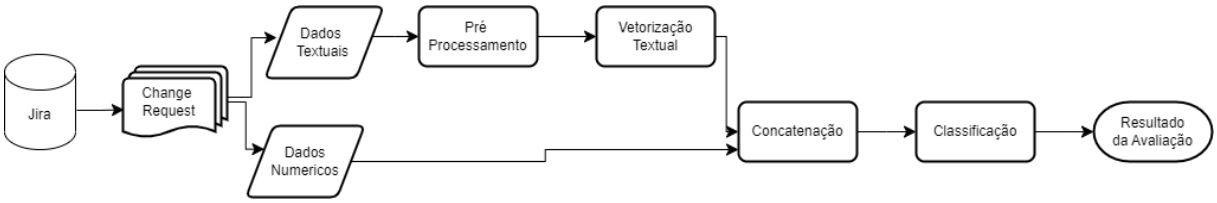
3.2 COLETA DE DADOS

A análise dos dados realizada foi baseada nas fontes de dados internas da empresa parceira. Mais especificamente, foram coletadas CRs armazenadas na ferramenta JIRA, sendo utilizada pela empresa parceira para gerenciar CRs abertas pelos vários times de teste. Como escopo de trabalho, a coleta de CRs foi delimitada para o Time de Regressão da empresa parceira. Foram coletadas CRs abertas por esse time e que já haviam sido reportadas como sendo DE ou não. Essa informação específica foi rotulada de forma manual pelos engenheiros de software que eram responsáveis pelas tarefas de EDA na empresa. Uma base de 3.767 CRs foi coletada neste trabalho.

A visualização destes dados foi realizada através da importação do Matplotlib, que é uma biblioteca utilizada para se poder ter uma percepção visual de dados no Python; sendo uma ferramenta de visualização de dados utilizada de frequentemente e amplamente em pesquisas científicas, análise de dados e AM. Após o processo de coleta de dados, as próximas etapas

puderam ser realizadas, conforme visualizado na Figura 2 (LINS et al., 2022), sendo adaptada para essa pesquisa. Após o processamento, estes dados são utilizados para o treinamento de um modelo de AM, para a tarefa de classificação de EDA.

Figura 2 – Processo de Construção de Classificadores



Fonte: autora, 2023 **Adaptação:** Adaptado de LINS et al.

3.2.1 Rótulos das CRs

Neste trabalho, cada CR está associada a um dos seguintes rótulos alvo para classificação, que pode ser visualizado na tabela 1.

Tabela 1 – CRs

Rótulo	Quantidade
Out of Scope	1614
Core Product Validation	1801
Escaped Defects	352

Fonte: Elaborado pela autora, 2023

Cada rótulo pode ser entendido conforme a definição:

- Core Validation: Esta categoria contém CRs abertas corretamente durante a execução do teste do Time de Regressão, i.e., são CRs associadas a falhas que deveriam de fato serem encontradas pelo time. Desta forma, não houve um DE. Houve um total de 1.801 CRs inclusas na categoria Core Validation;
- Out of Scope: Nesta categoria, as CRs são detectadas pelas equipes da empresa parceira, mas que não estão no escopo de teste de regressão, por não ser a especialidade designada para o time. Aqui, é a categoria geral onde as CRs que não são consideradas DE do

Time de Regressão estão associadas, ou seja, CRs que não deveriam ter sido abertas pelo time de Regressão, mas que foram descobertas pelo Time de Regressão. Houve um total de 1.614 CRs inclusas na categoria Out of Scope. Neste contexto, os problemas comumente encontrados são:

  - Questões que, de maneira geral, estão fora do escopo do Time de Regressão;
  - Problemas em aplicativos de terceiros;
  - Tarefas relacionadas a atividades que envolvem o desenvolvedor;
  - Questões específicas das Operadoras de Telefone;
  - Problemas que não podem ser avaliados onde há anomalia no MME[1] e no Teste do Sistema;
  - Problemas realmente difíceis de reproduzir, surgindo após atividades de longa duração em DUT[2], ou em cenários extremos;
  - Quando uma CR é encerrada de maneira incorreta, com uso inadequado do fluxo de trabalho de uma CR, ou até mesmo quando uma CR é duplicada indevidamente;
  - Preferência pessoal do usuário.
  - Ambiente de teste não disponível ou inadequado para reproduzir o problema do Core Product Validation;
  - Problemas presentes apenas em versões de fábrica, ainda não lançados para os usuários finais.

- Escaped Defects: Nesta categoria, há um total de 352 CRs, que abrangem as demais CRs que não se enquadram em nenhuma das categorias mencionadas anteriormente, já que são consideradas DEs e que são divididas em 7 subcategorias:

  - Before Product Validation Start: Esta categoria refere-se a problemas identificados antes da equipe de Validação do Produto iniciar os testes no recurso ou funcionalidade. Geralmente, as atividades de planejamento de testes e execução se enquadram nesta categoria. Houve um total de 113 CRs inclusas nesta categoria;

[1] Principal entidade controladora no Evolved Packet Core (EPC)

[2] Data Type Unit, onde há a ausência de um valor específico; o tipo de unidade possui apenas um único valor, que atua como espaço reservado quando nenhum outro valor existe ou é necessário.

- Breakage: Essa categoria se aplica quando uma falha é introduzida após a conclusão do Ciclo de Regressão[3], muitas vezes devido a efeitos colaterais ou integrações que não foram validadas imediatamente após os testes ou quando o ciclo foi executado em uma configuração diferente das execuções padrão da equipe. A ação recomendada é colaborar com a equipe de desenvolvimento ou componente para evitar a recorrência desses problemas. Houve um total de 74 CRs inclusas nesta categoria;
- Escaped by Coverage: Essa categoria é relevante quando não há cobertura para o defeito ou quando a cobertura existente não é adequada e requer aprimoramento. A ação subsequente envolve a criação ou atualização do caso de teste correspondente e sua execução. Houve um total de 54 CRs inclusas nesta categoria;
- Requirement Change: Esta categoria é utilizada quando um problema foi identificado devido a uma alteração ou melhoria nos requisitos. Nesse caso, o problema pode ser fechado como uma atualização de requisitos. Houve um total de 39 CRs inclusas nesta categoria;
- Escaped by Test Case: Essa categoria é aplicável quando há cobertura para o problema, mas os Casos de Teste não abrangem o cenário relatado por outros ou quando o Caso de Teste está desatualizado. A ação necessária é a atualização ou criação de um novo Caso de Teste para abordar essa deficiência e deve ser realizada. Houve um total de 30 CRs inclusas nesta categoria;
- Escaped by Planning: Essa categoria é empregada quando o problema resulta de falhas no planejamento, como a omissão de um Caso de Teste em um plano ou a escolha inadequada de uma construção, mesmo quando existe cobertura adequada. A ação recomendada é revisar o planejamento e a seleção de Casos de Teste para evitar esses problemas no futuro. Houve um total de 26 CRs inclusas nesta categoria;
- Escaped by Tester: Esta categoria é relevante quando o testador conseguiu identificar o problema, mas, por algum motivo, não o relatou. A ação recomendada é melhorar o processo de execução dos testes para garantir que tais problemas sejam relatados corretamente. Houve um total de 16 CRs inclusas nesta categoria.

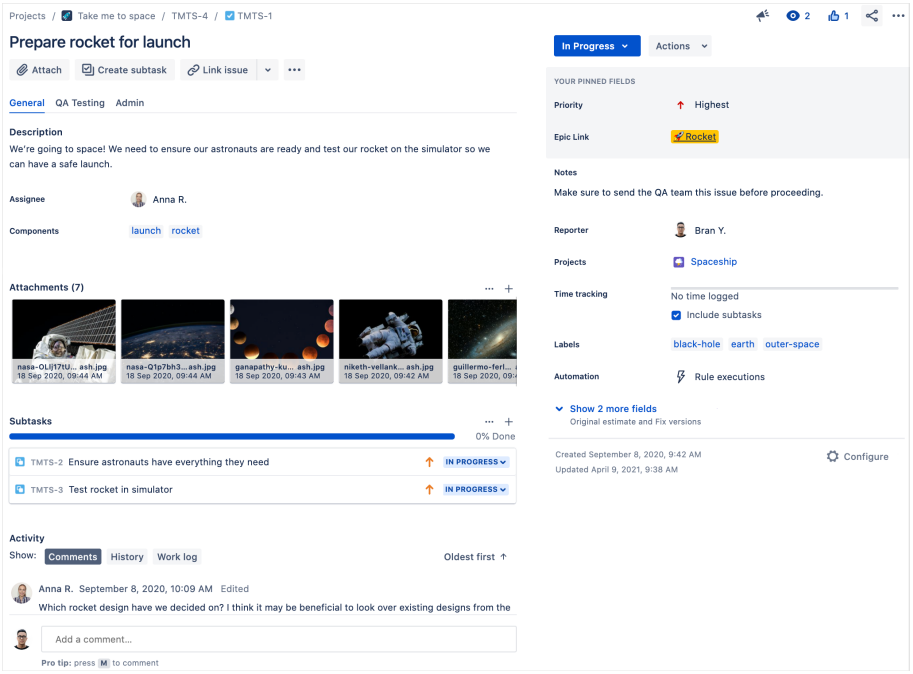
[3] Realizado unicamente pelo Time de Regressão da Empresa Parceira

3.3 PRÉ-PROCESSAMENTO

Com a finalização da etapa da coleta de dados, se faz necessário realizar o pré-processamento dos dados, que tem como principal objetivo produzir um vetor de características que possa ser utilizado no conjunto de treinamento para os algoritmos de AM.

Todas as CRs utilizadas nesta pesquisa foram armazenadas no repositório do JIRA e mantidas no banco de dados da empresa parceira, seguindo a organização por campos que as idenficam e descrevem o problema que foi encontrado pelo testador de software. Os campos contêm as seguintes informações das CRs: ID, nome, sumário, resumo e prioridade (que pode variar de 1 a 3, onde o estado crítico de urgência é caracterizado pelo número 1); conforme visualizado na Figura 3.

Figura 3 – Estrutura da CR no JIRA



Fonte: ATLASSIAN

Diferentes campos de uma CR podem ser utilizados para o processo de classificação de AM. Neste trabalho, no entanto, foi trabalhado com os dados contidos no campo de Sumário, que pode conter informações preciosas para serem pré-processadas e passar pelo processo de treinamento de AM, para que se tenha a possibilidade de identificação, caso a CR seja considerada um DE ou não.

Os sumários das CRs são dados textuais, e precisam então ser processados de maneira adequada para criar uma representação vetorial de cada CR. As seguintes operações foram

aplicadas para cada sumário de CR:

- Tokenização: caracterizado por ser o processo de divisão textual em pedaços menores, sendo chamados de tokens. Os tokens podem ser compostos por palavras, sílabas ou letras. Para o trabalho aqui realizado, foi utilizado tokens de palavras, com a aplicação do pacote NLTK[4] sendo utilizado, juntamente com o módulo Tokenize, para que o parâmetro de espaços em branco como separador fosse obedecido;

- Normalização: os tokens foram convertidos para letras minúsculas, para que a duplicação de palavras se tornasse possível evitar, para não ocorrer uma diferenciação de letras maiúsculas e minúsculas, evitando assim problemas de Case Sensitive[5]. Símbolos, sinais de pontuação, quebras de linha, números e caracteres não ASCII[6] foram removidos, de forma que apenas os caracteres alfanuméricos fossem considerados. Essa etapa tem grande importância, já que auxilia na redução de ruídos e garante a consistência dos dados;

- Remoção de Stopwords: consideradas palavras sem importância por não serem discriminatórias por se caracterizar como irrelevantes ao contexto aqui trabalhado, mas ocorrem com certa frequência nos dados textuais. Foram removidas as palavras que estevam contidas no pacote NLTK, tendo poucas exceções (on, off, not);

- Lematização: a lematização ajuda a consolidar palavras com significados semelhados e a reduzir a dimensão dos dados, já que estas foram agrupadas de acordo com o seu lema, que é a sua forma canônica. Desta forma, plurais foram reduzidos, por exemplo a sua forma singular. Para isso, foi necessário utilizar o pacote NLTK, com o módulo WordNedLemmatizer;

- Stemming: os tokens foram reduzidos ao seu radical comum, removendo os afixos, ou seja, os prefixos e sufixos. Nesta etapa, há um mapeamento de palavras para a sua forma raiz, possibilitando um melhor desempenho e agrupamento de palavras com significados semelhantes. Para isso, foi utilizado Krovetz Stemming[7];

- Filtro de Frequência: os tokens com menos de 10 ocorrências foram removidos, pois foram consideradas como palavras raras ou com erros ortográficos.

[4] www.nltk.org
[5] Caracteres maiúsculos e em minúsculos são tratados de modo diferente
[6] Sistema de representação de letras, algarismos e sinais de pontuação e de controle
[7] pypi.org/project/KrovetzStemmer

Após o pré-processamento, foram observados 843 tokens distintos, que foram utilizados para representar as CRs. Nesta etapa, duas técnicas de vetorização foram aplicadas, para gerar as representações finais das CRs, que serão brevemente apresentadas abaixo:

- TF-IDF[8]: técnica utilizada para se poder representar a importância de dada palavra em um documento. É considerado a frequência no documento, bem como a raridade, em todo o conjunto de dados. Para que o cálculo do TF-IDF seja realizado, duas medidas, bem como o cálculo de ambas de forma agrupada, são consideradas:
  - TF: é a frequência da palavra o documento, que pode ser calculada de acordo com o número de vezes que a palavra aparece no documento, dividido pelo número total de palavras contidas;
  - IDF: é a medida que se refere a raridade da palavra no documento, tendo o seu cálculo realizado através do logaritmo do número de total de documentos dividido pelo número de documentos que contém a palavra;
  - Valor Final do TF-IDF: o valor final para uma palavra em um documento é dado pelo produto do TF e IDF, que tem como resultado uma matriz em que cada linha representa um documento e cada coluna representa o termo, sendo os valores medidos por pesos TF-IDF das palavras que estão contidas no documento.
- Representação Binária: é a forma mais simples de vetorização de dados. Cada documento tem sua representação por um vetor binário, do qual há uma representação por um vetor binário, tendo cada posição correspondente a uma palavra que está contida no vocabulário formado. Se a palavra estiver contida no documento, o valor correspondente atribuído será 1; caso contrário, o valor atribuído será 0.

As representações vetoriais são usadas para a construção dos exemplos de treinamentos utilizados pelos algoritmos de AM. O tipo de representação pode ser de grande importância, tendo em vista que a escolha da representação pode impactar a qualidade dos modelos de AM treinados. No caso desta pesquisa, ambas as vetorizações obtiveram resultados satisfatórios, conforme apresentado nos experimentos.

[8] Term Frequency-Inverse Document Frequency

3.4 CLASSIFICADORES

A seleção dos classificadores empregados neste estudo foi baseada na utilização de uma variedade de modelos e na obtenção de desempenho significativamente superior aos valores de referência para cada rótulo, tais valores serão detalhadamente elucidados na Seção 4.2.

De maneira geral, os dados de treinamento foram padronizados para todos os algoritmos, utilizando o StandardScaler[9], de tal maneira que as variáveis independentes fossem ajustadas para se ter uma média zero e variância unitária. O mesmo padrão de transformação também foi aplicado e no conjunto de dados de testes.

Neste trabalho, sucedeu o trabalho de algoritmos de AM que foram empregados os quais serão elencados abaixo:

- Extra Trees Classifier (ETC): Este algoritmo gera um comitê de árvores de decisão combinadas para realizar predições. Cada árvore é treinada usando uma amostra aleatória do conjunto de dados de treinamento. A predição final para um exemplo fornecido como entrada é uma votação majoritária das predições geradas pelo comitê. Por se tratar, de um meta-estimador que usa um conjunto diversificado de árvores no comitê, o ETC resulta em predições mais robustas. Esse algoritmo é semelhante ao algoritmo de Random Forest (popular em AM), porém tem um menor custo computacional. Cabe mencionar que ocorreu uma customização de alguns hiper-parâmetros, a saber: o número total de árvores no conjunto foi fixado em 100, enquanto a profundidade máxima das árvores foi limitada a 10. Adicionalmente, foi estabelecido um valor de 2 como o número mínimo de amostras requeridas para efetuar a divisão de um nó interno;

- Naive Bayes (NB): Este algoritmo utiliza o teorema de Bayes[10] para calcular a probabilidade de classe dado um conjunto de atributos de cada exemplo a ser classificado. Esse algoritmo assume a independência entre características de entrada, simplificando substancialmente o cálculo das probabilidades de classe e o tornando leve em termos computacionais. Essa característica a torna especialmente aplicável para tarefas de classificação de texto, que envolvem normalmente um número alto de características. É importante notar que os métodos relacionados ao NB não apresentam uma vasta gama

[9] Calcula a média e o desvio padrão em um conjunto de treinamento, de forma em que o mesmo cálculo possa ser replicado posteriormente para a transformação do conjunto de teste

[10] Descreve a probabilidade de um evento, baseado em um conhecimento inicial que pode estar relacionado ao evento.

de hiper-parâmetros disponíveis para ajuste, portanto, o algoritmo está sendo utilizado em sua forma padrão de hiper-parâmetros;

- K-nearest Neighbors (KNN): Este classificador é muito utilizado em aplicações de AM. Dado um exemplo a ser classificado, o algoritmo recupera as instâncias de treinamento mais similares (i.e., seus vizinhos mais próximos), a partir do uso de uma medida de distância ou de similaridade. Se os vizinhos mais próximos forem em sua maioria de uma determinada classe, o exemplo será classificado com essa mesma classe. O aprendizado consiste apenas em armazenar os exemplos de treinamento e recuperar as instâncias mais similares a cada exemplo a ser classificado, sendo definido assim como uma abordagem de aprendizagem baseada em instâncias ou uma aprendizagem não generalizante (i.e., não é criado um modelo de dados como uma árvore de decisão). Os métodos KNN possuem diversos hiper-parâmetros que podem ser personalizados conforme as necessidades do problema em questão. No entanto, é importante notar que no código trabalhado, o modelo está sendo instanciado sem a especificação de hiper-parâmetros personalizados. Isso implica que o algoritmo está utilizando os valores padrão dos hiper-parâmetros predefinidos pelo sistema;

- Support Vector Machine (SVM): Este é um conjunto de métodos de AM comumente utilizados em problemas de classificação e de regressão. Para problemas linearmente separáveis, tendo a utilização de métodos que otimizam uma superfície de separação linear com máxima margem para os exemplos de treinamento. Para problemas não-lineares, o algoritmo utiliza de funções de kernel para mapear os exemplos originais em um espaço de atributos de alta dimensão em que eles sejam então linearmente separáveis. O SVM é particularmente eficaz em espaços de alta dimensão e pode lidar com casos em que o número de dimensões excede o número de amostras. Um modelo SVC (Support Vector Classifier), que é uma implementação do SVM para tarefas de classificação, foi instanciado. No código trabalhado, é relevante destacar que o parâmetro *probability=True* foi habilitado. Essa configuração permite que o modelo calcule as probabilidades de pertencimento a cada classe. É importante ressaltar que os valores padrão dos demais hiper-parâmetros estão sendo empregados no modelo;

- Decision Trees (DT): É um método de AM supervisionado utilizado para problemas de classificação e regressão. A partir dos exemplos de treinamento, o algoritmo cria

um modelo de dados estruturado como uma árvore hierárquica de decisão. Cada nó interno da árvore representa um teste para um atributo específico e cada folha da árvore é associada a uma classificação final, de forma em que os exemplos são classificados realizando os testes de atributos desde a raiz da árvore até alcançar uma folha com a classe a ser associada ao exemplo. As DT é um modelo interpretável diretamente que possui um baixo custo computacional e lida tanto com dados numéricos como categóricos. O modelo foi criado sem a especificação de hiper-parâmetros personalizados, o que indica que ele está sendo utilizado com os valores padrão dos hiper-parâmetros predefinidos;

- Multi-Layer Perceptron (MLP): É uma Rede Neural Artificial (RNA) amplamente eficaz na modelagem de relações não lineares entre os dados de treinamento de classificação e regressão. Uma MLP consiste em pelo menos três camadas de nós de processamento:
  - A camada de entrada, onde os atributos dos exemplos a serem preditos são recebidos e transferidos para as camadas seguintes;
  - As camadas ocultas, contendo nós de processamento (lineares ou não-lineares), aplicados a partir dos dados recebidos da camada anterior;
  - A camada de saída que recebe os resultados obtidos das camadas anteriores e gera as predições finais do modelo para cada exemplo de entrada da rede.

  Cada camada contém pesos (parâmetros) sendo ajustados por algoritmos de aprendizado específicos (e.g., algoritmo Backpropagation), para ajustar os dados de treinamento. Os modelos de MLP apresentam uma variedade de hiper-parâmetros que podem ser ajustados, incluindo o número de camadas e neurônios nas camadas ocultas, a função de ativação, a taxa de aprendizado e outros. No código trabalhado, é relevante observar que o modelo está sendo criado sem a especificação de hiper-parâmetros personalizados, o que implica que ele está utilizando os valores padrão dos hiper-parâmetros predefinidos pelo sistema.

4 EXPERIMENTOS

Nesta seção, serão apresentados os resultados dos experimentos dos classificadores de AM construídos para classificação em EDA. Como os dados reais de uma empresa parceira foram utilizados nos experimentos, não é possível exibir diretamente todas as informações (e.g., características das CRs) por restrições de confidencialidade. Sendo assim, todas as informações confidenciais da empresa foram ocultadas dos resultados da análise.

Foram trabalhados um total de 3.767 CRs distribuídas em três principais categorias, conforme explicadas na seção 3.2.1, seguindo a distribuição da tabela 1 já visualizada na seção 3.2.1.

4.1 METOLODOGIA DE AVALIAÇÃO

Para o trabalho, foi adotada a abordagem Stratified K-Fold Cross Validation (CV) para testar a aplicação dos algoritmos de AM. Nessa abordagem, o conjunto de exemplos é dividido em K subconjuntos. Em cada rodada avaliativa, um modelo de AM é treinado utilizando K-1 subconjuntos, sendo avaliado no subconjunto restante. Esse procedimento é repetido usando cada um dos K subconjuntos de exemplos como base de teste. Para o experimento realizado nessa pesquisa, foi aplicado um valor de K=5.

4.1.1 Métricas de Avaliação

Para a avaliação dos modelos de AM, foram utilizadas duas medidas de desempenho preditivo comumente adotadas na literatura de AM. A primeira medida foi a acurácia, que consiste no cálculo avaliativo de desempenho de um modelo de classificação de AM, que calcula a proporção dos exemplos classificados corretamente pelo modelo de classificação em relação aos número total de exemplos que está contido no conjunto de dados do teste, conforme a equação abaixo:

$$Acurácia = \frac{Total\ de\ acertos}{Número\ de\ exemplos} \tag{4.1}$$

A métrica de acurácia varia de 0 a 1, sendo 0 a acurácia mínima (nenhuma previsão correta) e 1 a acurácia máxima (todas as previsões corretas).

No caso de algoritmos de AM que retornam probabilidades de classe, neste trabalho, foi adotado um limiar de decisão de 0.5 para a classificação de exemplos, i.e., se a probabilidade da classe positiva retornada pelo modelo de AM for maior que 0.5, então a instância de teste é predita como positiva, caso contrário, ela é predita como negativa. Esse limiar adotado é um valor padrão que retorna predições para classe positiva apenas quando a probabilidade da classe positiva retornada por um modelo é maior que a probabilidade da classe negativa.

A segunda métrica de avaliação escolhida foi a área sob a curva ROC, referida na literatura como AUC. Essa métrica é utilizada para avaliar um modelo para diferentes limiares de decisão adotados. A curva ROC representa graficamente a relação entre a taxa de verdadeiros positivos (Sensibilidade) e a taxa de falsos positivos (Especificidade) para diferentes limiares de decisão para a classificação. A medida AUC agrega a taxa de verdadeiros positivos em relação à taxa de falsos positivos. A AUC varia entre 0 e 1 e tem a seguinte interpretação probabilística: o valor da AUC é uma estimativa da probabilidade de uma instância positiva escolhida de maneira aleatória da base de exemplos seja de fato classificada como positiva em relação a uma instância negativa também escolhida de maneira aleatória. Considerando a probabilidade da classe positiva como referência, um valor de AUC igual a 1 é obtido apenas quando todas as probabilidades dos exemplos de teste da classe positiva forem maiores que todas as probabilidades dos exemplos de teste da classe negativa, ou seja, o modelo consegue ordenar de maneira perfeita as instâncias de teste.

4.1.1.1 Avaliação

Os Classificadores foram utilizados neste trabalho de maneira Binária, isto é, de acordo com a Classe de Interesse, sendo esta alocada como Classe Positiva para a representação Binárias, e as demais como Classe Negativa. Neste contexto, foram desenvolvidos três classificadores, conforme serão detalhados seus resultados nas seções 4.2.1. 4.2.2 e 4.2.3; onde cada Classificador teve como Classe Positiva a sua correspondente.

4.2 RESULTADO DOS EXPERIMENTOS

Nesta seção serão abordados os resultados obtidos nos experimentos, tanto usando os algoritmos de maneira binária para predizer cada classe de interesse, conforme as seções 4.2.1, 4.2.2 e 4.2.3.

4.2.1 Classificador - Out of Scope

Para o classificador Out of Scope, o conjunto de dados pode ser dividido da seguinte maneira: Out of Scope (classe positiva) vs ≠Out of Scope (classe negativa), que teve um total de 1801 exemplos da classe positiva vs 1966 exemplos da classe negativa. Nos dados que compõe os dados de treinamento, que representa 70% dos dados, houve um total de 2636 exemplos, sendo estes 1249 da classe positiva vs 1387 da classe negativa. Enquanto que, nos dados que compõe os dados de teste, que representa 30% dos dados, houve um total de 1131 exemplos, sendo estes 552 da classe positiva vs 579 da classe negativa.

Para a realização do cálculo do valor de referência para a acurácia, foi utilizado os valores contidos na base de dados de teste. Contudo, o cálculo foi baseado na classe majoritária que, no caso, foi correspondente a classe negativa, conforme a equação abaixo:

$$\text{Acurácia} = \frac{\text{Total de acertos}}{\text{Número de exemplos}} = \frac{579}{1131} \approx 0,51 \tag{4.2}$$

Os resultados obtidos com cada algoritmo de AM no conjunto de dados Out of Scope demonstraram um desempenho extremamente positivo, já que todos os modelos apresentaram valores de acurácia e AUC superiores aos seus valores de referência (0,51 para acurácia e 0,5 para a AUC). O melhor desempenho em termos de acurácia foi obtido pelo modelo MLP (0,78), seguido pelo ETC (0,75) e DT (0,75). Para a AUC, o SVM obteve o melhor resultado (0,87), seguido pelo ETC (0,85) e MLP (0,86), como pode ser observado na Tabela 2. Apesar de existir uma variação de desempenho dependendo do algoritmo, os resultados do AM foram em geral consistentes, superiores às métricas de referência, indicando uma capacidade robusta desses modelos em classificar corretamente as instâncias de interesse.

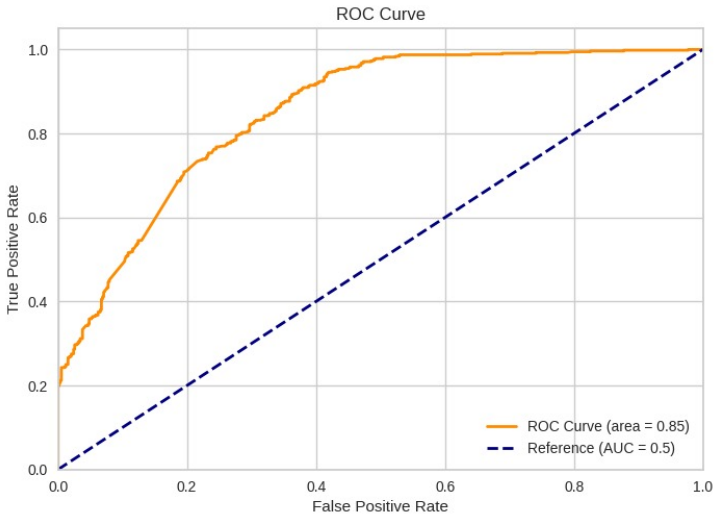
Tabela 2 – Resultados dos algoritmos de AM - Out Of Scope

Classificador	Acurácia	AUC
ETC	0,75	0,85
NB	0,72	0,79
KNN	0,72	0,81
SVM	0,71	0,87
DT	0,75	0,76
MLP	0,78	0,86

Fonte: Elaborado pela autora, 2023

Ao realizar uma análise detalhada da curva ROC representada na Figura 4, é possível observar algumas características relevantes. A curva ROC começa a partir de valores muito rígidos de limiares de decisão adotados pelo modelo de AM. Nesse ponto, CRs são classificadas como Out of Scope somente quando o modelo considera suas previsões bastante confiáveis. Assim, ambos os valores das taxas de verdadeiros positivos e falsos positivos são baixos, indicando que o modelo está sendo conservador na identificação de instâncias como Out of Scope.

Figura 4 – Curva ROC obtida pelo melhor algoritmo de AM - Out of Scope



Fonte: autora, 2023

A curva ROC se inicia com valor de taxa de verdadeiros positivos de aproximadamente 0,2, o que significa que, nesse ponto de decisão, o modelo está capturando corretamente cerca de 20% das instâncias que realmente estão fora do escopo (Out of Scope). Nesse limiar, não ocorrem falsos positivos. Em termos de aplicação, isso significa que o modelo pode ser usando para identificar 20% das CRs que são Out of Scope, sem nenhum falso positivo. Ou seja, é possível descartar do processo do EDA um montante significativo de CRs que são fora de escopo, sem descartar nenhuma CR relevante.

A parte final da curva ROC está relacionada ao uso de limiares de decisão mais baixos. Essa abordagem é aplicada quando o objetivo é maximizar a filtragem de CRs consideradas Out of Scope. No entanto, ao adotar limiares mais baixos, existe o risco de rejeitar CRs que são relevantes para a EDA. Portanto, a seleção do limiar de decisão deve ser cuidadosamente ponderada para equilibrar a detecção eficiente de CRs Out of Scope com a manutenção da relevância das CRs para o EDA.

Essa análise detalhada da curva ROC proporciona uma visão mais clara de como o modelo está se comportando em diferentes pontos de decisão e quais são as taxas de verdadeiros positivos e falsos positivos associadas a esses pontos, auxiliando na interpretação do desempenho do modelo e na escolha do limiar mais adequado para a aplicação específica. A curva ROC indica que, ao se buscar taxas de verdadeiros positivos próximas a 1, ou seja, buscando identificar a maioria das CRs Out of Scope, há um aumento significativo no número de falsos positivos, atingindo 0,5 dos dados. Isso implica que uma quantidade substancial de CRs que são, na verdade, defeitos (DEs) poderia ser erroneamente classificada como Out of Scope e, portanto, excluída do processo.

Nesse contexto, é importante ressaltar que limiares intermediários podem ser adotados para equilibrar sensibilidade e especificidade (por exemplo, ambos com valores de 0,5). A resolução final desse trade-off deve considerar a disponibilidade de recursos e a necessidade de garantir que o processo de EDA seja mais confiável. Em outras palavras, o objetivo é evitar a perda de CRs que podem ser importantes para a análise, ao mesmo tempo em que se controla a quantidade de falsos positivos para otimizar os recursos disponíveis.

4.2.2 Classificador - Core Product Validation

Para o classificador Core Product Validation, o conjunto de dados pode ser dividido da seguinte maneira: Core Product Validation (classe positiva) vs $\neq$Core Product Validation (classe negativa), que teve um total de 1614 exemplos da classe positiva vs 2153 exemplos da classe negativa. Nos dados que compõe os dados de treinamento, que representa 70% dos dados, houve um total de 2636 exemplos, sendo estes 1135 da classe positiva vs 1501 da classe negativa. Enquanto que, nos dados que compõe os dados de teste, que representa 30% dos dados, houve um total de 1131 exemplos, sendo estes 479 da classe positiva vs 652 da classe negativa.

Para a realização do cálculo do valor de referência para a acurácia, foi utilizado os valores contidos na base de dados de teste. Contudo, o cálculo foi baseado na classe majoritária que, no caso, foi correspondente a classe negativa, conforme a equação abaixo:

$$\text{Acurácia} = \frac{\text{Total de acertos}}{\text{Número de exemplos}} = \frac{652}{1131} \approx 0,57 \tag{4.3}$$

Os resultados obtidos pelos algoritmos de AM para a classe Core Validation também foram

geralmente satisfatórios e superiores aos valores de referência (0,57 para acurácia e 0,5 para AUC). O algoritmo NB foi uma exceção, com um desempenho inferior (0,62), enquanto os melhores resultados foram alcançados pelo ETC (0,78) e MLP (0,78), como indicado na tabela 3. Quanto à métrica AUC, os resultados também foram considerados satisfatórios. O SVM (0,88) e o ETC (0,87) apresentaram as melhores pontuações nesse quesito. Por outro lado, o NB (0,67) manteve um resultado abaixo dos outros algoritmos de AM.

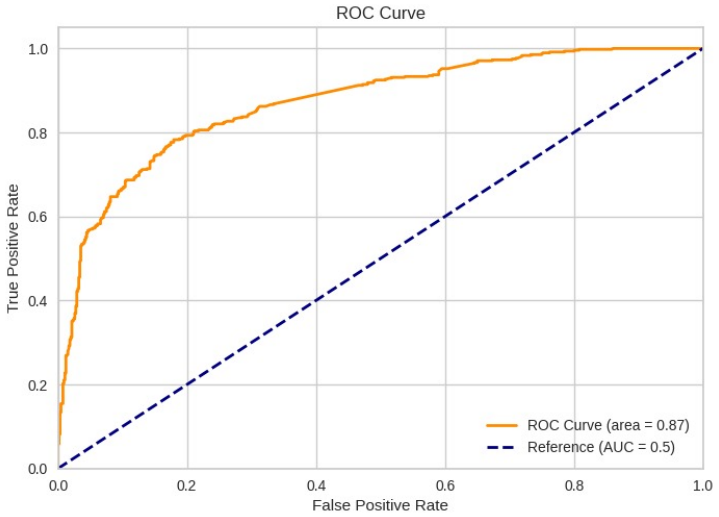
Tabela 3 – Resultados dos algoritmos de AM - Core Product Validation

Classificador	Acurácia	AUC
ETC	0,78	0,87
NB	0,62	0,67
KNN	0,74	0,80
SVM	0,74	0,88
DT	0,78	0,78
MLP	0,78	0,85

Fonte: Elaborado pela autora, 2023

Os resultados sugerem que a maioria dos algoritmos de AM teve sucesso na classificação de CRs em relação ao Core Product Validation, com MLP e ETC apresentando desempenho particularmente forte em várias métricas avaliadas. No entanto, o KNN ficou abaixo dos outros algoritmos, sugerindo que talvez não seja a escolha mais ideal para essa tarefa específica.

Figura 5 – Curva ROC obtida pelo melhor algoritmo de AM - Core Product Validation



Fonte: autora, 2023

Neste experimento, a curva ROC para o rótulo de Core Product Validation também foi

analisada, como mostrado na Figura 5. Assim como no rótulo de Out of Scope, foi observada uma taxa de verdadeiros positivos em torno de 0,2, com um nível baixo de falsos positivos. Isso indica que, nesse modelo, há uma proporção razoável de verdadeiros positivos em relação aos falsos positivos. Por exemplo, uma taxa verdadeira positiva de 0,5 pode ser alcançada com menos de 0,1 de falsos positivos. Portanto, cerca de 50% dos valores correspondentes às CRs de Core Product Validation não estão relacionados a DEs, o que permite que eles sejam descartados do processo de EDA com poucos falsos positivos. Isso economiza tempo de análise para as equipes de teste e desenvolvimento.

Entretanto, a parte final da curva ROC levanta algumas questões sobre a confiança do classificador, semelhante ao que foi observado para o modelo Out of Scope. Nessa região da curva, 100% dos verdadeiros positivos são obtidos apenas a uma taxa de falsos positivos de 0,8. Isso indica que, ao optar por uma abordagem muito rigorosa para minimizar os falsos positivos, pode haver a perda de muitos verdadeiros positivos. Novamente, é recomendado estabelecer um equilíbrio entre Sensibilidade e Especificidade, possivelmente em torno de valores intermediários (0,5), para obter um resultado mais confiável e útil no processo de EDA.

4.2.3 Classificador - Escaped Defects

Para o classificador Escaped Defects, o conjunto de dados pode ser dividido da seguinte maneira: Escaped Defects (classe positiva) vs $\neq$Escaped Defects (classe negativa), que teve um total de 352 exemplos da classe positiva vs 3415 exemplos da classe negativa. Nos dados que compõe os dados de treinamento, que representa 70% dos dados, houve um total de 2636 exemplos, sendo estes 252 da classe positiva vs 2384 da classe negativa. Enquanto que, nos dados que compõe os dados de teste, que representa 30% dos dados, houve um total de 1131 exemplos, sendo estes 100 da classe positiva vs 1031 da classe negativa.

Para a realização do cálculo do valor de referência para a acurácia, foi utilizado os valores contidos na base de dados de teste. Contudo, o cálculo foi baseado na classe majoritária que, no caso, foi correspondente a classe negativa, conforme a equação abaixo:

$$\text{Acurácia} = \frac{\text{Total de acertos}}{\text{Número de exemplos}} = \frac{1031}{1131} \approx 0,91 \tag{4.4}$$

Em relação à classe Escaped Defects, os resultados obtidos para acurácia não foram satisfatórios, uma vez que todos os algoritmos não alcançaram o valor de referência (0,91),

exceto o algoritmo ETC, que conseguiu atingir a acurácia padrão de referência. Para AUC, no entanto, todos os algoritmos ultrapassaram o valor de referência de 0,5. O algoritmo que obteve o melhor desempenho foi novamente o ETC (0,73), conforme visualizado na Tabela 4. Esses resultados sugerem que, para a classe Escaped Defects, o algoritmo ETC demonstrou um desempenho superior em comparação com outros algoritmos. Isso indica que o ETC pode ser uma escolha mais adequada para classificação dessa classe específica.

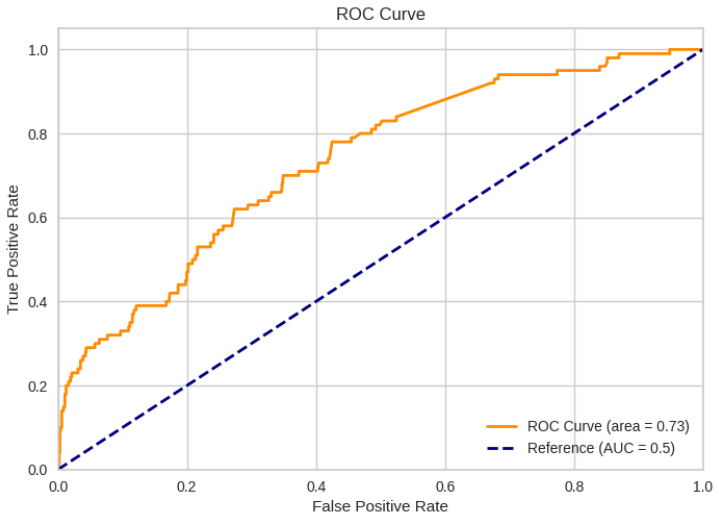
Tabela 4 – Resultados dos algoritmos de AM - Escaped Defects

Classificador	**Acurácia**	**AUC**
ETC	0,91	0,73
NB	0,64	0,56
KNN	0,90	0,65
SVM	0,90	0,67
DT	0,88	0,60
MLP	0,88	0,70

Fonte: Elaborado pela autora, 2023

A análise da curva ROC para a classe Escaped Defects, conforme representado na Figura 6, segue um padrão semelhante ao observado nas classes Out of Scope e Core Product Validation. Ao adotar limiares de decisão menos rigorosos para classificar uma CR como Escaped Defects, é possível obter uma alta taxa de verdadeiros positivos, o que significa que muitas CRs legítimas podem ser identificadas corretamente. No entanto, essa abordagem também resultará em um número significativo de falsos positivos, ou seja, CRs classificadas como Escaped Defects quando, na verdade pertencem a outras classes. Portanto, a escolha do limiar de decisão deve ser equilibrada com base nas necessidades e objetivos específicos do processo de EDA, considerando o trade-off entre a identificação precisa de CRs Escaped Defects e a minimização de falsos positivos.

Figura 6 – Curva ROC obtida pelo melhor algoritmo de AM - Escaped Defects



Fonte: autora, 2023

5 CONCLUSÕES

Este trabalho apresentou uma abordagem para a identificação automática das classes das CRs, com base em um estudo empírico realizado nos casos de teste de cada CR fornecida pelo conjunto de dados da empresa parceira. O objetivo principal deste estudo de caso real foi automatizar a análise das CRs, visando reduzir o tempo e os recursos necessários para essa tarefa e, em simultâneo, identificar a causa raiz dos problemas relacionados às TS para preveni-los no futuro.

No Capítulo 4, foram detalhadas as técnicas utilizadas para a escolha do algoritmo de AM que apresentou os melhores resultados na tarefa de classificação, como descrito na Seção 4.2.

Os resultados dos Classificadores mostraram que os algoritmos ETC e MLP obtiveram os melhores resultados para as classes "Out of Scope"e "Core Product Validation"em todas as métricas avaliadas, como visto nas Seções 4.2.1 e 4.2.2. No entanto, quando se tratou da classe "Escaped Defects", apenas o algoritmo ETC conseguiu atingir o valor de referência, conforme apresentado na Seção 4.2.3, embora tenha obtido um desempenho mediano na métrica AUC.

Para integrar os modelos de AM na ferramenta de EDA da empresa parceira, foi essencial estabelecer uma comunicação direta com a equipe de Engenheiros de Software. Esse diálogo foi crucial para viabilizar a aplicação direta dos algoritmos de AM no código-fonte do EDA. Estima-se, ainda, que a economia de tempo e recurso venha a ser considerável, tendo em vista que o algoritmo de AM consegue rótulas as CRs, contudo, cabe ao testador a resposta final se uma CR é um DE ou não. Até o momento em que este trabalho foi entregue, os algoritmos de AM ainda estava em etapa de testes para a verificação de harmonia entre a classificação das CRs entre homem e máquina.

Quanto à aplicação de algoritmos de AM na resolução de problemas, é importante destacar uma certa limitação, como a necessidade de ajustar os dados de acordo com a funcionalidade específica de cada algoritmo. Essa questão pode ser superada por meio da aplicação de técnicas de pré-processamento, conforme abordado nesta pesquisa, visando a melhoria da qualidade dos dados. Além disso, a base de exemplos necessita ser construída e atualizada, sendo estimado um ciclo periódico de seis meses, dado que é o período típico para as atualizações dos dados das CRs.

5.1 TRABALHOS FUTUROS

Um dos principais trabalhos futuros é a implementação dos algoritmos de AM na ferramenta de EDA e a observação do desempenho dos classificadores em tempo real. É crucial verificar se os resultados obtidos na fase de experimentação se manterão satisfatórios quando os algoritmos estiverem em produção, lidando com novos casos de CR à medida que são criados.

Incluir a descrição das CRs no treinamento de dados é uma excelente sugestão para melhorar a precisão do classificador. A descrição contém informações valiosas que podem ser relevantes para a identificação das classes das CRs. Ao considerar a descrição, o classificador terá acesso a um conjunto mais amplo de características textuais que podem ser úteis para tomar decisões mais informadas.

Além disso, a utilização das labels das CRs como parte do treinamento também é uma estratégia promissora. As labels podem conter informações contextuais importantes sobre o conteúdo das CRs e ajudar o classificador a entender melhor o contexto em que essas CRs foram criadas. Isso pode ser particularmente útil quando as CRs não têm informações suficientes em seu resumo ou quando os resumos são ambíguos.

Portanto, incluir a descrição das CRs e explorar as labels como recursos adicionais para treinamento são abordagens valiosas para melhorar a qualidade das previsões do classificador. Essas etapas adicionais podem levar a resultados mais precisos e robustos na identificação das classes das CRs.

Outra direção para o trabalho futuro é a combinação dos três melhores classificadores identificados, ou seja, ETC, MLP e SVM. Embora o SVM tenha apresentado um desempenho um pouco inferior em termos de acurácia em comparação com os dois primeiros, ele ainda pode contribuir para melhorar a precisão e a assertividade do sistema de classificação. A combinação de classificadores pode ser realizada com base nos rótulos de classe, utilizando diferentes métodos, como voto majoritário, borda count, soma, média, produto, máximo e mínimo. Cada método de combinação pode produzir resultados diferentes, e a escolha do método mais adequado dependerá das características específicas do problema e dos objetivos. Vale destacar que, com exceção do voto majoritário e borda count, os classificadores são capazes de fornecer saídas com probabilidades, o que pode ser útil na combinação dos resultados (KITTLER et al., 1998; JAIN; DUIN; MAO, 2000). Esse processo de combinação deve ser realizado após a implementação dos classificadores na ferramenta de EDA.

A expectativa é que essas etapas adicionais podem aprimorar ainda mais o sistema de

classificação, tornando-o mais robusto e eficaz na identificação das classes das CRs.

REFERÊNCIAS

ABRAN, A.; MOORE, J. W.; BOURQUE, P.; DUPUIS, R.; TRIPP, L. Software engineering body of knowledge. *IEEE Computer Society, Angela Burgess*, v. 25, 2004.

ANVIK, J. Automating bug report assignment. In: *Proceedings of the 28th international conference on Software engineering*. [S.l.: s.n.], 2006. p. 937–940.

ATLASSIAN. *What is the issue view? | Jira Work Management Cloud | Atlassian Support*. <https://support.atlassian.com/jira-work-management/docs/what-is-the-new-jira-issue-view/>. (Accessed on 09/01/2023).

BRUNIALTI, L.; PERES, S.; FREIRE, V.; LIMA, C. Aprendizado de maquina em sistemas de recomendacao baseados em conteudo textual: Uma revisao sistematica. *Anais do XI Simpósio Brasileiro de Sistemas de Informação*, SBC, p. 203–210, 2015.

CAI, D.; HE, X. Manifold adaptive experimental design for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 24, n. 4, p. 707–719, 2011.

CAVALCANTI, Y. C.; NETO, P. A. da M. S.; LUCRÉDIO, D.; VALE, T.; ALMEIDA, E. S. de; MEIRA, S. R. de L. The bug report duplication problem: an exploratory study. *Software Quality Journal*, Springer, v. 21, p. 39–66, 2013.

CAVALCANTI, Y. C.; NETO, P. A. da M. S.; MACHADO, I. d. C.; VALE, T. F.; ALMEIDA, E. S. de; MEIRA, S. R. d. L. Challenges and opportunities for software change request repositories: a systematic mapping study. *Journal of Software: Evolution and Process*, Wiley Online Library, v. 26, n. 7, p. 620–653, 2014.

CORCOVIA, L. O.; ALVES, R. S. Aprendizagem de máquina e mineração de dados: avaliação de métodos de aprendizagem. *Revista Interface Tecnológica*, v. 16, n. 1, p. 90–101, 2019.

CRUZ, F. *Scrum e Agile em Projetos (2a. edição): guia completo*. [S.l.]: Brasport, 2018.

DALAL, S. R.; JAIN, A.; KARUNANITHI, N.; LEATON, J.; LOTT, C. M.; PATTON, G. C.; HOROWITZ, B. M. Model-based testing in practice. In: *Proceedings of the 21st international conference on Software engineering*. [S.l.: s.n.], 1999. p. 285–294.

DAWSON, M.; BURRELL, D. N.; RAHIM, E.; BREWSTER, S. Integrating software assurance into the software development life cycle (sdlc). *Journal of Information Systems Technology and Planning*, v. 3, n. 6, p. 49–53, 2010.

DUARTE, D.; STÅHL, N. Machine learning: a concise overview. *Data Science in Practice*, Springer, p. 27–58, 2019.

EASTWOOD, A. Firm fires shots at legacy systems. *Computing Canada*, v. 19, n. 2, p. 17, 1993.

GOYAL, A.; SARDANA, N. Empirical analysis of ensemble machine learning techniques for bug triaging. In: IEEE. *2019 Twelfth International Conference on Contemporary Computing (IC3)*. [S.l.], 2019. p. 1–6.

HERZIG, K.; JUST, S.; ZELLER, A. It's not a bug, it's a feature: how misclassification impacts bug prediction. In: IEEE. *2013 35th international conference on software engineering (ICSE)*. [S.l.], 2013. p. 392–401.

HIEW, L.; MURPHY, G. C.; ANVIK, J. Who should fix this bug? In: IEEE COMPUTER SOCIETY. *Software Engineering, International Conference on*. [S.l.], 2006. p. 361–370.

IHARA, A.; OHIRA, M.; MATSUMOTO, K.-i. An analysis method for improving a bug modification process in open source software development. In: *Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops*. [S.l.: s.n.], 2009. p. 135–144.

INDEZEICHAK, V. et al. *Análise do controle estatístico da produção para empresa de pequeno porte: um estudo de caso*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2005.

JAIN, A. K.; DUIN, R. P. W.; MAO, J. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, Ieee, v. 22, n. 1, p. 4–37, 2000.

KAGDI, H.; POSHYVANYK, D. Who can help me with this change request? In: IEEE. *2009 IEEE 17th International Conference on Program Comprehension*. [S.l.], 2009. p. 273–277.

KITTLER, J.; HATEF, M.; DUIN, R. P.; MATAS, J. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 20, n. 3, p. 226–239, 1998.

KÖKSAL, Ö.; ÖZTÜRK, C. E. A survey on machine learning-based automated software bug report classification. In: IEEE. *2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. [S.l.], 2022. p. 635–640.

LINS, R. F.; BARROS, F. A.; PRUDÊNCIO, R. B.; MELO, W. N. Automatic classification of bug reports for mobile devices: An industrial case study. In: SBC. *Anais do XIX Encontro Nacional de Inteligência Artificial e Computacional*. [S.l.], 2022. p. 728–739.

LUCCA, G. A. D.; PENTA, M. D.; GRADARA, S. An approach to classify software maintenance requests. In: IEEE. *International Conference on Software Maintenance, 2002. Proceedings.* [S.l.], 2002. p. 93–102.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. *Estudos Avançados*, SciELO Brasil, v. 35, p. 85–94, 2021.

MAFRA, J. N. D. Regression test selection to reduce escaped defects. 2008.

MAGALHÃES, C.; ANDRADE, J.; PERRUSI, L.; MOTA, A.; BARROS, F.; MAIA, E. Hsp: A hybrid selection and prioritisation of regression test cases based on information retrieval and code coverage applied on an industrial case study. *Journal of Systems and Software*, Elsevier, v. 159, p. 110430, 2020.

MAGALHÃES, C.; BARROS, F.; MOTA, A.; MAIA, E. Automatic selection of test cases for regression testing. In: *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing*. [S.l.: s.n.], 2016. p. 1–8.

MARTINS, P. G.; LAUGENI, F. P. Administração da produção. Saraiva São Paulo, 2005.

MILLER, K. Data-driven decision making: A primer for beginners. *Northeastern University Graduate Programs*, 2019.

MITCHELL, T. M. *Machine learning*. [S.l.]: McGraw-hill, 1997.

MOAD, J. Maintaining the competitive edge. *Datamation*, CAHNERS-DENVER PUBLISHING CO 8773 S RIDGELINE BLVD, HIGHLANDS RANCH, CO . . . , v. 36, n. 4, p. 61, 1990.

MURPHY, G.; CUBRANIC, D. Automatic bug triage using text categorization. In: CITESEER. *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*. [S.l.], 2004. p. 1–6.

MYERS, G. J.; BADGETT, T.; THOMAS, T. M.; SANDLER, C. *The art of software testing*. [S.l.]: Wiley Online Library, 2004. v. 2.

PAN, J. Software testing, dependable embedded systems. *Electrical and Computer Engineering Department, Carnegie Mellon University*, 1999.

PEREZ, Q.; JEAN, P.-A.; URTADO, C.; VAUTTIER, S. Bug or not bug? that is the question. In: IEEE. *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*. [S.l.], 2021. p. 47–58.

PIGOSKI, T. M. *Practical software maintenance: best practices for managing your software investment*. [S.l.]: Wiley Publishing, 1996.

RIBEIRO, R. H. Identificação de bugs em código-fonte usando aprendizagem de máquina. Universidade Federal da Fronteira Sul, 2021.

SAAPUNKI, K. Root cause analysis and escape defect analysis improvement at continuous delivery by data-driven decision-making. 2023.

SARAWAN, K.; POLPINIJ, J.; LUAPHOL, B. Machine learning-based methods for identifying bug severity level from bug reports. In: SPRINGER. *International Conference on Computing and Information Technology*. [S.l.], 2023. p. 199–208.

TOUATI, A.; BOSIO, A.; GIRARD, P.; VIRAZEL, A.; BERNARDI, P.; REORDA, M. S.; AUVRAY, E. Scan-chain intra-cell aware testing. *IEEE Transactions on Emerging Topics in Computing*, v. 6, n. 2, p. 278–287, 2018.

UDDIN, K. A.; KOWSHER, M.; SAKIB, K. Bsdrm: A machine learning based bug triaging model to recommend developer team. In: SPRINGER. *International Conference on Machine Intelligence and Emerging Technologies*. [S.l.], 2022. p. 256–270.

VANDERMARK, M. A. *Defect Escape Analysis: Test Process Improvement*. [S.l.]: STAREAST, 2003.

VICCARI, L. D. *Automacao de teste de software atraves de linhas de produtos e testes baseados em modelos*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, 2009.

WEISS, A. E. *Key business solutions: essential problem-solving tools and techniques that every manager needs to know*. [S.l.]: Pearson UK, 2012.

YU, J.; SHEN, Y.; XIE, J. Mining user interest and its evolution for recommendation on the micro-blogging system. In: SPRINGER. *Web-Age Information Management: 14th International Conference, WAIM 2013, Beidaihe, China, June 14-16, 2013. Proceedings 14*. [S.l.], 2013. p. 679–690.

ZHANG, J.; WANG, X.; HAO, D.; XIE, B.; ZHANG, L.; MEI, H. A survey on bug-report analysis. *Sci. China Inf. Sci.*, Citeseer, v. 58, n. 2, p. 1–24, 2015.